SLEZSKÁ UNIVERZITA
FILOZOFICKO-PŘÍRODOVĚDECKÁ FAKULTA V OPAVĚ

Šárka Vavrečková

Skripta do předmětu

Počítačové sítě a decentralizované systémy

Ústav informatiky
Filozoficko-přírodovědecká fakulta v Opavě
Slezská univerzita v Opavě

Opava
11. května 2020

Anotace: Tato skripta jsou určena pro studenty předmětu *Počítačové sítě a decentralizované systémy* pro navazující magisterské studium na Ústavu informatiky Slezské univerzity v Opavě. V předmětu se zabýváme prostředky a postupy používanými v počítačových sítích, navazujeme na obdobný předmět z bakalářského stupně studia.

Počítačové sítě a decentralizované systémy

RNDr. Šárka Vavrečková, Ph.D.

Dostupné na: http://vavreckova.zam.slu.cz/sitedec.html

Ústav informatiky
Filozoficko-přírodovědecká fakulta v Opavě
Slezská univerzita v Opavě
Bezručovo nám. 13, Opava

Sázeno v systému LaTeX

Předmluva

Co najdeme v těchto skriptech

Tato skripta jsou určena pro studenty Ústavu informatiky Slezské univerzity v Opavě. Obsahují látku vyučovanou na přednáškách předmětu *Počítačové sítě a decentralizované systémy*, ve kterém se zabýváme (jak název napovídá) počítačovými sítěmi.

Některé oblasti jsou „navíc“ (jsou označeny ikonami fialové barvy), ty nejsou probírány a ani se neobjeví na zkoušce – jejich úkolem je motivovat k dalšímu samostatnému studiu či pokusům nebo pomáhat v budoucnu při získávání dalších informací. Pokud je fialová ikona před názvem kapitoly (sekce), platí pro vše, co se v dané kapitole či sekci nachází.

Značení

Ve skriptech se používají následující barevné ikony:

- Nové *pojmy*, značení apod. jsou značeny modrým symbolem, který vidíme zde vlevo.
- Konkrétní *postupy* a nástroje, způsoby řešení různých situací, do kterých se může správce počítačového vybavení dostat, atd. jsou značeny také modrou ikonou.
- Některé části textu jsou označeny fialovou ikonou, což znamená, že jde o *nepovinné úseky*, které nejsou probírány (většinou; studenti si je mohou podle zájmu vyžádat nebo sami prostudovat). Jejich účelem je dobrovolné rozšíření znalostí studentů o pokročilá témata, na která obvykle při výuce nezbývá moc času.
- Žlutou ikonou jsou označeny odkazy, na kterých lze získat *další informace* o tématu. Nejčastěji u této ikony najdeme webové odkazy na stránky, kde se dané tématice jejich autoři věnují podrobněji.
- Červená je ikona pro *upozornění* a poznámky.

Pokud je množství textu patřícího k určité ikoně větší, je celý blok ohraničen prostředím s ikonami na začátku i konci, například pro definování nového pojmu:

Definice

V takovém prostředí definujeme pojem či vysvětlujeme sice relativně známý, ale komplexní pojem s více významy či vlastnostmi.

Podobně může vypadat prostředí pro delší postup nebo delší poznámku či více odkazů na další informace. Mohou být použita také jiná prostředí:

Příklad

Takto vypadá prostředí s příkladem, obvykle nějakého postupu. Příklady jsou obvykle komentovány, aby byl jasný postup jejich řešení.

Úkol

Otázky a úkoly, náměty na vyzkoušení, které se doporučuje při procvičování učiva provádět, jsou uzavřeny v tomto prostředí. Pokud je v prostředí více úkolů, jsou číslovány.

Obsah

Předmluva **iii**

1 Pojmy a standardy k počítačovým sítím **1**
1.1 Přehled základních pojmů a postupů . . . 1
1.2 Aktivní síťové prvky . . . 2
1.3 Co a jak přenášíme po počítačové síti . . . 3
1.3.1 Protokoly . . . 3
1.3.2 Vlastnosti protokolů . . . 4
1.4 Standardy a standardizační instituce . . . 5
1.4.1 Standardizační instituce pro počítačové sítě . . . 5
1.4.2 Volně dostupné standardy . . . 8
1.5 Referenční model ISO/OSI . . . 11
1.5.1 Přehled modelu . . . 11
1.5.2 Spolupráce protokolů . . . 14
1.5.3 Protokolové zásobníky . . . 18
1.6 Síťový model TCP/IP . . . 19
1.7 Charakteristiky přenosu a přenosových cest . . . 21
1.7.1 Přenos v základním a přeloženém pásmu . . . 21
1.7.2 Multiplex . . . 25

2 Lokální sítě – Ethernet **28**
2.1 Co je to Ethernet . . . 28
2.2 Komunikace v Ethernetu . . . 29
2.3 Ethernet na linkové vrstvě . . . 31
2.3.1 Adresace na vrstvě L2 . . . 31
2.3.2 Podvrstvy L2 . . . 32
2.3.3 EtherType . . . 33
2.3.4 Formát ethernetového rámce . . . 34
2.3.5 Tabulka MAC adres na switchi . . . 38
2.4 Ethernet na fyzické vrstvě . . . 40
2.4.1 Křížení a krimpování . . . 41

2.4.2 Parametry . . . 42
2.4.3 Implementace fyzické vrstvy . . . 44
2.5 Průběh přenosu . . . 45
2.5.1 Přenos v polovičním duplexu . . . 46
2.5.2 Přenos v plném duplexu . . . 49
2.6 Technologie související s Ethernetem . . . 50
2.6.1 Autonegociace . . . 50
2.6.2 Napájení přes Ethernet . . . 50
2.6.3 EFM . . . 51
2.6.4 Strukturovaná kabeláž . . . 51

3 Další témata k lokálním sítím **54**
3.1 Adresy v rámcích a paketech . . . 54
3.2 Přehled rodiny standardů IEEE 802 . . . 55
3.3 Další lokální sítě . . . 57
3.3.1 Token Ring . . . 57
3.3.2 100VG-AnyLAN . . . 58
3.4 VLAN . . . 59
3.4.1 VLAN na jednom switchi . . . 61
3.4.2 VLAN rámce na cestě přes trunk . . . 63
3.4.3 Komunikace mezi různými VLAN sítěmi . . . 65
3.5 STP – kostra v grafu . . . 67
3.5.1 Protokol STP . . . 68
3.5.2 Konvergence sítě switchů . . . 70
3.5.3 Varianty protokolu STP . . . 73
3.6 EtherChannel . . . 75
3.6.1 Vlastnosti . . . 75
3.6.2 Řízení toku . . . 75
3.6.3 Vytvoření kanálu . . . 76
3.7 SAN sítě . . . 78
3.7.1 Fibre Channel . . . 78
3.7.2 iSCSI . . . 79

4 Rozlehlé sítě a telekomunikace **80**
4.1 WAN sítě . . . 80
4.1.1 Struktura WAN sítí . . . 80
4.1.2 Protokoly vrstvy L2 pro WAN sítě . . . 81
4.2 Nejznámější WAN sítě . . . 83
4.2.1 Frame Relay . . . 83
4.2.2 ATM . . . 86
4.2.3 MPLS – přepínání značek . . . 87
4.3 Infrastruktura optických sítí . . . 92
4.3.1 SONET/SDH . . . 92
4.3.2 WDM . . . 92

4.4 Protokol PPP . . . 94
4.4.1 Vlastnosti PPP . . . 94
4.4.2 Rozšíření PPP . . . 95
4.5 Telekomunikační sítě . . . 96
4.5.1 Využití telekomunikačních sítí pro přenos dat . . . 96
4.5.2 Slučování linek . . . 97
4.6 Přístupové telekomunikační sítě . . . 98
4.6.1 ADSL . . . 99
4.6.2 VDSL . . . 103
4.6.3 Další xDSL technologie . . . 103
4.7 Pobočkové ústředny . . . 104

5 Bezdrátové a mobilní sítě 106
5.1 Bezdrátové technologie . . . 106
5.2 Wi-fi . . . 107
5.2.1 Access point . . . 108
5.2.2 Fyzická vrstva . . . 109
5.2.3 Interference . . . 114
5.2.4 Diagnostické nástroje pro fyzickou vrstvu . . . 115
5.2.5 Signál a antény . . . 117
5.2.6 Identifikátory a služby . . . 118
5.2.7 Přístupová metoda . . . 119
5.2.8 Wi-fi rámce . . . 120
5.3 Zabezpečení bezdrátové komunikace . . . 124
5.3.1 AAA . . . 124
5.3.2 WPS . . . 127
5.3.3 Jak tedy zabezpečit bezdrátovou síť . . . 128
5.4 Centrálně řízená bezdrátová síť . . . 128
5.5 WiMAX . . . 129
5.6 Mobilní technologie . . . 131
5.6.1 Struktura mobilní sítě . . . 131
5.6.2 První generace (1G) . . . 132
5.6.3 Druhá generace (2G) . . . 132
5.6.4 Přechodová generace (2.5G) . . . 133
5.6.5 Třetí generace (3G) . . . 134
5.6.6 Čtvrtá generace . . . 135

6 Síťová vrstva 137
6.1 Síťová vrstva a logické adresy . . . 137
6.2 Protokol IPv4 . . . 138
6.2.1 Adresy IPv4 . . . 138
6.2.2 Speciální adresy podle IPv4 . . . 139
6.2.3 IPv4 pakety . . . 139
6.2.4 Pole TTL a životnost paketu . . . 141

6.2.5 MTU a fragmentace IPv4 paketu . . . 141
6.2.6 NAT a soukromé adresy . . . 144
6.2.7 Jak získat IPv4 adresu . . . 146
6.3 Protokol IPv6 . . . 146
6.3.1 Adresy IPv6 . . . 147
6.3.2 Speciální adresy podle IPv6 . . . 149
6.3.3 IPv6 pakety . . . 150
6.3.4 Jak získat IPv6 adresu . . . 152
6.4 Adresní rozsah IPv4 . . . 154
6.4.1 Třídy . . . 154
6.4.2 Podsíťování . . . 156
6.4.3 VLSM . . . 158
6.4.4 CIDR . . . 160
6.5 Protokol ICMP a zprávy . . . 162
6.5.1 Účel protokolu ICMP . . . 162
6.5.2 ICMP verze 6 . . . 164
6.5.3 Testování dosažitelnosti . . . 165
6.6 Správa skupin . . . 167
6.7 Objevování sousedů . . . 169
6.7.1 Tabulky sousedů . . . 169
6.7.2 K protokolům . . . 170
6.7.3 Bezpečnost . . . 171
6.8 Jak funguje router . . . 171

7 Transportní vrstva **173**
7.1 Komunikace typu klient-server . . . 173
7.2 Čísla portů . . . 173
7.3 Protokoly transportní vrstvy . . . 174
7.4 Protokol TCP . . . 175
7.4.1 TCP segment . . . 175
7.4.2 Průběh TCP spojení . . . 179
7.5 Protokol UDP . . . 182

8 Některé aplikační protokoly **183**
8.1 Služba WWW a protokol HTTP . . . 183
8.1.1 Adresace . . . 183
8.1.2 HTTP zprávy . . . 184
8.1.3 Komunikace podle HTTP . . . 184
8.1.4 Informační kódy HTTP . . . 187
8.2 Služby elektronické pošty . . . 188
8.2.1 Infrastruktura . . . 188
8.2.2 Protokoly . . . 189
8.2.3 Komunikace a nastavení . . . 190
8.2.4 MIME . . . 191

8.3 Souborové služby . . . 191
8.3.1 Protokol FTP . . . 191
8.3.2 Sdílení prostředků v lokální síti . . . 193
8.4 Přidělování IP adres a protokol DHCP . . . 193
8.5 Další typy serverů . . . 195
8.6 Vzdálená konfigurace . . . 196
8.6.1 Telnet . . . 196
8.6.2 SSH . . . 199
8.7 Přehled protokolů a portů . . . 200

9 Decentralizované a distribuované systémy **201**
9.1 Pojmy . . . 201
9.1.1 Typy systémů . . . 201
9.1.2 Distribuované systémy . . . 202
9.2 Bridging, switching, routing . . . 203
9.3 Směrování . . . 204
9.3.1 Jak směrujeme . . . 204
9.3.2 Autonomní systém . . . 206
9.3.3 Směrovací algoritmy . . . 206
9.3.4 Směrovací protokol RIP . . . 210
9.3.5 Směrovací protokoly IGRP a EIGRP . . . 211
9.3.6 Směrovací protokol OSPF . . . 213
9.3.7 Směrovací protokol IS-IS . . . 215
9.3.8 Směrovací protokol BGP . . . 215
9.3.9 Softwarový směrovač . . . 216
9.4 Mechanismus DNS . . . 217
9.4.1 Domény a jmenné adresy . . . 217
9.4.2 Zóny a DNS servery . . . 218
9.4.3 DNS dotazy . . . 219
9.4.4 Tabulka hostitelů . . . 221
9.4.5 Protokol DNS a DNS paket . . . 222
9.4.6 Bezpečnost v DNS . . . 223
9.4.7 Databáze WHOIS . . . 224
9.5 Adresářové služby a Active Directory . . . 225
9.6 QoS . . . 227
9.7 VoIP a videotelefonie . . . 229
9.7.1 Princip . . . 229
9.7.2 Protokoly . . . 229
9.7.3 Videotelefonie a videokonference . . . 231
9.8 CESNET2 . . . 233
9.9 Bezpečnostní týmy . . . 234

10 Bezpečnost a správa **236**
10.1 Typy útoků . . . 236

10.2 Síťový analyzátor . . . 239
10.3 Firewall . . . 242
10.3.1 Princip firewallu . . . 242
10.3.2 Typy filtrování . . . 243
10.3.3 OpenWRT . . . 247
10.4 VPN . . . 247
10.4.1 IPSec VPN – na síťové vrstvě . . . 248
10.4.2 GRE tunely . . . 250
10.4.3 L2TP tunely . . . 251
10.4.4 OpenVPN . . . 252
10.4.5 SSH . . . 252
10.4.6 MPLS VPN . . . 253
10.4.7 Další možnosti řešení VPN . . . 253
10.5 Správa sítě . . . 255
10.5.1 NMS . . . 255
10.5.2 ISO model řízení sítě . . . 256
10.5.3 Management v ISO/OSI . . . 256
10.6 Management v TCP/IP . . . 257
10.6.1 SNMP . . . 257
10.6.2 MIB-II . . . 258
10.6.3 Používání protokolu SNMP . . . 261
10.7 Syslog . . . 262
10.8 Monitorování sítě . . . 266
10.8.1 Protokol RMON . . . 266
10.8.2 Snort . . . 267
10.8.3 NetFlow . . . 269
10.9 WBEM . . . 270
10.9.1 Princip WBEM . . . 270
10.9.2 WMI . . . 271
10.10 Dohledové systémy . . . 273
10.10.1 Nagios . . . 274
10.10.2 Zabbix . . . 275
10.10.3 OpenNMS . . . 275
10.10.4 Zenoss . . . 275
10.10.5 Cacti . . . 276
10.11 SIEM . . . 277

Seznam doporučené literatury **280**

Přílohy **282**

A Práce s adresami a sítěmi ve Windows **283**
A.1 Problémy při používání příkazů ve Windows . . . 283
A.2 Soubory související se sítěmi . . . 283

A.3 Práce s adresami a síťovými kartami . . . 284
A.3.1 Základní práce s IP adresami `--ipconfig` . . . 284
A.3.2 Překlad adres `--arp` a `nslookup` . . . 286
A.3.3 Směrování – `route` . . . 288
A.3.4 Malá síť a skupina (workgroup) . . . 290
A.4 Testování a statistiky . . . 292
A.4.1 Testování cesty a průchodnosti . . . 292
A.4.2 Statistiky v `netstatu` . . . 293
A.5 Služba WMI a program `wmic` . . . 295
A.6 Firewall ve Windows . . . 298
A.7 Další příkazy . . . 300

B Práce s adresami a sítěmi v Linuxu 301
B.1 Specializované distribuce . . . 301
B.2 Soubory související se sítěmi . . . 303
B.3 Starší příkazy pro práci s adresami . . . 305
B.4 Mechanismus `iproute2` (příkaz `ip`) – adresy, síť, směrování . . . 307
B.4.1 Konfigurace síťového rozhraní a adres . . . 307
B.4.2 Směrování a filtrování . . . 309
B.4.3 Objevování sítě . . . 314
B.4.4 Tunely . . . 316
B.5 Překlad názvů . . . 319
B.5.1 Název počítače . . . 319
B.5.2 Resolver a soubor `resolv.conf` . . . 319
B.5.3 Testování DNS . . . 320
B.5.4 Zjistění informací o doméně . . . 323
B.6 Testování a statistiky . . . 324
B.6.1 Základní nástroje . . . 324
B.6.2 Pokročilé testování . . . 327
B.7 Firewall v Linuxu . . . 327
B.7.1 Princip firewallu . . . 327
B.7.2 Základní vnitřní příkazy a parametry pro filtrování . . . 329
B.7.3 SPI . . . 333
B.7.4 NAT a maškaráda . . . 336
B.7.5 Označování paketů . . . 337
B.7.6 Skripty . . . 339
B.7.7 Uložení změn . . . 339
B.8 Vzdálený přístup . . . 340
B.9 Další příkazy . . . 342

Kapitola 1

Pojmy a standardy k počítačovým sítím

V této kapitole si pouze ujasníme či připomeneme pojmy, které bychom už měli znát z bakalářského studia.

Dále budeme používat pojem počítačová síť, ale to je nepřesné, ve skutečnosti zde ani zdaleka nepůjde pouze o počítače (nicméně ten pojem je poměrně vžitý).

1.1 Přehled základních pojmů a postupů

V síti máme samozřejmě zařízení nacházející se na začátku nebo konci komunikační cesty – koncová zařízení. Jsou to počítače, servery, tablety, mobily, síťové tiskárny, zařízení internetu věcí a další. Dále do počítačové sítě patří síťové prvky, přes které komunikace probíhá. Rozlišujeme

- *aktivní síťové prvky* – aktivně ovlivňují komunikaci, směrují, zesilují signál apod.,
- *pasivní síťové prvky* – pouze „pasivně přenesou“ data, například kabeláž.

Definice (Počítačová síť)

Počítačová síť je soustava zařízení – *uzlů* (node) vzájemně propojených *přenosovými cestami* (transmission path). Uzel v počítačové síti je buď koncové zařízení nebo aktivní síťový prvek.

Následuje přehled pojmů, o kterých bude v dalším textu předpokládáno, že je čtenář zná a chápe.

- spoj (link), přenosový kanál (channel), přenosový okruh (circuit), pronajatý okruh (leased circuit),
- poskytovatel síťové konektivity (ISP – Internet Service Provider),
- přenosový signál, vlnová délka, frekvence, amplituda, šířka pásma,
- fyzický/pevný okruh (physical circuit), virtuální okruh (virtual circuit),
  - pevný virtuální okruh (PVC, permanent virtual circuit)
  - přepínaný virtuální okruh (SVC, switched virtual circuit)
- sítě PAN, LAN, MAN, WAN,
- přenos simplexní, plně duplexní (full duplex), poloduplexní (half duplex),
- přenos proudový (stream) a paketový (blokový, block, packet),
- přepojování okruhů (circuit switching), přepojování paketů (packet switching),

- přenos spojovaný (služba se spojením – connection-oriented) nebo nespojovaný (connectionless),
- datagramová služba,
- přenos spolehlivý (reliable) nebo nespolehlivý (unreliable), best effort (nejlepší snaha),
- komunikace typu unicast, multicast, broadcast, anycast,
- fyzická a logická topologie, druhy: sběrnice, kruh, hvězda, strom, mesh, point-to-point,
- páteřní vedení,
- síťové rozhraní,
- referenční model ISO/OSI, síťový model TCP/IP (obojí bude ještě dále připomenuto),
- jednotlivé typy kabelů (twisted pair, optika, koaxiál, twin-ax) – typické vlastnosti, použití,
- u twisted pair kategorie, nestíněný (UTP), různé druhy stíněného, kabel typu lanko (licna) vs. drát.

Taktéž se předpokládá, že čtenář zvládá minimálně následující činnosti:

- jak se pracuje v binární a hexadecimální číselné soustavě,
- jak zjistit svou MAC a IP adresu v grafickém i textovém režimu,
- jak nastavit svou IP adresu, masku, adresu brány, adresu DNS serveru,
- subnetting – počítání rozsahů adres, alespoň pro IPv4,
- jak se používá Wireshark (https://www.wireshark.org/download.html),
- jak se krimpuje kabel typu twisted pair na RJ-45 (8p8c) včetně testování,
- jak se používá buď Packet Tracer nebo jiný obdobný program pro simulaci sítě a její konfigurace,
- jak se zjišťuje dostupnost zařízení na síti a cesta k němu (`ping`, atd.),
- jak se dá zjistit, které procesy naslouchají na některém portu, a jak se vypisují různé statistiky související se sítí,
- jak u Wi-fi zjistíte obsazenost kanálů a informace o AP.

1.2 Aktivní síťové prvky

Následuje přehled běžných aktivních síťových zařízení. Ke každému je taktéž přidána informace o tom, na kterou vrstvu ISO/OSI patří.

Repeater (opakovač) je jednoduchý aktivní síťový prvek se dvěma porty. Příchozí signál zesílí (případně znovu vygeneruje) a pošle dál. Pracuje na vrstvě L1.

Dnes se s repeatery setkáváme spíše u bezdrátových lokálních sítí, ale také např. HDMI, kde slouží ke zvýšení dosahu signálu.

Hub (rozbočovač) je jednoduchý aktivní síťový prvek, který má typicky více než dva porty. Cokoliv přijde na některý port, jen přepošle na všechny ostatní porty (tento signál zesílí nebo znovu vygeneruje). Pracuje pouze se signálem – zesílí nebo znovu vygeneruje signál. Pracuje na vrstvě L1.

Jeho výhodou je rychlost, nevýhodou je, že zbytečně zahlcuje síť (paket pošle všem kromě odesílatele). V běžných počítačových sítích se už dnes s huby moc nesetkáváme, používají se ale například v některých PAN sítích nebo u USB.

Switch (přepínač) je již inteligentnější aktivní síťový prvek, který si vede tabulku adres uzlů. U příchozího paketu zjistí adresáta, podle tabulky určí port, který k němu vede, a paket odešle jen na tento port. Pokud adresáta nemá v tabulce nebo když jde o broadcast, odešle ho na všechny porty. Switch rozděluje síť na segmenty – pokud mezi sebou komunikují uzly ze stejného segmentu, pak taková komunikace nepronikne do jiného segmentu. Pracuje na vrstvě L2, ale ve skutečnosti může obsahovat i funkcionalitu vyšších vrstev (multilayer switch).

Výhodou je, že tolik nezahlcuje síť, také je možné implementovat správní a bezpečnostní funkce. Nevýhodou je výpočetně náročnější provoz (což je technicky řešitelné, velká část funkčnosti je implementována hardwarově). Se switchi se běžně setkáváme v LAN, MAN i WAN sítích.

Bridge (most) je zařízení navzájem oddělující dva segmenty sítě, jako switch. Na rozdíl od switche mívá méně portů, jeho funkčnost je implementována softwarově a je standardizován (IEEE 802.1). V praxi se přímo s bridgi nesetkáváme. Most pracuje na vrstvě L2.

Router (směrovač) si vede směrovací tabulku, v ní však nemá adresy konkrétních uzlů, ale L3 adresy sítí a podsítí. Příchozí paket „pitvá“ o něco důkladněji než switch, přičemž podle cílové adresy zjistí, do které sítě patří adresát, ve směrovací tabulce ověří, který port je cestou do této sítě, a na ten port paket odešle. Broadcasty zahazuje.

Zatímco switch odděluje segmenty jedné sítě, router odděluje různé sítě. Routery pracují na vrstvě L3.

Výhodou routeru je možnost implementace pokročilých správných a bezpečnostních funkcí a schopnost oddělit komunikaci v různých sítích. Nevýhodou je ještě více výpočetně náročný provoz než u switche, proto typicky routery mají méně portů než switche, výkonnější procesor a více paměti, aby vyšší výpočetní zátěž unesly.

Brána (gateway) slouží k propojení dvou *různých typů* sítí, resp. slouží jako spojovací bod a zároveň „překladatel“. Například pokud máme v domácnosti VDSL router, pak toto zařízení má vestavěnou bránu pro komunikaci mezi naší lokální sítí a VDSL sítí poskytovatele internetu.

Brána typicky pracuje na vyšší vrstvě než protokoly, které má propojit.

1.3 Co a jak přenášíme po počítačové síti

1.3.1 Protokoly

Protokol určuje, jakým způsobem má komunikace začít, případně jak se dohodnout na určitých parametrech komunikace, jak sdělit druhé straně určitý typ informace, jak má druhá strana potvrdit příjem nebo sdělit, že je třeba přenos opakovat, jak se komunikace ukončuje, atd.

Definice (Protokol a jeho implementace)

Protokol je konvence, podle které probíhá určitý typ (většinou elektronické) komunikace. Definuje pravidla určující syntaxi (jak jsou které signály poskládány), sémantiku (význam) a synchronizaci komunikace.

Protokol je pouze předpis, který je třeba implementovat (naprogramovat, aby mohl být v praxi používán). Implementace může být softwarová, hardwarová (v obvodech) nebo kombinace softwarové a hardwarové.

Například protokol HTTP je implementován (naprogramován) v operačním systému běžícím na počítači, u kterého sedíte, a taky na druhé straně – v operačním systému WWW serveru, se kterým komunikujete.

Poznámka:

Pro návrh protokolu platí jedno důležité pravidlo (vlastně je zachováváno i jinde, například u systému UNIX): protokol by měl být jednoduchý, krátký, jednoznačně implementovatelný. Neměl by být moc složitý, protože čím větší složitost, tím větší pravděpodobnost chyb. Proto žádný protokol není moc univerzální – umí jednu konkrétní věc, a umí ji dobře. V angličtině se to vyjadřuje zkratkou KISS (Keep it Simple, Stupid – ať je to jednoduché, hloupé).

Aby tento princip mohl být dodržován, existují určité konvence pro spolupráci protokolů: to, co protokol neumí, předá ke zpracování jinému protokolu.

Princip protokolů není ani zdaleka jen záležitostí počítačových sítí – protokoly se používají v telekomunikacích, spotřební elektronice, strojírenství, ale například i „uvnitř“ počítače. Každý se setkal třeba s protokolem USB, SATA,...

1.3.2 Vlastnosti protokolů

Běžně používané protokoly mají obvykle (kromě jiných) tyto tři důležité vlastnosti:

- jsou všeobecně známé a (téměř) kdokoliv je může implementovat (kromě proprietárních),
- jsou spíše jednoduché, nepříliš komplexní,
- mohou spolupracovat s jinými protokoly.

Nejdřív se zaměříme na první vlastnost. Proč je důležitá? Představme si situaci, kdy výrobce síťového hardwaru přijde s novým zařízením a rozhodne se, že toto zařízení bude komunikovat podle nového protokolu, jehož specifikaci nikomu jinému neposkytne. Toto zařízení si ovšem bude „rozumět“ jen s takovými zařízeními, která budou podporovat tentýž protokol, takže nedokáže komunikovat se zařízeními od jiných výrobců. Pro dotyčného výrobce by to snad mohlo být výhodné, pokud by dokázal své potenciální zákazníky přesvědčit, aby kupovali jenom od něj, ale realita bývá jiná – zákazníci by radši kupovali taková zařízení, která by mohli bez problémů zařadit do své sítě, ve které už pravděpodobně mají nějaká zařízení od jiných výrobců.

Proto je většina protokolů buď volně dostupná (specifikace je zcela volně k nahlédnutí, implementovat může kdokoliv) nebo alespoň dostupná jiným způsobem (za poplatek, udělení licence).

Otevřený protokol je protokol volně dostupný, naopak proprietární protokol je takový protokol, jehož specifikace není nikde zveřejněna a jeho tvůrce si ji buď nechává jen pro sebe nebo licencuje za poplatek.

Poznámka:

O rozšířenosti volných specifikací mluví například to, že momentálně je na routerech nejběžnějším směrovacím protokolem otevřený protokol OSPF. Naopak IGRP (proprietární směrovací protokol společnosti Cisco) se téměř nepoužívá dokonce ani na zařízeních od samotného Cisca.

Pokud síťové zařízení podporuje některý proprietární protokol, obvykle pro danou funkci taky podporuje některý alternativní protokol s dostupnější specifikací, aby si mohlo „popovídat“ se zařízeními jiných výrobců.

Dále se k principu otevřenosti, jednoduchosti a spolupráce budeme často vracet.

1.4 Standardy a standardizační instituce

Definice (Standard, norma a standardizace)

Standard (v oblasti technologií) je požadavek na splnění určitých konkrétních vlastností pro určitý typ hardwaru, softwaru apod. Je to dokument popisující požadavky, specifikace, návody a popisy pro daný produkt, proces či službu. Rozlišujeme standardy

- *normativní (de iure, podle zákona)* – jejich dodržování je vyžadováno, obvykle je stanoví příslušná státní instituce,
- *popisné (de facto)* – jejich dodržování sice není striktně vyžadováno, ale je v zájmu výrobce.

V oblasti technologií obvykle používáme pojem *norma* pro normativní standardy, jinak prostě použijeme pojem *standard*, třebaže se taky často mluví o *doporučeních* (recommendation).

Standardizace je proces sjednocení vlastností a funkcí daného produktu pomocí stanovení standardu.

1.4.1 Standardizační instituce pro počítačové sítě

Následuje přehled nejdůležitějších standardizačních organizací a institucí, které vydávají standardy související s počítačovými sítěmi a technologiemi obecně. Většinou se jedná o standardy de facto (nejsou závazné, ale běžně bývají dodržovány), samozřejmě až náš vlastní (a částečně i evropský) normalizační institut.

Situace se mírně komplikuje tím, že některé standardy vydané jednou organizací bývají často adaptovány jednou či dokonce více dalšími organizacemi (například je běžná adaptace standardů českým normalizačním úřadem). Původce standardu obvykle poznáme podle začátku označení tohoto standardu, kombinaci více organizací pak díky kombinaci jejich označení.

Úřad pro technickou normalizaci, metrologii a státní zkušebnictví je národním standardizačním orgánem pro Českou republiku, zabývá se tvorbou českých norem. Normy vydané tímto ústavem poznáme podle toho, že začínají písmeny ČSN, za nimi následuje číslo normy. V současné době ve značné míře zabývá adaptací (přejmutím a přizpůsobením) norem vydaných Evropskou unií nebo některou standardizační organizací. Tyto normy poznáme podle toho, že za ČSN následuje zkratka původní organizace či instituce, například ČSN EN... jsou normy přejaté od Evropské unie.

Další informace:

Normy vydané Úřadem pro technickou normalizaci, metrologii a státní zkušebnictví jsou k nahlédnutí na http://www.unmz.cz/urad/csn-online (jen náhledy, za celé znění se platí).

ETSI (European Telecommunications Standard Institute) je sice původně evropská organizace, ale ve skutečnosti jsou její členové ze všech obydlených kontinentů (státy, významní výrobci komunikačních zařízení, poskytovatelé síťových služeb, výzkumné organizace, atd.). Náměty na nové standardy nebo změny stávajících vycházejí ze tří zdrojů – buď se domluví nejméně čtyři členové ETSI, nebo je přijat podnět od Evropské komise (EC, European Commission) nebo od EFTA (European Free Trade Association).

Rozlišuje se několik typů ETSI standardů – například EN (European Standard), ES (ETSI Standard), TS (ETSI Technical Specification) a další. K nejznámějším patří standardy související s mobilními sítěmi (především GSM, 3G, 4G sítě), chytrými sítěmi, komunikací machine-to-machine, ICT ve zdravotnictví, atd. ETSI standardy jsou dostupné volně bez poplatků.

https://portal.etsi.org

ITU (International Telecommunications Union) je celosvětová organizace pro informační a komunikační technologie. Je součástí OSN a sídlí ve Švýcarsku. Má tři části:

- ITU-R: radiokomunikační sektor (původně CCIR),
- ITU-T: sektor pro standardizaci telekomunikací (původně CCITT),
- ITU-D: sektor rozvoje telekomunikací.

ITU-T je součást ITU pro standardizaci telekomunikací, její činnost souvisí i s počítačovými sítěmi. Členy s hlasovacím právem jsou zainteresované země, členem bez hlasovacího práva může být kterákoliv organizace. Členství je placené.

ITU-T se dále člení do *studijních skupin* (SG, Study Group), které mají každá své vlastní zaměření. Například SG13 v poslední době vydala několik standardů o cloud computingu, SG16 se zabývá multimédii (přenos videa, obrázky, zvuk apod.), SG17 bezpečností, SG20 chytrými sítěmi (Internet of Things,...). Standardy vytvořené ITU-T jsou otevřené, volně dostupné.

Se standardy ITU-T se setkáváme poměrně běžně: například má „na svědomí“ protokol H.323 používaný v multimediálních přenosech a internetové telefonii, protokoly G.992.1 and G.992.2 pro přístupové sítě ADSL, standard X.509 používaný při šifrování a další.

http://www.itu.int/en/ITU-T/Pages/default.aspx

ITU-R je součást ITU pro radiokomunikace, a tedy se zabývá standardy pro bezdrátové vysílání (také datové). Momentálně se hodně probírají například standardy pro LTE Advanced (formálně IMT Advanced), jako je M.2012 – *Detailed specifications of the terrestrial radio interfaces of International Mobile Telecommunications Advanced (IMT-Advanced).*

Standardy začínající písmenem „M“ jsou pro mobilní radiokomunikační sítě, „RS“ pro bezdrátové sítě senzorů a jejich vzdálené ovládání, „SM“ pro správu frekvenčního spektra, atd.

http://www.itu.int/pub/R-REC

ISO (International Organization for Standardization) je nezávislá organizace vydávající standardy pro komunikační technologie. Jejími členy jsou standardizační instituce z různých zemí (každá země tam má jedno zastoupení), naopak organizace ISO je členem ITU. Sídlo ISO je ve Švýcarsku.

ISO má hierarchickou strukturu. Člení se na 200 TC (Technical Committee, technický výbor), každý technický výbor má svou oblast působnosti. Například ISO/TC 47 se zabývá chemií, ISO/TC 20 letadly a vesmírnými vozidly, ISO/TC 272 forenzními vědami, ISO/TC 299 robotikou, ISO/IEC JTC 1 informačními technologiemi.

Každý TC se člení na subkomise (SC, Subcommittee) a každá subkomise se člení na pracovní skupiny (WG, Working Group). Návrhy na standardy nebo jejich změnu jdou vždy zespoda, od pracovních skupin, postupně se přepracovávají a „probojovávají“ nahoru až ke schválení standardu. Pracovní skupina vytvoří *pracovní verzi* (Working Paper), ten je přepracován na *koncept* výboru (Committee Draft), následuje *koncept mezinárodního standardu* (Draft International Standard), posledním stupněm je *mezinárodní standard* (International Standard).

Nejznámějším standardem, se kterým se seznámíme už na následujících stranách, je referenční model ISO/OSI (standard ISO 7498 a hodně dalších s ním souvisejících), dále se ISO zabývá sítěmi senzorů, vzdáleným přístupem, UPnP, webovými službami a dalšími tématy.

http://www.iso.org/iso/home/standards_development/list_of_iso_technical_committees.htm

IEEE (Institute of Electrical and Electronics Engineers, čteme anglicky „Eye-triple-E“, tedy [aj tripl i:]) je největší světová organizace sdružující odborníky z oblasti elektroniky, elektrotechniky, informatiky a souvisejících oborů, je to tedy profesní sdružení. Nezabývá se jen standardy, ale pořádá různá setkání, konference, vzdělávací akce, vydává odborné časopisy a vyvíjí další aktivity. Sídlo IEEE je v USA. Jednou z aktivních sekcí IEEE je i ta česká.

Sdružení IEEE stojí především za skupinou standardů IEEE 802 pro lokální a částečně i rozlehlé sítě, do které patří například IEEE 802.11 pro bezdrátové lokální sítě (Wi-fi) nebo IEEE 802.3 pro Ethernet. Přístup k plnému znění standardů je placený.

http://www.ieee.org

IEC (International Electrotechnical Commission) se z oblasti počítačových sítí zabývá především standardy pro elektrická a elektronická zařízení, spolupracuje především s organizací ISO a hodně standardů nese označení „ISO IEC“.

http://www.iec.ch/

IETF (The Internet Engineering Task Force) se zabývá především standardy souvisejícími s Internetem, včetně například protokolů TCP/IP. Velmi úzce spolupracuje s dalšími organizacemi, například IANA (The Internet Assigned Numbers Authority), ISOC (Internet Society), IAB (Internet Architecture Board, Rada pro architekturu Internetu), W3C (World Wide Web Consortium, standardy pro web) a dalšími.

Standardy vydané IETF obvykle začínají zkratkou RFC (Request for Comments) a následuje číslo. Pokud je třeba standard aktualizovat, nemění se původní znění, ale vytvoří se RFC dokument (obsahující popis standardu) s novým číslem. Například pro výše zmíněný otevřený směrovací protokol OSPF bylo takto postupně vytvořeno několik dokumentů (verzí), z nichž jsou momentálně aktuální RFC 2328 (OSPF verze 2) a RFC 5340 (OSPF verze 3).

Další informace:

Oficiální stránky organizace jsou na https://www.ietf.org/. Všechny dokumenty RFC jsou volně dostupné, dokonce existuje řada webových stránek, které je zpřístupňují, například:

- https://tools.ietf.org/html/
- https://datatracker.ietf.org/doc/
- http://www.rfc-base.org/
- https://www.rfc-editor.org/

ANSI (American National Standards Institute) je normalizační úřad USA. Sdružuje státní instituce, komerční společnosti, členy z akademické sféry a další. Kromě vyvíjení vlastních standardů také reprezentuje USA v ISO, IEC a dalších nadnárodních organizacích. Standardy jsou dostupné za poplatek.

ANSI stojí například za standardem pro znakovou sadu ASCII, známý je také standard pro programovací jazyk C (ANSI C), v oblasti sítí například FDDI, CDMA, Frame Relay.

https://www.ansi.org/

TIA a EIA (Telecommunication Industries Association, Electronic Industries Alliance) jsou americké organizace zaměřující se především na fyzickou úroveň komunikace mezi zařízeními, v čemž hodně spolupracují také s ANSI.

TIA se zabývá standardy pro nejrůznější typy kabelů a konektorů, připojení antén, mobilní sítě, ICT ve zdravotnictví, chytré sítě. EIA stojí například za starým známým konektorem SCART, dále v sítích se setkáváme s konektorem RS-232. Existují standardy vzniklé při spolupráci těchto společností, například TIA/EIA 568, se kterým se setkáme v kapitole o Ethernetu.

Další informace:

- http://www.tiaonline.org/
- https://www.eia.gov/about/eia_standards.cfm

1.4.2 Volně dostupné standardy

RFC dokumenty. Pro fungování (nejen) Internetu jsou důležité také RFC dokumenty, ve kterých jsou popsány nejdůležitější protokoly a způsob jejich spolupráce. Připomeňme si základní vlastnosti těchto dokumentů:

- Každý RFC má jednoznačné číslo a také svůj název.
- Neexistují žádné aktualizace RFC dokumentů, aktualizace obsahu je vydána pod novým číslem (ale název bývá stejný).
- Jedno téma (případně protokol) může být popsáno ve více než jednom RFC dokumentu.

Výhodou druhé vlastnosti je, že se nemusíme ztrácet v různých verzích téhož dokumentu, nevýhodou je, že hledání aktuálního znění je o něco těžší. Výhodou třetí vlastnosti je, že RFC dokumenty nejsou obvykle až tak dlouhé, aby ztrácely na přehlednosti, nevýhodou je, že někdy musíme dohledávat související informace.

Příklad

Podíváme se na některé RFC dokumenty související s protokolem TCP. Tento protokol pracuje na transportní vrstvě a jeho úkolem je navázat a spravovat spojení se strojem, s nímž komunikuje náš stroj.

Na adrese https://tools.ietf.org/html/ nejdřív zadáme do vyhledávání číslo RFC dokumentu 7414. Tento dokument nepopisuje přímo protokol TCP, ale je jakýmsi rozcestníkem k RFC dokumentům, které s TCP mají něco společného.

Aby se předešlo problémům s formáty, jsou RFC dokumenty čistě textové. Všimněte si struktury dokumentu:

- Na začátku je krátké informační záhlaví dokumentu, pak název („A Roadmap for Transmission...“), následuje abstrakt se stručným popisem, stav dokumentu a licenční ujednání.
- Dále tu máme obsah a za ním kapitoly dokumentu. Poslední části jsou odkazy na další zdroje (většinou další RFC) a kontakty na autory.
- Třebaže si prohlížíme „dlouhou“ webovou stránku, je dokument rozčleněn na stejně dlouhé stránky a každá má své záhlaví a zápatí (mezi zápatím předchozí strany a záhlavím následující je čára).

Zaměřme se na informační záhlaví dokumentu. Vlevo se dozvíme, že se jedná o produkt IETF a je to RFC číslo 7414, na dalším řádku stojí „`Obsoletes: 4614`“. To znamená, že předchozí varianta tohoto dokumentu (zastaralý, obsolete) má číslo 4614. Všimněte si, že čím novější, tím vyšší číslo. Ve sloupci vpravo zjistíme, kdy tento RFC dokument vznikl: v únoru 2015.

V úvodní kapitole dokumentu je obvykle povídání o obsahu dokumentu a jakýsi souhrn. V další kapitole – Core Functionality – najdeme odkaz na hlavní dokument obsahující popis protokolu TCP, tedy RFC 793. Všimněte si nízkého čísla – tento dokument je platný už dlouho (od roku 1981), ale další RFC přidávají novou funkcionalitu.

Na to, že jde vlastně jen o komentovaný seznam RFC dokumentů souvisejících s protokolem TCP, je dokument opravdu hodně dlouhý. Ovšem celý dokument studovat nebudeme.

Příklad

V minulém příkladu jsme se dozvěděli, že RFC 7414 je novější variantou nahrazující zastaralý (obsolete) dokument RFC 4614. Podívejme se na tento starší dokument (stejným způsobem – na adrese https://tools.ietf.org/html/ zadáme do vyhledávacího okna číslo 4614, nebo prostě využijeme odkaz z novějšího dokumentu).

Vidíme, že název dokumentu („A Roadmap for Transmission...“) je stejný jako u novějšího, jen číslo je jiné. V levém sloupci záhlaví dokumentu je řádek „`Obsoleted by: 7414`“. Tento řádek je důležitý, protože pokud narazíme na RFC dokument, o jehož platnosti nic nevíme, tento řádek nám řekne, že je zastaralý a který dokument je jeho náhradou.

Dále je řádek „`Updated by: 6247`“. Nejedná se o novou verzi, pouze se mění „vztah k okolí“, v tomto konkrétním případě jsou některá nepoužívaná rozšíření protokolu TCP „odsunuta do hitorie“.

Příklad

Teď se podívejme na „Core“ RFC dokument o protokolu TCP. Z prvního příkladu víme, že jeho číslo je 793. Začátek dokumentu vypadá trochu jinak, jak je vidět, struktura RFC dokumentů se postupně obohacovala.

Všimněme si nákresů – všechny jsou čistě textové, nicméně pro tyto účely to dostačuje. V kapitole 1.1 je hrubý nákres vzhledem k okolním vrstvám (jak vidíme, pod TCP má být internetový protokol, tedy IP), a dále v kapitole 2.5 na straně 9 je podrobnější nákres zahrnující konkrétní spolupracující protokoly (samozřejmě ne všechny). V kapitole 3.1 je nákres záhlaví TCP segmentu, za nímž jsou všechny součásti vysvětleny. Na straně 23 najdeme stavový diagram popisující komunikaci s využitím

protokolu TCP.

Úkol

Prohlédněte si RFC 2460 a odpovězte na tyto otázky:

- Co tento dokument popisuje?
- Kdy byl publikován?
- Nahrazuje některý zastaralý (obsolete) dokument?
- Najděte kapitolu 2 o terminologii. Které pojmy vám něco říkají?
- Prohlédněte si nákresy v dokumentu uvedené.

ITU-T. Další organizací, která své standardy celé zveřejňuje, je ITU-T. Informace jsou dostupné na webu http://www.itu.int/en/ITU-T/Pages/default.aspx.

Příklad

Jedním ze standardů, které vznikly v ITU-T, je X.500. Je věnován adresářovým službám, což jsou síťové služby, které mají zjednodušovat správu sítí – v distribuované (rozkládající se na více zařízeních) databázi jsou evidovány jak jednotlivé objekty v síti (počítače, servery, prostředky na nich dostupné, aktivní prvky, atd.), tak i uživatelé a jejich přístupová práva k těmto objektům. Odlehčenou variantou protokolu X.500 je protokol LDAP, který je implementován jak ve Windows (jako Active Directory) tak i v Linuxu a dalších operačních systémech.

Na adrese http://www.itu.int/itu-t/recommendations/rec.aspx?rec=X.500 najdeme základní informace o protokolu a také odkaz na PDF soubor s celým zněním (stačí klepnout na červenou ikonu vpravo).

Úkol

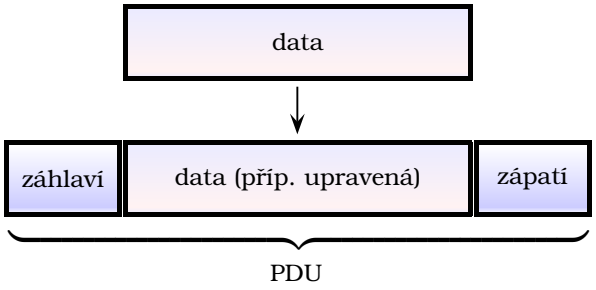
Prohlédněte si obsah PDF souboru se specifikací X.500, o který se jednalo v předchozím příkladu, zejména:

- Srovnejte strukturu dokumentu s RFC dokumenty.
- Na začátku kapitoly 6 (Overview of the Directory) si přečtěte, co je to Directory (adresář). Všimněte si, že zde je tento pojem chápán trochu jinak než v běžných operačních systémech (obdoba složky).
- Všimněte si nákresů. Většinou jde o nákresy komunikace s databází nebo vztahové diagramy (stromy). Na obrázku 3 (tisknutá strana 7, podle pořadí 13) je ukázková struktura adresářového stromu podle X.500, pokuste se pochopit vztahy mezi uzly tohoto stromu.

1.5 Referenční model ISO/OSI

1.5.1 Přehled modelu

Protokolová datová jednotka (PDU, Protocol Data Unit) je sekvence dat opatřená metadaty (informacemi o datech) vztahující se ke konkrétnímu protokolu. Protokoly obvykle obdrží data, podle potřeby je určitým způsobem zpracují (strukturují, rozdělí na menší části, zašifrují, komprimují, přeloží, určí adresu příjemce apod.) a přidají před ně *záhlaví* (header) se souvztažnými informacemi (délka dat, použitý šifrovací algoritmus, adresa odesílatele a příjemce, atd.). Některé protokoly také přidávají za data *zápatí* (trailer) obsahující například kontrolní součet. Data přenášená v datové jednotce taky nazýváme *payload*.



Obrázek 1.1: Datová jednotka (PDU)

K nejdůležitějším standardům z oblasti počítačových sítí patří skupina standardů popisujících *referenční model OSI* (Open Systems Interconnection) publikovaných organizací ISO, proto se označuje RM ISO/OSI. Původní označení standardu je ISO/IEC 7498-1, dnes je dostupný jako ITU-T X.200 na webu http://www.itu.int/rec/T-REC-X.200-199407-I.

OSI definuje sedm vrstev. Každá vrstva plní v komunikaci přes počítačovou síť konkrétní funkci a je přesně stanoveno, co se na dané vrstvě může dít, jedná se tedy o *konceptuální model* (tj. popisuje logiku návrhu, vztahy mezi součástmi).

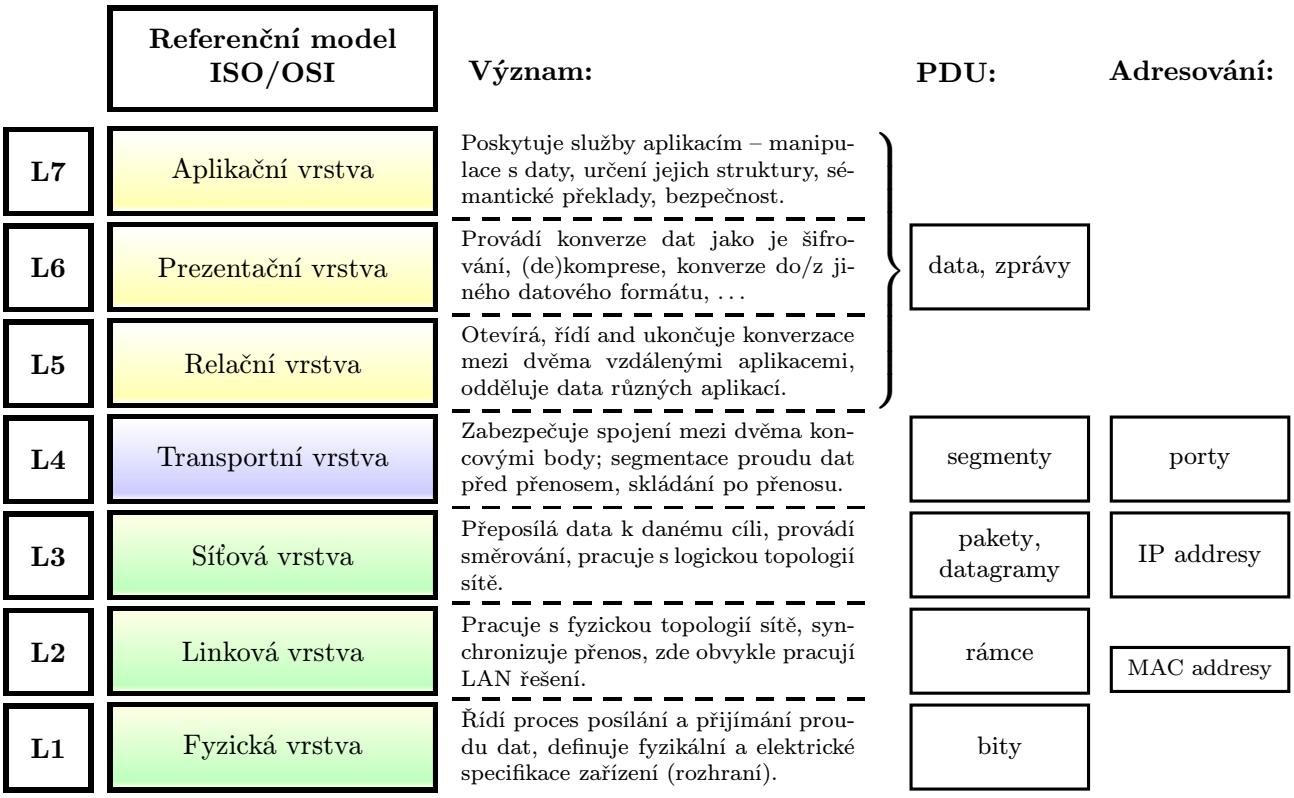
Pořadí vrstev a stručný popis najdeme na obrázku 1.2. Vrstvy zobrazené zeleně (L1 až L3) jsou závislé na přenosovém médiu, vrstva L4 (modrá) je přechodová, vrstvy zobrazené žlutě (L5 až L7) jsou nezávislé na přenosovém médiu a na samotném přenosu se podílejí jen nepřímo (přípravou dat a komunikací s aplikacemi).

Dále se budeme věnovat jednotlivým vrstvám. Ke každé potřebujeme znát její účel, typické protokoly, které na této vrstvě pracují a používané datové jednotky.

Fyzická vrstva (L1, Physical Layer). Vrstva L1 je odpovědná za samotný fyzický přenos dat. Definuje přenosové médium (kabely, rádiové vlny apod.), jak je reprezentován bit o hodnotě 0 nebo 1 (kódování), síťové rozhraní (porty, konektory), k čemu konkrétně je používán který vodič v kabelu, a obecně vše, co je potřeba pro konverzi sekvence bitů do formy přenášeného signálu. Na této vrstvě se nevyskytují žádné datové jednotky, vrstva pouze přijímá proud bitů a přetváří je na signál, který odesílá.

Fyzickou vrstvu mají implementovánu všechna zařízení v počítačové síti, tedy všechna, která jsou opatřena nějakým síťovým rozhraním.

Zařízení, která mají implementovanou pouze fyzickou vrstvu: hub a repeater. Výhodou těchto zařízení je jednoduchost a rychlost, nevýhodou je nemožnost implementovat pokročilejší funkce.



Obrázek 1.2: referenční model ISO/OSI

Linková vrstva (L2, Data-link Layer, spojová). Na této vrstvě se určuje vztah mezi příchozím proudem bitů (který vidí fyzická vrstva) a konkrétním uzlem v síti. Zařízení s implementovanou vrstvou L2 má přehled o zařízeních připojených do místní sítě (minimálně „vidí“ ta zařízení, ke kterým je přímo připojeno), vede si tabulku fyzických adres těchto zařízení (nazývá se obvykle MAC tabulka, CAM tabulka apod., podle konkrétního protokolu a konkrétního zařízení). Ke každé adrese v tabulce máme kromě jiného i port, přes který je dotyčné zařízení dosažitelné.

Na linkové vrstvě jsou také zajištěny funkce, které se sice vztahují k přenosovému médiu, ale zároveň vyžadují práci s datovými jednotkami. Například se zde určuje rychlost přenosu, protože tu je třeba řídit i podle toho, v jakém stavu je příjem datových jednotek. Pokud se sem tam nějaká datová jednotka ztratí (což by fyzická vrstva nepoznala), pak zřejmě celý mechanismus „nestíhá“ a je třeba snížit rychlost přenosu. K funkcím vrstvy L2 tedy patří i detekce přenosových chyb.

Datové jednotky, se kterými pracují protokoly této vrstvy, se nazývají *rámce*. Každý rámec především jednoznačně označuje začátek a konec dat a obsahuje (kromě jiného) fyzickou adresu příjemce a fyzickou adresu odesílatele v rámci sítě.

Vrstvu L2 implementují ta zařízení v síti, která potřebují přehled o uzlech v místní síti a pracují s adresami těchto uzlů. Switche a bridge implementují vrstvu L2, přičemž zde pracuje vždy některý protokol, který si vede tabulku fyzických adres. Ovšem existují i switche implementující vrstvu L3, ale to už je dodatečná funkcionalita.

Pokud se jedná o aktivní síťový prvek, pak takové zařízení dokáže *oddělovat segmenty v síti.*

Síťová vrstva (L3, Network Layer). Zatímco protokoly linkové vrstvy mají přehled o fyzické topologii sítě, protokoly síťové vrstvy pracují s logickou topologií sítě a „vidí“ i za hranice lokální sítě.

Úkolem síťových protokolů je stanovit skutečnou cestu (nebo úsek cesty, za který je dotyčné zařízení zodpovědné), tedy určit adresu vyšší úrovně a *směrovat*.

Datové jednotky na síťové vrstvě jsou označovány jako *pakety* nebo *datagramy* (záleží na konkrétním protokolu, a taky na dotyčném zdroji informace). Jaký je mezi nimi rozdíl?

- *Datagram* je datová jednotka posílaná v rámci nespojované (datagramové) služby.
- *Paket* je datová jednotka posílaná (nejen) v rámci spojované služby, ale často se používá i v obecnějším významu (prostě jako datová jednotka).

Síťová vrstva je implementovaná na koncových zařízeních a dále na těch aktivních síťových prvcích, které zajišťují směrování, pracují s logickou topologií sítě, propojují nejen segmenty, ale také celé sítě, což jsou routery (směrovače). Router tedy implementuje vrstvy L1, částečně L2 (jen to, co je nezbytné k propojení s vyšší vrstvou) a L3.

Na síťové vrstvě je vedena (minimálně jedna) *směrovací tabulka* (routing table). Ve směrovací tabulce je informace o tom, kam poslat datovou jednotku patřící do určité (pod)sítě – můžeme si to představit tak, že na řádku je adresa cílové (pod)sítě, brána (zde ve smyslu blízkého zařízení, přes které se dá do té (pod)sítě dostat, tedy určení dalšího kroku na cestě), síťové rozhraní, přes které mají data vyjít z tohoto zařízení a další informace.

Takže pokud směrovač dostane data k přeposlání, najde v záhlaví síťové vrstvy logickou adresu příjemce a postupně prochází jednotlivé řádky směrovací tabulky – zastaví se na prvním řádku takovém, že adresa příjemce patří do (pod)sítě na tomto řádku uvedené. V řádku si přečte směr, kterým má data poslat (buď adresu cílového zařízení nebo adresu dalšího směrovače na cestě, tedy bránu, a pak síťové rozhraní, přes které data odešle).

 **Transportní vrstva (L4, Transport Layer).** Tato vrstva je jakýmsi přechodem mezi vrstvami orientovanými na proces přenosu (L1–L3) a vrstvami orientovanými na aplikace a tedy nezávislými na procesu přenosu (L5–L7). Takže směrem nahoru potřebuje vazbu na konkrétní protokol protokol vyšší vrstvy (této vazbě, číslu, které je obdobou adresy, říkáme *port*) a směrem dolů uplatňuje funkce související s přenosem dat.

 **Poznámka:**

Pozor – pojem „port“ se v počítačových sítích používá ve dvou velmi odlišných významech:

- (fyzický) port jako součást síťového rozhraní, jak je vysvětleno na straně **??**,
- port (určený číslem) v záhlaví datové jednotky transportní vrstvy určující, se kterým protokolem vyšší vrstvy se právě komunikuje.

Například číslo 80 znamená při použití protokolů TCP nebo UDP komunikaci s protokolem HTTP.

Datovou jednotkou transportní vrstvy je *segment*, a pokud jde o nespojovanou službu, může se použít i pojem *datagram*. Hlavním úkolem protokolů transportní vrstvy (tedy kromě práce s číslem portu) je *segmentace dat* z vyšší vrstvy na dostatečně malé úseky, které bude možné přes síť přepravit. Data jsou rozdělena na úseky o stanovené délce, ke každému je přidáno záhlaví s údaji transportní vrstvy, čímž je vytvořen segment.

Na transportní vrstvě se rozhoduje, zda bude přenos realizován formou služby se spojením nebo služby bez spojení. U služby se spojením zajišťuje vrstva L4 navázání spojení (handshake), řídí průběh

spojení, potvrzování doručených segmentů, v případě ztráty či poškození dat zajistí opakování přenosu segmentu (tedy potvrzovaná, spolehlivá služba) a podle potřeby zajistí úpravu parametrů spojení, a pak ukončí spojení.

Transportní vrstva je implementovaná obvykle jen na koncových zařízeních.

Relační vrstva (L5, Session Layer). Na této vrstvě jsou oddělena data patřící různým aplikacím. Každá aplikace komunikující se sítí přes tuto vrstvu navazuje *relaci* (session) s nějakou aplikací na jiném systému, a v rámci této relace se například přenášejí data. Z toho vyplývá, že relace je logické spojení mezi dvěma aplikacemi navázané (většinou) za účelem výměny dat, které může vést přes síť.

Jaký je rozdíl mezi spojením na transportní vrstvě a relací na relační vrstě? Zatímco spojení se navazuje mezi dvěma zařízeními, relace se navazuje mezi dvěma aplikacemi na těchto zařízeních. Jak bylo výše uvedeno – spodní vrstva poskytuje služby vyšší vrstvě, a tedy pokud relační vrstva chce pro určitou aplikaci navázat relaci s aplikací na jiném systému, potřebuje k tomu kromě jiného také spojení, které jí zajistí transportní vrstva.

Na relační vrstvě je implementován například koncept socketů. Také zde bývá implementováno to, co nutně potřebujeme k navázání relace – autentizace a autorizace.

Relační vrstva, stejně jako všechny vyšší vrstvy, je implementovaná na koncových zařízeních.

Prezentační vrstva (L6, Presentation Layer). Vrstva L6 je zodpovědná za provádění konverzí dat (žádná z nižších vrstev do dat nezasahuje), například úpravy kódování textu (ASCII, EBCDIC apod.), komprese a dekomprese, šifrování a dešifrování, některé úkoly související se zpracováním multimédií.

Proč se tato vrstva vlastně nazývá prezentační? Protože má za úkol data prezentovat vyšším vrstvám v takové formě, které tyto vrstvy rozumí.

Aplikační vrstva (L7, Application Layer). Na této vrstvě pracují protokoly, které jsou využívány aplikacemi. Například aplikace webový prohlížeč využívá (kromě jiného) aplikační protokol HTTP. Funkcí této vrstvy je při odesílání dat převzat data od aplikace a předat nižším vrstvám; naopak při přijímání dat jsou tato data přijata od nižších vrstev a předána příslušné aplikaci.

Poznámka:

Pozor, na aplikační vrstvě jsou aplikační protokoly, nikoliv aplikace!

1.5.2 Spolupráce protokolů

Nadřízená vrstva vytvoří svou PDU (například paket) a odešle podřízené vrstvě. Pro podřízenou vrstvu je to, co takto obdržela, označováno jako *SDU* (Service Data Unit). Tedy celý postup je takový, že každá vrstva obdrží od nadřízené vrstvy SDU a přidáním záhlaví (a případně i zápatí) z ní vytvoří PDU a předá nižší vrstvě. To platí i pro aplikační vrstvu – nad ní sice žádná vrstva není, ale sekvenci dat, kterou obdrží od aplikace, která požaduje její služby, také označujeme SDU.

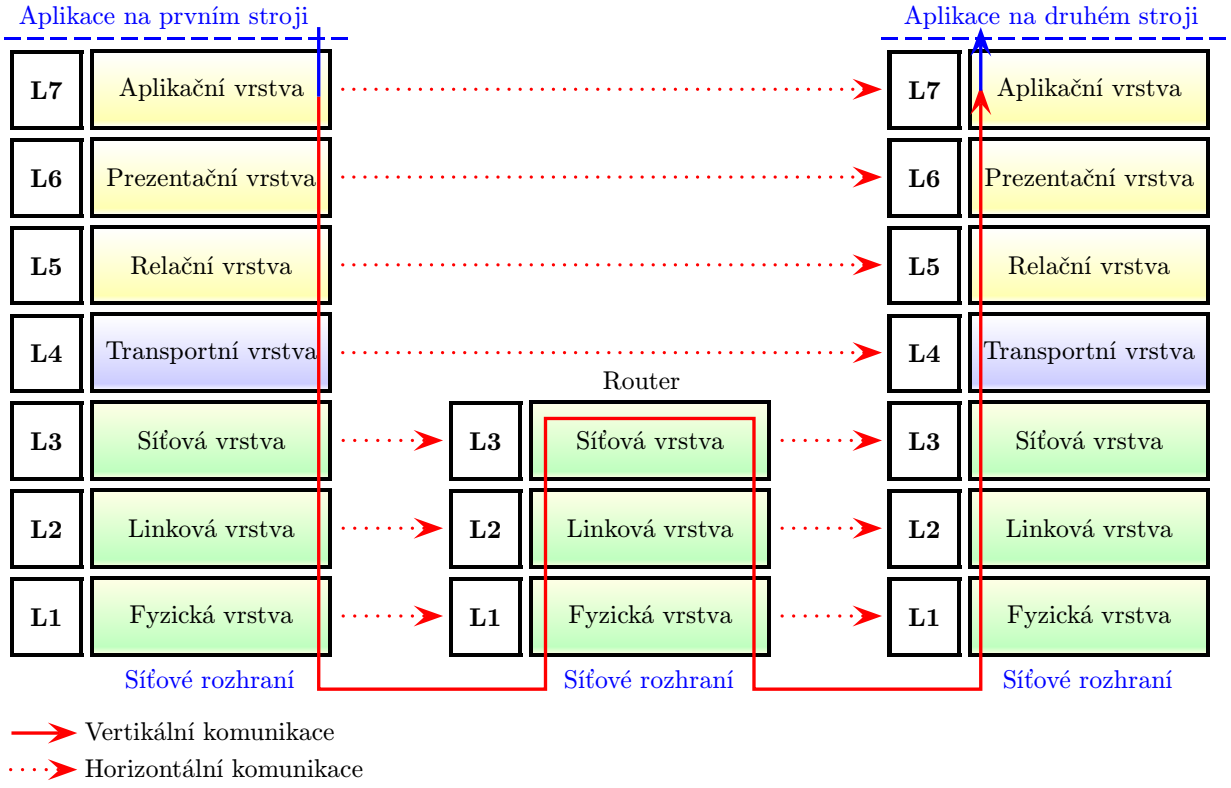
Víme, že protokoly nemají být moc komplexní, a tedy potřebují spolupracovat s jinými protokoly. Spolupráce může probíhat buď v rámci jedné vrstvy, nebo mezi sousedními vrstvami (nikdy ne na přeskáčku, vždy jen mezi přímými sousedy), přičemž platí, že spodní vrstva poskytuje služby horní vrstvě.

Entita je aktivní prvek na určité vrstvě v modelu ISO/OSI, který má definováno rozhraní – sadu

služeb, které může využívat entita z bezprostředně nadřízené vrstvy. Entitu vrstvy n označujeme *n-entita*. Na n-té vrstvě ISO/OSI je tedy sada n-entit.

Rozhraní mezi komunikujícími entitami (a tedy mezi vrstvami) se nazývá *SAP* (Service Access Point, přístupový bod služby). SAP tedy propojuje dvě entity v sousedních vrstvách – uživatele služby (service user) a poskytovatele služby (service provider).

Způsob komunikace v rámci jednoho systému (řetěz střídajících se entit a SAP) se v ISO/OSI nazývá *vertikální komunikace*. *Horizontální komunikace* v ISO/OSI je komunikace mezi dvěma stejnými vrstami umístěnými na různých strojích, a to na logické úrovni. Oba způsoby komunikace jsou naznačeny na obrázku 1.3.



Obrázek 1.3: Horizontální a vertikální komunikace v ISO/OSI

Pod pojmem entita si můžeme představit (spuštěnou) instanci některého konkrétního protokolu nebo jejich sady. Jeden SAP může být v jednom okamžiku využíván pouze jedním uživatelem a jedním poskytovatelem, ale jedna entita může zároveň používat více SAPů.

Protokol tedy při odesílání dat (podle obrázku 1.3 na stroji vlevo)

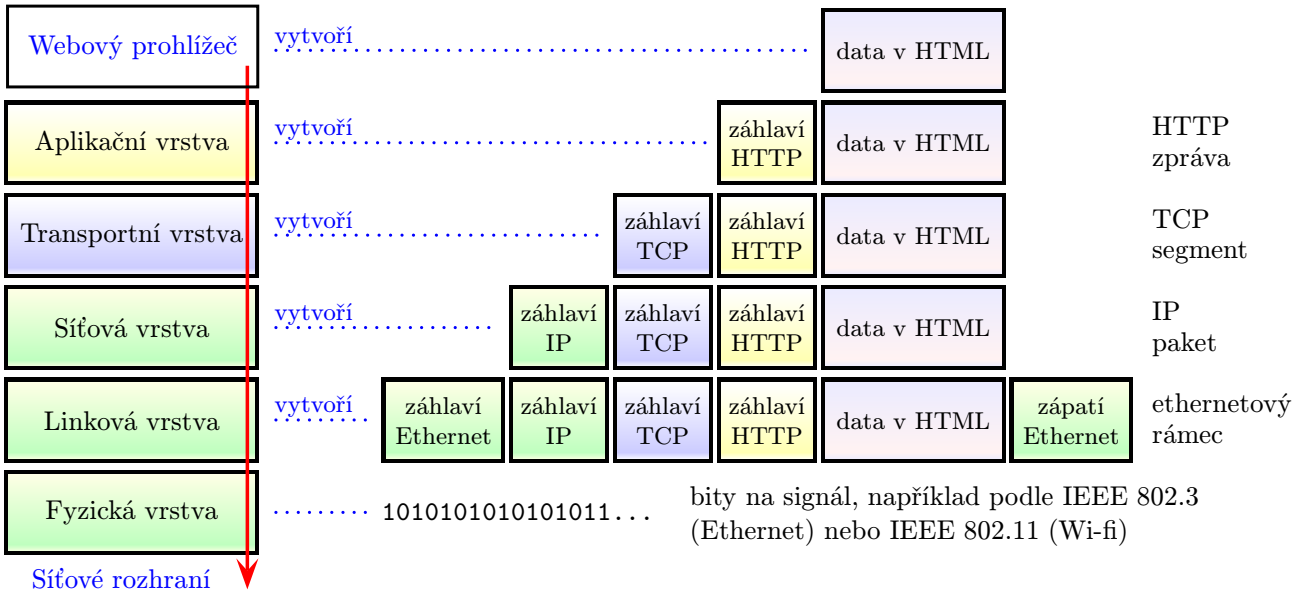
- obdrží data přes SAP od vyšší vrstvy,
- pokud je to nutné, stanoveným způsobem je zpracuje či rozdělí na menší bloky,
- stanoví příslušné metainformace (tj. informace o informacích), například adresy, velikost dat, informace o tom, jak se má po cestě s daty zacházet, apod.,
- přidá záhlaví (header) s metainformacemi a pokud je to třeba, pak i zápatí (trailer), přidá k datům, čímž data „zabalí“ do PDU (paket, rámec, datagram apod.),
- předá PDU přes SAP nižší vrstvě.

Jestliže jsou data naopak přijímána (podle obrázku 1.3 na stroji zcela vpravo), pak protokol

- přijme PDU přes SAP od nižší vrstvy,
- oddělí od PDU záhlaví a zápatí (pokud tam ovšem zápatí je),
- analyzuje metainformace v záhlaví a zápatí, stanoveným způsobem je zpracuje,
- pokud byla data před odesláním rozdělena na více bloků a jedná se o vrstvu, kde jsou bloky kompletovány (pozná se ze záhlaví), postupně shromáždí všechny bloky a zkompletuje,
- „rozbalená“ data předá přes SAP do vyšší vrstvy.

Není řečeno, že se nutně mají zúčastnit všechny vrstvy, některé nejsou pro konkrétní druh komunikace potřebné.

Při odesílání se tedy provádí *enkapsulace* (zabalení, zapouzdření, encapsulation) dat a při přijetí *dekapsulace* (rozbalení, decapsulation). To, co je pro nadřízenou vrstvu její vlastní PDU, to je pro jí podřízenou vrstvu pouze sekvence dat, ze kterých vytvoří vlastní PDU.



Obrázek 1.4: Zapouzdření PDU protokolu HTTP

Na obrázku 1.4 je naznačen proces enkapsulace. Postup při odesílání:

- data vyprodukovaná webovým prohlížečem jsou na vrstvě L7 zabalena do protokolové datové jednotky protokolu HTTP (do záhlaví se uloží například adresa serveru, informace o „žádajícím“ webovém prohlížeči, znakové sadě, časové údaje a další),
- následně se na transportní vrstvě dojedná spojení (nebo se využije už dojednané), pokud je to nutné, provede se segmentace (rozdělení na menší bloky) a následně se připojí TCP záhlaví (to obsahuje například číslo portu určující, že na nadřízené vrstvě komunikuje protokol HTTP, informace pro zajištění zkompletování celku při rozdělení na víc segmentů, kontrolní součet atd.), čímž je vytvořen TCP segment,
- na vrstvě L3 je tento segment předán entitě protokolu IP a je přidáno záhlaví protokolu IP (obsahuje například IP adresu zdroje a cíle, informaci o tom, který protokol sestavil to, co je „uvnitř“, atd.), výsledkem je IP paket,
- přes SAP mezi síťovou a linkovou vrstvou se IP paket dostane k entitě protokolu IEEE 802.3 (Ethernet), který před něj připojí ethernetové záhlaví (to obsahuje například synchronizační sek-

venci bitů, aby byl rozeznatelný začátek bitů paketu, fyzické adresy apod.) a za něj ethernetové zápatí (s kontrolním součtem),

- další na řadě je fyzická vrstva, která již zajistí převod sekvence bitů na signál podle určeného protokolu a připojeného přenosového média.

Na straně adresáta proběhne opačný proces – rozbalování.

Ve skutečnosti by se tohoto přenosu účastnily i další protokoly, například protokol DNS překládající adresy nebo při zabezpečené komunikaci protokol SSL.

Všimněte si, že některé vrstvy modelu ISO/OSI se tohoto procesu neúčastní, jsou tedy výjimky z pravidla komunikují pouze bezprostředně sousedící vrstvy – protokoly aplikační vrstvy mohou používat přístupové body SAP z vrstev L6, L5 i L4.

Jak se vlastně komunikuje přes takový SAP, co vše se na tomto přístupovém bodu děje? Ke každému SAPu jsou definována komunikační *primitiva*, což jsou jednoduché funkce, například *Request* (žádost o poskytnutí služby k nižší vrstvě, navazuje komunikaci přes SAP směrem dolů na straně odesílajícího stroje), *Indication* (upozornění na potřebu komunikace od nižší vrstvy k vyšší vrstvě, na straně přijímajícího stroje), *Response* (odpověď na Indication), *Confirm* (odpověď na Request).

Tato primitiva mohou mít, podobně jako běžné funkce, parametry – buď se jedná o SDU předávanou přes SAP z vyšší vrstvy, nebo se může jednat o dodatkové provozní informace, které nemají být součástí výsledné PDU.

Poznámka:

Zvídavého čtenáře určitě napadlo, že musí existovat způsob, jak se například síťová vrstva (L3) „dozví“, na kterou adresu má být vlastně dotyčný paket poslán. Přes záhlaví PDU to být nemůže, protože dovnitř záhlaví vyšších vrstev se protokol IP nedostane (je to pro něj prostě součást dat, která je třeba poslat), navíc v některých záhlavích tato informace ani není. Ano, dozví se to z parametrů primitiv. Další informace najdete na

http://www.erg.abdn.ac.uk/users/gorry/course/intro-pages/service-prim.html.

Socket je kombinace síťové adresy (používané na vrstvě L3) a čísla portu (používaného na vrstvě L4). Pokud tento pár zapisujeme „ručně“ (třeba do adresního řádku webového prohlížeče), umístíme mezi oba údaje tečku. Místo síťové (tedy číselné) adresy může být použita doménová adresa.

Příklad

Socket na server `www.neco.cz` na portu 8080 zapíšeme jako `www.neco.cz:8080`, tedy umístíme mezi adresu a port dvojtečku. Pro IPv4 adresu to bude podobné, například `193.84.214.5:8080`.

S adresou IPv6 to je složitější, protože ta obsahuje dvojtečky sama o sobě, ale přitom zápis musí být jednoznačný. Proto například při zápisu do adresního řádku webového prohlížeče IPv6 adresu umístíme do hranatých závorek: `[2001:718:2201:208:5]:8080`.

Sockety (sockets) najdeme na vrstvách L5–L7, víceméně fungují jako rozhraní mezi aplikačními protokoly a transportní vrstvou. Pro aplikace jsou dostupné jako Socket API (tj. rozhraní pro programování aplikací) ve formě knihoven obsahujících funkce pro práci se sockety (pro vytvoření socketu, akcepto-

vání na druhé straně spojení, naslouchání, čtení, zápis do socketu). Speciálním typem socketu je *stream socket*, který zaručuje dodání více bloků posílaných dat ve správném pořadí, a tedy na transportní vrstvě spolupracuje pouze s protokoly zajišťujícími spojovaný přenos (třeba TCP).

Se sockety se setkáme jak v UNIXových systémech (včetně Linuxu a MacOS X), tak i ve Windows (WinSock API).

1.5.3 Protokolové zásobníky

Sada protokolů (Protocol Suite) je definice (určení) skupiny protokolů, které vzájemně spolupracují. *Protokolový zásobník* (Protocol Stack) je implementace některé sady protokolů. Tyto dva pojmy se často používají jako synonyma.

Nemusí nutně jít o specifikaci protokolů pro naprosto všechny vrstvy modelu ISO/OSI, mohou být specifikovány pouze některé, přičemž se předpokládá spolupráce s jiným (doplňujícím) protokolovým zásobníkem.

TCP/IP Protocol Stack (taky se nazývá *Internet Protocol Suite*) je sada navzájem spolupracujících protokolů, kterou potřebujeme na zařízeních připojených k rozsáhlé síti (typicky Internetu). Kromě protokolů obsažených přímo v názvu (TCP, IP) zahrnuje ještě další, nejvíc na aplikační vrstvě. Přímo jsou určeny protokoly na vrstvách L3–L7, pro nižší vrstvy je pouze specifikováno rozhraní.

Tento protokolový zásobník je formalizován jako model TCP/IP, kterému se budeme podrobněji věnovat v následujícím textu (včetně protokolů).

IPX/SPX je konkurenčním protokolovým zásobníkem k TCP/IP od společnosti Novell vytvořeným pro operační systém Novell Netware – IPX pracuje na L3 místo protokolu IP, SPX na vrstvě L4 místo TCP. Byl projektován spíše pro menší sítě, zatímco TCP/IP je určen i pro rozlehlé sítě. V současné době se už téměř nepoužívá.

Protokolové sady pro lokální sítě jsou například Ethernet (IEEE 802.3), Wi-fi (IEEE 802.11) Token Ring (IEEE 802.5, už se nepoužívá) a další. Obvykle implementují pouze vrstvy L1 a L2, nad ně se nasouvá obvykle protokolový zásobník TCP/IP (ten pro změnu přímo nespecifikuje protokoly na L1 a L2).

Protokolové sady pro rozlehlé sítě jsou například ATM, Frame Relay, MPLS a další. Taky se obvykle napojují na TCP/IP, ale každá „trochu jinak“.

Například ATM se podsouvá pod síťovou vrstvu (L3), ale mezi ni a svou implementaci vrstvy L2 vsouvá speciální přizpůsobovací vrstvu. Frame Relay implementuje vrstvu L2, na L1 předpokládá některé vhodné fyzické rozhraní, většinou dle standardů EIA/TIA (spoléhá na ISO/OSI).

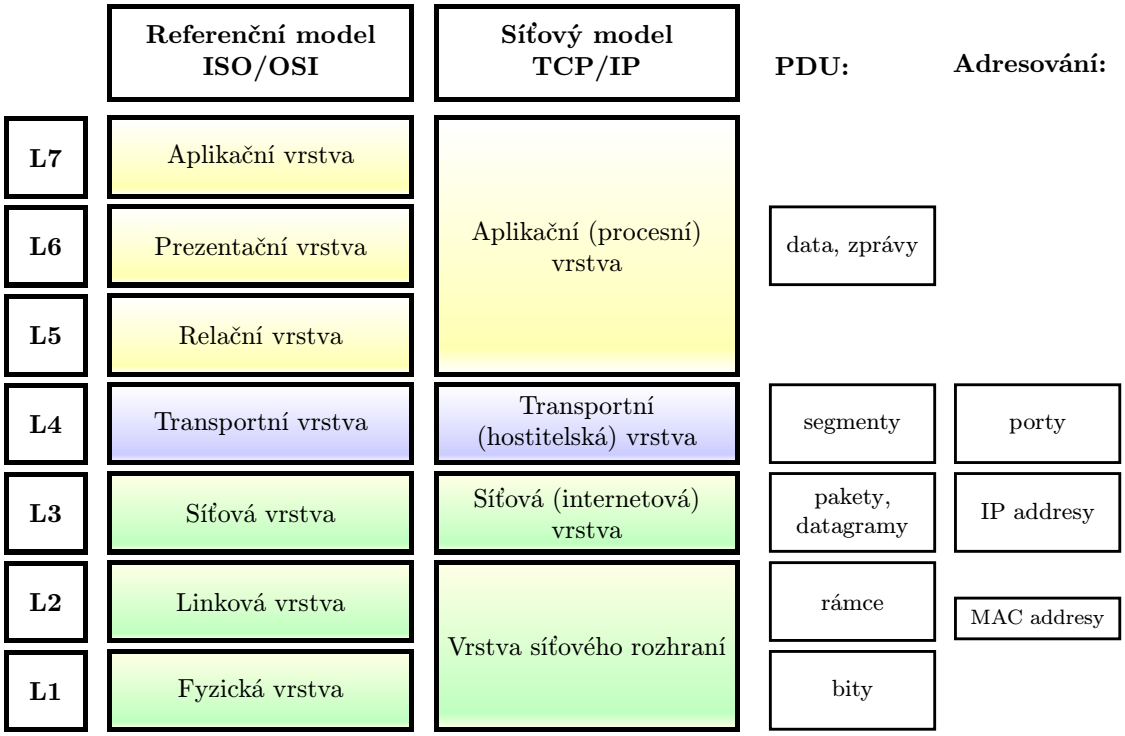
Oproti tomu MPLS se vsouvá mezi L2 a L3, tedy MPLS paket v sobě zabaluje paket z vrstvy L3 (většinou IP paket) a je zabalen do rámce vrstvy L2 (například do etnernetového rámce). Taky může běžet nad ATM nebo Frame Relay, a tedy lze využít technologie ze starších zařízení. Taky dokáže běžet nad PPP a dalšími protokoly pro přístupové sítě.

Protokolové sady pro mobilní sítě jsou například sady pro LTE, GPRS, CDMA, UMTS a další. Implementují obvykle vrstvy L1 a L2 a předpokládají některé konkrétní protokoly i na vyšších vrstvách, ale ve skutečnosti záleží, o jaký typ zařízení jde (koncové zařízení bude potřebovat jinou sadu protokolů než základnová stanice nebo jiná specializovaná zařízení v mobilní síti).

1.6 Síťový model TCP/IP

Referenční model ISO/OSI je velmi komplexní, tudíž složitý, a příliš teoretický. Postupně bylo vytvořeno několik zjednodušených variant, z nichž je nejznámější právě síťový model TCP/IP, který je také nazýván DoD model (USA Department of Defense Model, tedy model Ministerstva obrany USA). Jeho součásti jsou také standardizovány organizací IETF a dostupné v RFC dokumentech.

Síťový model TCP/IP je vlastně formální popis síťového zásobníku TCP/IP, jeho napojení na spolupracující zásobníky a obecně možnost zapojení jiných protokolů. Skládá se ze čtyř vrstev, jejichž vztah k vrstvám RM OSI je naznačen na obrázku 1.5.



Obrázek 1.5: Srovnání modelů RM ISO/OSI a TCP/IP

Vztahy vyjádřené na obrázku platí i co se týče funkčnosti – aplikační (procesní) vrstva TCP/IP plní tutéž roli jako vrstvy L5–L7 v ISO/OSI. TCP/IP je navržen tak, aby

- byl co nejvíc decentralizovaný (žádná centrální správa),
- byl co nejodolnější vůči různým (i kritickým) podmínkám provozu a co nejodolnější vůči přenosovým chybám,
- nechával co nejvíc práce na koncových zařízeních (jádro sítě musí být rychlé a pružné), a aby byl spravovatelný distribuovaně (každá část si spravuje „to svoje)“,
- dokázal propojit i sítě s hodně odlišnou síťovou architekturou a technologiemi (aby byl co nejuniverzálnější při zachování předchozích vlastností).

Postupně projdeme všechny vrstvy a na rozdíl od referenčního modelu se soustředíme především na protokoly pracující na těchto vrstvách.

Vrstva síťového rozhraní. Tato vrstva v sobě sdružuje funkčnost vrstev L1 a L2 referenčního modelu. Přímo v TCP/IP pro ni nejsou stanoveny žádné protokoly, je jen určeno, jak mají komuni-

kovat se síťovou vrstvou. Komunikuje s hardwarem (síťovým rozhraním), případně její část může být hardwarově implementovaná.

Obvykle se do této vrstvy napojují protokolové sady lokálních a rozlehlých sítí (jinými slovy – přímo v TCP/IP pro tuto vrstvu nejsou žádné protokoly uvedeny), například

- IEEE 802.3 (Ethernet),
- IEEE 802.11 (Wi-fi),
- protokolové sady WAN sítí nebo jejich části, xDSL, mobilních sítí.

Na nižší části této vrstvy (obdoba L1) najdeme jednoduchá zařízení pracující pouze se signálem typu hub (rozbočovač) nebo repeater (opakovač). Ve vyšší části této vrstvy (obdoba L2) pak o něco složitější zařízení pracující s rámci a spojující (či oddělující) jednotlivé segmenty sítě, tedy switch (přepínač) a bridge (most).

Pokud si budeme všímat jen vyšší části této vrstvy, pak ze probíhá proces *přepínání rámců*.

Síťová vrstva. Také se nazývá „internetová vrstva“, na této vrstvě probíhá *internetworking* (propojování sítí), včetně *směrování* mezi sítěmi. Pojem „internet“ (s malým počátečním písmenem) značí obecně síť sítí, tedy síť propojující nikoliv jen jednotlivé uzly, ale menší sítě. Typické zařízení této vrstvy je router (směrovač) nebo L3 switch.

Co se protokolů týče, bude nás zajímat především protokol IP (Internet Protocol), a to jeho verze IPv4 a IPv6, které jsou dnes v praxi používány, a dále směrovací protokoly (OSPF, EIGRP, RIP, BGP a další).

Z dalších protokolů to je například ICMP.

(Transportní vrstva). Také ji nazýváme „hostitelská vrstva“, protože je implementována pouze na hostitelích v síti. *Hostitel* je název pro koncové zařízení (počítač, server, tablet, chytrá televize, atd.), které „hostí“ data, aplikace, služby; každý hostitel má svůj název (hostname).

Poznámka:

Pozor – „hostitel“ se anglicky řekne „host“, kdežto anglickým ekvivalentem českého slova „host“ je „guest“. Tedy anglické „hostname“ se překládá jako „název hostitele“ (nebo prostě název koncového zařízení).

Nejznámější protokoly transportní vrstvy jsou TCP a UDP.

Aplikační vrstva. Také „procesní vrstva“, protože zdejší protokoly komunikují s procesy. Sdružuje v sobě vše, co je v RM ISO/OSI na nejvyšších třech vrstvách (L5–L7). Na této vrstvě najdeme velké množství protokolů, s většinou z nich komunikují aplikace potřebující přistupovat na síť. Příklady aplikačních protokolů: HTTP, SMTP, IMAP, POP3, DNS, DHCP, atd.

Poznámka:

Každý z modelů má své výhody a nevýhody. Většinou se používá terminologie z ISO/OSI (například označení vrstev, entity apod.), ale celkové rozvržení činností souvisejících se sítí a implementace postupů jsou obvykle podle TCP/IP.

Definice (Internetworking)

Internetworking je mechanismus propojování sítí, tedy zajišťování komunikace mezi (inter) sítěmi (networks). Tento mechanismus probíhá na vrstvě L3, a to na zařízeních typu router (směrovač) nebo switch (přepínač) s funkcionalitou vrstvy L3.

1.7 Charakteristiky přenosu a přenosových cest

1.7.1 Přenos v základním a přeloženém pásmu

Přenos se provádí buď v základním pásmu nebo v přeloženém pásmu.

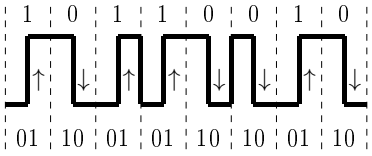
Přenos v základním pásmu (baseband). Přenos probíhá tak, že sekvenci jedniček a nul, která má být přenesena, přímo zakódujeme do frekvenčního spektra (emitujeme signál) a takto vzniklý signál je přenesen komunikačním kanálem. Obvykle nám stačí nízké frekvence. Baseband se používá pro metalické lokální sítě (Ethernet na měděném kabelu) a v základu pro optické sítě.

Nevýhodou je omezený dosah (menší vzdálenost pro přenos) a problematická synchronizace (pokud bychom sekvenci nul a jedniček kódovali tak jak je, pak by dlouhé sekvence nul nebo dlouhé sekvence jedniček byly špatně dekódovatelné, nebylo by možné stanovit jejich délku). Odesílající a přijímající strana potřebují mít správně seřízené časovače, aby bylo možné synchronizovat intervaly mezi úseky představujícími jednotlivé bity, ale u dlouhých sekvencí bitů se stejnou hodnotou to nestačí.

Tento problém se obchází tak, že posloupnost bitů před kódováním na signál pozměníme podle určitého klíče tak, aby se v posloupnosti nevyskytovaly dlouhé sekvence stejných číslic, a samozřejmě aby bylo možné data po ukončení přenosu vrátit do původního stavu. Původní sekvence bitů se nahradí jinou sekvencí bitů tak, aby se dostatečně často „střídaly“ hodnoty 0 a 1. Dále si představíme několik obvyklých kódování pro přenos v základním pásmu.

Kódování Manchester je velmi jednoduché. Spočívá v zakódování bitů do směru kmitu signálu – 0 je kódována jako přechod shora dolů, 1 je kódována jako přechod zdola nahoru (viz obrázek 1.6).

0:	High → Low	10
1:	Low → High	01



Obrázek 1.6: Kódování Manchester pro oktet $(10110010)_2$

Sekvence bitů 10111000 je při kódování Manchester zakódována na 01 10 01 01 01 10 10 10.

Jak vidíme, stejné symboly vedle sebe získáme pouze na těch místech, kde se v původní sekvenci měnily hodnoty bitů, a to nejvýše dva stejné symboly. To je výhodou kódování Manchester, nevýhodou je navýšení délky posílaných dat (délka se oproti původním zdvojnásobí). U vyšších rychlostí jsou další techniky jako data scrambling nebo samoopravitelný kód (Gigabit Ethernet).

Existují dvě základní varianty kódování Manchester:

- původní G.E. Thomas, kde jsou přechody opačně než na výše uvedeném obrázku (1: přechod dolů, 0: přechod nahoru),
- IEEE 802.3 pro 10b Ethernet s přechody přesně podle výše uvedeného obrázku.

Druhá varianta se tedy používá v 10Base-T, a také v jednom z typů spojení přes NFC.

Také se můžeme setkat s variantou differential Manchester, kde jsou přechody vždy jen při změně bitů (01 nebo 10). Tato varianta se používá například při zápisu na pevné disky.

Kódování 4B/5B: z každé čtveřice bitů se vytvoří pět bitů tak, aby v celé sekvenci bylo co nejvíce jedniček (nejméně dvě na pětici), na začátku pětice nejvýše jedna nula, na konci nejvýše dvě nuly. Kódy jsou v tabulce 1.1.

4B	5B	4B	5B	4B	5B	4B	5B
0000	11110	0100	01010	1000	10010	1100	11010
0001	01001	0101	01011	1001	10011	1101	11011
0010	10100	0110	01110	1010	10110	1110	11100
0011	10101	0111	01111	1011	10111	1111	11101

Tabulka 1.1: Tabulka kódů pro 4B/5B

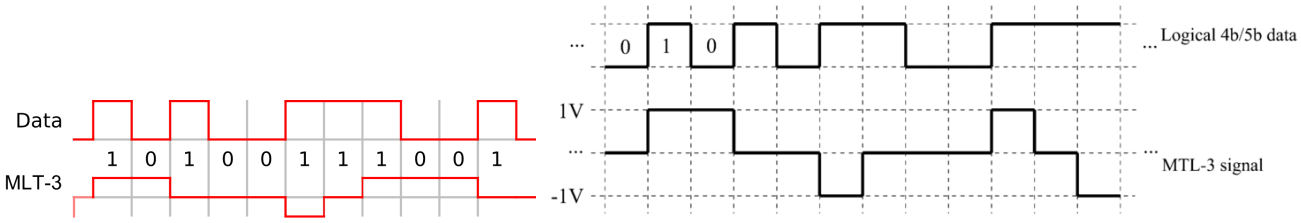
Sekvenci 10111000, kterou jsme v kódování Manchester zakódovali do 16 znaků, zde zpracujeme na 1011110010 o délce 10 znaků.

4B/5B bylo původně určeno pro FDDI, v Ethernetu se s ním setkáme například v 100Base-X.

Kódování 8B/6T znamená, že 8 bitů původní sekvence je zakódováno 6 změnami signálu (ternární symboly, tedy stavy -,0,+). Používá se na kroucené dvojlince kategorie 3 se 4 páry, data se rozdělí do 3 párů, přenášejí se vždy jedním směrem. Čtvrtý pár je používán pro indikaci kolizí.

Další „lomítkové“ varianty také spočívají v tom, že určitý počet bitů ze zdroje se kóduje do určitého počtu bitů/stavů v cílovém signálu. Například *kódování 8B/10B* mapuje 8bitové sekvence do 10bitových sekvencí, používá se např. v 1000Base-X.

Kódování MLT-3 používá tři úrovně napětí (předchozí kódování používala jen dvě úrovně), a to -, základna, +. Změna napětí proběhne pouze na signál 1, a to na „sinusovce“. Toto kódování se obvykle kombinuje s některým jiným, například signál pro MLT-3 může být předzpracován kódováním 4B/5B.



Obrázek 1.7: Kódování MLT-3 (vlevo: samotné, vpravo: kombinace s 4B/5B)[1]

Na obrázku vlevo vidíme kódování MLT-3 použité samostatně, vpravo v kombinaci s 4B/5B.

Oproti Manchestru má signál jen čtvrtinovou frekvenci, proto méně vyzařuje, generuje mnohem méně rušení do okolí.

Například 100Base-TX kombinuje 4B/5B, NRZI (viz dále) a MLT-3.

[1] Zdroj: podle https://cs.m.wikipedia.org/wiki/Linkov%C3%BD_k%C3%B3d, https://www.researchgate.net/figure/Mapping-from-a-4b-5b-data-stream-to-MLT-3-signal-level_fig1_3056686

Kódování PAM-5 používá čtyři úrovně napětí pro data a pátou pro řídící bit. Data se kódují následovně:

- `00` → `-1.0 V`
- `01` → `-0.5 V`
- `10` → `+0.5 V`
- `11` → `+1.0 V`

PAM-5 se používá například v 1000Base-T.

Kódování NRZ (Non Return to Zero) používá pro reprezentaci signálu dvě úrovně, z nich žádná není 0 (základna), například +1,-1. Existuje více variant – unipolární (1 je představována kladným napětím, 0 také kladným, ale menším), bipolárním (0 kladné, 1 záporné), NRZI, atd.

Kódování NRZI (Non Return to Zero – Inverted) znamená, že 1 vyvolá změnu signálu, 0 ne. Je to obdoba MLT-3, ale jen na dvou úrovních (kladné a záporné).

Poznámka:

Pokud vám z předchozího studia nic neříkají označení 100Base-T, 1000Base-X apod., nalistujte si tyto zkratky v následující kapitole o Ethernetu.

Přenos v přeloženém pásmu (broadband). Tento přenos probíhá tak, že sekvenci jedniček a nul sice zakódujeme také do frekvenčního spektra jako u basebandu, ale vzniklý signál přeložíme do takového pásma, které je pro tento konkrétní přenos určeno (nebo kódujeme data přímo do příslušného frekvenčního pásma). Tomuto překladu říkáme *modulace*.

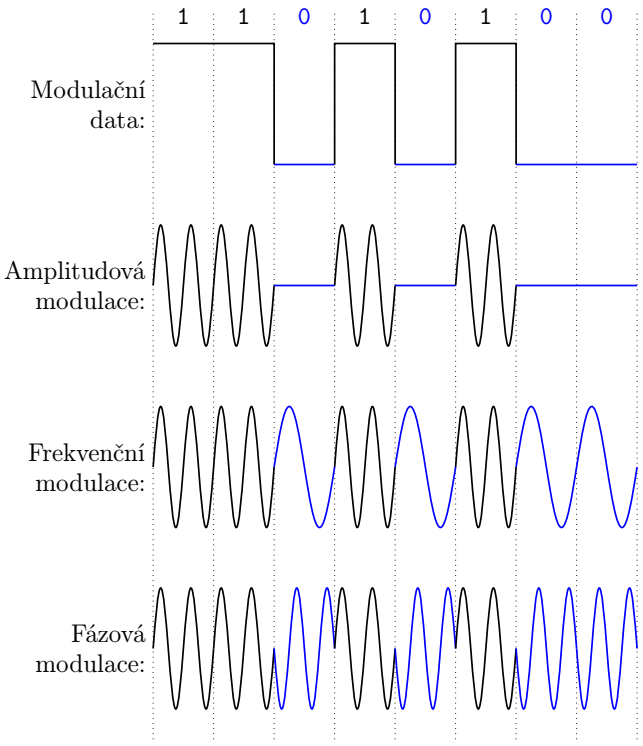
Modulace tedy probíhá následovně:

- Určíme *nosnou*, což je vhodný signál na té frekvenci, na které mají být data přenesena, volí se signál s harmonickou frekvencí, obvykle sinusoida.
- Pozměníme parametry (amplitudu, frekvenci nebo fázi) tohoto signálu podle toho, jaká data mají být přenášena – modulujeme data na signál.
- Podle potřeby vše sloučíme a odešleme.

Rovnice popisující průběh modulace je $s(t) = A\sin(\omega \cdot t + \varphi)$, kde t je čas (to je proměnná), A je amplituda signálu, ω je úhlový kmitočet a φ je fázový posun.

Na obrázku 1.8 na straně 24 jsou ukázky tří základních typů modulace – amplitudové (mění se amplituda), frekvenční (mění se frekvence signálu) a fázové (mění se fáze), vodorovná osa je proměnná t, svislá výsledek $s(t)$. Ve skutečnosti se často mění více než jen jeden z těchto tří parametrů, navíc různé modulace mohou tentýž parametr měnit s různou intenzitou. Podíváme se na několik nejznámějších modulací.

QAM (Kvadraturní amplitudová modulace) používá kombinaci fázového (PSK) a amplitudového (ASK) posunu. Dokáže modulovat jak analogový, tak i digitální signál do příslušného frekvenčního rozsahu ve výsledném analogovém signálu. Existuje více variant: 16-QAM, 64-QAM, 256-QAM. V každé variantě je u určitého počtu nosných použita amplitudová modulace, u zbývajících fázová, kombinací získáme určitý počet stavů reprezentujících modulační data či signál. Například u nejjednodušší varianty



Obrázek 1.8: Příklad amplitudové, frekvenční a fázové modulace digitálních dat

16-QAM se používají dvě nosné a existuje 36 kombinací, ale využíváno je pouze 16 (těch, které jsou od sebe nejsnáze rozlišitelné).

Modulace QAM v kombinaci s multiplexováním OFDM (viz dále) se dnes používá především v bezdrátových a mobilních sítích, například Wi-fi podle standardu IEEE 802.11n používá až 64-QAM, kdežto IEEE 802.11ac používá až 256-QAM (při špatném signálu „spadne“ na nižší variantu). V mobilních sítích čtvrté generace (LTE Advanced) se používá 256-QAM.

Z dalších modulací můžeme jmenovat například CAP (Carrierless Amplitude/Phase modulation), která se využívá v ADSL, taktéž s multiplexem.

Baseband	**Broadband**
posíláme digitální signál	výsledkem je analogový signál
signál přímo emitujeme	modulační signál modulujeme na nosnou
na kratší vzdálenosti	na delší vzdálenosti
signál využívá celou šířku pásma	signál lze omezit na stanovenou šířku pásma

Tabulka 1.2: Rozdíl mezi přenosem v základním a přeloženém pásmu

Další informace:

- http://www.cs.vsb.cz/grygarek/PS/lect/PREZENTACE/fyzPrincipy.pdf
- http://www.earchiv.cz/l226/slide.php3?l=4

1.7.2 Multiplex

Vezměme jeden přenosový kanál s určitými fyzikálními charakteristikami (frekvence apod.). Za normálních okolností bychom tímto přenosovým kanálem mohli přenášet jen jeden signál.

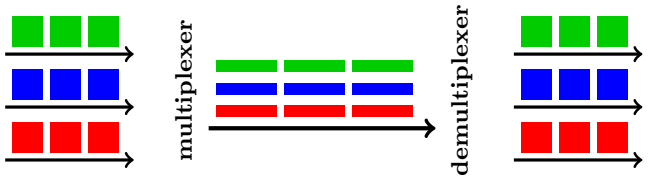
Multiplexing je technika, která umožňuje rozdělit přenosový kanál s dostatečnou šířkou pásma na více logických subkanálů, zdánlivě samostatných, a v každém přenášet jiné bloky dat. V reálu to znamená, že se více signálů sloučí do jediného signálu. Multiplex vlastně znamená „multiple access“, tedy vícenásobný přístup (současný přístup více uživatelů k přenosovému kanálu).

Komponenta, která provádí multiplexing, se nazývá multiplexer (multiplexor, MUX). Opačnou operací k multiplexingu je demultiplexing (DEMUX, DMX): Na odesílající straně přenosového kanálu se provádí multiplexing (rozdělení do subkanálů), na přijímající straně demultiplexing (odebrání ze subkanálů).

Existuje několik běžných typů multiplexování:

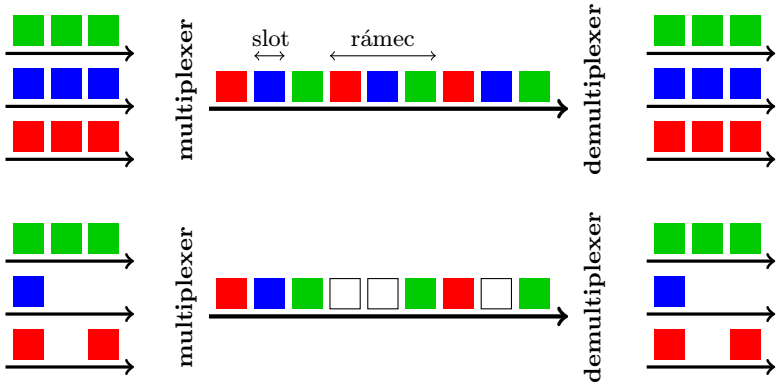
Frekvenční multiplex (FDM, Frequency Division Multiplex): každému přenášenému signálu je přidělena část šířky pásma (mezi přidělenými pásmy musí být odstup, aby nedocházelo k rušení), a tento signál je namapován do této přidělené části. Každý subkanál má tedy přidělen vlastní interval frekvencí.

FDM je použitelný pouze na analogový signál, a další jeho nevýhodou je, že je vhodný spíše pro „stabilní počet“ uživatelů (resp. existuje strop pro množství uživatelů).



Obrázek 1.9: FDM – Frekvenční multiplex

Frekvenční multiplex známe například z rozhlasu, kdy různé stanice mají přiděleny různé frekvence, dále například v satelitních přenosech a kabelových sítích. Také se používal na analogové telefonní síti – každý „telefonista“ měl vyhrazen subkanál o šířce 3.1 kHz.



Obrázek 1.10: TDM – Časový multiplex (nahoře plné vytížení, dole částečné vytížení)

Časový multiplex (TDM, Time Division Multiplex): přenosový kanál je střídavě přidělován různým konkrétním přenosům. Princip je naznačen na obrázku 1.10. Přenosový kanál je rozškálován na tzv. *rámce* (pozor, to nejsou tytéž rámce jako na vrstvě L2, jen shoda názvů), v každém rámci je pro každého odesílajícího vyhrazen jeden *slot* (zásuvka), do kterého lze umístit paket (nebo ho nechat prázdný).

Problém „čistého“ TDM je, že je taky vhodný spíše pro relativně konstantní počet uživatelů, navíc víceméně podobně komunikujících (co se týče počtu odesílaných paketů). Pokud některý uživatel odesílá výrazně méně paketů, jsou jeho sloty nevyužity, jak vidíme na obrázku 1.10 dole. Další nevýhodou časového multiplexu je nutnost neustálé synchronizace rámců – oběma komunikujícím stranám musí být jasné, kdy začíná a končí rámec a kdy začíná/končí který slot a komu patří. Nicméně na rozdíl od FDM je vhodný i pro digitální signál.

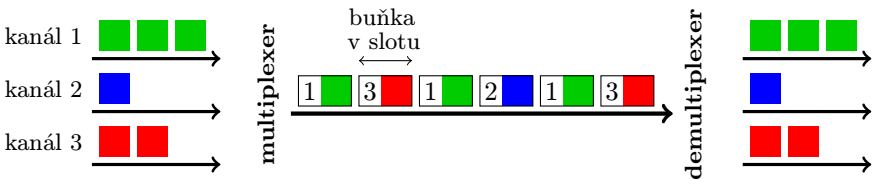
TDM se používá například v mobilních sítích při použití základního přesnosu GSM.

Kódový multiplex (CDM, Code Division Multiplex, také CDMA, Code Division Multiple Access, rozprostřené spektrum) je digitální metoda, jejíž vznik byl motivován především potřebou bezpečnosti (co nejvíc ztížit odposlech). Každý dílčí přenos je zakódován (kódování probíhá v koncových zařízeních, neprovádí je multiplexer), všechny paralelní (zakódované) přenosy jsou pak multiplexerem sloučeny a přenášeny sdíleným kanálem až ke koncovým zařízením. Cílové koncové zařízení pak s pomocí speciálního kódu dekóduje jen to, co mu ve skutečnosti patří, bez tohoto kódu se k obsahu subkanálu nelze dostat.

Problém CDM je potřeba složité synchronizace a horní limit pro počet subkanálů (příliš mnoho subkanálů by se navzájem rušilo a nebylo by možné je dekódovat). Také je nutné zajistit bezpečnou domluvu o kódu pro dekódování. Existuje víc různých variant CDM, většinou se používají v mobilních sítích třetí generace.

FDM, TDM a CDM jsou základní typy multiplexu, ale existují i velmi používané odvozené typy:

Statistický multiplex je podobný časovému multiplexu v tom, že přenosový kanál je členěn na sloty. Ovšem u statistického nejsou sloty napevno přiděleny konkrétním subkanálům, ale jsou přidělovány podle potřeby – vytíženější subkanál dostane víc slotů než méně vytížený. Aby bylo jasné, ke kterému subkanálu který slot patří, musí být k přenášeným blokům dat přidána informace o subkanálu (záhlaví).



Obrázek 1.11: Statistický multiplex

Výhodou statistického multiplexu je o něco nižší potřeba synchronizace (není nutné zabývat se rámci slotů), synchronizují se jen samotné sloty a celková komunikace zůstává asynchronní. Další výhodou je lepší vytěžování přenosového kanálu. Nevýhodou je nutnost opatřovat data záhlavími a s tím spojená režie.

Statistický multiplex se používá v některých WAN sítích, například v ATM (do slotů se skládají datové jednotky zvané buňky).

Vlnový multiplex (vlnové dělení, WDM, Wave Division Multiplex): je to obdoba frekvenčního multiplexu, ale u optického signálu, kde místo frekvencí používáme dělení podle vlnových délek neboli barev světla (což je, jak víme, vlastně podobné). Každý signál dostane přidělen určitý rozsah vlnových délek, které jsou použity při jeho přenosu. Podle příkladu v předchozí sekci je zřejmé, že u světla se s vlnovými délkami pracuje lépe než s frekvencemi, jsou to menší čísla.

WDM se používá například při přenosu optickým kabelem, tedy především v optických WAN sítích a dále například v technologii FTTx (optické vlákno se vede co nejblíž zákazníkovi). Na signál generovaný laserem nebo LED diodou je po multiplexování modulován signál.

Ortogonální multiplex (OFDM, Orthogonal Frequency-Division Multiplex) mapuje subkanály na různé frekvence podobně jako FDM, ale mnohem efektivněji. Zatímco FDM používá jedinou nosnou pro všechny subkanály, OFDM používá pro každý subkanál jinou nosnou, přičemž jednotlivé nosné jsou navzájem ortogonální, proto se mohou překrývat a přesto je lze na straně přijímače oddělit. Data přenášená přes subkanál jsou modulována na přidělenou nosnou některou vhodnou modulací, většinou se používá některá varianta modulace QAM.

OFDM (a taktéž jeho varianty) se dnes široce používá pro modulaci digitálních dat do analogového signálu v počítačových sítích (Wi-fi, WiMAX, LTE, xDSL apod.), ale také například pro přenos digitální televize.

Existují různé varianty OFDM přizpůsobené konkrétnímu způsobu využití v některé technologii. Například u mobilních sítí se často setkáváme s OFDMA (OFDM Access), kde jsou subkanály jednoho kanálu rozprostřeny po celém spektru a mohou být přidělovány různým klientům.

Další informace:

- http://fyzika.jreichl.com/main.article/view/156-harmonicke-kmitani
- http://www.cs.vsb.cz/grygarek/PS/lect/PREZENTACE/SdileniMedia.pdf
- http://www.wikiskripta.eu/index.php/Elektromagnetick%C3%A9_spektrum
- http://measure.feld.cvut.cz/cs/system/files/files/cs/vyuka/predmety/x38ssl/ofdm.pdf
- https://khanovaskola.cz/video/8/27/1423-amplituda-perioda-frekvence-a-vlnova-delka-periodickeho-vlneni

Kapitola 2

Lokální sítě – Ethernet

Opět budeme předpokládat určité znalosti z oblasti lokálních sítí, část zde uvedených informací je především pro připomenutí.

Pod pojmem lokální síť (LAN) budeme dále rozumět souhrn navzájem propojených zařízení (hostitelských/koncových zařízení, aktivních síťových prvků apod.), která patří do téže sítě (tj. jako aktivní síťové prvky jsou použity nejvýše switche/přepínače), obvykle v rámci jedné budovy či bytu, s typickou rozlohou jednotek až stovek metrů, výjimečně více.

V současné době jsou nejpoužívanějšími LAN technologiemi Ethernet (kabel) a Wi-fi (bezdrát). V této kapitole se zaměříme na Ethernet, Wi-fi budeme probírat v jedné z dalších kapitol.

2.1 Co je to Ethernet

Na původní specifikaci Ethernetu spolupracovaly společnosti Xerox, Digital a Intel, tato specifikace byla zveřejněna roku 1976 a označuje se jako DIX Ethernet (podle počátečních písmen společností).

Později byl Ethernet standardizován jako IEEE 802.3, ale v tomto standardu se název „Ethernet“ vůbec nevyskytuje a s původním DIX Ethernetem je nekompatibilní. Postupně se objevovaly různé varianty – pro různá přenosová média (metalické kabely různých kategorií, optické kabely) a také se navyšovala rychlost.

Definice (Ethernet, IEEE 802.3)

Síť Ethernet, resp. IEEE 802.3, je standard popisující souhrn technologií pro lokální počítačovou síť používající kabely (metalické nebo optické), kde hostitelská zařízení sdílejí stejnou šířku komunikačního pásma a vzájemně o ni soupeří.

Standard IEEE 802.3 popisuje implementaci pro fyzickou a linkovou vrstvu (tj. celou vrstvu síťového rozhraní podle TCP/IP) včetně metody komunikace a řešení kolizí.

Typické vlastnosti sítě Ethernet (nebo IEEE 802.3):

- Všechna zařízení v síti jsou rovnocenná, žádné nemá prioritu.
- Jako kabeláž se používá buď kroucená dvojlinka (twisted pair – nestíněná nebo stíněná) nebo optika, v datových centrech se můžeme setkat také například s twin-ax kabelem.

- Typickým aktivním síťovým prvkem je switch.
- Všechna zařízení na segmentu sdílejí přenosové médium, ovšem při použití switche je logická topologie segmentu vpodstatě point-to-point.
- Komunikuje se v poloviční nebo plném duplexu, dnes je typický plný duplex.

Poznámka:

V dalším textu budeme hovořit o Ethernetu, nicméně v standardu IEEE 802.3 ani jeho dodatcích se slovo „Ethernet“ vůbec nevyskytuje. Tento „rozpor“ bude dále vysvětlen.

2.2 Komunikace v Ethernetu

Přehled pojmů:

- *DTE* (Data Terminal Equipment, terminálové zařízení) – koncové zařízení v síti (počítač, server apod.),
- *DCE* (Data Circuit-terminating Equipment, komunikační zařízení, zařízení ukončující okruh) – zařízení v síti, které přeposílá provoz, obvykle není zdrojem ani cílem dat (obvykle switch),
- *kolizní doména* (segment) – souhrn zařízení v síti, která se navzájem „vidí“; pokud některé začne vysílat, všichni ostatní na segmentu jsou cílem a zároveň vysílat nemohou (jinak by došlo ke kolizi), vzhledem k aktivním síťovým zařízením:
  - hub neodděluje kolizní domény,
  - switch a router oddělují kolizní domény,
- *všesměrová doména* (broadcast doména) – souhrn zařízení v síti, kterým je doručeno broadcastové vysílání, jehož zdrojem je některé zařízení z tohoto souhrnu, vzhledem k aktivním síťovým zařízením:
  - hub ani switch neoddělují broadcastové domény,
  - router odděluje broadcastové domény.

Přístupová (kolizní) metoda určuje, jakým způsobem se rozhoduje, zda zařízení může či nemůže začít vysílat. Pro Ethernet je specifikována přístupová metoda CSMA/CD.

Definice (Přístupová metoda CSMA/CD)

V Ethernetu se používá technologie vícenásobného přístupu k přenosovému médiu s nasloucháním nosné a detekcí kolizních stavů – CSMA/CD:

- CS (Carrier Sense) – uzly v síti neustále naslouchají na nosné, zda nevysílá jiný uzel,
- MA (Multiple Access) – vícenásobný přístup, tedy kterýkoliv připojený uzel může začít vysílat, pokud nasloucháním zjistí, že nikdo jiný nevysílá,
- CD (Collision Detection) – pokud uzel v síti před vysíláním nestihne včas detekovat vysílání jiného uzlu (například tehdy, když oba začnou vysílat v přibližně stejné době), musí být schopen vzniklou kolizi signálů detekovat a patřičně na ni reagovat.

Použití přístupové metody má smysl tehdy, když se komunikuje v poloviční duplexu (nebo když používáme huby, ale to už je celkem historie).

Co se stane, když nastane kolize:

- Vysílající uzel buď chce vysílat a přitom zjistil, že vysílá jiný uzel, nebo detekoval kolizi (zjistil, že kromě něho vysílá i někdo jiný).
- Pokud už vysílal, přestane vysílat (ne okamžitě, musí umožnit i druhému vysílajícímu uzlu, aby kolizi detekoval). Vyšle *Jam signál*, což je signál oznamující kolizi na médiu.
- Podle speciálního algoritmu (Backoff algoritmus) určí dobu čekání a po uplynutí této doby se pokusí znovu vysílat.
- Jestliže i další pokus selže (buď ještě před vysíláním zjistí, že linka není volná, nebo opět zjistí kolizi), vrací se k předchozímu bodu (doba čekání bez vysílání a nový pokus).

Backoff algoritmus určuje, jak dlouho má uzel čekat s vysíláním, když zjistí, že vysílá jiný uzel. Účelem je zajistit, aby v případě více zájemců o vysílání každý z nich počkal jinou dobu (určenou náhodně generovaným číslem), čímž by se snížila pravděpodobnost další kolize.

Postup (Backoff algoritmus)

Existuje víc různých (různě složitých) variant tohoto algoritmu, základní (Exponential Backoff) je následující:

- Při kolizi vyšle „Jam“ signál, čímž informuje o kolizi a zrušení právě odesílaného rámce.
- Čeká po dobu $0 \ldots 51{,}2$ μs (náhodně vygenerované číslo z tohoto intervalu), pak se znovu pokusí o přenos.
- Jestliže přenos znovu selže, čeká po dobu $2 \times 0 \ldots 51{,}2$ μs (opět se generuje náhodné číslo) a pak se znovu pokusí o přenos.
- Pokud dojde k dalším selháním přenosu, čeká vždy po dobu $K \times 0 \ldots 51{,}2$ μs, kde K je náhodně vygenerované číslo z intervalu $0 \ldots 2^c - 1$, kde c je dosavadní počet kolizí (neúspěšných pokusů o odeslání).

Po 10 pokusech se již c nezvyšuje, pak je vždy K z intervalu $0 \ldots 2^{10} - 1$. Po 16 pokusech postup končí, rámec je považován za nedoručitelný.

Jak vidíme, po prvních dvou selháních přenosu se doba čekání určuje vygenerováním jednoho náhodného čísla, ale od třetího pokusu dále vlastně násobíme dvě náhodná čísla, přičemž pro první z nich se horní mez každým krokem exponenciálně zvyšuje. Tím se snižuje pravděpodobnost, že dva uzly opakují pokus o přenos ve stejnou dobu.

Poznámka:

Uvědomme si, že zatímco fyzická topologie se projevuje na vrstvě L1 (fyzické), logická topologie na vrstvě L2 (linkové). Původní Ethernet používající koaxiální kabel byl sběrnicový na fyzické i logické úrovni (na fyzické díky napojení zařízení na sdílené přenosové médium, na logické díky přístupové metodě CSMA/CD), novější je sběrnicový jen na logické úrovni, a to jen tehdy, když se používá CSMA/CD.

2.3 Ethernet na linkové vrstvě

2.3.1 Adresace na vrstvě L2

Na linkové vrstvě se používají MAC adresy (fyzické, hardwarové adresy). V každém rámci je MAC adresa cílového zařízení (adresáta) a MAC adresa zdrojového zařízení (odesílatele).

✎ Zařízení pracující na vrstvě L2 by mělo mít přehled o *fyzické topologii sítě*, tedy o adresách zařízení, s nimiž může komunikovat, a o tom, přes které síťové rozhraní (kterou cestou) je dotyčné zařízení dosažitelné. Tyto informace si každé zařízení vede ve své tabulce MAC adres (může se nazývat MAC tabulka, CAM tabulka apod., podle toho, jak ji nazve výrobce).

✎ *Hardwarová (MAC, fyzická, EUI-48) adresa* je vlastně nízkoúrovňovou adresou síťového rozhraní. Tuto adresu má každé síťové rozhraní, a to i tehdy, když zrovna není připojeno k síti.

Máme dva druhy MAC adres:

- adresa přidělená výrobcem (BIA – Burned-In-Address, „vypálená“), která je uložena „napevno“ v ROM nebo flash paměti síťového rozhraní,
- lokální (dočasná) MAC adresa, kterou jsme si nastavili sami.

V současných sítích se používají 48bitové MAC adresy (tj. 6 oktetů) a zapisují se většinou hexadecimálními číslicemi – MAC adresu zapíšeme pomocí 12 hexadecimálních číslic. Jednoznačnost je zajištěna takto:

- První polovinu adresy dostane výrobce přidělenou od sdružení IEEE (velcí výrobci mají přiděleno několik, menším stačí jedna), tedy první polovina je charakteristická pro výrobce a žádní dva výrobci nemají stejnou. Toto číslo se označuje OUI (Organizationally Unique Identifier).
- Druhou polovinu určuje již samotný výrobce, který si hlídá, aby byla tato čísla unikátní v rámci jemu přidělené první poloviny.

V tabulce 2.1 je výše popsaná struktura (dvě části) přehledněji naznačena.

24 bitů	+	**24 bitů**	=	**48 bitů**
OUI = identifikace výrobce	+	identifikace konkrétního výrobku	=	celá adresa
přiděluje IEEE	+	určuje výrobce	=	globálně jednoznačná

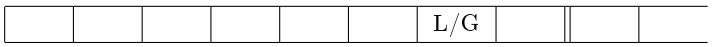
Tabulka 2.1: Struktura MAC adresy síťového rozhraní

Fyzickou adresu obvykle zapisujeme tak, že dvojice hexadecimálních číslic oddělíme dvojtečkou nebo pomlčkou, nebo dvojice oktetů oddělíme tečkou, případně je nijak neoddělujeme. Například:

`50:E5:49:A2:80:61` `50-E5-49-A2-80-61` `50E5.49A2.8061` `50E549A28061`

Změna MAC adresy není až tak běžnou věcí, ale někdy se hodí – například tehdy, když pro účely využití určité technologie či aplikace potřebujeme nutně MAC adresu v určitém konkrétním tvaru a není čas či možnost změnit konfiguraci, nebo ve virtualizovaném prostředí (třebaže i tam mohou být používány unikátní adresy). Hackeři používají pozměněné MAC adresy při pokusu o průnik do sítě chráněné blacklistem nebo whitelistem (seznamem zakázaných/povolených adres).

✎ Každá lokálně platná adresa (tj. ne BIA od výrobce) by správně měla mít nastavený *L/G bit*. Tento bit je v prvním oktetu adresy druhý zprava (v běžných síťových implementacích, například v Ethernetu), jak je vidět na obrázku 2.1.

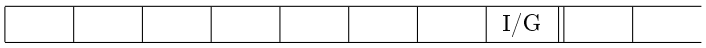


L/G bit: lokální (1) nebo globální (0) adresa

Obrázek 2.1: Umístění L/G (local/global) bitu v MAC adrese

Pokud jsou všechny hexadecimální číslice nastaveny na F, znamená to, že tato adresa je nejen lokálně platná (není globálně jednoznačná), ale navíc je *všeobecná* (broadcastová), tedy neoznačuje jedno konkrétní zařízení. Vypadá takto: `FF-FF-FF-FF-FF-FF`. Všeobecné MAC adresy se používají například tehdy, když je odesílán rámec určený pro všechna zařízení v lokální síti.

Pokud adresa má označovat skupinu zařízení (ale ne nutně všechna), nazývá se *skupinovou* (multicast) adresou. V Ethernetu a některých dalších sítích používajících MAC adresy poznáme skupinovou adresu podle bitu v prvním oktetu nejvíc vpravo (nejméně významný bit prvního oktetu). U skupinové adresy je nastaven na 1, což znamená, že první oktet bude liché číslo.



I/G bit: unicast (individuální, 0) nebo multicast či broadcast (group, 1)

Obrázek 2.2: Umístění I/G (individual/group) bitu v MAC adrese

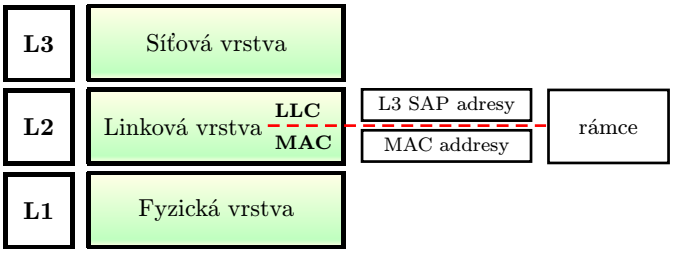
2.3.2 Podvrstvy L2

Na vrstvě L2 (linkové) pracují tzv. *spojové protokoly* (protokoly vrstvy L2). Takový protokol buď sám určuje fungování celé vrstvy L2, nebo se používají dva protokoly, které totéž zvládnou společně. Vrstva L2 se totiž ve skutečnosti dělí na dvě podvrstvy:

- LLC (Logical Link Control) – horní podvrstva, zajišťuje spolupráci s vrstvou L3 (síťovou), přidává do rámce informaci o SAP pro spolupráci s L3 a další řídící informace, zde se také provádí multiplexování.
- MAC (Media Access Control) – spodní podvrstva, zajišťuje spolupráci s vrstvou L1 (fyzickou), přidává do rámce fyzickou adresu příjemce a odesílatele, přizpůsobuje strukturu rámce pro stanovenou přenosovou rychlost a na začátek vloží synchronizační posloupnost bitů (aby bylo zřejmé, kde začíná rámec), zde je implementován mechanismus CSMA/CD.

Z funkce těchto podvrstev vyplývá, že na aktivních síťových prvcích, které implementují nejvýše vrstvu L2 (switche, mosty), vlastně ani není nutné mít celou podvrstvu LLC.

Na obrázku vpravo je naznačen vztah podvrstev linkové vrstvy k okolním vrstvám. Jak bylo výše řečeno, v protokolové sadě může buď obě podvrstvy „naplnit“ jeden protokol, nebo jsou použity dva protokoly – jeden pro podvrstvu LLC a druhý pro podvrstvu MAC.



Ethernet může být na vrstvě L2 implementován následovně (co se týče protokolů na L2):

1. Pro podvrstvu LLC je použit protokol IEEE 802.2 (přímo se nazývá LLC), data z nadřízené vrstvy se zapouzdří do LLC rámce. Ten je následně zapouzdřen do MAC rámce podle protokolu IEEE 802.3, čímž vznikne kompletní rámec vrstvy L2.

2. Podobně jako první bod, jen místo protokolu LLC se použije jeho rozšíření SNAP.
3. Data z nadřízené vrstvy (L3) se zapouzdří podle protokolu Ethernet II, který „zvládá“ obě podvrstvy – LLC i MAC.

První dvě možnosti mohou být využívány, ale momentálně se s nimi u Ethernetu téměř nesetkáváme. Protokol IEEE 802.2 se pro implementaci podvrstvy LLC používá spíše v bezdrátových sítích a také ve starších sítích typu Token Ring nebo FDDI. SNAP je vylepšením původního protokolu LLC, umožňuje napojovat do komunikace širší množinu protokolů. Nejčastěji se setkáváme právě s třetí možností, takové rámce se označují *Ethernet II*.

2.3.3 EtherType

Než se zaměříme na strukturu rámce, řekneme si něco o identifikačních kódech určujících, se kterým protokolem vlastně komunikujeme, tedy co konkrétně má být do rámce zapouzdřeno.

EtherType je identifikátor určující typ dat, která jsou zapouzdřována do rámce, většinou určuje protokol, který na síťové vrstvě vytvořil odesílaný blok dat. Zapisuje se čtyřmi hexadecimálními číslicemi, takže maximální hodnota je $(\mathrm{FFFF})_{16}$ (zapisujeme $0\times\mathrm{FFFF}$), to je v dekadické soustavě 65 535. Ve skutečnosti existuje i spodní limit – pro EtherType se používají čísla o minimální hodnotě 0×0600, to je 1536. To znamená, že EtherType je vždy z rozsahu $0\times0600\ldots0\times\mathrm{FFFF}$.

Poznámka:

V záhlaví rámce vytvářeného na vrstvě L2 jsou dva speciální oktety (odtud maximum $0\times\mathrm{FFFF}$), do kterých se obvykle EtherType ukládá. Jenže totéž pole může být použito i pro jiný typ údaje – délku rámce. Ovšem v technice musí být vždy vše naprosto jednoznačné, proto musí být možné rozlišit, zda v daném poli je EtherType nebo délka rámce.

Délka SDU zapouzdřeného do ethernetového rámce je (téměř) vždy menší než 0×05DC (dekadicky 1500), přičteme rezervu (zaokrouhlíme hexadecimálně nahoru) a získáme spodní mez pro hodnotu EtherType 0×0600.

Pokud je nutné přenést velmi velký rámec (nad stanovený limit), pak se jedná o *jumbo rámec* (jumbo frame) a pro ten je v daném poli speciální hodnota.

Hodnot EtherType je velmi mnoho, podíváme se jen na několik:

- 0×8870: jumbo frames
- 0×8100: VLAN rámce podle 802.1Q
- 0×0800: IPv4
- 0×86DD: IPv6
- 0×0806: ARP
- 0×0835: RARP
- 0×08137, 0×08138: IPX
- 0×8847: MPLS unicast
- 0×8914: FCoE Initialization Protocol
- 0×814C: SNMP

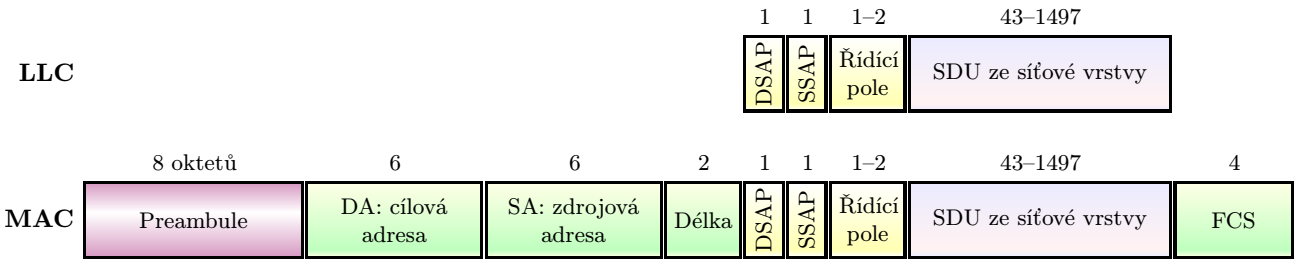
Poznámka:

Z názvu „EtherType“ by se mohlo zdát, že tento identifikátor je používán jen v Ethernetu. Sice jako první byl v Ethernetu použit, ale ve skutečnosti se s ním setkáváme i v jiných typech sítí.

2.3.4 Formát ethernetového rámce

Nyní se podíváme, jak vlastně vypadá rámec posílaný přes síť Ethernet. Postupně si projdeme všechny tři možnosti. Formát rámce (a později formát různých dalších PDU) budeme reprezentovat formou tabulky, kde v záhlaví je naznačena délka jednotlivých polí.

LLC rámec. Struktura LLC rámce (podle protokolu IEEE 802.2) je jednoduchá – k SDU přijaté z nadřízené vrstvy se přidá záhlaví se třemi poli.



Obrázek 2.3: Rámec podle IEEE 802.3/802.2: LLC rámec zapouzdřený v MAC rámci podle 802.3

Na obrázku 2.3 je naznačen formát rámce v případě, že byly zkombinovány protokol IEEE 802.2 na podvrstvě LLC (vznikl LLC rámec, jeho záhlaví je žluté) a protokol IEEE 802.3 na podvrstvě MAC (LLC rámec jsme zapouzdřili do MAC rámce podle 802.3, záhlaví a zápatí MAC rámce je zelené a fialové). Jednotlivá pole v LLC rámci (žlutém) mají tento význam:

- *SSAP, DSAP* (po 1 oktetu) – přístupové body (SAP) na cílovém (DSAP) a zdrojovém (SSAP) zařízení, které zabírají 7 bitů z každého oktetu; zbývající (nejméně významný, vpravo) bit v každém oktetu označuje:
  - u cílového údaje I/G bit (individual/group) určující, zda jde o skupinovou adresu SAP,
  - u zdrojového údaje C/R bit (command/response) pro typ paketu (příkaz nebo odpověď),
- *řídicí pole* (většinou 1 oktet) – určuje typ rámce z hlediska služby, pořadové číslo apod.,
- data z vyšší vrstvy (SDU), která jsou v tomto rámci zapouzdřena.

Vraťme se k polím DSAP a SSAP. Hodnoty pro tato pole jsou standardizovány, jmenujme si opět některé z nich:

- 0×06: DoD IP
- 0×42: IEEE 802.1 Bridge STP
- 0×98: Arpanet ARP
- 0×E0: Novell Netware (IPX)

Některé další hodnoty jsou používány například pro management LLC (tyto rámce jsou služební, s nadřízenou vrstvou nemají nic společného), další jsou vyhrazeny některým již nepoužívaným protokolům.

Příklad

Pokud LLC rámec obsahuje tyto údaje:

0×42	0×42	0×03	data

Pak to znamená, že na zdrojovém i cílovém zařízení vrstva L2 (její podvrstva LLC) komunikuje s protokolem STP (Spanning Tree Protocol). Číslo 0×42 je binárně 1000010, tedy I/G bit a C/R bit mají hodnotu 0 (individuální SAP, příkaz). Uvnitř (jako „data“) je PDU protokolu STP.

MAC rámec podle IEEE 802.3 přidává tato pole (na obrázku 2.3 zeleně a fialově):

- *Preambule* (8 oktetů) je identifikace začátku rámce, synchronizační informace. Prvních 7 oktetů preambule obsahuje střídající se jedničky a nuly, osmý oktet taky, až na poslední bit – ten je místo na nulu nastaven na jedničku. Celá preambule tedy vypadá takto:

  `10101010 10101010 10101010 10101010 10101010 10101010 10101010 10101011`

  V literatuře se setkáme taky s jinou charakteristikou – preambule na 7 oktetech se střídajícími se jedničkami a nulami, a pak jeden oktet, který toto pravidlo ve svém nejméně významném bitu porušuje. Tento oktet se nazývá SOF nebo SD (Start-of-frame Delimiter).
- *Destination Address* (DA, adresa cíle) a *Source Address* (SA, adresa odesílatele) jsou MAC adresy komunikujících uzlů v síti. Zdrojová adresa musí být vždy unicast, cílová může být unicast, multicast nebo broadcast.
- *Délka* vnořeného LLC rámce (takže délka dat z vyšší vrstvy plus LLC záhlaví).
- *FCS* (Frame Check Sequence) je kontrolní součet, který se vypočítává z polí adres (DA, SA), délka, záhlaví LLC a data z vyšší vrstvy (tj. ze všech kromě preambule, která je vždy stejná, a kromě FCS), plus obdoba preambule pro detekci konce rámce.

Kontrolní součet v rámci. Pro FCS se v Ethernetu používá algoritmus CRC-32, který se dá celkem jednoduše implementovat hardwarově, v obvodech. Výpočet je založen na operacích v Galoisově poli $GF(2^n)$, kde se jako operace využívá dělení polynomů nad binárními čísly modulo 2 (tj. zbytek po dělení dvěma).

Algebraické pole modulo dvěma funguje tak, že když jako výsledek operace vyjde číslo 2 nebo vyšší, nahradí se zbytkem po dělení dvěma. Například 1+1 není dvě, ale nula. A navíc místo čísel pracujeme s polynomy.

Například polynom stupně 2 nad binárními čísly má tvar $a_2 * x^2 + a_1 * x^1 + a_0 * x^0$, kde koeficienty a_2, a_1, a_0 jsou binární číslice (0 nebo 1), $x^0 = 1$, tj. $a_2 * x^2 + a_1 * x + a_0$, například $1 * x^2 + 0 * x + 1$, respektive $x^2 + 1$ je polynom nad binárními čísly stupně 2. My potřebujeme polynom stupně 31, tj. s mocninami 31 až 0 (to znamená 32 členů polynomu, potřebujeme 32 binárních číslic pro koeficienty).

Polynomy se dají dělit stejně jako čísla (ovšem zde zásadně tak, abychom pořád zůstali u binárních koeficientů), a pokud jsou nesoudělné, získáme zbytek po dělení (tak jako u běžných čísel při dělení čísla 15 dvojkou dostaneme zbytek 1). Zbytek po dělení je opět binární polynom, a ten můžeme převést na posloupnost bitů tak, že u členů polynomu s určitou mocninou vezmeme koeficienty (0 nebo 1) a zřetězíme je do 32bitového binárního čísla. Takže máme smyčku: binární číslo převedeme na polynom, vydělíme polynomem a zbytek převedeme zpět na binární číslo.

Postup výpočtu kontrolního součtu:

1. Posílané bity od pole pro cílovou adresu po data (tj. všechno mezi poli preambule a FCS) se rozdělí na posloupnost 32bitových slov (čísel), od toho CRC-32.
2. Tato 32bitová slova se dále chápou jako polynomy, kdy jednotlivé bity (0 nebo 1) určují, jestli určitý člen polynomu má nebo nemá být použit (0 – ne, 1 – ano). Například z posloupnosti 32 binárních číslic 1000111010000....0001 uděláme polynom $x^{31} + x^{27} + x^{26} + x^{25} + x^{23} + x + 1$.
3. Polynom z prvních 32 bitů vydělíme speciálním polynomem stupně vyššího než 31, kterému říkáme „ireducibilní“, v Galoisově poli binárních polynomů to je obdoba prvočísel, která jsou větší než

maximální hodnota pro dělence (než to číslo, které dělíme). Na rozdíl od aritmetických operací s běžnými čísly zde dostaneme jako zbytek po dělení binární polynom stupně nejvýše 31.

4. Vezmeme polynom z následujícího 32bitového slova a přičteme polynom, který je zbytkem po dělení předchozího 32bitového slova. Součet opět vydělíme ireducibilním polynomem. Vezmeme polynom z dalšího 32bitového slova, přičteme zbytek předchozího dělení, vydělíme... atd. postupně přes všechna 32bitová slova.
5. Když dojdeme na konec dat a dostaneme poslední zbytek po dělení, tento zbytek (polynom) převedeme na 32bitové binární slovo přesně opačným postupem, než jak jsme vytvářeli polynomy – jednotlivé koeficienty 0,1 u členů polynomu (v pořadí od stupně 31 do stupně 0) vezmeme, zřetězíme a dostaneme 32bitové číslo. To uložíme do pole FCS.

Galoisovo pole GF(2^n) má oproti běžným číslům jednu zvláštnost: když vezmeme dva polynomy a sečteme je, a tytéž dva polynomy od sebe odečteme, v obou případech získáme stejný výsledek. Taže sčítání a odčítání jsou zde stejná operace. Toho se využívá při kontrole kontrolního součtu. Takže když rámec prochází switchem, který opravdu pracuje s kontrolním součtem (tj. přepíná rámce metodou store-and-forward) nebo už přichází do koncového zařízení a je třeba zjistit, jestli rámec nebyl cestou poškozen, postupuje se takto:

1. Začneme úplně stejně jako při výpočtu CRC32, tj. bereme bity od adres včetně dál přes data po 32bitových slovech, postupně vytváříme binární polynomy, dělíme ireducibilním polynomem, zbytek přičteme k polynomu z dalších 32 bitů, vydělíme...
2. Místo abychom se zastavili na konci dat, sečteme výsledek posledního dělení s obsahem pole FCS (už nedělíme). Protože v předchozím algoritmu jsme na konci dat dostali binární číslo, které jsme uložili do FCS, teď bychom vlastně měli sčítat dvě stejné hodnoty. Protože sčítání a odčítání jsou zde shodné operace, je to totéž, jako když od sebe dvě stejná čísla odečítáme, tj. měli bychom dostat nulu.
3. Vyhodnocení: pokud je rámec ok, pak by výsledkem z bodu 2 měla být nula, ale pokud je rámec poškozený, vyjde něco jiného.

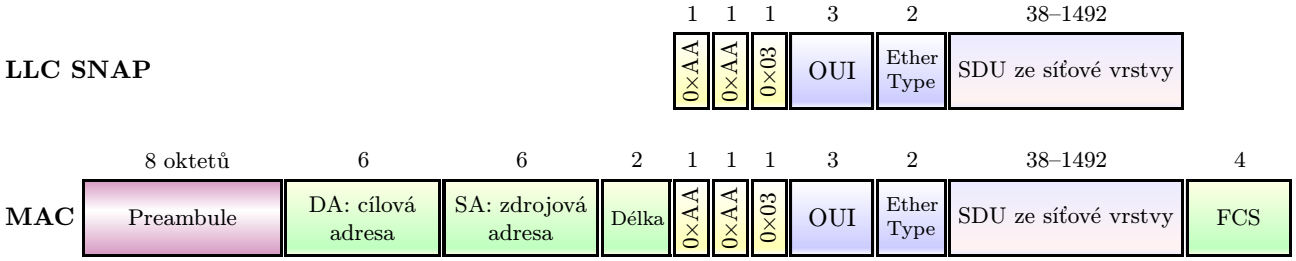
Výhodou je, že při kontrole, zda je rámec v pořádku, nemusíme nejdřív načíst rámec a pak se vracet na začátek, abychom spustili algoritmus, můžeme počítat souběžně s přijímáním rámce z rozhraní. Přijímáme, ukládáme do bufferu a speciální čip průběžně počítá. Po přijetí posledního 32bitového slova (což je FCS) stačí zkontrolovat výsledek. Když čip došel k výsledku 0, je rámec ok a přijmeme ho. Když došel k jinému výsledku, rámec zahodíme.

Další informace:

Další informace se dají najít na internetu, například
https://www.autohotkey.com/boards/viewtopic.php?f=7&t=35671

Na různých variantách Galoisova pole jsou založeny současné šifrovací algoritmy včetně známého AES, takže příslušné postupy najdete i v knihách o kryptografii, například:
Oulehla, Milan a Roman Jašek. *Moderní kryptografie*. Praha: IFP Publishing, 2017. ISBN 978-80-87383-67-4.

SNAP rámec. SNAP (SubNetwork Access Protocol) vznikl rozšířením protokolu IEEE 802.2 LLC, pozměněním a prodloužením záhlaví podvrstvy LLC. SNAP rámec vypadá takto:



Obrázek 2.4: Rámec IEEE 802.3/SNAP: SNAP rámec zapouzdřený v MAC rámci podle 802.3

Na obrázku 2.4 jsou pole přidaná rozšířením SNAP vybarvena modře. Jak vidíme, SNAP dosazuje do záhlaví protokolu LLC určité konstantní hodnoty – jako DSAP a SSAP se použije kód 0×AA (nebo je přípustný kód 0×AB) a do řídícího pole se uloží 0×03. Informaci, kterou by nám jinak tato pole dávala (tedy konkrétní protokoly na nadřízené vrstvě), najdeme právě v přidaných modrých polích.

Rozšíření SNAP přidává k LLC tato pole (na obrázku 2.4 modře):

- *OUI* (Organizationally Unique ID, organizace) je většinou nastaveno na 0. Pokud ne, pak se jedná o identifikátor organizace, v rámci jejíchž zařízení je tento rámec posílán. Například Cisco má OUI = 0×00 00 0C.
- *EtherType* (nebo obecněji Typ) určuje protokol nadřízené vrstvy, pokud je OUI = 0. Pokud není, pak je toto pole čistě v režii organizace, jejíž OUI je v předchozím poli, například pro přenos paketů podle proprietárních síťových protokolů.

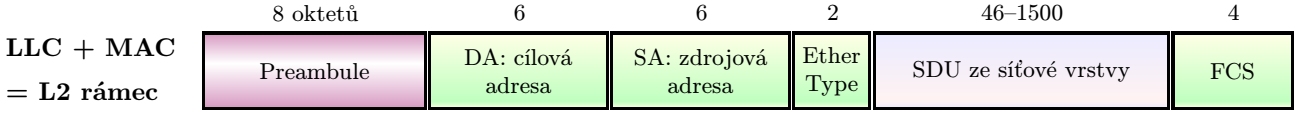
O dvě stránky výše je seznam několika běžných hodnot pro pole SSAP a DSAP v LLC rámci. K těmto hodnotám si můžeme přidat ještě 0×AA jako identifikátor rozšíření SNAP.

Poznámka:

Jak poznáme, jestli se jedná o rámec IEEE 802.3/LLC nebo IEEE 802.3/SNAP?

Několik polí (záhlaví MAC rámce podle 802.3) je stejných, odlišnost začíná na začátku vnořeného LLC/SNAP rámce. Pokud v „žluté části“, resp. třech oktetech za MAC záhlavím, najdeme hodnotu 0×AAAA03, pak je jasné, že jde o SNAP rámec. Jinak jde o LLC rámec.

Rámec Ethernet II. V současné době se v Ethernetu používá téměř výhradně tento typ rámce, a není divu – je nejjednodušší a nejefektivnější.



Obrázek 2.5: Rámec Ethernet II

Na obrázku 2.5 vidíme, že struktura rámce je vlastně velmi podobná tomu, co v předchozích případech na MAC podvrstvě dodával protokol IEEE 802.3. Pole pro preambuli, obě MAC adresy a kontrolní

součet mají stejný význam. Rozdíl je jen v poli, ve kterém byla v předchozích případech délka rámce – v rámci Ethernet II tam najdeme hodnotu EtherType. Proč?

- Evidovat délku rámce vlastně ani nepotřebujeme. Konec rámce jednoduše poznáme tak, že na přenosovém médiu nebude žádný signál (přenáší se v základním pásmu, kdy je signál na médiu generován, nikoliv modulován existující). Navíc je součástí zápatí obdoba synchronizační informace z preambule, třebaže kratší.
- Naopak nutně potřebujeme vědět, co je zapouzdřeno uvnitř rámce, aby bylo jasné, kterému protokolu vyšší vrstvy má být přenášený paket v cíli předán.

Proto místo pole s délkou máme pole s typem protokolu, tedy EtherType, o kterém byla řeč dříve.

Poznámka:

Podle čeho poznáme, že se jedná o rámec Ethernet II a ne žádný z předchozích?

V předchozí sekci je psáno o hodnotách a významu pole EtherType. Kromě jiného se tam v jedné poznámce dočteme, že EtherType a délka rámce jsou ukládány do téhož pole, přičemž číslo větší než 0×0600 znamená EtherType, menší znamená délku rámce. Takže přepínač během načítání prvních $8 + 6 + 6 = 20$ oktetů ještě typy rámců nedokáže rozlišit, ale hned podle 21. a 22. oktetu už dokáže rozlišit, zda se jedná o Ethernet II.

Další informace:

- http://www.infocellar.com/networks/ethernet/frame.htm
- http://ciscopub.blogspot.cz/2015_01_01_archive.html
- http://www.wildpackets.com/resources/compendium/ethernet/frame_formats

Poznámka:

Pokud se někde uvádí skutečná, minimální či maximální délka rámce, preambuli do ní obvykle nepočítáme. Když se vrátíme k nákresům struktury jednotlivých typů rámců a sečteme čísla uvedená nad jednotlivými poli (délky těchto polí) kromě preambule, zjistíme, že ve všech případech je minimální délka rámce $46 + 18 = 64$ oktetů, maximální délka je $1500 + 18 = 1518$ oktetů.

2.3.5 Tabulka MAC adres na switchi

Přepínací tabulce s MAC adresami zařízení mohou různí výrobci říkat různě, také se setkáváme s názvem CAM (Content Addressable Memory) tabulka.

V tabulce najdeme k jednotlivým zařízením především tyto údaje:

- MAC adresu zařízení,
- port, na kterém je zařízení dostupné,
- další informace (VLAN, časové razítko apod.).

Nás zatím budou zajímat první dva údaje. Shrňme si, jak switch zpracovává příchozí rámce.

- Pokud je adresátem rámce zařízení evidované v tabulce MAC adres, switch najde řádek tabulky s MAC adresou příjemce, na tomto řádku zjistí port, ke kterému je adresát připojen, a na ten port rámec pošle.
- Pokud je adresa příjemce neznámá (není v tabulce), je rámec odeslán na všechny porty kromě toho, ze kterého přišel. V tom případě se switch chová jako hub.
- Pokud je cílová adresa broadcast, je rámec poslán na všechny porty kromě toho, ze kterého přišel.

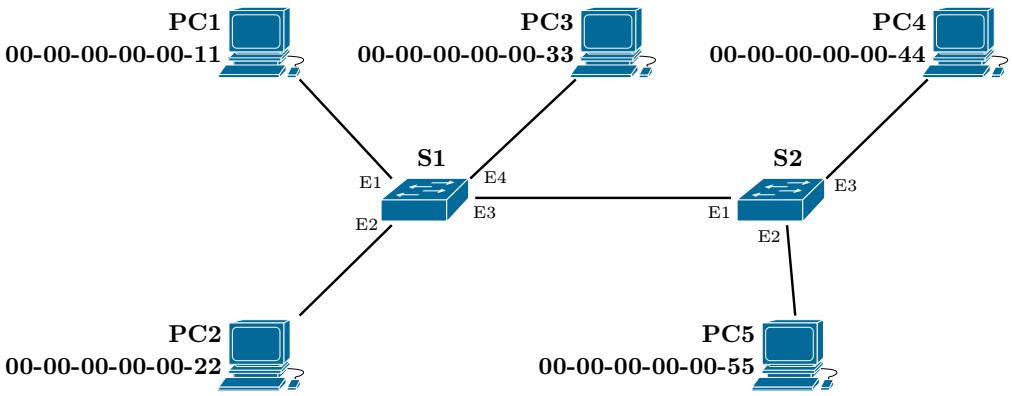
Ale jak se vlastně dotyčný adresát do té tabulky dostane?

Předpokládejme, že máme nový (nebo restartovaný) switch a poprvé ho zapojíme, tabulka je prázdná. V takovém případě jsou dvě možnosti, jak tabulku naplnit. Buď jsou do ní příslušné údaje „natvrdo“ přidány pomocí ruční konfigurace (či pomocí skriptu), nebo to jednoduše necháme na dotyčném switchi, ať se tedy v síti sám rozkouká a „seznámí se se sousedy“.

Ethernetové switche používají tento druhý způsob. Kdykoliv je doručen rámec, switch se zajímá nejen o cílovou adresu (ta je důležitá, aby věděl, kam rámec poslat), ale taky o zdrojovou adresu. U zdrojové adresy je víceméně jasné, na kterém portu je dotyčné (zdrojové) zařízení dostupné – na tom portu, ze kterého právě přišel rámec. Tedy pokud adresu zdroje v tabulce nemá, přidá ji s informací o portu, ze kterého rámec přišel.

Příklad

Podívejme se na následující obrázek. Jsou na něm dva switche, z nichž první má aktivní čtyři porty, druhý tři. Předpokládejme, že switch S1 má svou tabulku zatím prázdnou.



Následuje tento provoz (zkratka DA je Destination Address, tedy cílová adresa, SA je Source Address, tedy zdrojová adresa), vše na switchi S1:

- Na portu E2 je přijat rámec, v němž jsou tyto adresy:
  - DA = 00-00-00-00-00-33
  - SA = 00-00-00-00-00-22

  ⇒ switch usoudí, že zařízení s MAC adresou 00-00-00-00-00-22 je na portu E2, tabulka:

MAC adresa	Port	...
00-00-00-00-00-22	E2	...

  adresáta (adresu 00-00-00-00-00-33) nezná, takže rámec přepošle na porty E1, E3 a E4.

- Na portu E3 je přijat rámec, v němž jsou tyto adresy:
  - DA = 00-00-00-00-00-11
  - SA = 00-00-00-00-00-55
  - ⇒ zařízení s MAC adresou 00-00-00-00-00-55 je na portu E3, tabulka:

MAC adresa	**Port**	...
00-00-00-00-00-22	E2	...
00-00-00-00-00-55	E3	...

    adresáta (adresu 00-00-00-00-00-11) nezná, takže rámec přepošle na porty E1, E2 a E4.
- Na portu E1 je přijat rámec, v němž jsou tyto adresy:
  - DA = 00-00-00-00-00-22
  - SA = 00-00-00-00-00-11
  - ⇒ zařízení s MAC adresou 00-00-00-00-00-11 je na portu E1, tabulka:

MAC adresa	**Port**	...
00-00-00-00-00-22	E2	...
00-00-00-00-00-55	E3	...
00-00-00-00-00-11	E1	...

    adresáta (adresu 00-00-00-00-00-22) už v tabulce máme (hned na prvním řádku), takže rámec přepošleme pouze na takto zjištěný port E2.
- Na portu E1 je přijat rámec, v němž jsou tyto adresy:
  - DA = 00-00-00-00-00-44
  - SA = 00-00-00-00-00-11
  - ⇒ zařízení s MAC adresou 00-00-00-00-00-11 už v tabulce máme, takže teď do tabulky nebudeme nic přidávat,
    adresáta (adresu 00-00-00-00-00-44) nezná, takže rámec přepošle na porty E2, E3 a E4.

Tak bychom pokračovali i dál, MAC tabulka by byla kompletní až tehdy, když by se postupně „prozradila“ všechna zařízení v síti.

2.4 Ethernet na fyzické vrstvě

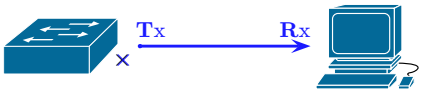
Na fyzické vrstvě v Ethernetu jsou zajištěny tyto operace:

- spolupráce s vrstvou L2, přebírání SDU z této vrstvy (tedy rámce), v opačném směru předávání dat vrstvě L2,
- kódování, multiplexing,
- vysílání a přijímání signálu, synchronizace se zařízením „na druhé straně“,
- auto-negociace (vyjednávání) – síťové rozhraní si potřebuje dojednat se síťovým rozhraním druhého zařízení veškeré přenosové parametry (rychlost, poloviční nebo plný duplex apod.), aby si navzájem rozuměly.

2.4.1 Křížení a krimpování

Zařízení na svém síťovém rozhraní vlastně provádí dvě základní operace:

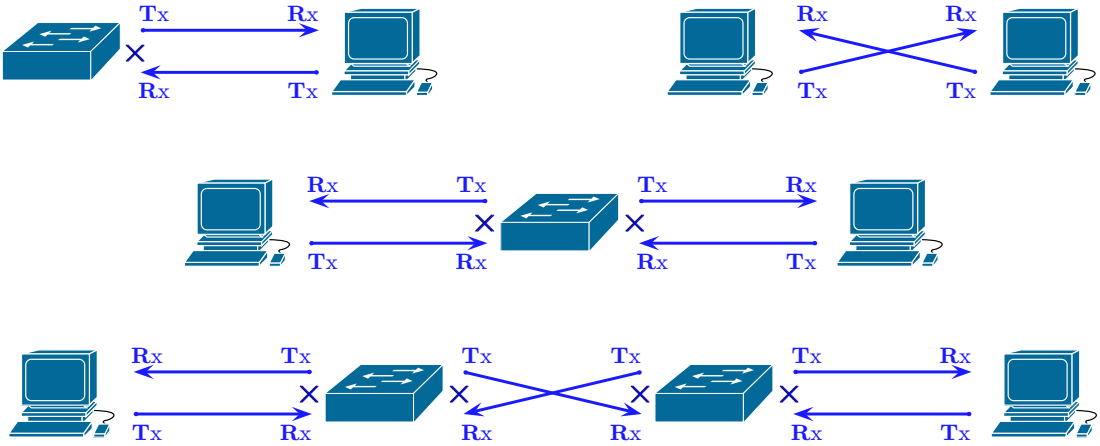
- Tx (Transmit) – odesílá,
- Rx (Receive) – přijímá (tj. naslouchá a když zjistí, že druhá strana vysílá, toto vysílání přijme).



V reálu to znamená, že například u kroucené dvojlinky (twisted pair) jsou páry vodičů vyhrazené pro operaci Tx a páry vodičů vyhrazené pro operaci Rx, případně je toto přiřazení určováno dynamicky (ale vždy platí, že v jednom okamžiku je toto přiřazení jednoznačné) – daný pár vodičů nemůže v jednom okamžiku provádět totéž. Jinými slovy – síťové rozhraní se skládá ze dvou částí – Tx a Rx, a je důležité, na kterou z těchto částí je který pár v kabelu napojen. U dvojice přímo komunikujících zařízení (tj. propojených tímtéž kabelem) musí platit, že na konkrétním páru jedno zařízení přijímá a druhé vysílá – kdyby na tomtéž páru obě zařízení odesílala, došlo by ke kolizi, a kdyby na tomtéž páru obě zařízení naslouchala, k žádnému přenosu by naopak vůbec nedošlo. Vysílající zařízení odešle signál do části Tx svého síťového rozhraní, a cílové zařízení tento signál přijme na části Rx svého síťového rozhraní, a tyto dvě části musí být fyzicky propojeny tímtéž párem vodičů.

Poznámka:

Z toho vyplývá, že někde musí být přenosová cesta *překřížena*. Pokud jsou dvě zařízení propojena kroucenou dvojlinkou, pak (minimálně jeden) pár vodičů vedoucí z Tx jednoho zařízení musí být napojen za Rx druhého zařízení a naopak.



Obrázek 2.6: Princip křížení na kroucené dvojlince

Na obrázku 2.6 vidíme několik řešení – v „první řadě“ křížení zajistí jedno ze zařízení (většinou aktivní síťové prvky) nebo je přímo v kabelu – křížení zajištěné zařízením je naznačeno křížkem. Níže jsou případy, kdy máme jedno nebo dvě mezilehlá zařízení (která zajišťují křížení).

Rozlišujeme dva typy kabelů typu kroucená dvojlinka:

- *přímý kabel* – žádné křížení neobsahuje, používá se obvykle k propojení DTE a DCE (tedy například počítače ke switchi),

- *křížený kabel* – obsahuje křížení, používá se k propojení zařízení stejného typu (tedy buď DTE a DTE, nebo DCE a DCE); pokud chceme propojit dva počítače přímo bez mezilehlého DCE, měli bychom použít křížený kabel.

Ve skutečnosti se v obou případech používá tentýž kabel; to, zda je přímý nebo křížený, se určuje nasazením koncovek (konektorů). Přímý kabel má na obou stranách koncovky nasazeny stejně (vodiče jsou na obou koncích ve stejném pořadí), kdežto křížený kabel má na jednom konci vodiče jinak seřazené než na druhém konci.

Poznámka:

V současné době máme situaci jednodušší – novější síťová rozhraní většinou dokážou automaticky rozpoznat, jestli mají nebo namají zajistit vnitřní křížení, takže prostě použijeme přímý kabel.

Krimpování (konektorování) je postup nasazování konektoru na kabel. V případě kroucené dvojlinky se většinou používají konektory a zásuvky typu *8p8c* (to znamená 8 pozic a 8 kontaktů), kterým se poněkud nepřesně říká RJ-45.

V konektoru (i zásuvce) jsou vodiče umístěny v řadě za sebou a jejich pořadí je určeno standardem TIA/EIA-568-B z roku 2001, nověji TIA/EIA-568-C z roku 2014. Standard ve skutečnosti určuje dvě pořadí – T568A a T568B (pozor, neplést si písmenka s verzí standardu), přičemž u přímého kabelu se na obou koncích použije pořadí podle T568B, kdežto u kříženého na každém konci jedno z těchto pořadí. Pozor, nejde jenom o obrácení pořadí!
V případě optických vláken je křížení zajištěno vždy v kabelu (zde jde o pořadí optických vláken).

2.4.2 Parametry

Naprostá většina druhů Ethernetu používá přenos v základním pásmu (baseband). Dalšími důležitými parametry jsou rychlost a typ přenosového média (kabelu), obojí má vliv na použité kódování a průběh multiplexování. Na fyzické vrstvě se v označení objevuje:

- rychlost: 10, 100, 1000 apod. – v Mb/s, u vyšších se píše s písmenem G, např. 10G,
- přenos v základním (base) nebo přeloženém (broad) pásmu,
- typ kabelu – většinou „T“ znamená kroucenou dvojlinku (twisted), „F“ optický.

Takže pojďme od historie, ale přeskočíme nejstarší standardy. Nebudeme se zabývat všemi, u každé rychlosti vybereme jen některé typické zástupce.

10Mb Ethernet. Všichni zástupci dovolují rychlost 10 Mb/s, konkrétně:

- 10Base-T: baseband, kabel nestíněná kroucená dvojlinka (UTP),
- 10Base-F: baseband, kabel optický,
- 10Broad-36: broadband (přeložené pásmo), kabel koaxiál o impendanci 75 Ω; nerozšířil se, ale stal se základem pro pozdější Ethernet provozovaný v sítích kabelových televizí (dnes se místo toho v kabelových televizích používá pro přenos dat standard DOCSIS).

Fast Ethernet. Rychlost 100 Mb/s a baseband, přibyla možnost auto-negociace (síťová rozhraní se dokážou automaticky dohodnout na parametrech přenosu). Zástupci:

- 100Base-TX: kabel UTP alespoň Cat5, využívá pouze dva páry (jeden pro vysílání a druhý pro příjem); vpodstatě pokračovatel 10Base-T, v síti bylo možné kombinovat síťové karty těchto typů,

- 100Base-FX: používá dvě optická vlákna, jedno pro každý směr. Není kompatibilní s 10Base-F, protože používá signál o jiné vlnové délce.

Gigabit Ethernet umožňuje rychlost 1 Gb/s, baseband. Zástupci:

- 1000Base-T: pro kabel UTP minimálně Cat.5 (doporučeno minimálně 5e), používají se všechny čtyři páry (takže žádný light se dvěma páry nelze použít), směr pro jednotlivé páry se určuje adaptivně – žádný není vyhrazen pro konkrétní směr,
- 1000Base-TX: jedná se o nepříliš úspěšný pokus o úpravu standardu 1000Base-T tak, že pro každý směr jsou vyhrazeny dva páry a je vyžadován kabel Cat.6; zatímco 1000Base-T je standard od organizace IEEE (IEEE 802.3ab), 1000Base-TX pochází od TIA (TIA/EIA-854); komerčně neúspěšný – nelze použít na starších rozvodech Cat.5 nebo 5e,
- 1000Base-X: souhrn standardů pro (většinou) optická vlákna
  - 1000Base-SX („S“ jako short) optická vlákna mnohavidová, typický dosah je ve stovkách metrů,
  - 1000Base-LX („L“ jako long) optická vlákna buď jednovidová (dosah v jednotkách kilometrů) nebo mnohavidová (stovky metrů),
  - 1000Base-CX je výjimka z pravidla – nepoužívá optiku, ale stíněnou kroucenou dvojlinku, konektory jsou podobné jako u UTP, ale jiné přiřazení pinů (jinak se krimpuje); typický dosah je do 25 metrů, používá se v datových centrech k připojení serverů ke switchům (například u některých produktů společnosti IBM).

Poznámka:

Velmi často se setkáváme se záměnou názvu – místo 1000Base-T je udáno 1000Base-TX. Obvykle to je proto, že u „dřívější generace“ (tedy Fast Ethernetu) byla přípona TX. U síťových rozhraní podporujících různé rychlosti bývá označení 100/1000Base-TX, třebaže u druhé uvedené rychlosti jde ve skutečnosti o „Téčkový“ standard.

10Gigabit Ethernet (10GE, 10GigE) umožňuje rychlost 10 Gb/s, přenos baseband. Přístupová metoda CSMA/CD je zcela odbourána, komunikuje se pouze ve full-duplexu.

- 10GBase-R je skupina standardů pro optická vlákna:
  - 10GBase-SR (short range) pro mohavidové vlákno, dosah do 62 m,
  - 10GBase-LR (long range) pro jednovidové vlákno, dosah cca 10 km,
  - 10GBase-LRM (long reach multimode) pro mnohavidové vlákno, dosah do 220 m,
  - 10GBase-ER (extended range) pro jednovidové vlákno, dosah 40 km,

  liší se nejen kabely, ale také vlnovými délkami, na kterých je vysílán signál,
- 10GBase-CX4 používá kabel twin-ax (podobně jako koaxiál, jen místo jednoho středového vodiče jsou dva), dosah je jen 15 metrů, ale na druhou stranu má velice dobré hodnoty odezvy (latence) a cena je vcelku příznivá; používá se v datových centrech pro připojení serverů k DCE,
- 10GBase-T je pokračovatelem 1000Base-T, používá kroucenou dvojlinku minimálně Cat.6; doporučuje se však použít vyšší kategorii, protože na Cat.6 má dosah jen max. 55 m,
- 10GBase-W už míří mimo LAN sítě – na vrstvu L1 přidává novou podvrstvu WIS, která umožňuje komunikovat s WAN sítí (SONET/SDH, viz dále); funguje podobně jako 10GBase-R, jen navíc ethernetový rámec zapouzdří do WIS rámce, ve kterém data procházejí napojenou WAN sítí.

Poznámka:

Zvídavý čtenář se určitě ptá: a kde jsou tedy ty standardy? Tak například 100Base-TX a 100base-FX (tedy kroucená dvojlinka a optika pro Fast Ethernet) jsou standardizovány v IEEE 802.3u, kdežto 1000Base-T (kroucená dvojlinka pro Gigabit Ethernet) je v standardu IEEE 802.3ab a Gigabit Ethernet na optice najdeme v IEEE 802.3z.

Standard IEEE 801.3 a jeho dodatky (písmenka na konci) definují vrstvu L1 a spodní část vrstvy L2 (MAC podvrstvu), přičemž se v praxi využívá spíše to, co platí pro vrstvu L1 (už dřív jsme si vysvětlili, že na L2 se setkáváme spíše s rámci podle Ethernet II).

40Gigabit Ethernet a 100Gigabit Ethernet (40GigE, 100GigE) předepisuje optické kabely s laserem jako zdrojem světla (na vzdálenost cca 100 m pro mnohavidová vlákna, resp. kilometry a desítky kilometrů pro jednovidová s různými parametry), na vzdálenost několika metrů kabel twinax (v datových centrech pro připojení výkonných serverů do sítě), a na vzdálenost do 30 m lze pro rychlost 40GigE použít kroucenou dvojlinku Cat.8.

2.5 a 5Gigabit Ethernet byly jako standardy zveřejněny až v roce 2016, tj. po „rychlejších“ sourozencích. Jedná se o komunikaci po kroucené dvojlince, formálně 2.5GBase-T a 5GBase-T, standard IEEE 802.3bz. Fyzická vrstva byla převzata z 10GBase-T a přizpůsobena „horší“ kabeláži. 2.5G-Base-T může využívat nestíněnou kroucenou dvojlinku kategorie Cat.5e, 5GBase-T používá kategorii Cat.5e (menší vzdálenost než 100 m) nebo Cat.6.

Zatímco 10GBase-T nepodporuje PoE, 2.5GBase-T a 5GBase-T by měly napájení přes Ethernet zvládat.

Poznámka:

Jak je možné řádově zvyšovat rychlost, když se pořád používají vpodstatě tytéž kabely (i když generaci od generace lepší)? Zvýšení se dosahovalo kombinací více způsobů, například:

- vytížením všech čtyř párů kroucené dvojlinky místo pouhých dvou,
- použitím kvalitnější kabeláže s lepším stíněním, kroucením s vyšší hustotou a větší šířkou přenosového pásma,
- použitím lepšího kódování dat na signál (v reálu se přenáší menší objem dat).

2.4.3 Implementace fyzické vrstvy

Fyzická vrstva je implementována typicky v obvodech, a to buď ve formě samostatného čipu (na základní desce nebo na síťové kartě NIC – Network Interface Card) nebo integrovaná v čipu s jinými komponentami. Může se skládat ze dvou částí.

MAU (Medium Attachment Unit) neboli *transceiver* (zkráceno ze slov transmitter/receiver) je prvek na NIC, který zajišťuje rozpoznání přítomnosti signálu, kolizi (signál jam) a vysílání/příjem signálu. Může být

- externí – u starého Ethernetu na koaxiálu a naopak u nových optických modulů (XENPAK, XFP, SFP, SFP+ apod.),
- integrovaný v čipu s ostatními částmi síťového rozhraní.

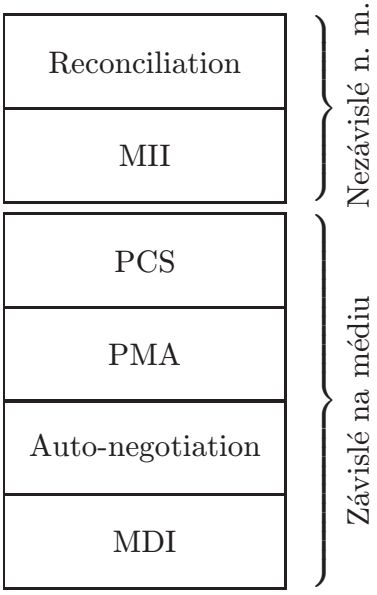
Na síťových prvcích (switche, routery) se můžeme setkat s pozicemi pro SFP+ moduly, které mohou být optické nebo metalické, s různou specifikací.

Obrázek 2.7: SFP+ transceivery pro 10GBase-LR/LW[1]

✎ Fyzická vrstva se také dělí na podvrstvy, a to závislé/nezávislé na médiu s vlastními komunikačními rozhraními (interface).

Podvrstvy nezávislé na médiu:

- vyrovnávací podvrstva (reconciliation),
- MII (10Mb a 100Mb), GMII (Gb) nebo XGMII (10Gb) – zvlášť odesílací a přijímací datové cesty o šířce 1 bit pro 10Mb (bit-serial), 4 bity pro 100Mb (nibble-serial) nebo 1 B (byte-serial).



Zajišťují logické spojení mezi MAC a různými sadami podvrstev závislých na médiu.

Podvrstvy závislé na médiu:

- PCS (Physical Coding Sublayer) – kódování, multiplexing, synchronizace odchozích a opačně u příchozích dat,
- PMA (Physical Medium Attachment) – vysílání a přijímání signálu, mechanismus zotavení časovače (zajištění sesynchronizování časovače na začátku proudu dat),
- Auto-negotiation (vyjednávací podvrstva) – tyto podvrstvy na začátku přenosu dohodnou konkrétní vlastnosti spojení vyhovující oběma NIC,
- MDI (Medium-Dependent Interface) – ke konektoru kabelu.

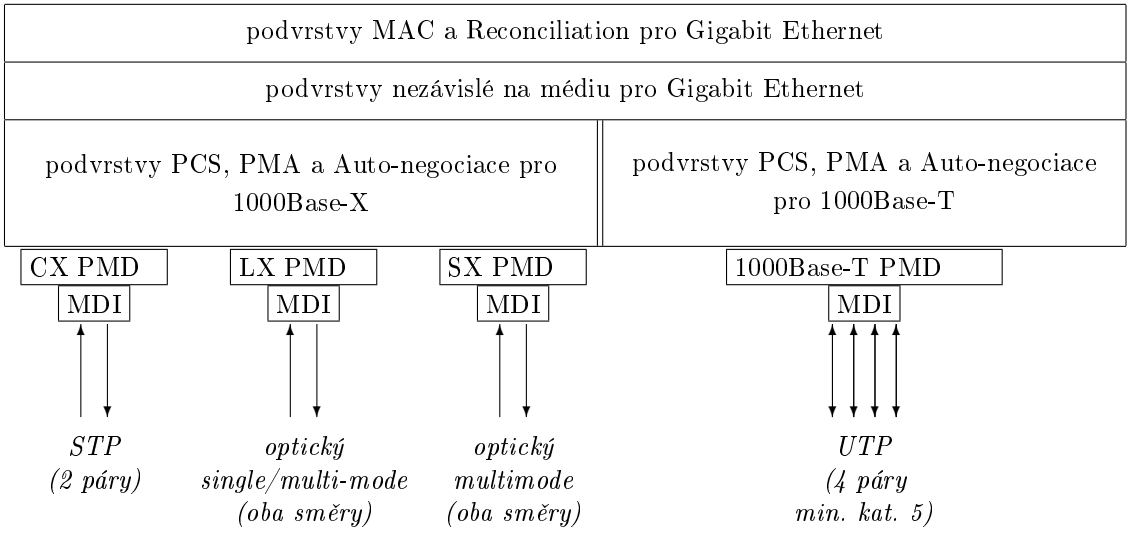
Příklad schématu vrstev (zde pro 1Gb Ethernet) vidíme na obrázku 2.8 na straně 46. Fyzická vrstva je zcela předefinována ve standardech od 10G Ethernetu.

2.5 Průběh přenosu

V předchozí kapitole jsme se dozvěděli, že obousměrný přenos mezi dvěma zařízeními může probíhat jedním z těchto způsobů:

- v poloviční duplexu (half duplex), kdy vysílat mohou obě zařízení, ale střídavě (v jeden okamžik může vysílat jen jedno zařízení), tedy sdílejí tentýž komunikační kanál,
- v plném duplexu (full duplex), kdy mohou vysílat třeba i obě zařízení najednou, protože pro každý směr je vyhrazen (minimálně) jeden komunikační kanál.

[1] Zdroje: https://www.alternetivo.cz, https://www.atcmarket.cz



Obrázek 2.8: Souhrnné schéma pro Gigabit Ethernet

2.5.1 Přenos v poloviční duplexu

Režim poloviční duplexu je možné používat maximálně do rychlosti 1 Gb/s. Protože zařízení na segmentu sdílejí komunikační kanál, používá se kolizní metoda CSMA/CD, která byla popsána výše, včetně Backoff algoritmu pro řešení kolizí.

Definice (Kolizní okno)

Kolizní okno (Collision Window, Slot Time) je doba, po kterou musí zařízení naslouchat na nosné, aby zachytilo vysílání jiného zařízení a dokázalo detekovat nebezpečí kolize.

Co se kolizního okna týče, platí tyto vztahy:

- největší možná vzdálenost mezi zařízeními v téže kolizní doméně (čím větší vzdálenost, tím déle musí zařízení naslouchat, protože signálu to „déle trvá“)– čím větší je maximální povolená vzdálenost, tím větší musí být kolizní okno,
- minimální délka rámce (zařízení se musí dozvědět ještě během vysílání rámce, že došlo ke kolizi, aby mohlo vysílání zrušit, počkat dle Backoff algoritmu a vysílat znovu) – čím menší je minimální délka rámce, tím menší musí být kolizní okno a menší kolizní doména,
- pokud je malé kolizní okno, musí být malá i kolizní doména.

Z toho vyplývá, že existuje velmi úzký vztah mezi kolizním oknem, velikostí kolizní domény (max. vzdáleností mezi zařízeními v síti) a minimální délkou rámce.

Poznámka:

Představme si tuto situaci: v kolizní doméně jsou dvě zařízení – Z_a a Z_b. Signálu trvá dobu t, než se dostane od jednoho zařízení k druhému. Zařízení Z_a začne vysílat v době t_a rámec, jehož délka je taková, že vysílání rámce trvá po dobu d_a.

Pokud zařízení Z_b začne vysílat ještě před tím, než k němu dojde signál od zařízení Z_a, dojde ke kolizi. Teď je nutné, aby se zařízení Z_a včas dozvědělo, že ke kolizi došlo, aby mohlo vysílání zrušit

a po čekání vysílat znovu. Musí se to tedy dozvědět ještě před tím, než skončí vysílání, tedy nejpozději v okamžiku $t_a + d_a$. Zařízení Z_b zjistí kolizi nejdřív v čase $t_a + t$, a pokud okamžitě zareaguje a pošle jam signál, tento signál se dostane k zařízení Z_a nejdřív v době $t_a + 2 \cdot t$. Takže tu máme nerovnici:

$$\begin{aligned} t_a + d_a &> t_a + 2 \cdot t \\ d_a &> 2 \cdot t \end{aligned}$$

To znamená, že minimální doba, po kterou má trvat vysílání jednoho rámce, musí být větší než doba, kterou potřebuje signál pro cestu tam a zpět mezi dvěma nejvzdálenějšími zařízeními na segmentu (v kolizní doméně). Proto pokud povolíme příliš velkou kolizní doménu (= chceme příliš dlouhé kabely), musíme přikázat vyšší minimální délku rámce.

Takže tady balancujeme mezi dvěma požadavky:

- Chceme sesíťovat co největší prostor, tedy jsou žádoucí dlouhé kabely a velká kolizní doména.
- Nechceme zbytečně zahlcovat linku v případě, že potřebujeme posílat i malé rámce (pokud je třeba poslat množství dat menší než je limit pro minimum, musí se přidat „vycpávka“ – data bez významu), takže je efektivnější mít spíše nízký limit pro minimální délku.

K tomu se přidává ještě třetí faktor – rychlost přenosu. Pokud pouze zvýšíme rychlost přenosu, musíme zvýšit limit pro minimální délku rámce nebo zmenšit doménu, protože zvýšení rychlosti ovlivní dobu přenosu.

Zaměřme se na různé rychlosti – 10, 100, 1000 Mb/s (u rychlejších se už nepoužívá CSMA/CD), ke každé si vysvětlíme, jak se tento problém řeší.

Pro 10 Mb/s je stanovena minimální délka rámce bez preambule na 512 bitů = 64 oktetů (podívejte se na strukturu rámce Ethernet II na straně 37), tedy odvysílání 512 bitů trvá 51,2 μs (to je doba d_a). Doba, po kterou může maximálně trvat cesta mezi dvěma nejvzdálenějšími zařízeními v kolizní doméně, je polovina této hodnoty. Vzdálenost v metrech pak záleží na tom, jaké přenosové médium je zvoleno (metalickými a optickými kabely se signál šíří odlišnou rychlostí).

Pokud je minimální délka MAC rámce bez preambule určena na 64 oktetů, pak by to, co je uvnitř zapouzdřeno, mělo být dlouhé minimálně $64 - 18 = 46$ oktetů (odečteme délku polí v záhlaví a západí MAC rámce).

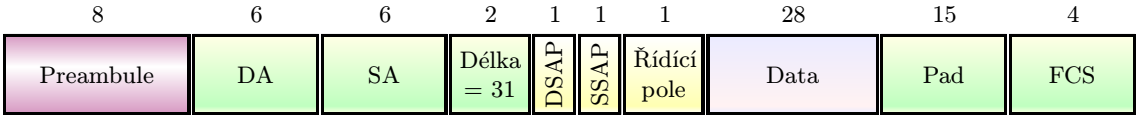
Teoreticky může nastat situace, že by uvnitř rámce mělo být přeneseno méně než 46 oktetů dat. Standard IEEE 802.3 ve spolupráci s LLC a SNAP to řeší „vycpávkou“ (pad) přidanou za data, přičemž délka vycpávky se dá odvodit z hodnoty v poli *Délka*. Řešení si ukážeme na příkladu.

Příklad

Pokud je v poli *Délka* LLC/SNAP rámce číslo < 46 (zde konkrétně 31), pak platí:

$$\begin{array}{rcrcrcl} \text{DSAP} & + & \text{SSAP} & + & \text{Řídicí pole} & + & \text{Data} & + & \text{Pad} & = & 46 \\ & & & & & & \textit{Délka} & + & \text{Pad} & = & 46 \\ & & & & & & & & \text{Pad} & = & 46 - \textit{Délka} \end{array}$$

Pro případ naznačený na obrázku 2.9 na straně 48 platí $\text{Pad} = 46 - (28 + 1 + 1 + 1) = 15$. Tento údaj je důležitý, aby bylo zřejmé, na kterém oktetu rámce začíná pole s kontrolním součtem. Pozor, pole *Délka* v sobě započítává i délku LLC záhlaví (na obrázku žlutě).



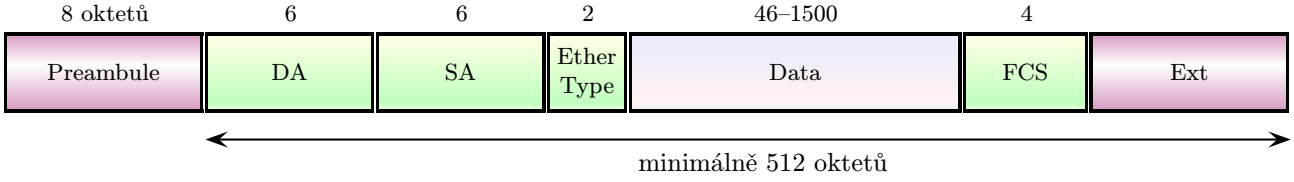
Obrázek 2.9: Velmi krátký LLC rámec v MAC rámci podle IEEE 802.3

Rámce podle Ethernet II toto neřeší, předpokládá se, že zapouzdřená data zachovávají určitou minimální délku. Protože je „uvnitř“ většinou IPv4 paket, jehož záhlaví je minimálně 20 oktetů dlouhé, na to, co je zapouzřeno uvnitř IP paketu, zbývá minimálně 26 oktetů, což není tak moc.

Při přechodu na vyšší rychlost, tedy na 100 Mb/s, bylo rozhodnuto zachovat minimální délku rámce (tj. pro formát rámce platí totéž, co bylo napsáno v předchozím odstavci) i velikost kolizního okna, tedy bylo třeba zmenšit velikost kolizní domény mezi DCE (přibližně 10×).

Na metalickém kabelu to znamenalo zmenšení kolizní domény z 2000 m na 200 m, což znamená, že mezi DTE a DCE je vzdálenost max. 100 m (to zůstalo) a mezi dvěma DCE max. 200 m.

Při přechodu na Gigabit Ethernet (1 Gb/s) toto řešení nebylo možné (20 m jako nejvyšší možná vzdálenost by bylo opravdu málo), tedy se pro změnu zvýšila minimální délka rámce bez započítání preambule na 512 oktetů (4096 bitů). Protože se často přenáší menší množství dat, je u rámců s délkou pod limitem nutné přidat další „vycpávku“ (Carrier Extension, nedatovou příponu), a to za kontrolní součet. Umístění je naznačeno na obrázku 2.10 (pole zcela vpravo).



Obrázek 2.10: Nedatová přípona u rámce pro Gigabit Ethernet

Nedatová přípona se při odesílání přidává až po vypočtení kontrolního součtu na MAC podvrstvě, při přijetí rámce je na MAC podvrstvě cílového systému odstraněna ještě před kontrolou podle kontrolního součtu. Skladba symbolů v nedatové příponě je patentovaná – musí být odlišitelná od pole s kontrolním součtem.

Údaje o patentu jsou dostupné na http://www.google.com/patents/US5940401.

Pokud by se posílalo velmi často větší množství rámců s malým množstvím zapouzdřených dat, linka by samozřejmě byla zbytečně vytěžována víc než je nutné. Takový případ se řeší jinak:

Burst mode (shlukový režim) je právě určen pro posílání sekvencí krátkých úseků dat. Je používán pouze u přenosů v rychlostech od 1 Gb/s výše, tj. od Gigabit Ethernetu. „Krátké“ rámce se posílají ve shluku, který má následující strukturu:

- první rámec ve shluku má výše popsanou strukturu, včetně případné nedatové přípony, pokud je menší než 512 oktetů,
- následuje *IFG* (InterFrame Gap) – speciální sekvence bitů vyplňující prostor mezi rámci, slouží k rozpoznatelnému oddělení rámců ve shluku,
- další rámce v posloupnosti navzájem oddělené IFG sekvencemi; pokud jsou kratší než 512 oktetů, není nutné je doplňovat o nedatovou příponu.

Celková délka shluku by neměla překročit přibližně 5,4násobek maximální možné délky rámce (přesněji 8 192 oktetů), ale pokud je tato hranice překročena až během odesílání posledního rámce ve shluku, může být jeho odesílání ještě dokončeno a shluk je ukončen až poté.

Poznámka:

To znamená, že během přenosu v burst modu je třeba sledovat, zda se neblížíme k hranici 5,4násobku maximální délky rámce. Pokud jí dosáhneme, pak rámec, který právě odesíláme, je posledním rámcem shluku. Je to důležité, protože během odesílání shluku rámců blokujeme linku a znemožňujeme odesílat ostatním zařízením v segmentu (téže kolizní doméně), a blokování nemůže trvat příliš dlouho.

2.5.2 Přenos v plném duplexu

V plném duplexu je situace mnohem jednodušší. Na logické úrovni se vpodstatě setkáváme jen se spoji typu point-to-point a veškeré spoje jsou vlastně vyhrazené, zařízení může vysílat kdykoliv. Nepoužívá se přístupová metoda CSMA/CD a parametry spojení nejsou omezeny vlastnostmi kolizní domény a kolizního okna, pouze fyzikálními vlastnostmi přenosové cesty (například útlumem).

Vysílání může prakticky pořád probíhat v burst modu tak, jak bylo popsáno u poloviční duplexu. Vysílání a příjem probíhají po jiné cestě, ale může jít o jeden společný kabel (například v mnoha standardech pro fyzickou vrstvu je pro UTP se čtyřmi páry kroucené dvojlinky vždy jedna dvojice vyhrazena pro příjem a druhá dvojice pro vysílání, nebo se směr určuje adaptivně). U optických kabelů je buď pro každý směr jeden kabel, nebo pravděpodobněji v kabelu máme jedno či více vláken zvlášť pro různé směry.

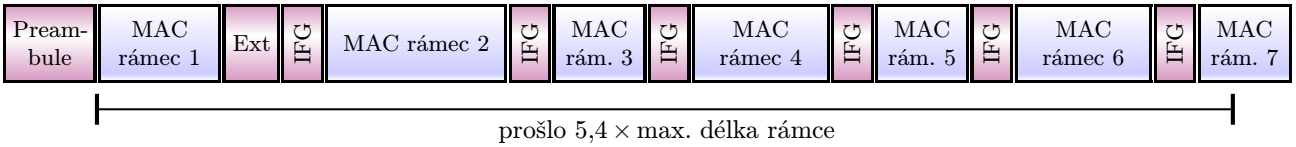
Jediný problém (který může nastat i u poloviční duplexu, ale pro plný duplex je typický) nastává tehdy, když vysílající zařízení neustále vyvíjí aktivitu, ale přijímající zařízení nestíhá rámce přijmout. Přijetí rámce totiž není až tak triviální, kromě zpracování na fyzické vrstvě je třeba také postupně rámec rozbalit a patřičně ho zpracovat, a procesor (který se na tom taky podílí) může mít i jinou práci než zpracovávat rámce od dotyčného zařízení. Tento problém se řeší kombinací těchto způsobů:

- zařízení má dostatečně velký *buffer* (vyrovnávací paměť), tam si ukládá to, co zatím nemohlo být zpracováno,
- odesílatele je třeba zpomalit, sdělit mu, že má určitou dobu počkat s vysíláním.

První možnost je celkem jasná, ale nemusí být dostačující. Druhá možnost znamená použití tzv. *pause frame*, tedy speciálního rámce, který je poslán „pilnému“ odesílateli; tento rámec obsahuje především informaci o tom, na jak dlouho je třeba přerušit vysílání.

Poznámka:

Plný duplex se dá použít pouze na spojích typu point-to-point (v kolizní doméně jsou tedy pouze právě



Obrázek 2.11: Přenos v burst (shlukovém) módu

dvě fyzicky propojená zařízení), což například naprosto vylučuje použití hubu (rozbočovače). Navíc obě propojená zařízení „musí umět“ plný duplex, aby bylo možné ho využít.

Od 10G Ethernetu (ale také 2.5G a 5G) se používá pouze plný duplex, poloviční není podporován. Ke kolizím z principu nemůže dojít, tedy nemá smysl využívat jakoukoliv přístupovou metodu (CSMA/CD se v tom případě nepoužívá).

2.6 Technologie související s Ethernetem

2.6.1 Autonegociace

Aby vůbec bylo možné zprovoznit přenos přes ethernetovou síť, je třeba, aby si síťová rozhraní komunikujících zařízení „rozuměla“, tedy aby se shodla na následujících parametrech:

- rychlost přenosu,
- pro danou rychlost konkrétní standard, pokud je víc možností pro použité přenosové médium (například aby bylo jasné, jak konkrétně budou jednotlivé páry UTP využívány),
- poloviční nebo plný duplex.

Zároveň je potřeba pravidelně provádět kontrolu funkčnosti spojení.

Pokud propojíme dvě zařízení kabelem a zprovozníme příslušná síťová rozhraní, pak se tato síťová rozhraní nejdřív dohodnou na výše uvedených parametrech – provede se *autonegociace* (tedy automatická dohoda o parametrech) a následně se pravidelně ozývají v době, kdy se nepřenášejí žádné rámce (aby bylo jasné, že dané zařízení je pořád připojeno a linka funguje). Autonegociace se provádí na fyzické vrstvě, je popsána přímo v standardu IEEE 802.3 a dodatcích.

První pokusy o autonegociaci (ale ještě ne zcela v plném smyslu toho slova) byly u Ethernetu 10Base-T, kdy síťové rozhraní vysílalo v pravidelných intervalech impulsy, které sloužily ke zjišťování, zda je linka funkční. Tento signál se označoval *NLP* (Normal Link Pulse). Pokud od zařízení po určitou dobu nepřišel žádný rámec ani NLP signál, linka byla považována za nefunkční.

Dnes, v době vyšších rychlostí, se odesílá signál *FLP* (Fast Link Pulse), kde se série původních NLP signálů prokládá dalšími informacemi (protože se postupně přidávaly další parametry, které bylo možné oznámit), a zatímco původně šlo o nepovinnou funkčnost, od Gigabit Ethernetu je pro metalické kabely autonegociace povinná.

Celý mechanismus je zpětně kompatibilní, a v reálu se pro linku použijí nejvyšší možné parametry z těch, které splňují obě komunikující strany. Například pokud jedno zařízení zvládá rychlost 100 Mb/s v poloviním duplexu a druhé rychlost 1 Gb/s s plným duplexem, použije se rychlost 100 Mb/s a poloviční duplex.

2.6.2 Napájení přes Ethernet

PoE (Power over Ethernet, IEEE 802.3af) je standard z roku 2003 popisující možnost napájení menších zařízení přes síťový (metalický) kabel. Výhodou je, že k dotyčnému zařízení nemusíme přivádět napájecí kabel. Takto napájené zařízení si samozřejmě musí vystačit s nižším odběrem, typicky jde například o IP telefony, IP kamery a některé Wi-fi access pointy připojené kroucenou dvojlinkou k zařízení, které může fungovat jako zdroj PoE (což je většinou switch).

Poznámka:

Ve skutečnosti se v praxi setkáváme s více různými řešeními – aktivní PoE je podle standardu IEEE 802.3af, a dále existuje několik nestandardizovaných starších implementací.

Switch sloužící jako zdroj PoE musí mít samozřejmě dostatečně dimenzovaný napájecí zdroj (což je ta komponenta, která stejně jako u běžných počítačů provádí transformaci střídavého napětí z napájecí sítě na stejnosměrné napětí), a i přesto obvykle nedokáže napájet zařízení na všech svých portech (většinou i méně než polovinu), zbývající připojená zařízení musí mít vlastní napájení.

Rozlišujeme *výkonové třídy* 0–4 podle toho, jaký proud a maximální příkon napájené zařízení potřebuje, přičemž třída 0 je pro zařízení nepodporující PoE. Čím vyšší příkon, tím vyšší třída.

Podobně jako při stanovování parametrů připojení se v rámci autonegociace dojednává rychlost a duplex, u PoE se při autonegociaci určuje třída. Nejdřív proběhne detekce, při níž se k připojenému zařízení pouští proud o nízkém napětí, které by v případě, že zařízení „neumí“ PoE, nemělo ublížit (odpovídá třídě 0). Jestliže zařízení zareaguje, pokračuje detekce určením odpovídající výkonové třídy.

Pro účely napájení se v kabelu používají některé z vodičů. Jestliže pro data potřebujeme jen dva páry (u Fast Ethernetu), pak jsou zbývající dva páry použity pro napájení (tj. páry 4+5, 7+8). Pokud pro data potřebujeme všechny čtyři páry (Gigabit Ethernet), pak slouží zároveň pro napájení.

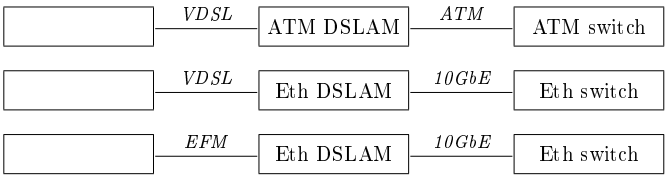
Další informace:

http://vyvoj.hw.cz/produkty/ethernet/princip-cinnosti-power-over-ethernet.html

2.6.3 EFM

EFM (Ethernet in the First Mile) (IEEE 802.3ah, rok 2004) je možnost využití Ethernetu na celé cestě přes WAN až ke koncovým počítačům. Počítá se jak s metalickými, tak i optickými spoji. Protože 10G a rychlejší Ethernet je použitelný (a také používán) v rozlehlých sítích, bylo by praktické použít Ethernet také na poslední (první) míli připojení k internetu („first mile“).

Pro připojení k ATM WAN se používá například VDSL. Při přechodu WAN od ATM k Ethernetu se uvažuje o nahrazení VDSL linek Ethernetem, protože by to znamenalo méně transformací dat na cestě při kombinování různých technologií.



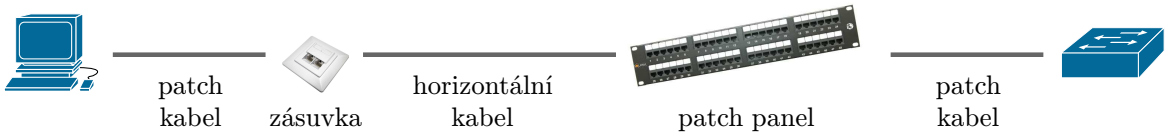
Obrázek 2.12: Postup zavádění EFM

2.6.4 Strukturovaná kabeláž

Rozvaděč (rack, rozvodná skříň) je speciální (otevřená nebo uzavřená) skříň poskytující zařízením (switchům, serverům apod.) upevnění, řízené chlazení, elektroinstalaci a konektivitu. Rozměry rozvaděčů jsou standardizované, resp. existuje několik běžných typizovaných rozměrů. Nejběžnější šířka je 19"(může být například 10”, 21”, 23”), hloubka 600 nebo 800 mm. Velikost zařízení do rozvaděče tomu samozřejmě odpovídá, přičemž výška se udává v násobcích jednotky 1 U = 1,75" = 4,4 cm.

Patch panel je pasivní síťový prvek, který obvykle umísťujeme mezi horizontální kabel a switch. Můžeme si ho představit jako předsunutou sadu portů pro „schovaný“ switch, přičemž patch panel taky obvykle připevňujeme do racku, typická výška je 1U.

Použití patch panelu nám zpřehledňuje a zjednodušuje přístup k portům switche (tedy pokud je propojení uděláno správně). Patch panel může být v „přístupnější“ výšce než switch, tedy se k němu snáze dostaneme, navíc do jednoho patch panelu mohou být zapojeny kabely z více switchů. Pokud je třeba změnit zapojení pro konkrétní horizontální kabel vedoucí z určité zásuvky, nemusíme provádět změnu na switchi, stačí přehodit kabel na patch panelu.



Obrázek 2.13: Horizontální kabeláž

Horizontální rozvody jsou rozvody sítě vedoucí víceméně vodorovně (horizontálně), obvykle propojují zařízení v rámci jednoho patra budovy. Kabel pro horizontální rozvod je obvykle veden zdí mezi zásuvkou na jedné straně a switchem (přesněji patch panelem) na druhé straně. Co se týče délky jednotlivých částí horizontální trasy, platí:

- Celá fyzická přenosová cesta od hostitele na levé straně ke switchi musí měřit maximálně 100 m.
- Fyzická délka horizontálního kabelu (od zásuvky k patch panelu) je maximálně 90 m.
- Délka patch kabelu mezi hostitelem a zásuvkou je maximálně 20 m.
- Délka ostatních patch kabelů (vč. mezi patch panelem a switchem) je maximálně 5 m.

Když sečteme hodnoty od druhého bodu seznamu dále, dostaneme víc než 100 m, ale ta hranice samozřejmě platí. Takže pokud se v některém bodě blížíme horní hranici, musíme ubrat jinde.

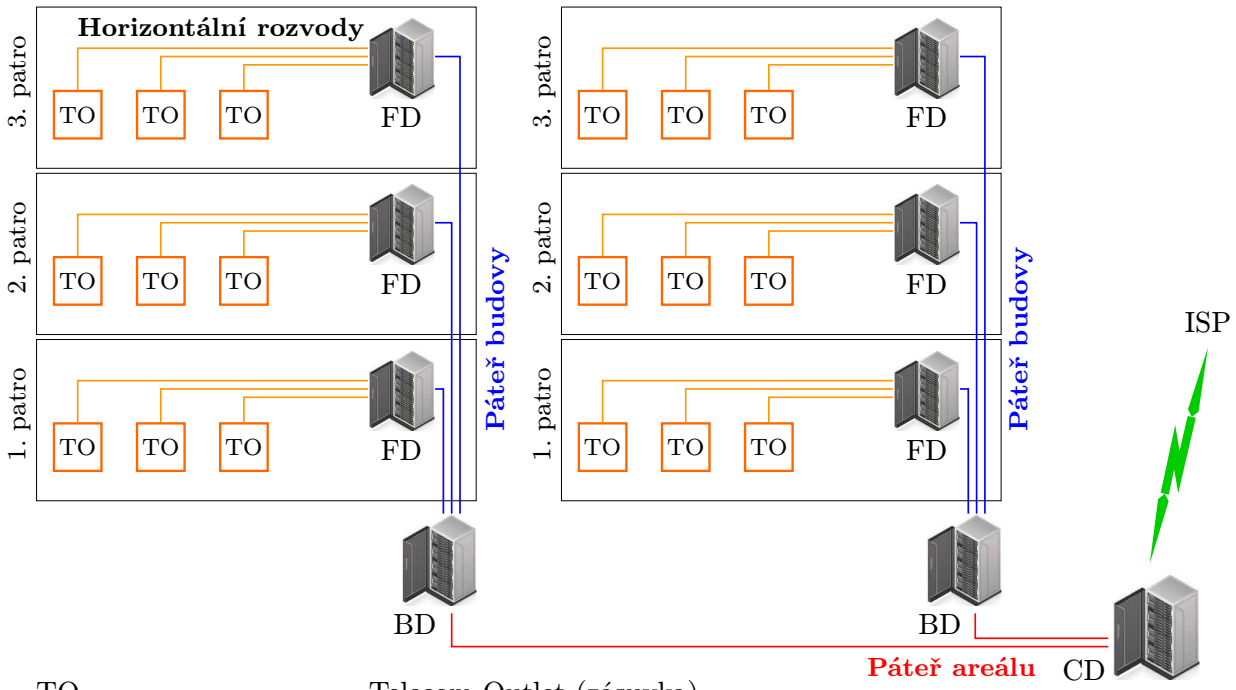
Síťová (telekomunikační) místnost. Na každém patře budovy by měl být dobře zabezpečený prostor (ideálně vyhrazená místnost, pokud je to možné), do kterého jsou svedeny horizontální rozvody od všech zásuvek. Pokud je to samostatná místnost, nazýváme ji *telekomunikační místnost* nebo *síťová místnost* patra. Je v ní především rozvaděč, do kterého ústí horizontální rozvody z celého patra, tedy *floor distributor* (FD).

Páteřní síť budovy (building backbone) propojuje všechna patra budovy, resp. všechny FD. Středem páteře budovy je *building distributor* (BD) – rozvaděč pro celou budovu.

Páteřní síť areálu (campus backbone). Pojem *campus* (areál) označuje skupinu budov, které určitým způsobem patří sobě (například areál patřící jedné firmě. *Campus distributor* (CD) je rozvaděč zastřešující síť v celém areálu, jednotlivé BD jsou s CD propojeny *páteřní sítí areálu*. Právě přes CD jde veškerá komunikace s poskytovatelem internetu (ISP) pro danou firmu.

Strukturovaná kabeláž jako celek. Pojem „strukturovaná kabeláž“ ani zdaleka nezahrnuje pouze to, jak mají být jednotlivá zařízení propojena. Je to komplexní systém zahrnující

- požadavky a doporučení týkající se rozvrstvení celé sítě,
- pasivních prvků (nejen kabelů, ale také například zásuvek a patch panelů), kabelů pro určité části sítě,
- požadavky na elektroinstalaci, redundantní napájení (pro případ výpadku) – UPS, ochranu proti přepětí různých úrovní,



TO ..... Telecom Outlet (zásuvka)
FD ..... Floor Distributor (rozvaděč pro podlaží)
BD ... Building Distributor (rozvaděč pro budovu)
CD ..... Campus Distributor (rozvaděč pro areál)

Obrázek 2.14: Schéma rozvodů strukturované kabeláže

- mechanické zabezpečení (například proti vniknutí neoprávněných osob),
- ochranu proti ohni a kouři, požární předpisy, atd.

Součástí sítě může být pobočková telefonní ústředna (pro vnitřní telefonní síť) a napojení na nejbližší místní telefonní ústřednu, přes které je možné řešit i přístup na Internet.

Strukturovaná kabeláž je u nás standardizována normami ČSN EN 50 173 (základní požadavky na strukturovanou kabeláž) a ČSN EN 50 174 (instalace kabelových rozvodů), a existují i další související normy a standardy. Z mezinárodních standardů de-facto se strukturovanou kabeláží zabývá ANSI/EIA-568-C (rozhodně není jen o barevných kódech pro vodiče kabelů), bezpečnostními aspekty strukturované kabeláže pak část skupiny standardů ISO 27000.

Poznámka:

V normě ČSN EN 50 173 se (ze záhadných důvodů) místo strukturované kabeláže mluví o „univerzálních kabelážních systémech“, ale ve skutečnosti jde o totéž.

Další informace:

- http://elektrika.cz/data/clanky/dsknn2
- https://www.anixter.com/content/dam/Anixter/Guide/12H0001X00-Anixter-Standard-Ref-Guide-ECS-US.pdf
- http://www.ladinn.cz/ostatni/technika/SKS.html

Kapitola 3

Další témata k lokálním sítím

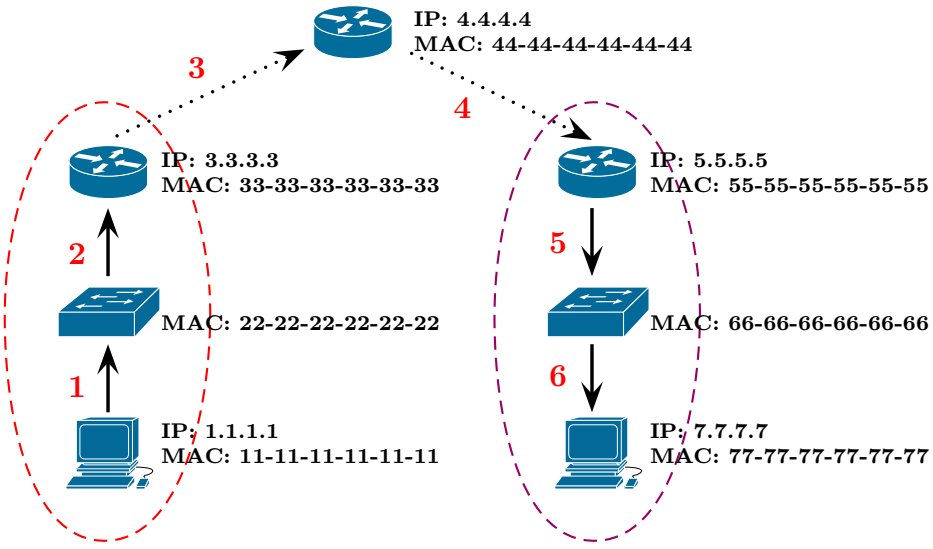
3.1 Adresy v rámcích a paketech

Když odesíláme IP paket, potřebujeme samozřejmě IP adresu zdroje (naši) a příjemce, ale musíme si uvědomit, že IP paket se má zapouzdřit do rámce (třeba ethernetového), a tam jsou uvedeny i MAC adresy zdroje a cíle.

Do rámce, který odesíláme, jako zdrojovou dáme vlastní MAC adresu, ale jakou dát tu cílovou? Určitě nemůžeme použít MAC adresu příjemce, protože tu neznáme. O příjemci (třeba serveru) máme totiž jen omezenou informaci (jenom IP nebo jmennou adresu), navíc pod toutéž jmennou či IP adresou může v reálu existovat více než jedno fyzické zařízení, například z důvodu vyrovnávání zátěže nebo kvůli údržbě. Jediná MAC adresa, kterou známe, je MAC adresa brány.

Příklad

Na obrázku 3.1 je naznačena struktura sítě, přes kterou máme poslat paket. Zdrojovým zařízením je počítač s IP adresou `1.1.1.1`, cílovým zařízením počítač s IP adresou `7.7.7.7`.



Obrázek 3.1: Vztah mezi L2 a L3 adresováním

Jak to proběhne:

- Na prvním úseku cesty směřuje PDU k switchi, který vidí jen do rámce; rámec a paket zapouzdřený v rámci obsahují adresy:

Zdroj:	IP 1.1.1.1	MAC 11-11-11-11-11-11	adresy odesílatele
Cíl:	IP 7.7.7.7	MAC 33-33-33-33-33-33	IP adresa příjemce, MAC adresa brány

  Všimněte si, že IP a MAC adresa cíle k sobě nepatří, ale to vůbec nevadí.

- Na druhém úseku se PDU dostává k bráně a adresy se zatím nezměnily:

Zdroj:	IP 1.1.1.1	MAC 11-11-11-11-11-11	adresy odesílatele
Cíl:	IP 7.7.7.7	MAC 33-33-33-33-33-33	IP adresa příjemce, MAC adresa brány

- Třetí úsek cesty již vede mimo síť odesílatele. Router, který plní roli brány, přijal a rozbalil rámec, rozbalil vnořený IP paket a znovu ho zabalil, tentokrát s jinými MAC adresami.

Zdroj:	IP 1.1.1.1	MAC 33-33-33-33-33-33	IP adresa odesílatele, MAC adresa začátku úseku
Cíl:	IP 7.7.7.7	MAC 44-44-44-44-44-44	IP adresa příjemce, MAC adresa konce úseku

- Podobně to bude na čtvrtém úseku:

Zdroj:	IP 1.1.1.1	MAC 44-44-44-44-44-44	IP adresa odesílatele, MAC adresa začátku úseku
Cíl:	IP 7.7.7.7	MAC 55-55-55-55-55-55	IP adresa příjemce, MAC adresa konce úseku

- Pak již jsme v síti příjemce. Brána v síti příjemce „zná“ MAC adresy uzlů ve své síti.

Zdroj:	IP 1.1.1.1	MAC 55-55-55-55-55-55	IP adresa odesílatele, MAC adresa cizí brány
Cíl:	IP 7.7.7.7	MAC 77-77-77-77-77-77	adresy příjemce

- Na šestém úseku už jen zůstáváme na vrstvě L2 a nemá smysl adresy měnit.

Zdroj:	IP 1.1.1.1	MAC 55-55-55-55-55-55	IP adresa odesílatele, MAC adresa cizí brány
Cíl:	IP 7.7.7.7	MAC 77-77-77-77-77-77	adresy příjemce

Z toho plyne, že opravdu nemusíme znát MAC adresu cíle, stačí nám IP adresa, a cíl zase nemusí znát naši MAC adresu. Cíl sice dostane rámec obsahující „nějakou“ MAC adresu, ale ta by byla naše pouze v případě, že jsme ve stejné síti. Taky z toho plyne další informace – switch, ke kterému jsme přímo připojeni, je pro nás vlastně „neviditelný“, k jeho MAC adrese se v rámcích nedostaneme (k jeho IP adrese taky ne – žádnou nemá).

3.2 Přehled rodiny standardů IEEE 802

Víme, že Ethernet je standardizován jako IEEE 802.3, a že existují standardy IEEE 802.2 (pro podvrstu LLC linkové vrstvy) a standard IEEE 802.1 (s jeho některými částmi jsme se setkali v této kapitole). Ve skutečnosti je rodina standardů IEEE 802 (Standardy pro místní sítě) mnohem širší. Obsahuje standardy definující vrstvu síťového rozhraní podle TCP/IP, resp. vrstvy L1 a L2 podle ISO/OSI, nebo část této vrstvy (vrstev). Obvykle se počítá s „podsunutím“ pod síťový zásobník TCP/IP.

V tabulce 3.1 je přehled momentálně existujících standardů z rodiny IEEE 802. Každý standard je vypracováván konkrétní pracovní skupinou (working group), například standard IEEE 802.3 má na starosti *Ethernet Working Group* a standard IEEE 802.11 má „na svědomí“ *Wireless LAN Working Group*.

Standard	Význam
802.1	Technologie mostů a správa sítí na vrstvě L2
802.2	Logické řízení spoje na vrstvě L2 – podvrstva LLC vrstvy L2 (pracovní skupina neaktivní)
802.3	Lokální síť Ethernet, přístupová metoda CSMA/CD (vrstva L1 a podvrstva MAC vrstvy L2)
802.4	Lokální síť Token Bus – sběrnicová LAN (pracovní skupina rozpuštěna)
802.5	Lokální síť Token Ring – kruhová LAN (pracovní skupina neaktivní)
802.6	Metropolitní síť DQDB (pracovní skupina rozpuštěna)
802.7	Širokopásmové (broadband) sítě LAN (pracovní skupina rozpuštěna)
802.8	Sítě LAN/MAN na optických vláknech (pracovní skupina rozpuštěna)
802.9	Integrované služby – izochronní sítě (pracovní skupina rozpuštěna)
802.10	Bezpečnost v sítích LAN/MAN (pracovní skupina rozpuštěna)
802.11	WLAN Working Group: Bezdrátová lokální síť – Wi-fi
802.12	Vysokorychlostní sítě – 100VG-AnyLAN (pracovní skupina rozpuštěna)
802.14	Širokopásmová síť na kabelech pro kabelové televize (pracovní skupina rozpuštěna)
802.15	WPAN Working Group: bezdrátové osobní sítě (například Bluetooth, ZigBee, UWB), včetně koexistence s IEEE 802.11
802.16	Širokopásmové bezdrátové MAN sítě (WiMAX) – neaktivní
802.17	RPR – Resilient Packet Ring, používá se v sítích SONET/SDH, neaktivní
802.18	Regulace rádiových vln
802.19	Koexistence nelicencovaných bezdrátových sítí
802.20	Mobilní širokopásmové bezdrátové LAN/MAN sítě včetně dopravní mobility
802.21	Handover Services Working Group – předávání mezi různými typy mobilních a bezdrátových sítí (GSM, GPRS, Wi-fi, Bluetooth, WiMAX, atd.)
802.22	Bezdrátové regionální sítě využívající nepoužívané frekvence televizního vysílání
802.23	Emergency Services Working Group (neaktivní)
802.24	Smart Grid Technical Advisory Group: vertikální aplikace – Internet of Things (IoT), komunikace Machine-to-Machine, Smart Grid, propojení dopravních prostředků…

Tabulka 3.1: Rodina standardů IEEE 802

Nejdůležitější standardy (tedy z našeho pohledu) jsou zvýrazněny tučným písmem. Některé z nich už známe, s jinými se seznámíme později.

Poznámka:

Všimněte si, že chybí řádek IEEE 802.13. Oficiálně je číslo 13 vyhrazeno pro další vývoj fast Ethernetu, ve skutečnosti je důvod podobný důvodu toho, proč v mnoha hotelech chybí třinácté patro.

Standard IEEE 802.1 je asi nejrůznorodější. Zatím jsme se setkali s těmito jeho podřízenými standardy (velké písmeno v označení) a dodatky (malé písmeno):

- IEEE 802.1Q – VLAN (Virtuální lokální sítě),
- IEEE 802.1p – stanovení priority (hodnota využívající 3 bity, tedy v rozsahu 0–7) používané například standardem IEEE 802.1Q, ve skutečnosti to není standard, pouze jednoduchý předpis,

- IEEE 802.1D – protokol STP (technologie mostů a síť bez smyček),
- IEEE 802.1w – původně protokol RSTP (opět není chápáno jako samostatný standard), později je specifikace přidána do revize IEEE 802.1D-2004,
- IEEE 802.1s – původně protokol MSTP, později přidán do revize IEEE 802.1Q-2005.

Z těch, kterým jsme se dosud nevěnovali, je zajímavý například IEEE 802.1X – autentizace zařízení přistupujícího do LAN. Autentizace podle tohoto standardu se často používá ve firemních sítích pro ověřování přístupu uživatele/zařízení do sítě a určování konkrétních oprávnění pro daného uživatele.

Poznámka:

Mnohé z pracovních skupin IEEE 802 jsou buď neaktivní nebo dokonce rozpuštěné. Neaktivní znamená, že dlouho nebyla vydána žádná aktualizace či dodatek (což není problém, když dodatky nejsou nutné), rozpuštěná znamená, že se ani v budoucnu neplánují žádné aktivity (typicky u technologií, které se neprosadily). Například sítě Token Ring, Token Bus a 100VG-AnyLAN byly postupně vytlačeny Ethernetem, Izochronní sítě nabízející možnost přidělovat prioritu některým typům dat (třeba při přenosu hlasu) taktéž (Ethernet s využitím IEEE 802.1p/Q to umí taky a je rychlejší). Jinými slovy, právě Ethernet stojí za upozaděním většiny standardů z rodiny IEEE 802.

3.3 Další lokální sítě

Pojem *přístupová metoda* už známe – víme, že se jedná o metodu pro stanovení vysílacího přístupu připojeného zařízení na sdílené přenosové médium, přičemž se může jednat i o kolizní metodu (určující buď způsob vyhýbání se kolizím nebo reakci na kolizi). Zatím jsme se seznámili s přístupovou/kolizní metodou CSMA/CD, která se používá v Ethernetu provozovaném v poloviční duplexu a jejím účelem je, aby zařízení dokázalo zjistit kolizi a správně na ni reagovat. V sítích IEEE 802.11 (Wi-fi) se pro změnu používá přístupová metoda CSMA/CA, kde se klade důraz na to, aby ke kolizi pokud možno vůbec nedošlo.

V této podkapitole se podíváme na další lokální sítě a jejich přístupové metody. Vzhledem k tomu, že tyto sítě byly prakticky vytlačeny Ethernetem, nebudeme zacházet do podrobností.

Přístupová metoda je *deterministická*, pokud zaručuje každému připojenému zařízení možnost vysílání, tedy že (při určitém čekání) dostane příležitost vyslat rámec, přístup všech zařízení k přenosovému médiu je spravedlivý a rovnocenný. Přístupová metoda, která toto nezaručuje, je *nedeterministická*.

Přístupové metody CSMA/CD a CSMA/CA jsou nedeterministické. Není stanoveno žádné pořadí, ve kterém by zařízení dostávala možnost vysílat, a při větším zarušení sítě se může stát, že rámec nebude možné vyslat vůbec (vzpomeňte si na Backoff algoritmus – po určitém počtu čekacích cyklů to odesílatel musí vzdát).

3.3.1 Token Ring

Token Ring je síť původně od společnosti IBM, později byla standardizována jako IEEE 802.5.

Fyzická topologie je hvězda (nebo více propojených hvězd), přičemž ve středu hvězdy máme speciální zařízení nazývané MSAU (Multistation Attachment Unit) – obdobu switche z Ethernetu. *Logická*

topologie je kruhová (proto je v názvu „Ring“), tedy připojená zařízení typu DTE jsou seřazena do logického kruhu, což souvisí právě s přístupovou metodou.

Používá se *přístupová metoda Token Passing* (předávání tokenu = peška, jako v dětské hře „chodí pešek okolo“). Sítí prochází v logickém kruhu speciální rámec zvaný *token*, vysílat může pouze to zařízení, které tento token právě „vlastní“. Když chce zařízení vyslat rámec, postupuje takto:

- počká, až obdrží token, čímž získá právo vysílat,
- vyšle rámec s daty,
- počká na potvrzení o doručení od cíle (tedy token s indikací potvrzení v záhlaví doputuje po kruhu zpět k odesílateli),
- vyšle do sítě původní token, čímž dovolí vysílat někomu dalšímu.

Pokud zařízení obdrží token a nechce vysílat, pak token přepošle sousedovi na kruhu. Naopak když chce vysílat, „pozdrží“ token u sebe po celou dobu od vyslání rámce do přijetí potvrzení.

Přístupová metoda Token Passing je deterministická – neobsahuje žádný náhodný prvek a zaručuje každému zařízení, že dostane možnost vyslat rámec.

Protokol IEEE 802.5 definuje pouze vrstvu L1 a podvrstvu MAC vrstvy L2 (podobně jako IEEE 802.3). Záhlaví a zápatí rámce je složitější než u Ethernetu, má o několik polí více.

MAC rámec podle IEEE 802.5 obsahuje v záhlaví bit indikace tokenu. Pokud je tento bit nastaven na 1, jde o krátký tokenový rámec a uvnitř není nic zapouzdřeno, kdežto pokud je tento bit nastaven na 0, máme uvnitř LLC nebo SNAP rámec.

	Ethernet	**Token Ring**
Přístupová metoda	nedeterministická CSMA/CD	deterministická Token Passing
Kolize	jen při half duplexu	žádné
Fyzická topologie	hvězda/strom	hvězda/strom
Logická topologie	sběrnice/point-to-point	kruh
Složitost	jednoduché rámce	složitější rámce
Cena	relativně levný hardware	mnohem dražší hardware

Tabulka 3.2: Srovnání sítí Ethernet a Token Ring

Z údajů v tabulce 3.2 plyne, proč nad sítěmi Token Ring zvítězil Ethernet – v prvních dvou řádcích je Ethernet v nevýhodě, ale nedeterminismus přístupové metody je obcházen přechodem na plný duplex. V praxi je jedním z nejdůležitějších kritérií cena, která také má svůj podíl na momentálním stavu.

3.3.2 100VG-AnyLAN

Síť 100VG-AnyLAN (standard IEEE 802.12) vznikla jako alternativa během vývoje Fast Ethernetu. Název 100VG AnyLAN znamená:

- rychlost 100 Mb/s (jako Fast Ethernet),
- VG je zkratka z „Voice Grade“ – hlasová kvalita (protože se počítalo s kabeláží UTP Cat3 používané také pro telefonní linky),
- AnyLAN je odkaz na to, že jde o LAN s tím, že se pro data používají všechny v té době nejpoužívanější typy rámců (Ethernet II, IEEE 802.3 a IEEE 802.5).

Z toho je zřejmé, že v IEEE 802.12 nejsou přímo specifikovány rámce pro data, ale používají se rámce z jiných typů sítí.

Fyzická topologie je hvězda nebo strom, jako DCE jsou použity huby (rozbočovače), případně mosty. Síť je přísně hierarchická – jeden hub je kořenem stromu, každý uzel má přehled o svých podřízených uzlech z nižší vrstvy a o svém nadřízeném DCE. *Logická topologie* je také hvězda či strom.

Používá se přístupová metoda *Demand Priority* (DPAM, Demand Priority Access Method), která je centralizovaná a vysílat může pouze takové zařízení, které k tomu dostane povolení. Pokud některé zařízení v síti chce vysílat, musí poslat nadřízenému uzlu speciální rámec s žádostí o vysílání s určením priority vysílaných dat (vysoká priorita pro multimediální data). Až po obdržení povolení vysílá rámec.

Kořenový DCE si vede *prioritní frontu žádostí o vysílání* (to znamená, že žádosti o vysílání dat s vyšší prioritou předbíhají nižší prioritu). Pokud žádost s nižší prioritou čeká na vyřízení déle než 300 ms, dostane vyšší prioritu. DCE, který není kořenový, pouze přeposílá obdržené žádosti směrem nahoru a obdržená povolení směrem dolů. Jak vidíme, rozhodování je plně centralizované.

Přístupová metoda DPAM je *deterministická* – řízení pomocí front neobsahuje žádnou náhodnost a zaručuje, že každé zařízení dříve či později dostane povolení vysílat.

	Ethernet	**100VG-AnyLAN**
Přístupová metoda	nedeterministická CSMA/CD	deterministická Demand Priority
Kolize	jen při half duplexu	žádné
Fyzická topologie	hvězda/strom	hvězda/strom
Logická topologie	sběrnice/point-to-point	strom, centralizovaná
Max. propustnost	50 % při použití hubů	i přes 90 %

Tabulka 3.3: Srovnání sítí Ethernet a 100VG-AnyLAN

Poznámka:

Principy použité při návrhu 100VG-AnyLAN byly původně zamýšleny pro Fast Ethernet – na vývoji Fast Ethernetu původně pracovala skupina lidí, která přišla právě s touto myšlenkou. Vedením však byli odmítnuti („to přece není Ethernet – vytvořte si vlastní pracovní skupinu“), a tak vznikla samostatná pracovní skupina pro IEEE 802.12 a nový typ sítě.

Proč zvítězil Ethernet? Princip 100VG-AnyLAN byl původně životaschopný a mohl Fast Ethernetu konkurovat (deterministická přístupová metoda, lepší chování při vysokém vytížení sítě). Jenže došlo ke zlevnění switchů, a při použití switche jako DCE se dá implementovat plný duplex při mnohem lepším využití spoje a nevýhody Ethernetu odpadají. Navíc může být na závadu centralizovanost řízení – decentralizovaný systém lépe odolává výpadkům.

3.4 VLAN

VLAN (Virtuální LAN, Virtual LAN) je logické seskupení určitých zdrojů (například počítačů) nacházejících se v jedné nebo několika běžných lokálních sítích. Členění na virtuální sítě se sice konfiguruje

na vrstvě L2, ale „táhne se“ na vrstvu L3, protože každá VLAN odpovídá některé podsíti naší sítě. Zařízení patřící do určité VLAN má přiřazenu IP adresu spadající do příslušné podsítě.

Kdyby nešlo o VLANy, ale o „obyčejné“ podsítě, měli bychom router (nebo L3 switch) a za každým jeho L3 portem by byla jiná podsíť, tedy všechna zařízení nacházející se za tím jedním portem by musela patřit do stejné podsítě. VLANy nám dávají možnost oprostit příslušnost k podsíti od fyzického umístění (od toho, ke kterému switchi/portu routeru je zařízení připojeno: k témuž switchi mohou být připojena zařízení patřící do různých VLAN.

V rámci téže VLAN (a v důsledku podsítě) je možné komunikovat jednoduše bez jakéhokoliv filtrování a přeposílání, přes switche. Pokud mají komunikovat zařízení patřící do různých VLAN, pak tyto pakety musí jít přes L3 zařízení, které má jedno rozhraní v jedné VLAN/podsíti a druhé rozhraní v druhé VLAN/podsíti. L3 zařízení pak může například filtrovat, určovat, jestli ta komunikace má či nemá proběhnout.

Proč něco takového dělat? Například z těchto důvodů:

- Zabezpečení – máme možnost omezit přístup ke konkrétnímu prostředku (třeba serveru s citlivými firemními informacemi) jen na někoho (členy jedné konkrétní VLANy).
- Omezení komunikace mezi VLAN sítěmi platí i pro broadcastové vysílání.
- Máme možnost zpřehlednit si si síť rozdělením do částí bez toho, abychom toto rozdělení museli řešit na úrovni kabelů a umístění switchů.

Rozdělení VLAN má smysl nejen kvůli bezpečnosti, ale například taky pro nastavení kvality služby (QoS), tedy priorit, zároveň s dalšími možnostmi řízení provozu. Například je typicky používána voice VLAN (pro IP telefony v rámci firmy).

Členství ve skupině (VLAN) se dá stanovit podle různých kritérií, záleží na tom, na které vrstvě ISO/OSI chceme VLAN implementovat, tedy na konkrétní technologii.

1. *VLAN podle portů* – na přepínačích v síti jsou konkrétní porty (a tedy stanice k nim připojené) roztříděny do jednotlivých VLAN. Tato metoda je poměrně jednoduchá. Při změně struktury sítě (například stanice bude přenesena a připojena k jinému portu) je však nutné provést rekonfiguraci, a další nevýhodou je nemožnost zařazení stanice (portu) do více než jedné VLAN.
2. *VLAN podle MAC adres* (na L2) – tato metoda je náročnější na počáteční konfiguraci, protože MAC adresy musejí být do skupin přiřazeny ručně. Odpadá však ruční rekonfigurace při změně umístění stanice a jedna stanice může být i ve více VLAN. Nevýhodou je větší časová náročnost přepínání a s tím související snížená propustnost sítě (zvláště pokud jedna MAC adresa přísluší do více VLAN). Je třeba dát pozor na možnost pozměnění MAC adresy.
3. *VLAN na síťové vrstvě* – seskupujeme stanice podle IP adresy podsítě. To lze pouze u mezilehlých prvků sítě, které pracují i na síťové vrstvě (přepínač samotný jen na L2).
4. *VLAN podle multicast adresy* – členství v VLAN závisí na příjmu konkrétní multicast komunikace. Na rozdíl od předchozích typů VLAN jsou rámce posílané uvnitř této VLAN vždy doručeny všem členům skupiny, ne jen jednomu (adresace je vždy skupinová).
5. *VLAN podle politiky* – příslušnost do VLAN je určena kombinací více kritérií (tj. politikou, „zásadou“) a může být dynamicky měněna podle chování stanice v síti (umístění, typu a množství zasílaných rámců, atd.). Chování stanic a provoz na síti jsou automaticky monitorovány a politiky mohou být přizpůsobovány.

6. *VLAN s autentizací* – můžeme použít IEEE 802.1X pro autentizaci uživatele (RADIUS) a autorizace pak obsahuje zařazení uživatele do příslušné VLAN. Je také možné nastavit, že uživatel, který nebyl autentizován (host) bude zařazen do speciální VLAN pro hosty.

V současné době se často kombinuje VLAN podle portů na L2 pro komunikaci v rámci téže VLANy a VLAN na síťové vrstvě pro zajištění (řízené) komunikace mezi VLANami.

V praxi se pro VLAN téměř výhradně používá implementace protokolu IEEE 802.1Q (určuje, jak se má informace o příslušnosti zdroje dat k určité VLAN distribuovat mezi switchi). V dalším textu budeme popisovat základ daný zmíněným protokolem, ten platí vždy, ať použijeme jakékoliv další technologie.

Poznámka:

Dále popisovaný algoritmus VLAN sice souvisí s Ethernetem, ale není povinný (ne každý ethernetový switch ho implementuje, nicméně dnes těžko najdete managovatelný switch, který by neuměl VLANy) a je standardizován zvlášť.

Každé koncové zařízení by mělo do některé VLAN patřit, což se obvykle určuje nastavením příslušné VLAN na portu switche, přes který je zařízení přímo dostupné. VLANy se rozlišují především svým číslem, ale každá VLAN může mít i slovní označení, aby se nich lidé lépe orientovali.

Používáme tato termíny:

- *datové* VLAN – VLAN pro běžný provoz, do nich přiřazujeme zařízení (vlastně porty),
- *default* – pokud na portu není nastaveno nic jiného, je tam defaultní VLAN (u většiny výrobců je to VLAN číslo 1),
- *native* – pokud se mezi dvěma switchi přenáší něco, co nemá žádnou VLAN nastavenou, spadne to do „kanálu“ nativní VLAN (přes nativní VLAN bývají přenášeny například také STP rámce); jestliže jsme v konfiguraci žádnou nativní nenastavili ručně, VLAN 1 se použije jako nativní,
- *management* – je určená pro management rámce (SSH, Syslog, SNMP,...), výchozí nastavení je VLAN 1, ovšem rozhodně je dobré ji přenastavit.

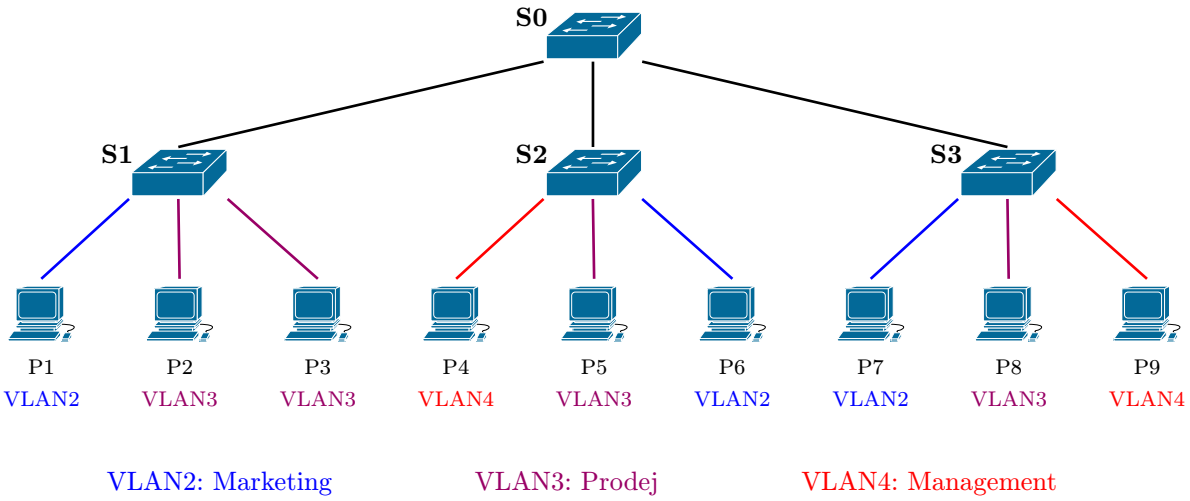
Konkrétní port na switchi je

- *přístupový* – za ním je přímo jedno zařízení, má nastavenu příslušnost do jediné VLAN,
- *trunkový* – za ním je další switch (tedy potenciálně celá řada různých zařízení patřících do různých VLAN), sousední switch má příslušný port taky nastavený jako trunkový.

Přes přístupový port přenášíme pouze rámce patřící do na něm nastavené VLAN, přes trunkový port lze přenášet rámce patřící do více různých VLAN (nastavuje se seznam „povolených“ VLAN, seznam musí být stejný na obou koncích trunku).

3.4.1 VLAN na jednom switchi

Princip VLAN je jednoduchý – rozdělíme zařízení v síti do skupin, každé skupině přiřadíme identifikátor (číslo VLAN) a zajistíme oddělení komunikace mezi těmito skupinami. Na obrázku 3.2 je naznačeno, jak by to mohlo vypadat (poněkud zjednodušeně).



Obrázek 3.2: Ukázka využití VLAN

Na switchi je pro každý port určeno, do které VLAN patří zařízení přes tento port přístupné. Tedy v našem případě je na switchi S1 první port přiřazen do VLAN2 a další dva porty do VLAN3. Port vedoucí směrem nahoru ke switchi S0 bude přenášet rámce z různých VLAN, proto jde o trunk.

Zatímco spoje ke koncovým zařízením jsou znázorněny různými barvami podle příslušnosti k VLAN, trunk je černě, ale můžeme si ho představit jako celý svazek různobarevných virtuálních linek (pro rámce patřící do různých VLAN) a k tomu „dodatkovou“ linku pro nativní VLAN, po které jdou rámce nepatřící do žádné VLANy.

Na každém switchi potřebujeme tabulku MAC adres. Protože přepínání má fungovat pouze mezi zařízeními patřícími do stejné VLAN (kromě trunku), musí být ke každé MAC adrese a portu ještě přidána informace o číslu VLAN.

Přepínání pak bude fungovat takto:

- Switch přijme na daném portu rámec, přičemž tento port patří do konkrétní VLAN.
- V tabulce MAC adres se zaměří pouze na záznamy patřící k téže VLAN a trunky, jiné nekontroluje.
- Mezi těmito „profiltrovanými“ záznamy najde ten, který odpovídá adrese cílového zařízení.
- Pokud vyjde přístupový port (s tímtéž číslem VLAN), pak na něj rámec jednoduše přepošle. Jestliže vyjde trunk (tj. k cíli vede cesta přes jiný switch), upraví rámec (později si ujasníme jak) a pošle do daného trunku.

Zjednodušeně si to můžeme představit tak, že na switchi je pro každou VLAN samostatná tabulka MAC adres (ve skutečnosti tomu tak není, protože bychom pak neměli kam zařadit ta zařízení, která do běžných „komunikačních“ VLAN nepatří). Pokud komunikace zůstává v rámci jednoho switche, je přepínání rámců celkem jednoduché. Pokud například počítač P2 posílá rámec počítači P3, přijme switch S1 tento rámec na „fialovém“ portu (VLAN3) a tedy se řídí podle řádků tabulky pro VLAN3, kde je přímo uvedeno, na kterém portu je cíl (P3) dostupný.

Poznámka:

V předchozí kapitole jsme si ukázali, že pokud cíl není v tabulce MAC adres, switch použije flooding (záplavu) – pošle rámec na všechny porty kromě toho, ze kterého přišel. Ale když se používají VLANy, tak to asi bude jinak. Kam myslíte, že bude takový rámec poslán? A jak se řeší broadcasty?

Zmínka o broadcastech je nápovědou – již dříve tu bylo zmíněno, že různé VLANy odpovídají různým broadcastovým doménám, tedy broadcast se pošle jen na porty, které příslušejí do stejné VLAN jako odesílatel, a do trunků s touto VLAN nakonfigurovanou (jinými slovy, broadcast se nedostane do jiné VLANy). No, a rámec, jehož cílovou adresu nemáme v tabulce, dopadne stejně – pošle se na porty patřící do téže VLAN jako odesílatel (a na trunky s touto VLAN).

Poznámka:

Uvědomte si, že koncová zařízení naprosto netuší nic o nějaké virtualizaci. Když odesílají rámec, vědí jen to, na kterou MAC adresu má dojít, ale číslo VLAN neznají, toto číslo (zatím) není ani součástí odesílaného rámce. VLAN dokáže určit až switch, protože ten si ke každému portu eviduje informaci o VLAN, do které port (a k němu připojené zařízení) patří.

3.4.2 VLAN rámce na cestě přes trunk

Dále potřebujeme mechanismus, který nám umožní správně přepínat rámce, které mají jít od zdroje k cíli přes více než jeden switch. Dokud jsme zůstali v rámci jediného switche, bylo to jednoduché – switch určil VLAN podle portu, ze kterého rámec přišel, a tím byly stanoveny i konkrétní řádky tabulky, na kterých má hledat adresu a port cílového zařízení.

Pokud však je cíl připojen k jinému switchi, pak je třeba rámec nejdřív odeslat na tento switch, který pak již zajistí předání cíli. Jenže jak „tomu druhému“ switchi předat informaci o tom, do které VLAN patřil zdroj, aby následující switch věděl, na které řádky se podívat?

Zaměřme se opět na obrázek 3.2. Předpokládejme, že počítač P1 posílá rámec počítači P6. Nějak to jít musí, protože oba počítače patří do VLAN2.

Switch S1 přijme rámec od počítače P1, podle tabulky zjistí, že ho má předat switchi S0, ale musí *přidat informaci o číslu VLAN*. Switch S0 přijme rámec včetně této dodatečné informace a obojí předá dál switchi S2. Až switch S2 zjistí, že tento rámec může předat přímo na port s přiřazeným číslem VLAN, a tedy dodatečnou informaci o číslu VLAN sice použije (aby určil tabulku adres), ale na cílový port odešle pouze rámec bez dodatečné informace.

Spoj, kterým posíláme rámce patřící do různých VLAN, se nazývá *trunk*. Vede obvykle mezi dvěma switchi nebo mezi switchem a routerem, výjimečně mezi switchem a serverem. Z toho vyplývá, že switche S1–S3 mají oba druhy portů – přístupové i trunkové.

Při komunikaci mezi dvěma počítači patřícími do téže VLAN je vždy prvním a posledním spojem na cestě spoj u přístupového portu, mezilehlé spoje jsou trunkové. Pouze přes trunkové spoje se přenáší dodatečná informace o VLAN.

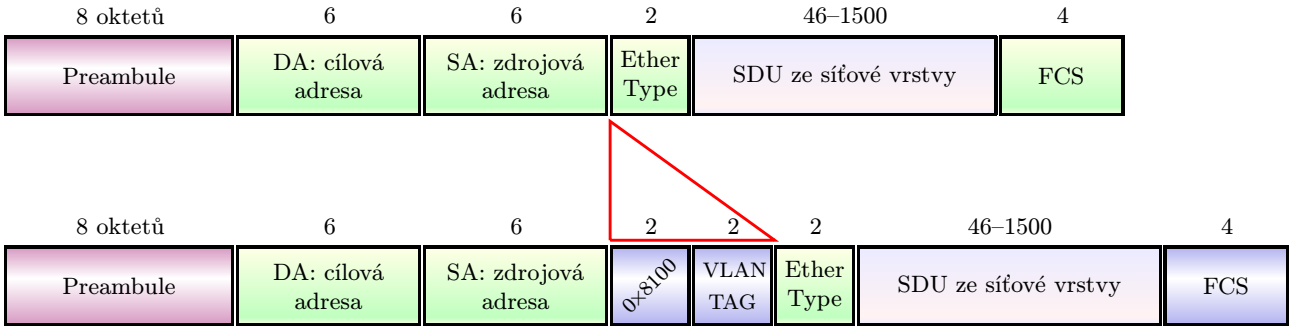
Nyní k té dodatečné informaci o VLAN. Už jsme si řekli, že při komunikaci v rámci jednoho switche (tj. pouze přes přístupové porty) něco takového není zapotřebí (dokonce platí, že pokud přijde přes přístupový port rámec s VLAN informací, bude zahozen, protože pravděpodobně se někdo pokouší „vpašovat“ do VLAN, do které nepatří), ale při komunikaci přes trunkový port je tato informace nutná, protože další switch na cestě nemůže vědět, ze kterého přístupového portu (a tedy VLAN) dotyčný rámec původně přišel.

Existují dva přístupy – první mění záhlaví rámce (přidá tam informaci o VLAN), kdežto podle druhého přístupu rámec neměníme a místo toho jej zabalíme do speciálního rámce s informací o VLAN. První přístup je popsán protokolem IEEE 802.1Q, druhý přístup je používán některými proprietárními protokoly.

VLAN podle IEEE 802.1Q. Podle tohoto standardu se informace o VLAN vkládá přímo do záhlaví přenášeného rámce, a to tak,

- aby bylo jasné, že to je VLAN rámec, a zároveň
- aby se ze záhlaví neztratily žádné původní položky.

Takže všechna původní pole necháme, ale dvě pole s informací o VLAN přidáme. Na obrázku 3.3 vidíme, kam jsou tato dvě pole vsunuta – hned za adresy.



Obrázek 3.3: VLAN rámec podle IEEE 802.1Q

Obsah nových polí je následující:

- *EtherType hodnota pro VLAN* (podle toho swich pozná, že doručený rámec obsahuje VLAN informaci). Pro VLAN to je hodnota 0×8100.
- *VLAN tag*, který v sobě zahrnuje
  - prioritu podle IEEE 802.1p (3 bity), obvykle je tady 0 (to znamená Best Effort – bez priorit),
  - indikace kanonického tvaru adresy (1 bit), obvykle 0,
  - číslo VLAN (12 bitů).

Protože se tímto změnil formát rámce, je třeba znovu vypočíst kontrolní součet, a tedy se mění obsah posledního pole celého rámce (FCS).

Pokud switch (například switch S2 podle obrázku 3.2) přijme na trunkovém portu rámec, zkontroluje pole EtherType. Pokud tam najde číslo 0×8100, je jasné, že se jedná o VLAN rámec, a v následujícím poli hledá identifikátor té VLAN, do které patří odesílatel.

Switch má přepínat rámce mezi porty (z jednoho rámec přijme, na druhý ho pošle) podle tabulky MAC adres, a totéž je třeba provést i zde.

- Switch přijme rámec (bez VLAN ID) na přístupovém portu, podle portu určí VLAN. Pokud zjistí podle tabulky (vyfiltrováním řádků podle VLAN), že cílem nebude jiný přístupový port, přidá do záhlaví obě pole související s VLAN, vypočte kontrolní součet a odešle hotový VLAN rámec na trunkový port.
- Pokud switch přijme rámec na trunkovém portu a zjistí, že cíl je dostupný na přístupovém portu (určil VLAN podle záhlaví rámce a podle toho vyfiltroval řádky tabulky), odstraní ze záhlaví obě VLAN pole, vypočte kontrolní součet a odešle na dotyčný přístupový port.

- Pokud máme switche v stromové hierarchii, pak nejvyšší úrovně obvykle pracují pouze v režimu trunku, tedy nekontrolují VLAN ID – aby páteřní síť zbytečně nezdržovala provoz. Takže takový switch přijme rámec, přečte adresu cíle a jednoduše přepne na příslušný port.

Poznámka:

Pojmy *přístupový port* a *trunkový port* patří do terminologie Cisco. U jiných výrobců se můžeme setkat s jinou terminologií (ale jinak to funguje úplně stejně, podle IEEE 802.1Q), například na zařízeních HP mluvíme o *untagged member* a *tagged member*.

K přístupovému portu (resp. untagged member) je připojeno zařízení, které „nerozumí“ VLAN (běžný počítač, server apod.), kdežto na trunkovém portu (resp. tagged member) je připojeno zařízení, které „rozumí“ VLAN (typicky switch) a umí pracovat s VLAN rámci.

Další protokoly. Druhý přístup implementace VLAN rámců spočívá v tom, že se do původního rámce nezasahuje, ale je zapouzdřen do speciálního rámce podle určitého protokolu – přidá se nové záhlaví obsahující kromě jiného identifikaci VLAN. Na (některých) zařízeních Cisco je k tomuto účelu používán *protokol ISL*.

Kromě ISL existují i další protokoly používané k tomuto účelu (proprietární od určitého výrobce, navzájem nekompatibilní). Vznikly ještě v době, kdy IEEE 802.1Q neexistoval, a proto tehdy pro jejich používání existoval důvod.

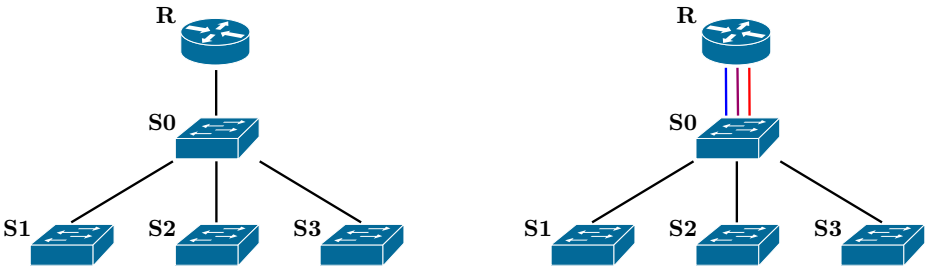
V současné době se pro identifikaci VLAN na trunkových portech téměř výhradně používá IEEE 802.1Q, tedy se jinými protokoly (včetně ISL) ani nebudeme zabývat. Navíc pokud konkrétní zařízení podporuje některý z proprietárních VLAN protokolů, pak obvykle podporuje taky IEEE 802.1Q.

3.4.3 Komunikace mezi různými VLAN sítěmi

Jenže my sice chceme tyto podsítě navzájem oddělit, ale ne úplně. Určitou možnost komunikace bychom chtěli zachovat i mezi různými VLANami, ale zároveň chceme tuto komunikaci určitým způsobem ovlivňovat (například skladníkovi nedovolíme navázat spojení se serverem s osobními údaji zaměstnanců), případně monitorovat, logovat. A právě k tomu nám může sloužit router, který je na obrázku připojený ke kořenovému switchi.

Obecně – komunikaci mezi různými VLANami může zajistit pouze a jenom zařízení pracující na vrstvě L3:

- router,
- switch s funkcionalitou vrstvy L3.



Obrázek 3.4: Možnosti pro zapojení L3 zařízení pro propojení VLAN sítí

Dále budeme pro zjednodušení zmiňovat a zobrazovat většinou router.

Při komunikaci mezi zařízeními z různých VLAN je rámec posílán přes zařízení vrstvy L3. Jestliže v tomto případě zašle počítač P1 z „modré“ VLAN2 rámec počítači P5 z „fialové“ VLAN3, bude cesta v síti následující:

P1 — S1 — S0 — R — S0 — S2 — P4

přičemž na routeru proběhne určité testování a směrování a také se změní identifikátor VLAN v rámci (ve skutečnosti se na jakémkoliv L3 zařízení vždy odstraní původní záhlaví L2 a vytvoří nové). Proběhne směrování paketu vnořeného v rámci z podsítě určené pro VLAN zdroje do podsítě určené pro VLAN cíle, což lze jen na vrstvě L3.

Ovšem je třeba si uvědomit, že jakákoliv komunikace mezi zařízeními z různých VLAN opravdu půjde přes zařízení vrstvy L3, se všemi důsledky včetně prodloužení cesty rámce v síti a případně i snížení propustnosti spojů v síti (i kdyby zdroj a cíl byly připojeny k témuž switchi).

Poznámka:

Pro místo, které může brzdit provoz (včetně výše popsaného trunkového spoje) se vžil název *úzké hrdlo* (bottleneck). Pokud je síť špatně navržená, může L3 zařízení být úzkým hrdlem.

Různé spoje na L3 zařízení nutně musí být v různých podsítích (samozřejmě s nepřekrývajícím se adresním prostorem), tj. každý port by měl mít IP adresu ze „své“ vlastní podsítě. Ale není nutné, aby to platilo pro fyzické porty a fyzické spoje.

Jsou dvě možnosti, jak napojení VLAN na L3 zajistit. V prvním případě povedeme mezi L2 zařízením a L3 zařízením jediný fyzický kabel, který je na logické úrovni rozdělen na virtuální spoje – pro každou VLAN vlastní virtuální spoj, v druhém případě povedeme mezi těmito zařízeními tolik kabelů, kolik je VLAN (tj. každá VLAN bude mít vlastní fyzický kabel). V prvním případě hovoříme o virtuálních subrozhraních jednoho fyzického síťového rozhraní, a místo abychom nastavovali IP adresu na síťovém rozhraní, budeme ji nastavovat zvlášť na jednotlivých subrozhraních. K oběma možnostem podrobněji:

Pokud router podporuje vytváření subrozhraní: centrální switch a router propojíme *z pohledu switche* trunkovým spojem, *na straně routeru* vytvoříme na daném síťovém rozhraní tolik virtuálních subrozhraní, kolik VLAN bude moci projít přes trunk od switche. Každá VLAN bude mít své subrozhraní s vlastním označením a IP adresou. Situace je naznačena na obrázku 3.4 vlevo.

Konkrétně – na switchi tento port nakonfigurujeme jako trunkový, na routeru jde o síťové rozhraní s definovanými subrozhraními (například pro port `g0/1` to budou subrozhraní `g0/1.1`, `g0/1.2`,..., `g0/1.10`,... podle potřeby, za tečkou je číslo VLANy). IP adresu přiřazujeme jednotlivým subrozhraním, například subrozhraní `g0/1.10` bude mít IP adresu patřící do podsítě, kterou jsme určili pro VLAN 10.

Pokud zařízení vrstvy L3 nepodporuje vytvoření subrozhraní: Tento postup se týká jak routerů, které „neumí“ IEEE 802.1Q, tak i L3 switchů, které sice daný protokol zvládají, ale obvykle nepodporují vytvoření subrozhraní na jednom fyzickém rozhraní.

Poznámka:

Možná se to zdá jako paradox, ale L3 switche různých výrobců včetně Cisca opravdu obvykle neumožňují vytvořit na jednom rozhraní více subrozhraní, ale když se nad tím zamyslíte – switch (i s funkcionalitou

L3) mívá desítky nebo dokonce stovky rozhraní, tak proč bychom si měli komplikovat život nějakými subrozhraními?

Mezi hlavním switchem a zařízením vrstvy L3 je třeba vést tolik fyzických spojů, kolik existuje VLAN. Tyto spoje pak budou na straně switche fungovat jako přístupové porty (tj. na každé je definována jedna konkrétní VLAN) a směrem k zařízení L3 (kde je síťové rozhraní s IP adresou patřící do podsítě pro příslušnou VLAN) půjdou již rámce bez VLAN polí.

Takže zatímco za trunkovým portem bývají obvykle L2 switche, za přístupovým (access) portem bývají buď koncová zařízení, nebo L3 zařízení bez definovaných subrozhraní.

Poznámka:

Na předchozích stránkách bylo uvedeno, že koncová zařízení neobsahují implementaci protokolu IEEE 802.1Q (tedy „nerozumějí VLAN“). Existují však i výjimky. U serverů může být výhodné zařazení do více než jedné VLAN (abychom se vyhnuli neustálému přeposílání provozu přes zařízení vrstvy L3), což se právě dá udělat implementací IEEE 802.1Q přímo na tomto zařízení. Jak se to dá provést:

- Pokud se jedná o server s Linuxem, pak je to jednoduché, protože Linux samotný obsahuje implementaci IEEE 802.1Q.
- U serverů s Windows je situace složitější, protože samotné Windows tento protokol nepodporují. Jediná možnost, jak tam rozjet podporu VLAN, je pořídit speciální síťovou kartu, která IEEE 802.1Q implementuje.

Výše bylo uvedeno, že existuje *management VLAN* (tj. VLAN určená pro správu, jsou po ní přenášeny například rámce s konfiguračními příkazy, např. SSH). Přístup na zařízení pro konfiguraci s pomocí Telnetu a SSH (vlastně i webový přes HTTP a HTTPS) funguje jen tehdy, když má zařízení nastavenu alespoň nějakou IP adresu (ta se uvádí jako parametr při navazování spojení), a měla by to být právě adresa z podsítě navázané na management VLAN.

Na L2 zařízení nenastavujeme IP adresu na konkrétním fyzickém rozhraní (to děláme jen na L3 zařízení), ale na virtuálním rozhraní (Switch Virtual Interface – SVI). Stejně jako koncová zařízení, i switch odpovídá na ARP/NDP dotazy typu „kdo má tuto IP adresu?“ ohlášením své IP a MAC adresy, takže v tomto smyslu není problém Telnet/SSH spojení navázat.

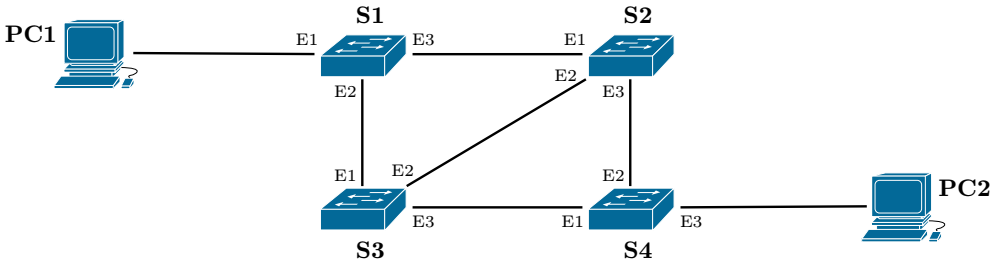
Nás spíše zajímá opačný problém – jak zajistit, aby „nepovolané osoby“ naopak to spojení navázat nemohly? Například právě tak, že určíme management VLAN a na L3 zařízení (třeba routeru) budeme určovat, kdo do podsítě navázané na management VLAN může a kdo ne. Ale pozor, tento postup není všemocný, dá se obejít, nejen využitím IP spoofingu (podvržení IP adresy).

3.5 STP – kostra v grafu

V následujícím textu se zaměříme na postup, který je standardizován pro zařízení typu bridge, nicméně my budeme hovořit o switchi.

3.5.1 Protokol STP

Představme si větší síť, ve které máme několik switchů. Protože chceme, aby síť pracovala i v případě, že některá přenosová cesta bude přerušena nebo některý switch přestane být funkční, jsou switche fyzicky propojeny více než je nutné – některé cesty jsou *redundantní* (znásobené).



Obrázek 3.5: Síť se smyčkami

Redundantní cesty jsou praktické pro případ výpadku části sítě, ale mohou přinést určité problémy. Představme si několik možných scénářů:

1. Počítač PC1 odesílá rámec počítači PC2. První část cesty (switch S1) je jednoznačná, ale co potom? Má být rámec odeslán ze switche S1 přes port E2 nebo přes port E3?
2. Počítač PC2 odesílá broadcastový rámec. Switch S4 rámec přijme na portu E3 a odešle na porty E1 a E2. Co provedou ostatní switche?
   - Switch S2 rámec přijme na portu E3 a odešle na porty E1 a E2.
   - Switch S3 rámec přijme na portu E3 a odešle na porty E1 a E2. Tentýž rámec (o čemž neví) přijme i na portu E2 (od switche S2) a odešle na porty E1 a E3.
   - Switch S2 opět přijme rámec na portu E2 (od switche S3) a odešle na porty E1 a E3.
   - Switch S1 tentýž rámec přijme dvakrát z portu E2 a dvakrát z portu E3, odešle ho postupně celkem dvakrát na port E2, dvakrát na port E3 a čtyřikrát na port E1, čímž se rámec dostane k počítači PC1.
   - Mezitím se broadcastový rámec několikrát dostal zpět ke switchi S4, počítači PC2 a následně opět k dalším switchům...
3. Předpokládejme, že se počítač PC1 připojuje do sítě, tedy zatím není v MAC tabulce žádného switche. Počítač PC2 mu odešle rámec (MAC adresa počítače PC2 je cílová). Switch S4 tuto adresu nezná, tedy rámec odešle na porty E1 a E2, atd. – rámec putuje sítí stejným způsobem jako broadcastový. K počítači PC1 se sice dostane, ale ještě dlouho budou jeho kopie bloudit sítí.

   Tato situace nemusí nastávat jen tehdy, když připojujeme nové zařízení do sítě. Uvědomte si, že v tabulce MAC adres na switchi jsou záznamy obvykle drženy jen po dobu jednotek minut (obvykle 5 minut, záleží na výrobci a případné změně konfigurace).

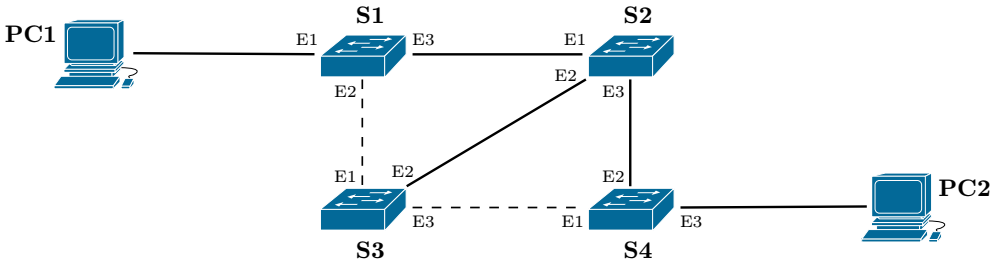
Poznámka:

Scénář popsaný v bodu 2 se nazývá *všesměrová bouře* (broadcast storm). Vyznačuje se tím, že sítí neustále bloudí totožné kopie téhož broadcastového rámce, čímž je síť defacto zahlcena a po určité době nepoužitelná.

Z toho vyplývá, že něco je špatně. Redundance cest je dobrá, ale zároveň bychom měli zabránit nekontrolovanému opakovanému bloudění rámců sítí. Tomu zabráníme, pokud ze sítě *odstraníme smyčky* (co se týče komunikace; ve fyzické topologii mohou smyčky kvůli redundanci zůstat).

Protokol STP (Spanning Tree Protocolol, také protokol kostry v grafu) je standardizován jako IEEE 802.1D jako mechanismus odstranění (logických) smyček v grafu sítě mostů (tedy v našem případě switchů). Tento protokol „vidí“ pouze mosty (resp. switche), žádná jiná zařízení ho nezajímají (takže v naší síti bude počítače PC1 a PC2 ignorovat).

Účelem je překonfigurovat switche v síti takovým způsobem, aby některé porty nebyly pro komunikaci používány (jsou buď deaktivovány nebo používány jen pro servisní účely), čímž ze sítě odstraníme smyčky. Protokol STP vpodstatě řeší „problém hledání kostry v grafu“, jehož účelem je v grafu nechat pouze takové spoje, aby mezi kterýmikoliv dvěma uzly sítě (switchi) existovala právě jedna cesta (ne více, ne méně), ideálně ta nejrychlejší.

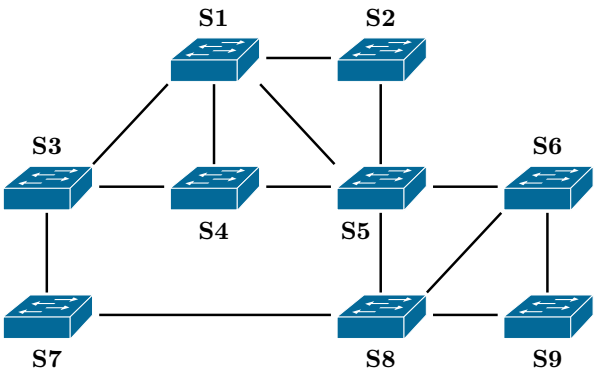


Obrázek 3.6: Síť po odstranění smyček

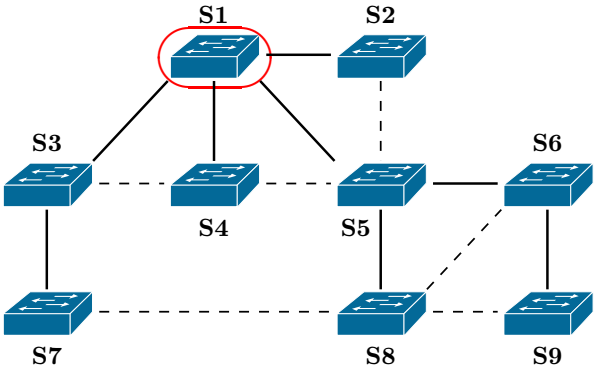
Vlastně ze sítě switchů vytvoříme *strom* („spanning tree“ znamená pokrývající strom). Jeden ze switchů je prohlášen za kořen stromu (*kořenový switch*) a z každého dalšího switche je hledána nejrychlejší cesta ke kořenovému switchi. Všechny ostatní cesty jsou deaktivovány. V našem případě by mohl být kořenovým switchem třeba switch S2 a v síti by zůstaly aktivní pouze spoje S1–S2, S3–S2 a S4–S2. Spoje S1–S3 a S3–S4 by byly deaktivovány.

Příklad

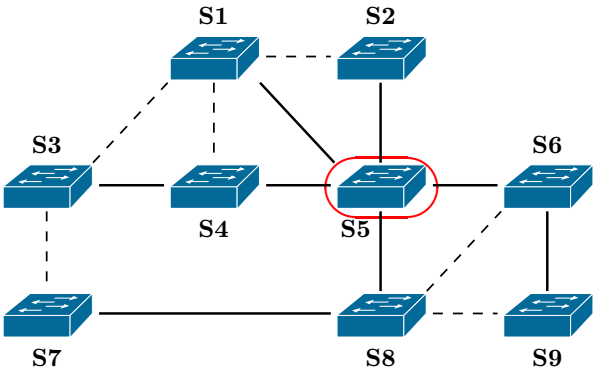
V síti na následujícím obrázku vidíme celkem devět switchů, které jsou propojeny ve fyzické topologii mesh. Máme tady smyčky, které je třeba vyřešit pomocí protokolu STP.

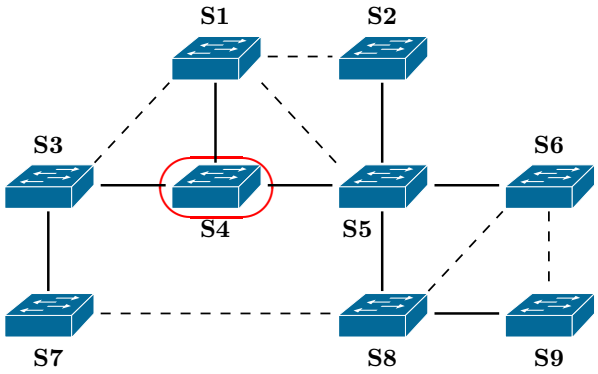


Prvním úkolem je zvolit kořenový switch. Máme na výběr, ovšem v reálné situaci bychom pro tuto volbu zřejmě měli určitá kritéria. Zde jednoduše zvolíme switch S1. Některé porty blokujeme, čímž pro běžnou komunikaci znepřístupníme celkem šest cest.



Jsou i další možnosti – jako kořenový switch můžeme zvolit třeba switch S5 nebo S4:





Také v těchto případech se některé porty deaktivovaly pro běžný provoz, přičemž od každého switche existuje právě jedna (pokud možno nejoptimálnější) cesta ke kořenovému switchi.

Všimněte si, že v každém z těchto tří řešení jsou cesty mezi některými dvojicemi switchů různě dlouhé. Také je třeba brát v úvahu, že tyto nákresy jsou pouze topologie, neberou v úvahu skutečné délky spojů a jejich vytížení, takže v reálné situaci by administrátor při volbě kořenového switche promýšlel různá kritéria.

3.5.2 Konvergence sítě switchů

Podle čeho určuje protokol kořenový switch? Každý switch, který „rozumí“ protokolu STP (je na něm tento protokol implementován), má přiřezen speciální identifikátor (bridge ID, resp. switch ID) o délce 8 oktetů, který se skládá ze dvou částí:

- Priorita (2 oktety)
- MAC adresa (6 oktetů).

1	2	3	4	5	6	7	8
Priorita		MAC adresa					

Standardně je v poli pro prioritu číslo 0×8000, což je desítkově 32 768. Přesto žádné dva switche nemají tento identifikátor stejný (liší se alespoň v MAC adrese, každý switch má jinou).

Protokol STP vybere jako kořenový strom ten switch, jehož switch ID je *nejmenší číslo*. Pokud bychom spoléhali jen na tento postup, mohl by být jako kořenový zvolen ten switch, u kterého se nám to moc nehodí – řešením je změnit prioritu námi vybraného switche na *menší hodnotu*. Díky tomu, že v switch ID je nejdřív priorita a pak až MAC adresa, má tato změna vždy očekávaný účinek.

Zbývá stanovit, jak určíme, které spoje (resp. porty) mají být blokovány a které mají zůstat aktivní. Pokud ke kořenovému switchi vede cesta přes víc než jeden port, pak se pro každý port vypočte

optimalita cesty (záleží na šířce pásma/rychlosti, vytížení, hovoříme o ceně cesty). Aktivní bude jen port pro nejoptimálnější cestu. Pokud pro oba porty vyjde stejná cena cesty, použije se při rozhodování switch ID sousedů, ke kterým tyto porty vedou – soused s nižším switch ID vyhrává, spoj k němu vedoucí zůstane aktivní.

✎ Takže port buď je nebo není aktivní z pohledu přeposílání rámců s běžným provozem. Tuto vlastnost označujeme jako *stav*, rozlišujeme tyto stavy portu:

- *předávání* (forwarding) – funguje, přeposílá veškerý provoz,
- *blocking* – nepřeposílá běžný provoz, ale přijímá správní rámce protokolu STP (tzv. BPDU rámce), aby se v případě potřeby dokázal aktivovat,
- *naslouchání* (listening) a *zjišťování* (learning) – mezilehlé stavy pro případ, že port má přejít ze stavu blokování do stavu předávání; ve stavu listening switch zpracovává BPDU rámce, ve stavu learning přijímá (ale nepřeposílá, zahazuje) rámce s běžným provozem a začíná si tvořit tabulku MAC adres,
- *disabled* – zcela vypnutý port, nepřeposílá vůbec nic.

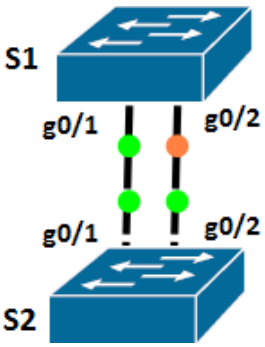
Porty se mezi těmito stavy přesouvají, třebaže většinu času stráví právě ve stavech předávání nebo blokování.

✎ Čas od času (především při změnách v síti) probíhá rekonfigurace – *proces konvergence*. Účelem je zajistit, aby byla síť v konvergentním stavu (tj. v našem případě aby v síti nebyly žádné smyčky a aby byly aktivní pouze porty pro nejoptimálnější cestu). Může být nutné určit nový kořenový switch, a také stavy portů se mohou měnit (například pokud má port přejít ze stavu blokování do stavu předávání, stráví určitý čas ve stavech naslouchání a zjišťování, aby si dokázal nastavit správně všechny parametry a doplnit tabulku MAC adres).

✎ Každý port plní určitou roli, která je dána pozicí switche v hierarchii. Podle role se v dané chvíli určuje, v jakém stavu tento port bude. Jaké role tedy existují?

- *root port* – port, který směřuje k root switchi, tedy každý switch kromě kořenového má jeden root port mířící „nahoru“ (předpokládejme, že máme switche nakresleny tak, že root je nahoře), root port je vždy aktivní, tj. ve stavu forwarding,
- *designated port* – port, který je aktivní a v hierarchii míří dolů, tedy k podřízeným switchům (takže za designated portem máme podstrom switchů); kořenový switch má všechny své porty kromě blokovaných v roli designated, naopak switche v hierarchii zcela dole (takové, od kterých je ke kořenovému switchi nejdelší cesta) nemají žádné designated porty,
- *blocked port* – tento port nebyl vybrán, nepoužívá se pro běžný provoz (tedy obvykle je ve stavu blocking), vyšší verze STP místo tohoto stavu rozlišuje stavy
  - backup port – blokovaný port, který je záložním portem k root portu (tj. když selže cesta nahoru ke kořenovému switchi, tento port se může stát root portem),
  - alternate port – blokovaný port, který je záložním portem k některému designated portu (tj. když selže některá cesta vedoucí k podřízeným switchům, alternate port se stane designated portem a začne předávat provoz).

Speciálním případem je situace, kdy právě mezi dvěma sousedními switchi máme natažené redundantní spoje (prostě mezi dvěma switchi vedou dva kabely), situace je naznačena na obrázku 3.7. Tuto situaci nazýváme *kravata (tie)*. To, který switch bude nadřízený (tj. z těch dvou potenciálně root), se určí tak, jak bylo výše naznačeno (tj. switch ID), ale co s těmi dvěma spoji, který z nich bude aktivní?



Obrázek 3.7: STP tie

Pro oba tyto spoje platí logicky stejná cena cesty a nedá se rozhodnout ani podle switch ID souseda, protože za oběma porty vlastně mám téhož souseda. V tomto případě se použije další kritérium – i na portech totiž máme dvojici údajů: prioritu portu a číslo portu. Priorita portu obvykle bývá číslo 128, takže rozhodující je číslo portu. Prioritu samozřejmě můžeme změnit a tím ovlivnit určení aktivního portu.

Příklad

Na obrázku 3.7 máme switche S1 a S2. Podle LED světel můžeme usoudit, že kořenovým switchem se stal S2 a spoj mezi porty g0/2 na obou switchích bude blokovaný. Výpis základního nastavení STP na switchi S1:

```
S1#show spanning-tree
VLAN0001
  Spanning tree enabled protocol ieee
  Root ID     Priority    32769
              Address     0007.EC99.A154
              Cost        4
              Port        25(GigabitEthernet0/1)
              Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID   Priority    32769  (priority 32768 sys-id-ext 1)
              Address     000D.BDC5.2941
              Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec
              Aging Time  20

Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Gi0/1            Root FWD 4         128.25   P2p
Gi0/2            Altn BLK 4         128.26   P2p
```

V části výpisu Bridge ID vidíme prioritu a MAC adresu switche S1, nad tím v části výpisu Root ID jsou tyto údaje o kořenovém switchi (každý switch musí znát „svého roota“). V částech Root ID a Bridge ID jsou různé MAC adresy, z toho plyne, že S1 není kořenovým switchem. Dole je pak tabulka se všemi používanými porty, ke každému vidíme roli, stav, cenu spoje, prioritu portu s číslem portu a typ linky.

Port g0/1 je root port, tj. vede ke kořenovému switchi, a je ve stavu forwarding (takže předává veškerý provoz, běžně funguje). Port g0/2 má roli Altn, což je „alternated“, neboli je blokovaný (právě u něj je oranžové světlo). Čísla těchto portů nejsou 1 a 2, ale 25 a 26 (předposlední sloupec) – určují se globálně jednoznačně za celý switch, nižší čísla mají fast-ethernetové porty f0/1–f0/24.

Tentýž výpis na switchi S2:

```
S2#show spanning-tree
VLAN0001
  Spanning tree enabled protocol ieee
```

```
  Root ID    Priority    32769
             Address     0007.EC99.A154
             This bridge is the root
             Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec

  Bridge ID  Priority    32769  (priority 32768 sys-id-ext 1)
             Address     0007.EC99.A154
             Hello Time  2 sec  Max Age 20 sec  Forward Delay 15 sec
             Aging Time  20

Interface        Role Sts Cost      Prio.Nbr Type
---------------- ---- --- --------- -------- --------------------------------
Gi0/1            Desg FWD 4         128.25   P2p
Gi0/2            Desg FWD 4         128.26   P2p
```

Oba switche mají stejnou prioritu (32769), liší se adresou. MAC adresa switche S2 je menší číslo, tedy S2 se stává rootem.

Porty g0/1 a g0/2 jsou oba v roli designated a oba předávají, což se může zdát zvláštní, když vlastně protějšek toho druhého je blokovaný. Jenže toto je běžný případ – blokovaná je jen jedna strana linky, aby v případě nutnosti bylo možné ji co nejrychleji aktivovat. Takže na straně nadřízeného switche je port v roli designated ve stavu přeposílání, na druhé straně je port v roli alternate ve stavu blokovaný. Běžný provoz neprojde (je zahozen na blokovaném portu), takže smyčka se nevytvoří.

Aby to vše mohlo fungovat, switche se navzájem domlouvají pomocí speciálních rámců, které označujeme *BPDU* (Bridge PDU). Slouží k údržbě a případné aktualizaci informací o síti potřebných pro STP, posílají se na multicast adresu označující všechna zařízení se spuštěným STP (`01:80:C2:00:00:00`). BPDU se odesílají pravidelně každé 2 sekundy, a pak při každé změně sítě switchů.

Součástí BPDU je kromě jiného informace o tom, že se jedná o rámec STP a jaké verze, který switch je kořenový (jeho bridge ID/switch ID), dále ID odesílajícího switche, údaje o době platnosti rámce, cena cesty od odesílatele ke kořenovému switchi, atd. Tyto údaje jsme viděli v příkladu na výpisech. BPDU se zapouzdřuje do LLC rámce podle IEEE 802.2 a následně do MAC rámce IEEE 802.3.

https://wiki.wireshark.org/STP

Pokud se do sítě switchů začleňuje nový switch, musí se také zapojit do STP stromu. Nový switch nejdřív sám sebe považuje za kořenový switch, ale své porty zatím nechává ve stavu naslouchání (přijímá BPDU, ale nic neodesílá). Pokud v přijatém BPDU najde ID kořenového switche, které je nižší než jeho ID, přestane se považovat za kořenový switch a postupně se zařadí do struktury (pro každý port si z přijatých BPDU vypočte cenu cesty k ostatním switchům a pro každý switch si vybere jen jeden port, ostatní blokuje).

3.5.3 Varianty protokolu STP

Původní protokol STP byl standardizován jako IEEE 802.1D.

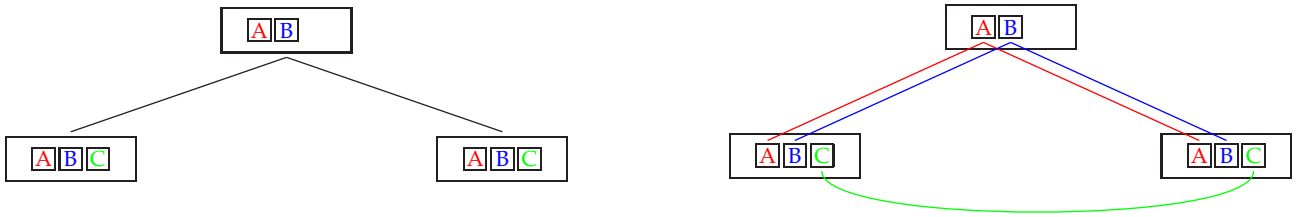
RSTP. Protože u rychlejších standardů přestala původní specifikace stačit, byl vytvořen nový dodatek standardu nazvaný RSTP (Rapid STP) jako IEEE 802.1w, nicméně nyní je zahrnut do revize IEEE 802.1D-2004 (takže původní STP a novější RSTP splynuly v jednom standardu, třebaže někteří výrobci v konfiguraci rozlišují původní STP a novější RSTP).

Zatímco u původního STP trval proces konvergence 50 sekund (po tuto dobu byla celá síť nefunkční, což je strašně dlouho), u novějšího RSTP to trvá kolem 1 sekundy, i méně. Mezi původním STP a RSTP jsou sice i jiné rozdíly, ale tento rozdíl je nejvýraznější.

MSTP. Varianta MSTP (Multiple STP) byla standardizována jako dodatek IEEE 802.1s, také se označuje jako MST (Multiple Spanning Tree). Účelem je zefektivnit činnost protokolu STP pro případ, že jsou v síti definovány virtuální sítě (VLAN). Taktéž byla dodatečně přidána do standardu, ale tentokrát IEEE 802.1Q, a to jako revize IEEE 802.1Q-2005 jako rozšíření RSTP pro VLAN sítě.

V principu jde o to, aby pro každou VLAN nebo skupinu VLAN mohla (nemusela) existovat samostatná instance protokolu STP, tedy pro každou VLAN (či několik) by mohla být trochu jiná struktura aktivních linek a taky jiný kořenový switch. Účelem je jak zefektivnění komunikace v rámci jednotlivých VLAN (aby například zbytečně nebyla vedena komunikace přes takové switche, které s dotyčnou VLAN nemají co do činění), ale také možnost vyrovnání zátěže v síti (určitý port bude pro jednu VLAN blokován a pro jinou aktivní, u jiného portu to může být naopak, každá bude využívat jiné cesty).

Pokud bychom v prostředí s VLAN sítěmi použili pouze STP, je možné, že bychom některé VLANy na určitých spojích prakticky vyřadili z činnosti nebo bychom některé z linek zbytečně zatížili příliš. Na obrázku 3.8 vidíme podobný problém: VLAN C (zelená) je aktivní jen na spodních dvou switchích. Pokud bychom použili klasický STP bez podpory VLAN, zbytečně bychom rámcům z VLAN C prodloužili cestu. Jestliže ale použijeme MSTP, můžeme pro zelenou VLAN určit jiný kořenový switch než pro ostatní, třeba jeden ze spodních switchů, a spoj mezi nimi necháme aktivní (jen pro tuto VLAN).



Obrázek 3.8: Přerušená cesta pro jednu VLAN při použití STP

Další informace:

http://www.samuraj-cz.com/clanek/cisco-ios-10-rapid-spanning-tree-protocol/

Kromě standardizovaných variant existují i proprietární varianty vytvořené různými výrobci. Se standardy to je totiž tak, že nepřicházejí tak rychle, jak by si komerční prostředí představovalo, a tak se často stává, že nejdřív výrobce používá svůj vlastní proprietární protokol (pouze pro zařízení jeho značky), a až se objeví standard, doplní ho do specifikace, takže zařízení „umí“ více různých vzájemně alternativních protokolů.

Například u Cisca se můžeme setkat s protokolem PVST+, který do původního IEEE 802.1D přidává podporu VLAN (pro každou VLAN vlastní STP instance), a Rapid PVST+, který zároveň s podporou VLAN urychluje konvergenci jako v RSTP. Ale pozor – Rapid PVST+ a MSTP nemají úplně stejné vlastnosti: zatímco Rapid PVST+ opravdu vytváří pro naprosto každou VLAN samostatnou STP instanci v vlastním kořenovým switchem apod., MSTP dovoluje vytvořit více než jednu instanci, ale nevynucuje si vytvoření tolik instancí, kolik je VLAN (může být v jedné instanci i více než jedna VLAN), což v případě velkého množství VLAN šetří výpočetní a přenosové kapacity sítě switchů.

3.6 EtherChannel

3.6.1 Vlastnosti

EtherChannel je technologie umožňující spojit (agregovat) až 8 fyzických ethernetových linek do jedné. Dva uzly (typicky servery, přepínače nebo směrovače podporující tuto technologii) jsou propojeny více aktivními fyzickými ethernetovými spoji, PDU procházejí některým z těchto spojů. Dále je podporováno navíc až 8 záložních fyzických linek (failover) pro případ, že aktivní linky budou nefunkční.

Poznámka:

V případě této technologie se používá dvojí terminologie – zatímco u firmy Cisco se mluví o EtherChannel, v systému Solaris se pro totéž používá pojem *trunk*, což ale u Cisco znamená něco úplně jiného. Proto když slyšíme pojem „trunk“, měli bychom mít jasno v tom, čí terminologie je vlastně právě používána. U serverů se také používá pojem *NIC Teaming*, což však není úplně totéž (je to širší termín, sdružování síťových karet do jedné „virtuální“, logické).

Spoje, které jsou součástí EtherChannelu, jsou typu Fast Ethernet nebo rychlejší, a všechny musejí být stejného typu (tj. například nelze kombinovat spoje Fast Ethernet a 10Gb Ethernet – stejná rychlost, dále stejný duplex apod.). Pokud používáme VLAN (virtuální sítě), měly by spoje být v rámci jedné VLAN nebo v trunk módu se stejnými parametry.

Celková propustnost se však nerovná součtu propustností jednotlivých fyzických cest. Pokud se v rámci jedné komunikace směřující k jednoho uzlu k jinému uzlu (ve výchozím nastavení) posílá více PDU, všechny jsou posílány po téže fyzické cestě, tedy komunikaci nelze rozdělit.

3.6.2 Řízení toku

Výchozí nastavení je takové, že spoje v rámci EtherChannelu jsou rozdělovány podle cílové MAC adresy, tedy PDU, které mají konkrétní cílovou MAC adresu, jsou všechny směrovány na tentýž spoj. To však ne vždy vyhovuje, proto je možné nastavit i jiné dělení:

- dvojici [zdrojová MAC, cílová MAC],
- pouze podle zdrojové MAC,
- podobně pro IP adresy a porty.

Rozdělení zátěže (tj. jednotlivých spojení) mezi linky v rámci EtherChannelu se provádí podle hashovacího vyvažovacího algoritmu, který třídí PDU podle zadaného kritéria (například podle cílové MAC adresy) a celou komunikaci dělí podle pravidla naznačeného v tabulce 3.4.

Každému „proudu dat“ určenému podle zvoleného kritéria (například podle cílové adresy) je hashováním přiděleno číslo z intervalu 0–7 (tj. jedno z 8 čísel), a to bez ohledu na skutečný počet spojů v EtherChannelu. Toto číslo pak určuje konkrétní fyzický spoj.

Například pokud máme EtherChannel nad třemi fyzickými spojeními (EC porty), čemuž odpovídá předposlední sloupec tabulky

Rozdělení komunikačních proudů mezi EC porty \ Počet EC portů	8	7	6	5	4	3	2
	1	2	2	2	2	3	4
	1	1	2	2	2	3	4
	1	1	1	2	2	2	
	1	1	1	1	2		
	1	1	1	1			
	1	1	1				
	1	1					
	1						

Tabulka 3.4: Hashování komunikace v EtherChannelu

vpravo, budou na první spoj směrovány PDU se třemi konkrétními přiřazenými čísly (0, 1, 2), jiná tři čísla (3, 4, 5) budou směrována na druhý spoj a zbylá dvě (6, 7) na třetí spoj. Je zřejmé, že nejoptimálněji budou spoje vyváženy, pokud do EtherChannelu sdružíme 2, 4 nebo 8 spojů.

3.6.3 Vytvoření kanálu

Pro EtherChannel (EC) existují dva *protokoly vyjednávající vlastnosti* EC spojení:

- *LACP* (Link Aggregation Control Protocol) definovaný IEEE 802.3ax[1],
- *PAgP* (Port Aggregation Protocol), což je proprietární protokol firmy Cisco.

První z těchto protokolů je obecně použitelný ve všech zařízeních podporujících EtherChannel (včetně zařízení společnosti Cisco), druhý se používá pouze při propojování dvou zařízení Cisco. V jedné EC síti musí být použit vždy jen jediný z těchto protokolů (a všechna zařízení mu musí „rozumět“).

Oba protokoly umožňují nastavit zařízení do jednoho ze dvou stavů – aktivního (*active* u LACP, *desirable* u PAgP), ve kterém zařízení aktivně vyjednává sestavení EtherChannelu, a pasivního (*passive* u LACP, *auto* u PAgP), ve kterém zařízení pasivně vyčkává na iniciaci vyjednávání o EtherChannelu od druhé strany. Ve dvojici by mělo být každé zařízení v jiném z těchto stavů.

EtherChannel lze také vytvořit manuálně napevno, stav je označován *on*. Pak je EtherChannel vytvořen námi a žádné vyjednávání zařízení není potřeba a tedy se vlastně nepoužije žádný z předchozích dvou protokolů.

Protokoly se nemohou míchat.

- Pokud zvolíme protokol LACP, musí být porty na jedné straně spoje v módu active (všechny porty ve svazku), na druhé straně je u portů přípustný buď passive nebo taky active, jinak se neaktivuje EtherChannel.
- Pokud zvolíme protokol PAgP, musí být port na jedné straně nastaven do módu desirable, na druhé straně je přípustný buď auto, nebo taky desirable (jinak spoj nebude fungovat).
- Pokud jsou porty ve svazku na jednom zařízení nastaveny do módu on, na druhé straně spoje taky musí být on (tato situace znamená, že jsme pro vyjednání kanálu nepoužili žádný protokol, ale ručně jsme určili, že má jít o EtherChannel).

Porty na prvním zařízení	**Porty na druhém zařízení**	**⇒ vybrali jsme protokol**
active	passive nebo active	LACP
desirable	auto nebo desirable	PAgP
on	on	žádný

Tabulka 3.5: Módy pro vyjednání EtherChannelu

Sebemenší chybička nebo nekonzistence způsobí, že EtherChannel nebude fungovat. Musíme si pohlídat jak rychlost, duplex a správný mód, tak i například nastavení VLAN. Pokud jde o přístupové porty (což je pro spoj mezi dvěma switchi méně běžné), musí být na všech portech ve svazku a na obou stranách nastaveno totéž číslo VLAN. Pokud jde o trunkové porty (přeposílající rámce podle IEEE

[1] Správně je IEEE 802.3ax, ale v literatuře se můžeme setkat s nesprávným (významově posunutým) IEEE 802.3ad, což je NIC Teaming.

802.1Q, to je u spoje mezi switchi nejběžnější), musí být trunk na obou stranách a taktéž musí sedět seznam povolených VLAN, které mohou do trunku být poslány.

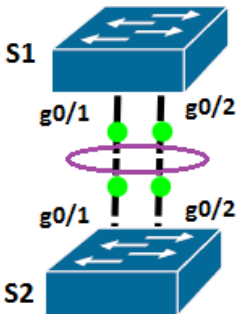
Poznámka:

Uvědomte si, že se nám tady míchají dva protichůdné mechanismy: zatímco STP zajišťuje, že pokud je mezi dvěma zařízeními více než jedna fyzická linka, bude aktivní jenom jedna z nich, kdežto EtherChannel potřebuje, aby mezi dvěma zařízeními vedlo více aktivních fyzických linek (které na logické úrovni budou chápány jako jediná). Takže pokud propojíme dvě zařízení více než dvěma kabely, okamžitě najede STP (ten je ve výchozím stavu zapnutý), a až po správné konfiguraci EtherChannel budou opravdu aktivní všechny fyzické linky.

Postup

Obrázek 3.9 je podobný obrázku 3.7 s tím rozdílem, že zelená LED je na všech portech, tedy opravdu všechny porty mohou přeposílat, všechny fyzické linky jsou aktivní a STP nám tu nic neblokuje.

Podíváme se na zkrácený výpis nastavení etherchannelu na switchi S1:



Obrázek 3.9: Etherchannel mezi dvěma switchi

```
S1#show interface etherchannel
GigabitEthernet0/1:
Port state    = 1
Channel group = 1      Mode = Active      Gcchange = -
Port-channel  = Po1    GC   =   -         Pseudo port-channel = Po1
Port index    = 0      Load = 0x00        Protocol =   LACP
... (zkráceno)

GigabitEthernet0/2:
Port state    = 1
Channel group = 1      Mode = Active      Gcchange = -
Port-channel  = Po1    GC   =   -         Pseudo port-channel = Po1
Port index    = 0      Load = 0x00        Protocol =   LACP
... (zkráceno)
```

Takže na switchi S1 jsme na portech g0/1 a g0/2 nastavili mód active (tj. vybrali jsme protokol LACP pro dojednání parametrů), takže na portech druhého switche musí být passive nebo active (lepší je passive, ať se „nehádají“, kdo začne vyjednávat). Tentýž výpis na switchi S2:

```
S2#show interface etherchannel
GigabitEthernet0/1:
Port state    = 1
Channel group = 1      Mode = Passive     Gcchange = -
Port-channel  = Po1    GC   =   -         Pseudo port-channel = Po1
Port index    = 0      Load = 0x00        Protocol =   LACP
... (zkráceno)

GigabitEthernet0/2:
Port state    = 1
Channel group = 1      Mode = Passive     Gcchange = -
Port-channel  = Po1    GC   =   -         Pseudo port-channel = Po1
Port index    = 0      Load = 0x00        Protocol =   LACP
... (zkráceno)
```

Na switchi S1 jsme postupovali takto:

```
S1>enable
S1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
S1(config)#vlan 10
... (vytvoříme všechny VLAN, které chceme přenášet)

S1(config)#interface range g0/1-2
S1(config-if-range)#shutdown
S1(config-if-range)#channel-group 1 mode active
S1(config-if-range)#interface port-channel 1
S1(config-if)#switchport mode trunk
S1(config-if)#switchport trunk native vlan 98
S1(config-if)#switchport trunk allowed vlan 10,20,30,98,99
S1(config)#interface range g0/1-2
S1(config-if-range)#no shutdown
```

Pokud to je možné, je dobré konfigurovat vypnuté porty. Abychom měli jistotu, že všechny porty v kanálu jsou nastaveny naprosto stejně, nejdřív vytvoříme etherchannel a pak nastavujeme (třeba VLAN trunk) na tomto kanálu.

Etherchannel sice podle názvu vypadá, že by se měl vztahovat k vrstvě L2 a Ethernetu, ale ve skutečnosti lze vytvořit Etherchannel kanály i na vrstvě L3.

Další informace:

- http://www.samuraj-cz.com/clanek/cisco-ios-21-etherchannel-link-agregation-pagp-lacp-nic-teaming/
- http://www.fit.vutbr.cz/~ivesely/publikace/etherchannel.pdf
- http://www.cisco.com/en/US/tech/tk389/tk213/technologies_tech_note09186a0080094714.shtml
- http://www.ciscozine.com/2008/11/04/configuring-link-aggregation-with-etherchannel/

3.7 SAN sítě

SAN (Storage Area Network) jsou sítě určené k propojení datových úložišť (diskových polí, NAS, souborových serverů apod.) s aktivními prvky sítě, jde tedy o zajištění vysoké dostupnosti úložišť. K nejznámějším SAN technologiím patří Fibre Channel (FC) a iSCSI.

3.7.1 Fibre Channel

Fibre Channel je převážně optické rozhraní (tj. na optických kabelech) pro rychlé přesuny dat na (relativně) krátké vzdálenosti, které pracuje na stejných vrstvách jako Ethernet, většina logiky technologie je v MAC podvrstvě. Původně bylo určeno pro použití „uvnitř počítače“ (konkrétněji serveru) ke komunikaci mezi I/O zařízeními a procesory, ale postupně se prosadilo jako síťový standard. Pod pojmem *kanál* (channel) se zde rozumí přímé nebo přepínané predikovatelné propojení dvou zařízení.

Jde o plně duplexní sériové rozhraní pro point-to-point spoje (i více point-to-point spojů s vhodným hardwarem, přepínačem). Přepínání je zde velmi jednoduché a rychlé, bývá realizováno hardwarově. Výhodou je tedy vysoká rychlost přepínání a tedy i dostupnosti dat.

Podle názvu by se mohlo zdát, že jako přenosový prostředek je možné použít pouze optický kabel, ale ve skutečnosti je možné použít také koaxiál nebo STP (stíněnou dvojlinku), což ale má vliv na dosažitelné vzdálenosti a rychlost. Zatímco s jednovidovým optickým vláknem dosáhneme rychlosti až 1 Gb/s a vzdálenosti až 10 km (s mnohavidovými vlákny samozřejmě méně, vzdálenost je od uzlu k přepínači), při použití STP to je 100 m při rychlosti 132 Mb/s nebo 50 m při 265 Mb/s. Nejnovější standard umožňuje dosažení rychlosti 4 Gb/s.

Levnější a jednodušší variantou je *Fibre Channel se smyčkou*, jehož topologie je kruhová bez použití mezilehlých prvků, přístupová metoda je podobná využívání tokenu. Toto řešení je vhodné jen pro malý počet zařízení, maximálně 30.

FCoE (Fibre Channel over Ethernet) je pokus o sloučení jinak rozdílných sítí Fibre Channel a Ethernet (10Gb). Zatímco samotný FC dokáže pracovat jen v rámci jednoho ethernetového segmentu (k nejbližšímu směrovači), FCoE díky napojení na technologii Ethernetu se dostane i za směrovače. FCoE vyžaduje podporu Jumbogramů (Jumboframe).

3.7.2 iSCSI

iSCSI (Internet Small Computer System Interface) je obdobou FCoE určenou spíše pro menší SAN sítě. Jedná se vlastně o možnost posílání SCSI příkazů přes IP síť. Klient (iniciator) potřebuje přes síť přistupovat k serveru (disku), z pohledu klienta je funkčnost stejná jako kdyby šlo o interní disk v počítači (typu SCSI, SAS apod.), jen nízkoúrovňové příkazy jsou posílány na síť. Na serverové straně jsou tyto příkazy rozbaleny z paketu a předány příslušnému úložišti.

Zatímco FC pracuje na ethernetových vrstvách ISO/OSI, iSCSI pracuje až na síťové vrstvě.

Technologie iSCSI může být implementována softwarově nebo hardwarově. Softwarová implementace je levnější (konkrétně – zdarma), příslušný software si lze stáhnout od výrobce/distributora každého operačního systému nebo je dokonce součástí instalace systému (jen je obvykle nutné aktivovat příslušnou službu či démona).

Hardwarová implementace znamená podporu na NIC (za tu se připlácí). Výhodou hardwarové implementace je vyšší rychlost, protože pomocný procesor na NIC odlehčuje hlavním procesorům při zpracovávání (nejen) iSCSI požadavků. Obecně platí, že do menší SAN stačí softwarová implementace, do větší nebo hodně vytížené SAN je dobré investovat do síťových karet s hardwarovou implementací iSCSI (samozřejmě pokud máme SCSI disky).

Další informace:

- Článek o FCoE (první díl seriálu): http://www.abclinuxu.cz/clanky/hardware/fcoe-fibre-channel-over-ethernet
- Článek o iSCSI: https://www.samuraj-cz.com/clanek/iscsi-san-sit-a-konfigurace-na-windows-server-2012/

Další možná řešení SAN sítí jsou InfiniBand, ATAoE (ATA over Ethernet), apod.

Kapitola 4

Rozlehlé sítě a telekomunikace

WAN (Wide Area Network) je rozlehlá síť zabírající obvykle rozlohu státu, kontinentu či celého světa. Propojuje nikoliv koncová zařízení, ale různé sítě typu LAN, MAN či menší WAN sítě.

4.1 WAN sítě

4.1.1 Struktura WAN sítí

✎ Jakou topologii vlastně používají WAN sítě? Záleží, jaký pohled použijeme.

- WAN sítě jsou navzájem uspořádány do (převážně) hierarchické struktury velmi podobné hierarchii podle protokolu IP. Souvisí to s tím, že lokální a regionální poskytovatelé konektivity tuto konektivitu také odněkud musejí získat, od velkých nadnárodních poskytovatelů. Celková struktura má charakter *páteřní sítě*.
- Když se podíváme na vnitřní strukturu WAN sítě, obvykle zjistíme, že používá architekturu *mesh*: redundantní spoje, které zajišťují vysokou dostupnost, odolnost proti výpadku.

✎ Zatímco u lokálních sítí víceméně platí, že data po těchto sítích posílaná souvisejí s vlastníkem sítě, u MAN a zejména WAN sítí tomu tak není. Provozovatelé WAN sítí jsou prostě majiteli síťové infrastruktury, kterou pronajímají svým zákazníkům, je to páteř, na kterou jsou napojeny zákaznické sítě.

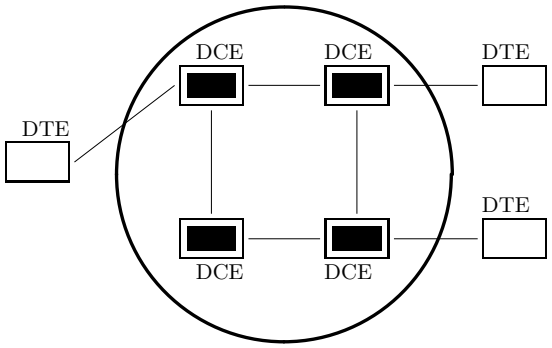
Síťová konektivita (případně v určité definované kvalitě) je jedinou službou, kterou WAN poskytují, funkčnost je podřízena také tomu, aby bylo možné poskytnuté služby tarifikovat. Dokonce může být problém se směrováním na jiné než unicast adresy, protože pro WAN je typická spojovaná komunikace přes virtuální okruhy.

✎ Ve WAN sítích obvykle rozlišujeme tyto typy zařízení:

- *DCE (Data Circuit Equipment)* jsou zařízení uvnitř WAN sítě (jádro). Obvykle přímo nekomunikují s ničím, co by do WAN nepatřilo, prostě dokážou přes sebe navázat komunikační okruh (circuit) a pak přes něj přenášet data. Pro zákazníka jsou „neviditelné“. Jejich typickou vlastností je rychlost přepínání dat (obvykle s hardwarovou podporou), je to obdoba switchů v LAN sítích. DCE se nezdržují prohlížením přenášených PDU, zajímá je jen „lokalizační část“ záhlaví.
- *DTE (Data Terminal Equipment)* jsou zařízení na rozhraní mezi WAN sítí a tím, co je k ní připojeno, jsou to tedy hraniční zařízení. Potřebují „rozumět“ jak protokolům pracujícím ve WAN,

tak i protokolům z připojené sítě (vzhledem k připojené síti obvykle potřebují funkcionalitu vrstvy L3, IP). Pracují jako překladatelé mezi dvěma druhy sítí.

Některé WAN sítě rozlišují i další typy svých zařízení, ale nám budou tyto pojmy stačit.



Obrázek 4.1: Zařízení v síti Frame Relay

Poznámka:

V některých WAN se používá trochu jiná terminologie:

- zařízení P (Provider) odpovídají zařízením DCE, jsou tedy uvnitř WAN sítě,
- zařízení PE (Provider Edge – na hranici poskytovatele) jsou na hranicích WAN sítě,
- zařízení CE (Customer Edge – na hranici zákazníka) jsou na straně zákazníka a zprostředkovávají pro něj přístup k WAN síti.

Zařízení typu PE a CE tedy mohou dohromady tvořit DTE.

4.1.2 Protokoly vrstvy L2 pro WAN sítě

Pro protokoly pracující na vrstvě L2 se vžil název *spojové protokoly*. Kromě těch, které jsme probírali dříve, zde najdeme také protokoly pro WAN sítě (protože většina WAN sítí je implementována především na této vrstvě).

Obvykle platí, že každá WAN síť má „svůj“ spojový protokol, nicméně u těchto protokolů můžeme vysledovat určité společné vlastnosti (protože ve skutečnosti jsou mezi nimi vztahy předek-potomek).

Uzly v síti a přenosové režimy. Spojové protokoly obvykle rozlišují primární uzel (začíná komunikaci, určuje parametry spojení) a sekundární uzly (pouze odpovídají). Pokud se jedná o point-to-point komunikaci, pak máme jeden primární a jeden sekundární uzel, když je to point-to-multipoint komunikace, je třeba jeden z uzlů prohlásit za „nejdůležitější“ – primární.

Uzly mezi sebou komunikují v konkrétním režimu:

- *normální režim odpovědi* (NRM, normal response mode) – sekundární uzel musí počkat na vyzvání od primárního, aby mohl komunikovat,
- *asynchronní režim odpovědi* (ARM, asynchronous response mode) – sekundární uzel může začít komunikovat s primárním bez povolení,

- *asynchronní vyvážený režim* (ABM, asynchronous balanced mode) – každý uzel může začít komunikaci, ale zároveň se dynamicky určuje, kdo je právě primárním uzlem; kdo začne komunikaci, stává se primárním uzlem.

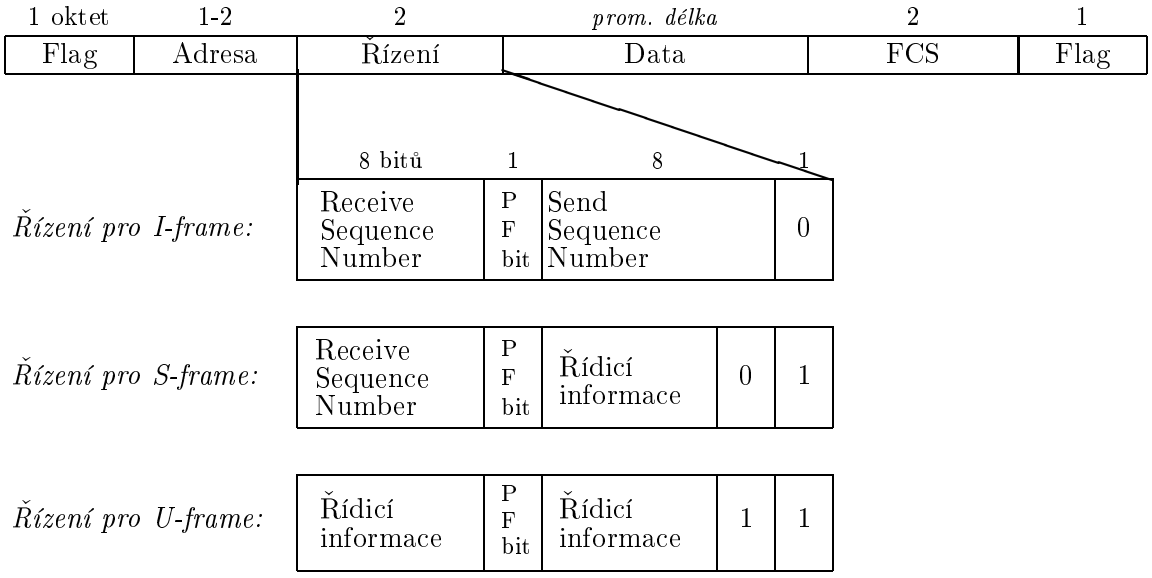
Novější spojové protokoly typicky fungují v režimu ABM.

Rámec spojového protokolu. Rozlišujeme několik typů rámců:

- *I-frame* (Information Frame, informační, datový) – nese informace z vyšší vrstvy a případně řídicí informace, tyto rámce podporují řazení, řízení toku, detekci chyb, zotavení. Určení: přenos dat.
- *S-frame* (Supervisory Frame, také služební či dohlížecí rámec) – obsahuje řídicí informace, je používán pro požadavek zahájení nebo ukončení spojení, hlášení o stavu, potvrzení přijetí I-rámce.
- *U-frame* (Unnumbered Frame, nečíslovaný) – pro řídicí informace (dohoda režimu provozu), na rozdíl od předchozího nepodporují číslování rámců do posloupnosti, a může také obsahovat data z vyšší vrstvy. Obecně: pro to, co se vejde do jediného rámce.

Rámce typu I-frame používají čísla Sequence Number pro oba směry (podobný mechanismus jako u TCP), S-frame mají čísla pouze pro jeden směr (přenášený obsah má totiž smysl pouze pro jeden ze směrů komunikace), U-frame nepoužívá žádná taková čísla (je nečíslovaný, nedělí komunikaci na části, které by bylo třeba kompletovat).

Čím méně místa zabírají čísla Sequence Number, tím více místa zbývá na speciální významové bity (typ rámce, určení konkrétního požadavku nebo odpovědi na požadavek, atd.).



Obrázek 4.2: Typický formát WAN rámce, zde pro protokol HDLC

Obecná (zjednodušená) struktura WAN rámce je tato:

- flag (příznak) na začátku rámce – synchronizační informace ve tvaru `01111110`,
- adresa – formát podle toho, jak se v konkrétní WAN adresují okruhy,
- pole pro řízení – Sequence Numbers a speciální významové bity podle typu rámce,
- data (pokud jsou v tomto rámci přenášena),
- FCS – kontrolní součet,
- flag stejně jako na začátku rámce.

Jak bylo výše naznačeno, spojové protokoly pro WAN sítě se vlastně vyvíjely jeden z druhého. Prapředkem těchto protokolů je SDLC od společnosti IBM, jeho nástupcem byl HDLC, následoval LAPB používaný v sítích X.25, další byl LAPD pro ISBN, pak LAPF z Frame Relay a další. O něco známější je protokol PPP a jeho odvozeniny.

Protokol IEEE 802.2 (LLC) známe už z kapitoly o Ethernetu, také se jedná o spojový protokol, takže se k němu krátce vrátíme i zde. Je určen pro point-to-point spoje, pracuje v asynchronním vyváženém režimu (ABM). Opět se rozlišují tři typy rámců – I-frame, S-frame a U-frame. Formát rámce je celkově jednodušší, protože LLC rámec se vždy zapouzdřuje do MAC rámce. Používá se jiná adresace (přístupové body služeb).

LLC nabízí tři typy služeb:

- *nepotvrzovaná služba bez spojení* (typ 1) – nejpoužívanější, funguje podobně jako doručování datagramů (paket je vybaven všemi dostupnými informacemi včetně adres zdroje a cíle a pořadí paketu ve zprávě), žádné ošetření chyb, v tomto ohledu spoléhá na protokoly vyšších vrstev,
- *služba se spojením* (typ 2) – pro virtuální okruhy, zahrnuje navázání spojení, potvrzování po doručení omezené skupiny paketů, atd., pouze první paket obsahuje informace o adresách,
- *potvrzovaná služba bez spojení* (typ 3) – pakety jsou po skupinách potvrzovány, tato služba je jen málo využívaná.

Koncové stanice podporují vždy nejméně typ 1, a pak obvykle ještě jednu ze zbývajících služeb.

4.2 Nejznámější WAN sítě

4.2.1 Frame Relay

Specifikace Frame Relay původně vznikla v rámci specifikace ISDN, ale protože se ukázala dobrá životaschopnost tohoto řešení, v r. 1989 se osamostatnila.

Obecně platí, že typickým použitím Frame Relay je z principu propojování lokálních sítí, implementace páteřní sítě. Počítá se s přenosem po kvalitním a rychlém vedení, tedy Frame Relay poskytuje sice službu se spojením, ale nespolehlivou (poskytuje detekci chyb, ale bez možnosti opravy).

V síti Frame Relay se nacházejí tato zařízení:

- *DTE* –vlastníkem může být zákazník nebo jde o pronájem, jsou to vpodstatě hraniční zařízení.
- *DCE* (Data circuit-terminating equipment) – poskytují služby synchronizace a přepínání provozu (obvykle paketové přepínače), vnitřní zařízení sítě.

U obou existuje komponenta fyzické vrstvy a komponenta linkové (spojové) vrstvy. Na fyzické jde většinou o některé sériové rozhraní, na linkové pracuje protokol LAPF.

FRAD (Frame Relay Access Device, Frame Relay Assembler/Disassembler) je zařízení, které slouží k přístupu do sítě Frame Relay. Může to být buď samostatné zařízení, anebo je to součást směrovače, přes který je lokální síť připojena do Frame Relay sítě. FRAD je tedy představitel DTE nebo je součástí DTE.

Komunikuje se po virtuálních okruzích, obvykle přepínaných (SVC). Pro SVC se používají operace navázání spojení (vytvoření okruhu), přenos dat, Idle a ukončení spojení (zrušení okruhu). U PVC se používá jen operace přenosu dat a Idle.

Garance informačního toku. *Parametr CIR* (Committed Information Rate, zaručený informační tok) určuje propustnost, kterou má spoj zaručovat. Hodnota CIR je pro okruhy PVC dohodnuta mezi poskytovatelem (telekomunikační společností) a uživatelem (vlastníkem lokální sítě, která má být připojena), pro okruhy SVC je dohodnuta při navazování spojení. Rámce posílané nad hodnotu CIR mohou být při větším zatížení sítě zahazovány.

Pro uživatele má hodnota CIR tento význam:

- čím vyšší CIR, tím je služba dražší,
- čím menší CIR, tím větší pravděpodobnost zahazování paketů a potenciálně pomalejší přenos.

Nárazový informační tok (také EIR, Excess Information Rate) představuje maximální informační tok, který dokáže přenosová cesta zvládnout (přesněji to, co je navíc nad CIR).

Rámce zasílané nad hodnotu CIR, které nepřekračují hranici EIR, jsou označeny jako *vhodné k zahození* (DE, Discard-Eligible). Jsou předávány přepínači Frame Relay (tj. do FR sítě) tak dlouho, dokud není k dispozici dostatečná šířka pásma (dokud se nevejdou do CIR). Označení rámce jako vhodného k zahození obvykle neznamená, že rámec nebude doručen, ale že se jeho doručení může (i dost hodně) zdržet.

SLA (Service Level Agreement) = dohoda o úrovni poskytovaných služeb (tento pojem se používá obecně u tohoto typu služeb, nejen pro Frame Relay). Jedná se o dohodu o parametrech služby, zde také zahrnuje CIR a EIR.

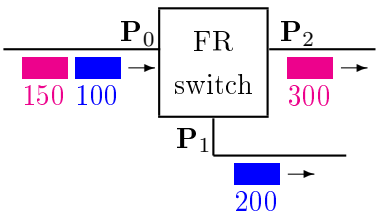
Adresace a přepínací tabulky. Po navázání okruhu se používá lokální adresace pomocí *čísel DLCI* (Data Link Connection Identifier). Ve skutečnosti nejde o identifikátory zařízení, jsou to identifikátory okruhů. Hodnoty DLCI pro DTE v naší LAN získáme od poskytovatele služby Frame Relay (typicky telekomunikační společnost), samozřejmě pokud si FremeRelay implementujeme sami, tak si DLCI taky sami přidělíme.

V celé síti WAN (propojující různé lokální sítě) však může existovat více okruhů se stejným DLCI, navíc se hodnota DLCI při průchodu přes přepínače mění, na každém konci mívá okruh jiné DLCI.

Některé hodnoty DLCI jsou vyhrazeny pro účely signalizace stavů, problémů apod., především z rozsahu 0–16. Dále rozmezí 1019–1022 je určeno pro skupinové vysílání.

Na jednotlivých zařízeních jsou rámce přepínány podle *přepínacích tabulek*. Uvnitř FR sítě najdeme FR switche (= DCE), jejich tabulka je jednoduchá, jak vidíme na obrázku 4.3 (případně by byl ještě další sloupec – informace o tom, zda je daná cesta aktivní). Přepínání je implementováno v hardwaru.

	Přijato z		**Přepnout na**		
	IN_Port	IN_DLCI	OUT_Port	OUT_DLCI	
■	P_0	100	P_1	200	■
■	P_0	150	P_2	300	■
	⋮				



Obrázek 4.3: Ukázka jednoduché přepínací tabulky Frame Relay switche

Na hranici sítě jsou FR routery (v roli DTE), které na jedné straně komunikují se zařízeními lokální sítě zákazníka, na druhé straně komunikují s FR switchi. Tato zařízení obsahují dvě tabulky:

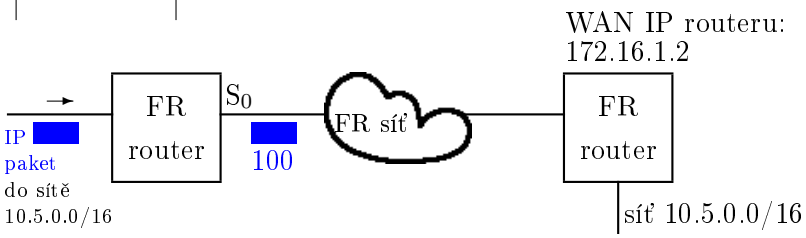
- směrovací tabulku určující, kam se rámec směřující do sítě s danou IP adresou má poslat,
- tabulku mapování DLCI.

Routing Table:

Network	Next Router	Interface
10.5.0.0/16	172.16.1.2	S_0
10.27.0.0/16	172.16.1.8	S_1
⋮		

FR Map:

Next router	DLCI
172.16.1.2	100
172.16.1.8	200
⋮	



Obrázek 4.4: Směrování na FR routeru (DTE), paket přichází do FR sítě

Obsah a význam obou tabulek vidíme na obrázku 4.4. Na třetí vrstvě je paket nasměrován na daný DCE (na druhé straně FR sítě) a příslušné rozhraní, na druhé vrstvě je nastaveno DLCI.

Linková vrstva. Na linkové vrstvě pracuje *protokol LAPF* (Link Access Procedure – Frame). Do (datového) rámce protokolu LAPF se zapouzdřuje IP paket.

Adresové pole má 2 oktety (může být rozšířeno až na 4) a zcela změněnou strukturu, řídicí pole bylo vyřazeno. V adresovém poli je uloženo číslo *DLCI*. V řídícím poli najdeme především bity pro řízení toku dat a kontrolu chyb:

- C/R – command/response bit, určuje, zda jde o příkaz (command) nebo odpověď či reakce na dříve zaslaný příkaz (response),
- *bit FECN* (Forward-explicit congestion notification, dopředné oznámení o přetížení), DCE na cestě oznamuje, že tento rámec byl poslán po přetíženém spojení, a tedy DTE, které je cílem rámců komunikace, by mělo snížit frekvenci příjmu rámců (pokud tak dokáže vyšší vrstva reagovat),
- *bit BECN* (Backward-explicit congestion notification, zpětné oznámení) – oznámení zdroji vysílání o přetížení linky, zdrojové zařízení by mělo snížit objem vysílání,
- *bit DE* (Discard-Eligible) – v rámcích, které mohou být při přetížení zahozeny.

Když DCE zjistí přetížení, nastaví v posílaném rámci bit FECN. Takto přijímající DTE zjistí, že na lince došlo k zahlcení spoje. Naopak když DCE přepíná rámec s nastaveným FECN, v nejbližším rámci posílaném v okruhu v opačném směru nastaví bit BECN. Takto vysílající DTE zjistí, že na lince došlo k zahlcení spoje.

LMI (Local Management Interface) je rozšíření specifikace Frame Relay, které určuje komunikaci mezi zařízeními DTE a DCE, tedy určuje, jakým způsobem se má k FR přistupovat z lokální sítě.

Existují tři LMI protokoly, typ použitého protokolu obvykle určuje DCE:

1. cisco
2. ansi (Annex D)
3. q933a (Annex A)

První používá pro signalizaci DLCI 1023, další dvě používají DLCI 0. Tedy po spojení jsou přenášeny buď datové rámce s DLCI podle cíle, a nebo služební LMI rámce (obvykle jeden ze dvou typů zprávy – dotaz na stav sítě směřující od zákazníka nebo stav sítě od DCE).

Další informace:

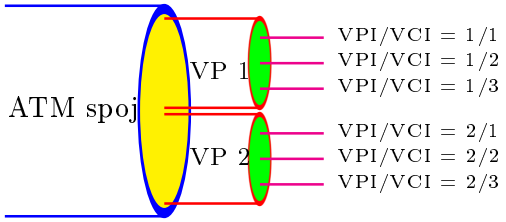
- http://www.cs.vsb.cz/grygarek/TPS/FrameRelay/frame.html
- http://books.google.cz/books?id=tROBDX-OBrEC&printsec=frontcover&dq=frame+relay
- http://www.cs.vsb.cz/grygarek/PS/lect0304/ps1lect11.html

4.2.2 ATM

Sítě ATM (Asynchronous Transfer Mode) byly původně považovány za velmi nadějnou technologii, nicméně v praxi byly postupně vytlačeny jednoduššími a pružnějšími sítěmi MPLS. „Asynchronní“ v názvu neznamená asynchronní přenos, ale nepravidelné zasílání „plných“ buněk na cestě, buňky jsou obsazovány daty podle potřeby, ne podle algoritmu.

ATM pracuje na principu komunikace se spojením, nespolehlivá služba (tj. bez oznámení chyb), na plném duplexu, v statistickém multiplexu. Důležitou vlastností je garance kvality služeb pro různé typy dat – vlastně jednou z nejdůležitějších charakteristik ATM jsou právě široké možnosti řízení kvality služeb.

Adresace okruhů je na rozdíl od FR dvouhodnotová. *Virtuální cesta* a *virtuální kanál* jsou obdobou DLCI. V jedné fyzické přenosové cestě může vést více virtuálních cest, v jedné virtuální cestě vede víc virtuálních kanálů. Číslo virtuální cesty označujeme VPI (Virtual Path Identifier), číslo virtuálního kanálu označujeme VCI (Virtual Channel Identifier). *Hodnota VPI/VCI* plně identifikuje přenosovou cestu mezi dvěma sousedními uzly, na každém ATM přepínači se mění.



Obrázek 4.5: Virtuální cesty a virtuální kanály v ATM (zjednodušeně)

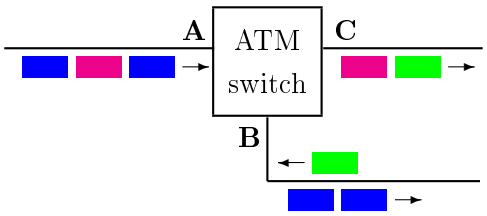
ATM switch si vede *tabulku připojení*, ve které jsou přidruženy příchozí a odchozí VPI/VCI, záznam se tvoří během navázání spojení (vytvoření okruhu).

Na obrázku 4.6 je zjednodušená a zkrácená přepínací tabulka ATM switche. Jsou v ní tři virtuální cesty. Každý řádek obsahuje jednu přepínací informaci, například v prvním řádku zjistíme, že buňka přicházející z portu A po cestě 1 v kanálu 29 je přepnuta na port B cestu 2 kanál 42. Porty jsou označeny písmeny spíše pro snadnější odlišení, většinou se setkáme s označením čísly nebo řetězci, podle výrobce.

PDU se nazývají *buňky*, jsou velmi malé s konstantní délkou 53 oktetů (z důvodu jednoduchého a rychlého přepínání). Aby se do buněk vůbec vešla nějaká data, je záhlaví velmi malé, pouze 5 B, na data zbývá 48 oktetů. V záhlaví máme kromě adres VPI/VCI také například jednobitový příznak CLT (Cell Loss Priority, priorita ztráty buňky, 1 bit), který plní stejnou roli jako u FR bit DE.

Protože se komunikuje v statistickém multiplexu, jsou v přenosu definovány sloty, do kterých se „sázejí“ buňky. Zde je právě výhodou, že buňky mají konstantní velikost odpovídající velikosti slotů.

	Přijato z			Přepnout na			
	Port	VPI	VCI	Port	VPI	VCI	
■	A	1	29	B	2	42	■
■	A	3	42	C	1	8	■
■	B	1	18	C	2	36	■
	⋮						



Obrázek 4.6: Ukázka jednoduché přepínací tabulky ATM switche

Poznámka:

O ATM bychom mohli ještě dlouho pokračovat (zejména architektura této sítě je velmi složitá – kromě vrstev se zde totiž vyskytují i roviny a několik různých druhů přizpůsobovacích vrstev k napojení na jiné protokoly), ale vzhledem k tomu, že dnes se s ATM nesetkáme, by to nemělo smysl.

4.2.3 MPLS – přepínání značek

Dosud jsme se zabývali přepínáním paketů (X.25, no vlastně tuto síť jsme přeskočili, už se nikde nepoužívá), rámců (Frame Relay) a buněk (ATM), teď se podíváme na přepínání značek.

Technologie byla představena firmou Ipsilon Networks pod názvem *IP Switching*. Později společnost Cisco vytvořila proprietární standard *Tag Switching*, který síť zbavil závislosti na ATM, podobný, ale otevřený standard později vydalo sdružení IETF.

MPLS (MultiProtocol Label Switching) je síť založená na *přepínání značek* (label, také návěští). Účelem je přesunout co nejvíce režie směrování (včetně administrace, QoS apod.) na okraj sítě tak, aby vnitřní oblast sítě byla co nejrychlejší. MPLS dokáže velmi rychle přenášet nejen běžná data, ale také hlas a video, a to se zajištěním QoS. Další výhodou je snadnější implementace virtuálních sítí.

Obrovskou výhodou MPLS je mnohotvárnost. Nemá přímo definovánu adresaci ani směrování, dokáže spolupracovat s více odlišnými protokoly. Původně MPLS pracovala jako nástavba nad ATM, ale v současné době pracuje také nad Ethernetem, Frame Relay, SONET/SDH, DWDM a dalšími. Takže výhody MPLS můžeme shrnout takto:

- rychlost přepínání,
- zajišťování řízení provozu (vyvažování zátěže) a QoS (odlišné zacházení s různými PDU),
- podpora VPN,
- schopnost spolupráce s mnoha různými protokoly, pod MPLS mohou fungovat různé technologie.

V ISO/OSI modelu bychom mohli MPLS zařadit někam mezi druhou (spojovou) a třetí (síťovou) vrstvu, také se označuje jako vrstva 2.5 nebo 2+.

Každý paket, který vstupuje do MPLS sítě, je na okrajovém směrovači opatřen jedním nebo více *MPLS záhlavími* uspořádanými do *zásobníku* (stack), a to na rozhraní mezi záhlavími druhé a třetí vrstvy. Záhlaví protokolu MPLS má tuto strukturu:

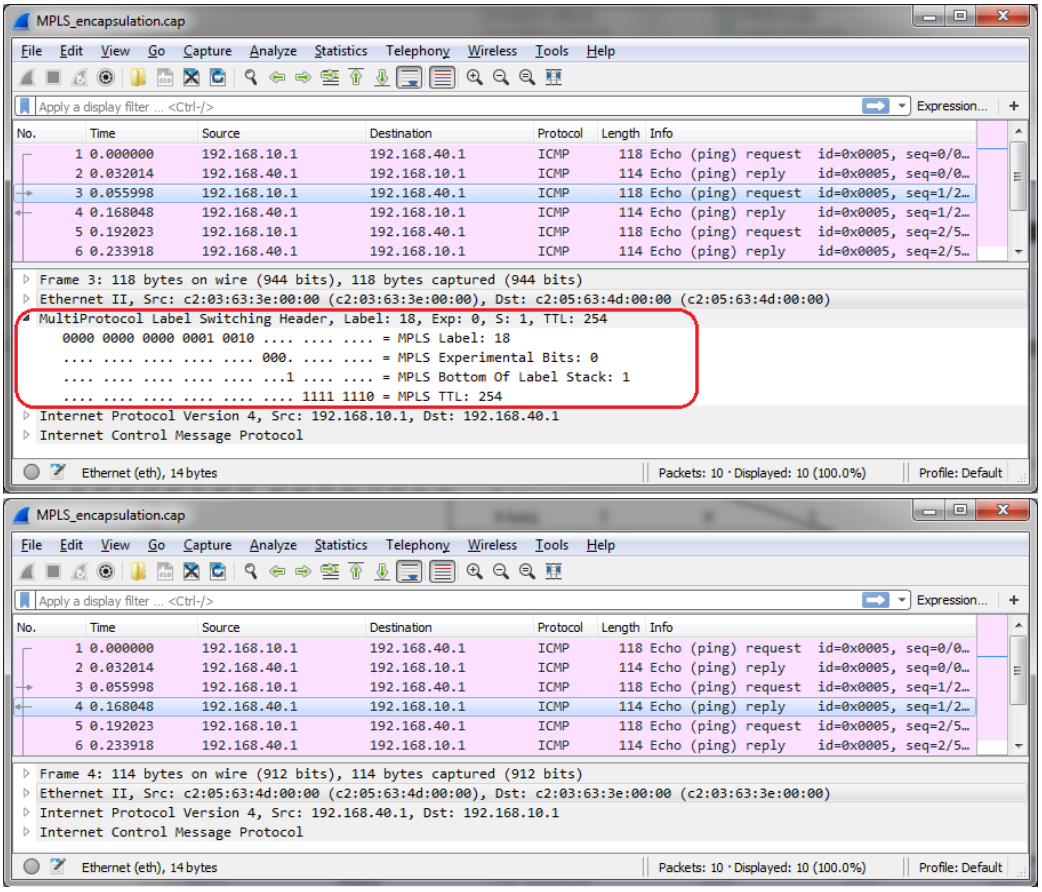
- samotná značka (návěští, label, 20 bitů),
- QoS informace (3 bity), v případě MPLS se setkáme s názvem CoS (Cathegory nebo Class of Service) nebo také ve významu Experimental (Exp.), do tohoto pole se případně mapuje QoS informace ze zapouzdřeného paketu,

- příznak konce zásobníku (1 bit, end of stack, bottom of label stack); pokud nenásleduje další záhlaví, je nastaven na 1 (to znamená, že následuje už přímo záhlaví IP paketu),
- TTL (Time to Live, 8 bitů) platné pro síť MPLS, může se mapovat z IP paketu.

Značka uložená v MPLS záhlaví jednoznačně určuje směrování paketu na cestě k následujícímu směrovači, v podobném smyslu jako DLCI u FR nebo VPI/VCI u ATM.

Příklad

Na obrázku 4.7 nahoře vidíme zprávu ICMP Echo request zapouzdřenou do IPv4 paketu, ten je uvnitř MPLS rámce, který je zapouzdřen v ethernetovém rámci.



Obrázek 4.7: Nahoře ping request posílaný přes MPLS spoj, dole odpověď došlá po běžném ethernetovém spoji[1]

Tento MPLS rámec má v zásobníku pouze jedno záhlaví (může mít více), label je v něm nastaven na 18, QoS (experimental) bity nejsou použity (jsou nastaveny na 0), je nastaven bit označující poslední záhlaví v zásobníku (bottom of label stack) a TTL je 254.

Záhlaví tedy může být v rámci více. *Vnější záhlaví* na zásobníku je určeno právě ke stanovení cest. Nenabízí vpodstatě o moc víc než IP záhlaví (s jedním důležitým rozdílem – rychleji se zpracovává).

[1] Obrázek podle rámce v cap souboru z packetlife.net

Další záhlaví hlouběji v zásobníku mají trochu jiný, specifický, účel, například záhlaví VPN pro určení virtuální sítě, záhlaví pro QoS nebo záhlaví pro řízení provozu.

Záhlaví rámce 2. vrstvy	MPLS Label 1, ES bit = 0	MPLS Label 2, ES bit = 0	MPLS Label 3, ES bit = 1	Záhlaví paketu 3. vrstvy (např. IP)	Datová část paketu

Tabulka 4.1: Zásobník značek v MPLS

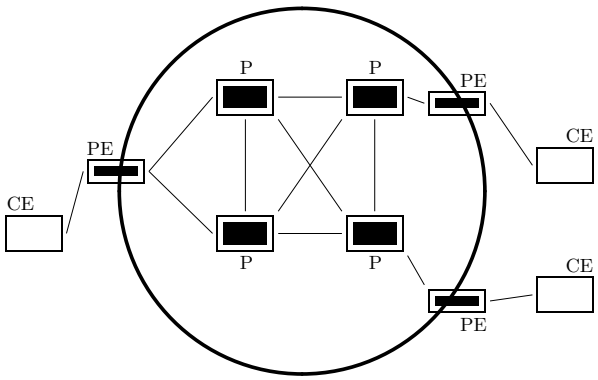
Přidělování značek (ve všech záhlavích ve stacku) probíhá formou třídění datových jednotek do *tříd* (FEC – Forward Equivalence Class), datové jednotky patřící do téže třídy jsou z uzlu odesílány se stejným záhlavím. Třída je stanovena na základě několika kritérií – především podle prefixu IP adresy a dále například podle některých charakteristik VPN.

Směrování v MPLS. Směrovače (spíše přepínače) v síti MPLS jsou nazývány *LSR* (*Label Switching Router*). LSR na okraji sítě (tj. komunikující také s uzly v lokálních sítích, které jsou sítí MPLS propojeny), jsou označovány jako *ELSR* (*Edge LSR*, hranové směrovače), a jsou buď vstupní (*Ingress ELSR*) nebo výstupní (*Egress ELSR*) podle směru provozu. U společnosti Cisco se můžeme setkat s odlišnou terminologií – LSR jsou nazývány provider/core router, ELSR se nazývají provider edge router (ale v mnoha dokumentech se používá „původní“ terminologie).

Velice často se také setkáváme s trochu jiným označením:

- P (Provider) – přepínače uvnitř MPLS sítě, vždy ve vlastnictví poskytovatele (nebo jeho poskytovatele),
- PE (Provider Edge) – na hranici MPLS sítě,
- CE (Customer Edge) – na hranici sítě zákazníka, buď v jeho vlastnictví nebo pronajat od poskytovatele (komunikuje s PE).

Na obrázku 4.8 je naznačena obvyklá struktura MPLS sítě s uzly P, PE a CE. CE je součástí lokální sítě zákazníka, k jednomu PE může být připojeno více CE (tj. více LAN).

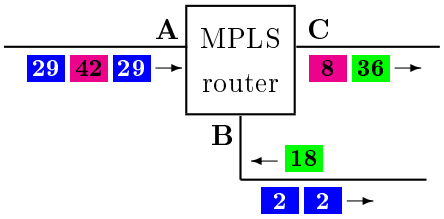


Obrázek 4.8: Struktura MPLS sítě

Směrování (přepínání značek) probíhá podobně jako v ATM, ale ještě jednodušeji. Směrovače si vedou jednoduchou *tabulku* (také se nazývá Label Forwarding Database – LFD), která je podobná

tabulce pro ATM, ale místo dvojice VPI/VCI je zde pouze hodnota značky (datagram z portu PPP opatřený značkou XXX je opatřen značkou YYY a poslán na port QQQ, apod.). Jedná se jen o určení vazby mezi příchozí a odchozí značkou, cesty jsou jednosměrné.

	In port	In label	Prefix adresy	Out label	Out port	
■	A	29	128.95.0.0/16	2	B	■
■	A	42	128.101.0.0/16	8	C	■
■	B	18	141.62.0.0/16	36	C	■
	⋮					



Obrázek 4.9: Ukázka jednoduché přepínací tabulky MPLS směrovače

Na obrázku 4.9 je ukázka tabulky na LSR uvnitř sítě. Pokud by se jednalo o vstupní Ingress ELSR, hodnoty ze sloupce se vstupní značkou a portem by nebyly používány, obdobně u Egress ELSR by nebyly používány hodnoty ze sloupce s výstupní značkou a portem.

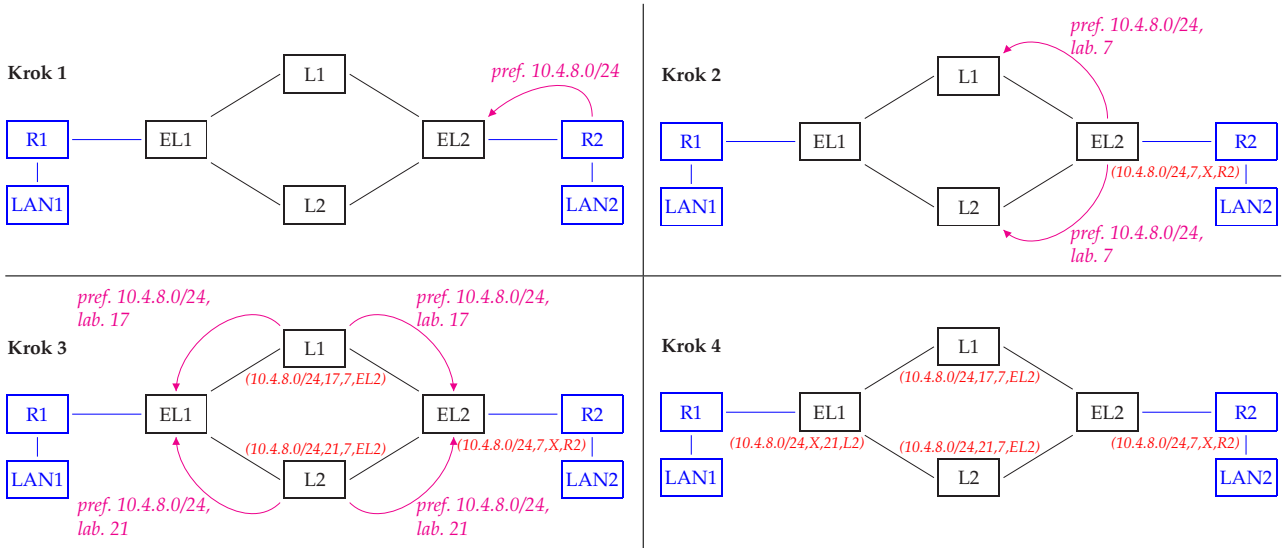
✎ Značka uložená v MPLS záhlaví se při průchodu směrovači neustále mění, podle toho, jak to bylo nastaveno při navazování spojení, podle obsahu tabulek na průchozích LSR. Tento proces se nazývá *Label Swapping* (výměna značek/návěští).

Konkrétní obsah tabulky na LSR závisí také na protokolu, se kterým MPLS spolupracuje (tj. na síti, nad kterou pracuje). Jestliže například MPLS pracuje nad ATM, jsou MPLS značky používány v ATM jako čísla VPI/VCI.

✎ **Vytváření cest.** *Protokol LDP* (Label Distribution Protocol) slouží k výměně informací o značkách mezi směrovači, výměna informací o prefixech adres probíhá podle *některého směrovacího protokolu* obecně označovaného zkratkou IGP (Interior Gateway Protocol), což může být prakticky kterýkoliv, velmi často OSPF (nyní spíše OSPFv2). U MPLS se však setkáváme i s dalšími protokoly, například při implementaci virtuálních sítí (VPN) se používá BGP. O směrovacích protokolech se budeme učit později.

✂ Postup přidávání záznamů do tabulek v základní variantě MPLS je naznačen na obrázku 4.10. Oznamování probíhá „proti směru“ cesty, každý uzel pro příchozí požadavek stanoví číslo (značku), které má právě volné, a pošle požadavek na všechny sousední uzly (i na ten, ze kterého požadavek přišel). Například podle obrázku 4.10 je *výstupním* LSR směrovač EL2:

- EL2 připojí LAN směrovač R2, prefix adresy je 10.4.8.0/24, přiřadí značku 7, do tabulky přidá záznam o tom, že cokoliv přijde na tuto značku a prefix, odešle směrovači R2,
- EL2 odešle sousedním MPLS směrovačům (L1 a L2) informaci „cestu k síti s prefixem 10.4.8.0/24 najdete na značce 7“,
- L1 obdrží tuto informaci, najde volnou značku 17, přidá do tabulky záznam o tom, že cokoliv přijde na značku 17 a daný prefix, v tom změní značku na 7 a pošle na směrovač EL2, odešle informaci směrovačům EL1 a EL2,
- podobně L2, ale našel volnou značku 21,
- EL1 obdrží dvě informace, ale informace z L2 přišla dřív, tedy uloží výstupní značku 21.



Obrázek 4.10: Připojení nové LAN do MPLS

Další informace:

- https://www.juniper.net/documentation/en_US/junos/topics/topic-map/mpls-overview.html
- http://www.ietf.org/dyn/wg/charter/mpls-charter.html
- http://www.cisco.com/en/US/products/ps6557/prod_white_papers_list.html
- http://www.cs.vsb.cz/grygarek/TPS/MPLS/Introduction_to_MPLS.htm
- http://www.cs.vsb.cz/grygarek/TPS/MPLS/Introduction_to_MPLS_II.htm

GMPLS (Generalized MPLS, také G-MPLS) je rozšíření MPLS, které zobecňuje MPLS na další vrstvy ISO/OSI a další rozhraní. Účelem je odstranit co nejvíce „komunikačních mezikroků“ a umožnit nasazení MPLS technologie i nad takovými řešeními, nad kterými to dřív nebylo možné (nejen nad řešeními používajícími pakety a rámce). Návěští už nemusí být jen číslo, ale může to být například vlnová délka v optické síti, fyzický port, časový slot v TDM apod., cokoliv, podle čeho lze rozšilit různé komunikace, tedy MPLS lze nasadit například i nad DWDM.

Podobné vlastnosti jako GMPLS mají sítě *ASON* (Automatically Switched Optical Network), také běží nad optickými cestami či SDH, také zahrnují podporu QoS, VPN a rychlého transportu.

Další informace:

- http://www.faqs.org/rfcs/rfc6002.html
- Farrel, A., Bryskin, I. *GMPLS: Architecture and Applications*. San Francisco, Elsevier, 2006. Většina stran dostupná na http://books.google.cz/books?id=cCWS75MlUpcC&printsec=frontcover
- http://wh.cs.vsb.cz/mil051/index.php/Any_Transport_over_MPLS_(AToM)

4.3 Infrastruktura optických sítí

4.3.1 SONET/SDH

SONET (Synchronous Optical Network, ANSI T.105) je digitální optický (na optických vláknech) přenosový systém využívající časový multiplex, centrálně synchronizovaný atomovými hodinami. Je standardizován a používán v USA, Kanadě a Japonsku.

SDH (Synchronous Digital Hierarchy, ITU-T G.707) je obdoba sítě SONET ve zbytku světa, rozdíly jsou jen malé (mj. v řídicích protokolech). Často se setkáme s označením SONET/SDH, odkazujícím na technologii, kterou mají obě řešení společnou.

SONET/SDH byla vytvořena pro hlasovou komunikaci a pro tento účel je také optimalizována, sama o sobě není efektivním řešením pro přenos dat. Proto nad SDH obvykle pracuje ještě jiný typ WAN/MAN sítě (nad SDH může pracovat například ATM, POS – Packet over SONET, MPLS, EoS – Gigabit Ethernet over SONET/SDH).

Zákazník je nucen zvolit si z několika málo typů služby podle propustnosti (2 Mb/s, 34 Mb/s, 130 Mb/s), za tuto propustnost také platí, i když ji zrovna nevyužívá (pásmo nelze pronajmout jinému zákazníkovi).

Podobně jako některé předchozí WAN, i zde je používána PDU o konstantní délce – *blok* má délku 810 oktetů. Pracuje se v časovém multiplexu, tedy přenášené bloky dat jsou umísťovány do volných *slotů* vysílaných v pravidelných intervalech 125 μs.

Další informace:

Článek o SONET/SDH: http://electrosofts.com/sonet/

4.3.2 WDM

WDM (Wavelength Division Multiplexing) je technologie multiplexování světla v optických spojích do vlnových délek, první řešení jsou už ze 70. let 20. století. Bez použití WDM je možné jedním optickým spojem vést jen jeden signál, s použitím WDM více signálů zároveň na různých vlnových délkách, na každé vlnové délce podle jiného protokolu a o jiné rychlosti.

WDM je záležitost fyzické vrstvy (L1).

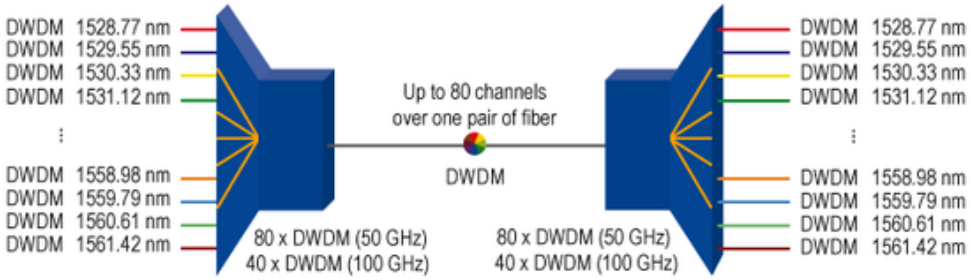
CWDM (Coarse WDM, standard ITU-T G.694.2 a dále pro kabely novější standardy G.652.C a G.652.D) je řešení multiplexování do optického vlákna používající poměrně široké kanály (šířka 20 nm). Typický dosah je v desítkách kilometrů a slučuje až 16 různých vlnových délek do jednoho signálu (záleží na konkrétním standardu), což je dostačující pro metropolitní sítě a hlavně celkem levné.

CWDM se dnes používá například v sítích poskytovatelů kabelové televize (přenáší se jak televizní, tak datový signál), v Ethernetu se s CWDM setkáme na fyzické vrstvě v 10GBase-LX4, také v sítích FTTx (Fibre to the Node/Building/Home/...) nebo v průmyslu na páteři při sesíťování velkých areálů.

DWDM (Dense Wavelength Division Multiplexing) umožňuje přenášet kanály na vlnových délkách s velmi malým odstupem (hustěji), i pod 1 nm. Čím menší odstup mezi vlnovými délkami, tím více na jednu stranu lze přenášet kanálů paralelně, ale na druhou stranu se může zhoršit kvalita přenosu,

proto obvykle není využíván plně. Do jednoho optického vlákna lze poskládat až 128 kanálů, ale ve frekventovaných datových páteřních sítích se spíše setkáme s 32 kanály v jednom vlákně (jmenovitě – CESNET2). Počet kanálů také závisí na použitých optických multiplexorech (které provádějí samotné mapování signálu do kanálu a zpět).

Obvyklá rychlost na jeden kanál (jednu vlnovou délku) je přibližně 10 Gb/s (nebo 40 Gb/s), a celková propustnost jednoho vlákna je (téměř) součtem propustností jednotlivých kanálů.



Obrázek 4.11: Skládání signálu v DWDM[2]

Na DWDM (mezi DWDM a TCP/IP) jsou pak implementovány dodatečné technologie, které mají urychlit a obecně zefektivnit přenos, v současné době především IP/MPLS a na okrajích rozlehlé sítě pak například EoMPLS (Ethernet over MPLS), který velmi dobře spolupracuje s připojenými ethernetovými LAN.

OTN (Optical Transport Network) v sobě spojuje výhody SONET/SDH a DWDM. Z SDH bere výhodu transparentnosti, pokročilého managementu (včetně monitorování, detekce chyb) a možnost vytvoření point-to-multipoint spojů, z DWDM má efektivní multiplexování velkého množství původních signálů do optického vlákna.

OTN se dá podsunout pod Ethernet, MPLS, IP a další.

Pro implementaci existuje více různých řešení fyzické vrstvy lišících se propustností a tím, se kterými protokoly vyšších vrstev je třeba spolupracovat. Například při spolupráci s Ethernetem existují různá řešení pro 1000Base-X, 10GBase-W, 40Gb Ethernet, 100Gb Ethernet a další. Princip je podobný skládání linek v SONET/SDH, například ODU4 (pro 100Gb Ethernet) se dá složit z až dvou ODU3 nebo deseti ODU2 nebo čtyřiceti ODU1 nebo osmdesáti ODU0 modulů.

Další informace:

- http://www.cables-solutions.com/capacity-expansion-and-flexibility-dwdm-network.html
- https://www.optcore.net/introduction-to-dwdm/
- https://www.itu.int/itu-t/recommendations/rec.aspx?rec=13076
- https://pon.fibrain.com/artykuly-techniczne/optical-transmission-lexicon-catv-cwdm-dwdm-ftth,44.html
- http://www.fiber-optic-cable-sale.com/dwdm-vs-otn-whats-difference.html

[2] Zdroj: https://www.pandacomdirekt.com/en/technologies/wdm/what-is-dwdm.html

4.4 Protokol PPP

Na začátku této kapitoly byl zmíněn protokol PPP. Je to jeden ze spojových protokolů, ale na rozdíl od jiných je poněkud univerzálnější a jednodušší (je osekaný jen na ten základ, který je při jeho používání potřebný). S protokolem PPP a jeho odvozeninami se dnes setkáváme zejména v přístupových sítích, kterými se budeme zabývat dále.

4.4.1 Vlastnosti PPP

Protokol *PPP* (Point-to-Point Protocol) je dvoubodový (point-to-point). Nerozlišuje primární a sekundární uzly, podporuje synchronní i asynchronní přenos. Dokáže provádět automatickou konfiguraci při navazování spojení, přičemž se testuje kvalita spoje. Součástí rámce je i určení protokolu vyšší vrstvy, proto dokáže nejen zapouzdřovat IP pakety, ale i jiné typy paketů vyšší vrstvy.

Specifikace PPP se skládá ze dvou částí (přesněji – obsahuje dvě vrstvy):

- *protokoly řízení sítě* (NCP, Network Control Protocol) – protokoly zajišťující komunikaci s vyšší vrstvou; existuje více těchto protokolů pro různé síťové architektury (TCP/IP, NetWare, AppleTalk apod.), tato vrstva zasahuje částečně i do síťové vrstvy v ISO/OSI modelu,
- *protokol řízení spoje* (LCP, Link Control Protocol) – navazování a udržování spojení, dohodnutí konfigurace na začátku spojení (negociace), může zajišťovat také testování spoje a autentizaci.

Účelem je maximálně omezit množství funkcí vlastního spojového protokolu (tj. LCP) a takto zefektivnit jeho fungování.

Autentizace je volitelnou vlastností, lze ji provádět některým z těchto protokolů:

- *PAP* (Password Authentication Protocol) – textové heslo se posílá po spoji, tedy nejde o bezpečnou metodu autentizace,
- *CHAP* (Challenge Handshake Authentication Protocol) – používá se asymetrická šifra (klíč), iniciující stanice odešle žádost, obdrží náhodně vygenerovanou sekvenci znaků, kterou zpracuje pomocí tajného klíče a odešle zpět, po ověření příjemcem může začít komunikace,
- *EAP* (Extensible Authentication Protocol) – zcela odděluje fázi navazování spojení od fáze autentizace, samotná autentizace může probíhat třemi různými způsoby:
  1. MD5 (funguje podobně jako CHAP, také pomocí klíče),
  2. heslo na jedno použití,
  3. token card.

PPP také podporuje šifrování a komprimaci.

Rámec protokolu PPP je do značné míry podobný rámcům jiných spojových protokolů:

- příznak (opět binárně 01111110),
- adresa (1 B) – toto pole existuje, ale má vždy konstantní hodnotu (binárně samé 1),
- řídicí pole (1 B) – obsahuje vždy hodnotu 0000011 (indikuje nečíslovaný rámec ve službě bez spojení),
- protokol (2 B, toto pole se u jiných protokolů nevyskytuje) – obsahuje identifikaci protokolu vyšší vrstvy (například pro IPv4 je to 0×0021, pro IPv6 0×0057, pro IPX je zde 0×002B), k nalezení v RFC dokumentech,

- data – maximálně 1500 B,
- FCS,
- příznak.

To byl rámec obsahující data pro přenos.

Používají se také služební LCP pakety, které je třeba zapouzdřit do PPP rámce následovně:

- v poli protokolu je hodnota 0×C021,
- v poli data je samotný LCP paket obsahující tyto informace:
  - kód – identifikuje funkci LCP paketu (požadavky na změnu parametrů, potvrzení přijetí požadavků na změnu, odmítnutí požadavků, ukončení spojení, atd.),
  - identifikátor – u příkazového paketu obsahuje náhodně vygenerovanou hodnotu, u odpovědi musí obsahovat tutéž hodnotu jako paket s příkazem, na který odpovídá,
  - délka celého LCP paketu v oktetech,
  - data – obvykle posloupnost uspořádaných dvojic (volba,hodnota), například typ autentizačního protokolu, vyšší max. délka dat, atd.

4.4.2 Rozšíření PPP

Pro rozšíření protokolu PPP platí vpodstatě totéž co pro PPP (vrstva L2, přibližně účel, šifrování, většinou i formát rámce). Tato rozšíření jsou navíc přizpůsobena konkrétnímu způsobu využití.

Multilink PPP (PPP po více spojích, také se označuje MP, MPPP, MLP; RFC 1990) je varianta PPP pro vícebodové spoje. Není tím míněno ani tak více uzlů, jako spíše více sériových spojů pro jedinou komunikaci. Lze zkombinovat více sériových spojů do jediného logického kanálu s vyšší propustností (agregace), protokol dokáže data rozdělit, vhodně přeuspořádat a pak na druhém konci linky opět uvést do původního stavu.

Tunneling PPP (PPTP, Point-to-point Tunneling Protocol, RFC 1171) je určen pro virtuální privátní sítě a k připojení vzdáleného klienta k firemní síti (obecně LAN). Ovšem dnes je podporován prakticky už jen ve Windows a jeho podpora se bude omezovat i zde, doporučuje se přechod na jeho následovníka *L2TP* (Layer 2 Tunneling Protocol, RFC 3931).

L2TP: Do rámce protokolu L2TP se zapouzdřují rámce PPP. Sám o sobě nemá tento protokol zabezpečovací funkce, proto se často kombinuje se zabezpečením pomocí IPSec nebo jiných metod. K L2TP se vrátíme v kapitole o bezpečnosti, je to jeden z protokolů pro vytváření VPN (tunelů).

Připojení k Internetu se řeší napojením PPP (iniciační fáze připojení) na protokoly WAN. Využívá se také v technologiích DSL pro přenos přes WAN poskytovatele, a to v těchto variantách:

- PPPoA (PPP over ATM, RFC 2364) – PPP paket je zapouzdřen do ATM buňky,
- PPPoE (PPP over Ethernet, RFC 2516) – PPP paket je zapouzdřen do Ethernetového rámce, dnes běžné.

Tyto varianty umožňují navázat autentizované spojení s ISP a dynamicky získat IP adresu. Jsou potřeba u typů komunikace, kde se navazuje spojení, nevyužívají se u trvalého připojení k Internetu (resp. využívají se tehdy, když se navazuje spojení; když už je navázáno, nejsou potřeba).

4.5 Telekomunikační sítě

4.5.1 Využití telekomunikačních sítí pro přenos dat

Shannonův teorém. Claude Elwood Shannon je zakladatel moderní teorie informace. Shannonův teorém udává strop pro maximální rychlost přenosu.

> „Aby bylo možné rekonstruovat (na cílové stanici) spojitý frekvenčně omezený analogový signál ze vzorků, musí být vzorkován s frekvencí alespoň dvakrát vyšší než je maximální frekvence tohoto signálu při přenosu.“

Důsledkem je ohraničení maximální dosažitelné rychlosti přenosu, a to

- šířkou pásma (počet stavů v rámci pásma nelze zvyšovat do nekonečna, ztratili bychom možnost jeho rozlišení),
- kvalitou signálu (vyplývá z fyzikálních vlastností linky, zhoršená šumem, také „odstup signálu od šumu“),

ale nezávisí na použité technologii (včetně použité modulace). Vzorec:

$$\max(\text{rychlost přenosu}) = \text{šířka pásma} * \log_2(1 + \text{signál/šum})$$

Telekomunikační (telefonní) síť je řešena hierarchicky:

- *účastnické zařízení* (telefon, modem apod.), připojuje se přes *účastnickou přípojku*,
- (veřejná) *místní telefonní ústředna* (MTO) – v pravidelných intervalech na ulicích,
- *uzlová telefonní ústředna* (UTO), *tranzitní telefonní ústředna* (TTO).

V rámci firemního areálu může také fungovat *pobočková ústředna* (PBX, Private Branch Exchange), která bývá připojena na některou veřejnou místní telefonní ústřednu (tj. funguje jako brána).

Při přenosu dat přes telefonní síť, dimenzovanou pro přenos zvuku (především hlasu), bylo hlavní motivací využití již existující husté sítě těchto linek, tedy snadná dostupnost koncových bodů.

V telefonní síti platí: pro přenos hlasu plně postačují frekvence z intervalu 300–3400 Hz, ořezání okolních frekvencí prakticky neovlivní kvalitu hovoru. Přenos musí být multiplexován, v analogové síti je volen *frekvenční multiplex* s šířkou pásma pro jeden přenos zaokrouhlenou nahoru (s rezervou) – 4000 Hz.

Z toho vyplývá, že vyšší frekvence se v telekomunikačním kabelu dají využít pro přenos dat. Pokud bychom netrvali na přenosu analogového hlasu (na čemž v poslední době většinou netrváme), mohli bychom pro data využít i nižší frekvence.

Podle Shannonova teorému můžeme rychlost přenosu (nebo dosah) zvýšit buď tím, že rozšíříme frekvenční pásmo (směrem nahoru), nebo zvýšíme kvalitu signálu – kabely asi vyměňovat nebudeme, ale existují různé možnosti ovlivňování šumu. V každém případě pro přenos dat je schůdnější spíše první možnost.

Pojmy. Abychom snáze pochopili následující text, vysvětlíme si několik pojmů souvisejících s telekomunikační sítí.

- PSTN (Public Switched Telephone Network) – běžná telefonní síť dimenzovaná pro přenos hlasu, původně čistě analogová s frekvenčním multiplexem, po digitalizaci se používá časový multiplex

s rezervací komunikačního pásma. Pozor, digitalizace se týká především telekomunikačních ústředen, část spoje u zákazníka je analogová.

Po digitalizaci bylo nutno (v principu analogový) hlas v MTO digitalizovat, používal se některý kodek, tedy CODEC (COder-DECoder). Podobný princip se dnes používá i ve VoIP přímo v koncovém zařízení.

- POTS (Plain Old Telephone Service) je technologie využívající PSTN pro přenos analogového hlasu a digitálních dat. Protože PSTN je zkraje analogová a až od ústředny digitální, je třeba digitální data modulovat na analogový signál, používá se MODEM (MODulator-DEModulator).
- Modem je tedy zařízení, které moduluje digitální signál na analogový a v cíli ho zpětně demoduluje do digitální podoby. Modemy byly dříve analogové, dnes používáme digitální modemy (pro ADSL, VDSL apod.) – telekomunikační linka na straně zákazníka je prostě pořád analogová.
- ISDN (Integrated Services Digital Network) byla prvním pokusem o digitalizaci přístupových sítí, ale tak trochu na půl cesty. Šlo o vytáčené připojení (tarifikace podle času), protože správa sítě stavěla na původní POTS, vlastně to byla jen digitální nástavba POTS.
- xDSL je skupina technologií, které se oprostily od většiny součástí POTS, z telekomunikační sítě se využívá pouze spoj mezi účastnickou přípojkou a nejbližší ústřednou MTO. Pak komunikační cesta odbočuje do plně digitální počítačové WAN sítě poskytovatele.

4.5.2 Slučování linek

Předchůdcem sítě SDH, se kterou jsme se seznámili v předchozím textu, je síť PDH (Pleisochronous Digital Hierarchy). Zatímco SDH je „synchronní“ (komunikace je synchronizována v celé síti, s časovým multiplexem), PDH je „téměř synchronní“ – pleisochronní, tedy komunikace je téměř synchronizována (tolerují se malé odlišnosti v rychlosti).

V PDH se začalo využívat *slučování (agregace) linek* za účelem zvýšení kapacity přenosu. Značení a některé další prvky se odlišují v normách platných pro Evropu od norem platných v Americe a Japonsku. Zatímco v Americe a Japonsku se setkáme s T-linkami (T-carrier, normy podle ITU-T), v Evropě se používají E-linky (E-carrier, normy podle CEPT – Evropské rady pro správu pošt a telekomunikací).

Princip vychází ze sdružování linek do svazků, které mají logicky vyšší přenosovou kapacitu a tím i rychlost přenosu. V obou normách jsou linky řádu 0 (nula) tvořeny jedinou linkou o rychlosti 64 kb/s. V řádu 1 se sdružuje buď 24 nebo 30 linek datových a 2 linky signální (nepřenášejí data, ale řídicí signály), vzniká T1-linka (Amerika a Japonsko) nebo E1-linka (Evropa). Sdružením čtyř T1 linek vznikne T2 linka, podobně u E-linek. Postup a navyšování přenosové rychlosti najdeme v tabulce 4.2.

Značení	Dat. kanálů	Přen. rychlost
T1	24	1,54 Mb/s
T2	96	6,31 Mb/s
T3	672	44,38 Mb/s
T4	4032	274,18 Mb/s

Značení	Dat. kanálů	Přen. rychlost
E1	30	2,05 Mb/s
E2	120	8,45 Mb/s
E3	480	34,37 Mb/s
E4	1920	139,27 Mb/s
E5	7680	565,15 Mb/s

Tabulka 4.2: Srovnání T-carrier (Amerika, Japonsko) a E-carrier (Evropa) linek

SONET	SDH	Přen. rychlost
STS-1	STM-0	52 Mb/s
STS-3	STM-1	156 Mb/s
STS-12	STM-4	622 Mb/s
STS-48	STM-16	2 488 Mb/s

SONET	SDH	Přen. rychlost
STS-192	STM-64	9 953 Mb/s
STS-768	STM-256	39 813 Mb/s
STS-3072	STM-1024	159 252 Mb/s

Tabulka 4.3: Úrovně a rychlosti v sítích SONET a SDH

Fyzicky bývají sdružovány spoje na kroucených dvoulinkách (UTP), což početně souhlasí: 30 + 2 = 32, což je dělitelné číslem 4 (počtem drátů v UTP).

Rámec E1 se skládá z 32 timeslotů („oken“ o stejné délce – 125 µs), přenášených paralelně. Jeden timeslot obsahuje prostor pro jeden oktet (8 bitů), od toho se odvíjí přenosová kapacita.

SDH se od PDH liší především využitím optických kabelů (jednovidových) a dokonalejší synchronizací, která se provádí přes celou síť pomocí atomových hodin. Velikost timeslotu je stejná (125 µs), ale multiplexování funguje trochu jinak (protože už se neodvíjí od UTP kabelu).

Na slučování linek stála i technologie ISDN – přenosová kapacita je v této technologii rozdělena na B-kanály a D-kanál, existují dva druhy rozhraní

- BRI (u nás euroISDN2) pro domácnosti a malé kanceláře, slučuje dvě datové a jednu signální linku,
- PRI (u nás euroISDN30) pro větší firmy, dražší, slučuje 30 datových a jednu signální linku.

V BRI je bitový tok rozdělen na časové rámce po 48 bitech (0,25 ms) – 2 oktety pro každý B-kanál, 4 bity pro D-kanál a 12 doplňkových bitů, rychlost u BRI je maximálně 160 kb/s. B-kanály pracují s přepojováním okruhů (proto tarifikace podle času – platí se za dobu existence přepínaného okruhu SVC) a mají vyšší přenosovou rychlost, D-kanály pracují s pakety, a to se spojením i bez spojení.

Poznámka:

Zde sice o agregaci linek hovoříme spíše vzhledem k telekomunikačním sítím, ale linky je možno agregovat i lokálně. V předchozím textu jsme se setkali například s technologií EtherChannel pro sloučení více ethernetových linek do jednoho virtuálního spoje, další termíny jsou Link Aggregation a NIC Teaming. Používají se například pro navýšení přenosové kapacity spoje mezi serverem (nebo jiným zařízením) a switchem, což je užitečné hlavně u velmi vytěžovaných zařízení. Ovšem je třeba mít na daném zařízení více ethernetových portů a podporu příslušné technologie (na obou stranách spoje).

http://www.samuraj-cz.com/clanek/pripojeni-rychlejsi-a-spolehlivejsi/

4.6 Přístupové telekomunikační sítě

Přístupové sítě jsou sítě sloužící pouze a jenom k napojení lokálních (nebo jiných menších) sítí na WAN/MAN sítě (třeba patřící některému ISP) a zprostředkovaně do Internetu. Také hovoříme o sítích *první míle* (First Mile) nebo *poslední míle* (Last Mile), což vlastně znamená jakousi hranici mezi menší sítí a WAN/MAN sítí. V anglické literatuře se spíše setkáme s „optimističtějším“ Fist Mile.

Pro přístupové sítě existují různé technologie – může jít o bezdrátové sítě (například WiMAX) nebo telekomunikační sítě (většinou ADSL/VDSL nebo podobná technologie), v některých případech se přístupová síť dokonce jeví jako speciální součást WAN sítě (mobilní sítě, třeba LTE).

Specifikem přístupové sítě je, že spojuje velice odlišné technologie vzhledem ke způsobu komunikace (v LAN sítích máme typicky paketový přenos dat, kdežto ve WAN sítích se komunikuje přes okruhy), a také vzhledem k fyzické vrstvě a zacházení se signálem.

V této kapitole se zaměříme spíše na komunikaci přes telekomunikační linku, na bezdrátové a mobilní sítě se podíváme v další kapitole.

4.6.1 ADSL

xDSL (Digital Subscriber Line) je skupina širokopásmových technologií, která v sobě sdružuje několik různých technologií (ADSL, SDSL, HDSL, HDSL-2, G.SHDL, IDSL, VDSL). Jedná se o vyhrazenou službu, point-to-point.

Je to služba se spojením, ale tarifikace není závislá na čase.

ADSL (Asymmetric Digital Subscriber Line) je asymetrická služba. Asymetrická proto, že downstream (stahování dat do DTE) je rychlejší než upstream (odesílání dat do sítě).

Teoreticky lze dosáhnout rychlostí až 24 Mb/s (v novějším standardu dokonce až 48 Mb/s). U nás nabízená rychlost je 8 Mb/s nebo 16 Mb/s *pro downstream*, ale reálná rychlost může být nižší. Upstream bývá na rychlosti 1 Mb/s.

Rychlost je ovlivněna více různými kritérii. Je to nejen použité zařízení a přenosové protokoly, ale také délka spoje – čím větší vzdálenost k místní ústředně, tím pomalejší spojení.

Rozsah	Šířka	Účel
0 kHz –4 kHz	4 kHz	přenos hlasu (telefon)
4 kHz –26 kHz	22 kHz	nárazníkové pásmo
26 kHz –138 kHz	112 kHz	upstream
138 kHz –1100 kHz	962 kHz	downstream

Tabulka 4.4: Využití přenosového pásma v ADSL

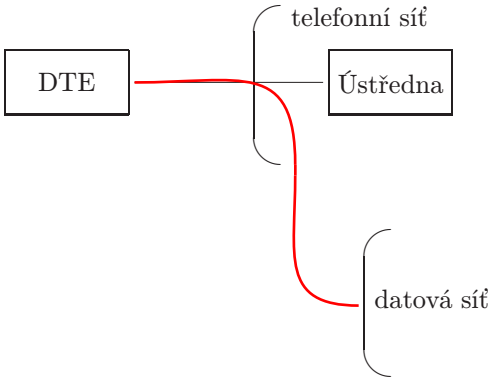
V technologii ADSL (a taky v dalších xDSL technologiích) se navýšení rychlosti dosahuje kromě jiného i odkloněním datového toku z telefonních linek na datovou síť poskytovatele (obvykle některou WAN síť na optických kabelech). Princip je naznačen na obrázku 4.12.

Modulace v ADSL – používá se CAP nebo DMT.

CAP (Carrieless amplitude phase modulation) je velmi podobná modulaci QAM. CAP používá pro celé přenášené pásmo (1,1 MHz) jediný kmitočet, fakticky nedochází k dělení na kanály.

DMT (Discrete MultiTone) rozdělí celé pásmo na kanály přibližně po 4 kHz (celkem kolem 255 kanálů), kanály 7–32 pro upstream, od 33 pro downstream. Průběžně se sleduje kvalita přenosu v jednotlivých kanálech, přenos se adaptivně rozkládá na ty kanály, které jsou považovány za právě nejkvalitnější. Na rozdíl od CAP se používá více nosných kmitočtů (to jsou ty tóny v názvu, také se nazývá Multicarrier Modulation).

V současné době se setkáme spíše s DMT.



Obrázek 4.12: Provedení ADSL

http://access.feld.cvut.cz/view.php?cisloclanku=2004072903

Agregace = sdílení přípojky ADSL. Každý ISP rozdělí kapacitu linky, kterou má k dispozici, formou časového multiplexu. Vzorec pro agregační poměr je následující:

$$agregační\ poměr = \frac{rychlost(ISP)}{\sum_i rychlost(i)}$$

Typické hodnoty agregace mohou být například $1:50,\ 1:20$.

FUP (Fair User Policy) je omezení toku dat (většinou se odvozuje od množství přenášených dat). V současné době není u ADSL FUP uplatňováno.

FUP v ADSL obvykle souvisí s protokolem PPPoA. Proto pokud zjistíme, že FUP je na naší lince uplatňováno, měli bychom projít konfiguraci ADSL modemu a zjistit, jestli je možné nastavit místo PPPoA (PPP over ATM) protokol PPPoE (PPP over Ethernet). U starších ADSL modemů to nejde, ale u novějších by to neměl být problém.

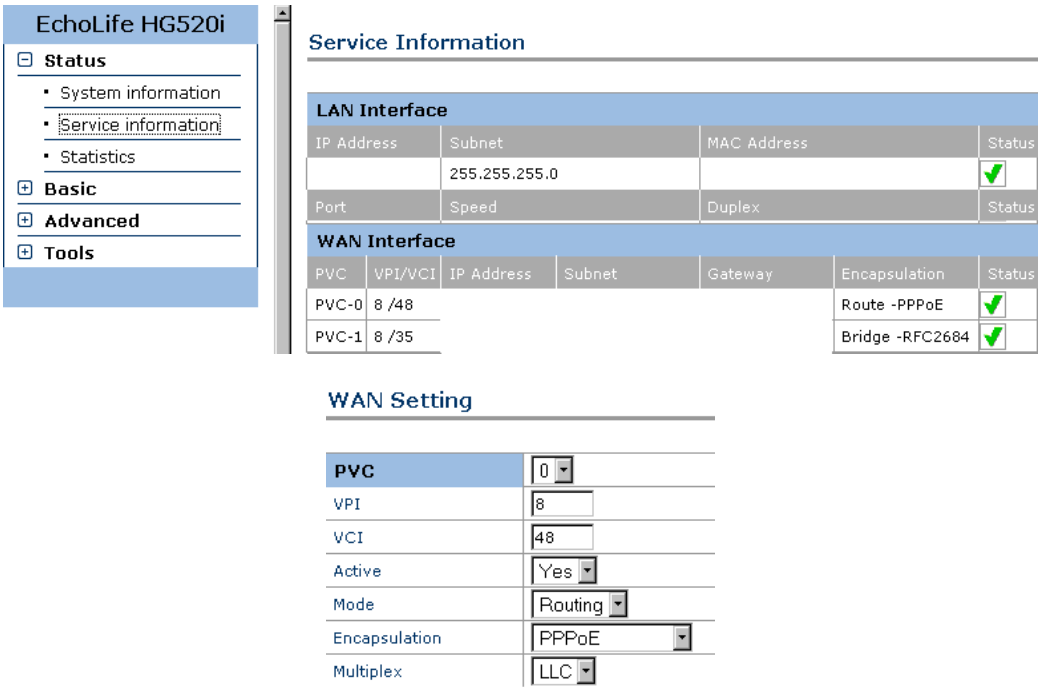
Standardy. U jednotlivých standardů se rozlišují varianty Annex A (over POTS – přes analogové linky), Annex B (over ISDN) a ADSL Lite (odlehčená verze pro nekvalitní spoje). U nás lze provozovat pouze Annex B, proto je třeba si dát pozor, když pořizujeme takové zařízení.

V současné době existují následující standardy, které odpovídají verzím této technologie:

- ADSL (původní) – Annex A a Annex B s propustností až 8 Mb/s (download), resp. 1 Mb/s (upload), a dále ADSL Lite (ještě nižší propustnost – do 1.5 Mb/s),
- ADSL2 – druhá generace, rychlost na downloadu až 12 Mb/s,
- ADSL2+ – zvýšení rychlosti až na 24 Mb/s (reálně až 16 Mb/s),
- ADSL2++ – opět zvýšení rychlosti, teoreticky až 48 Mb/s.

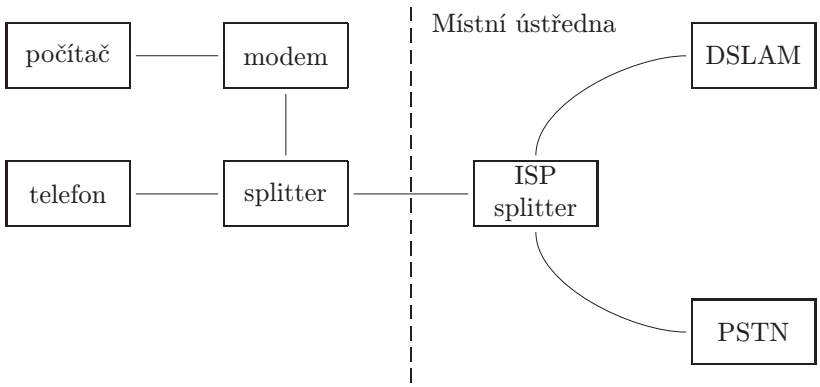
Zařízení v síti ADSL. Když nepočítáme koncová zařízení v síti zákazníka (která ve skutečnosti do sítě ADSL vůbec nepatří), v ADSL se používají tato zařízení:

- *ADSL modem* (MODulator/DEModulator) – moduluje digitální signál na analogový (šířka přibližně 1,1 MHz) a zpětně demoduluje, připojují se k němu datová koncová zařízení,
- *splitter* – sloučení analogového hlasového přenosu a modulovaných dat, připojuje se k němu modem a hlasová koncová zařízení (analogová),
- *DSLAM* (DSL Access Multiplexer) – na straně ISP, sdružuje spojení z ISP splitterů pro jednotlivá připojení.



Obrázek 4.13: Zapouzdření do PPPoE

V praxi je u účastnických přípojek mohou být modem a splitter v jednom zařízení (interní, integrovaný), na straně ISP bývají odděleny ve dvou zařízeních.[3] Je výhodnější mít splitter zvlášť. Pokud nepoužíváme žádná analogová zařízení (analogový telefon), měli bychom se splitteru zbavit, protože může být zdrojem mírných posunů signálu a tím i poruch.



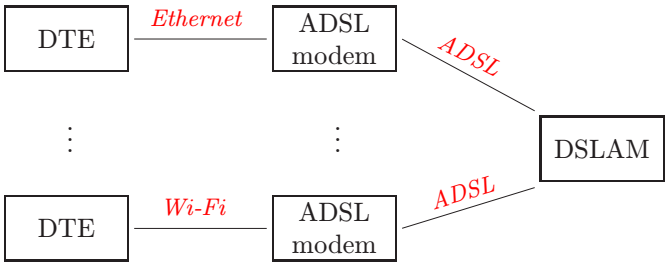
Obrázek 4.14: Zařízení v síti ADSL

Digitální telefony samozřejmě ke splitteru nepřipojujeme!

ADSL modem je v SOHO (Small Office, Home Office) často jen modulem v složitějším zařízení, například jsou běžné ADSL Wi-fi routery (tj. ADSL modem s Wi-fi routerem plus další funkcionalita: hardwarový firewall, DHCP server, atd.). Připojuje se k účastnické přípojce přes RJ-11 (telefonní kabel).

Jeho funkce jsou jak navazování spojení (obvykle při zapnutí nebo resetu přístroje), tak i zajišťování samotné komunikace. Pakety, které je třeba odeslat z lokální sítě přes přístupovou síť, zapouzdřuje do

[3] Modem a splitter v jednom zařízení najdeme spíše u levnějších řešení pro SOHO (Small Office–Home Office), tj. pro domácnosti a malé firmy.



Obrázek 4.15: ADSL je technologie první míle

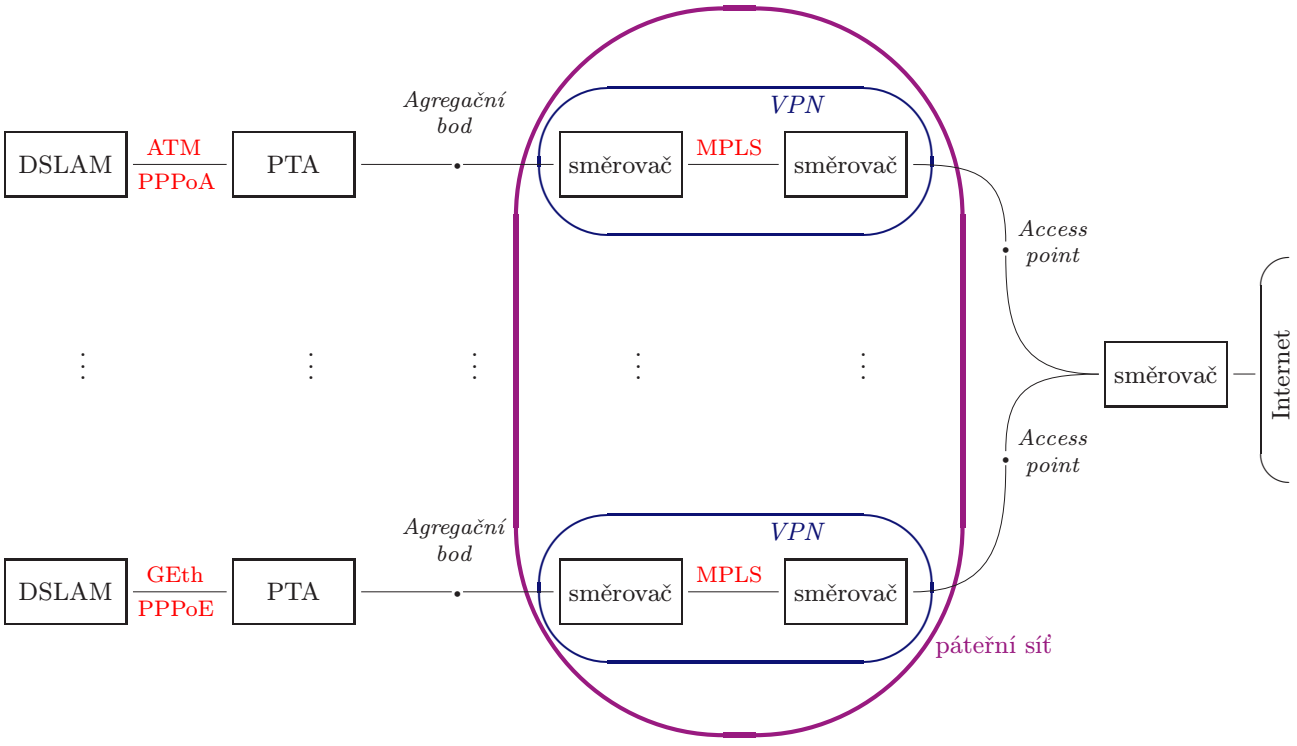
PDU protokolu PPPoE, PPPoA nebo IPoA (ten první je dnes nejběžnější).

DSLAM je brána do datové sítě poskytovatele. Existují menší DSLAMy pro připojení 24 nebo 48 DSL linek (a obvykle stejný počet pro POTS – analogový telefon), a pak větší pro tisíce linek. Obvykle podporuje protokoly PPPoE i PPPoA.

DSLAM má rozhraní RJ-11 směrem k zákazníkovi a pak další rozhraní – většinou dvě RJ-45 pro Gigabit Ethernet nebo rychlejší, případně optiku. Administrace se provádí přes webové rozhraní, anebo přes konzoli, záleží na konkrétním výrobci.

Na obrázku 4.16 je zjednodušený nákres struktury sítě na straně poskytovatele Internetu. Pojmy k tomuto obrázku:

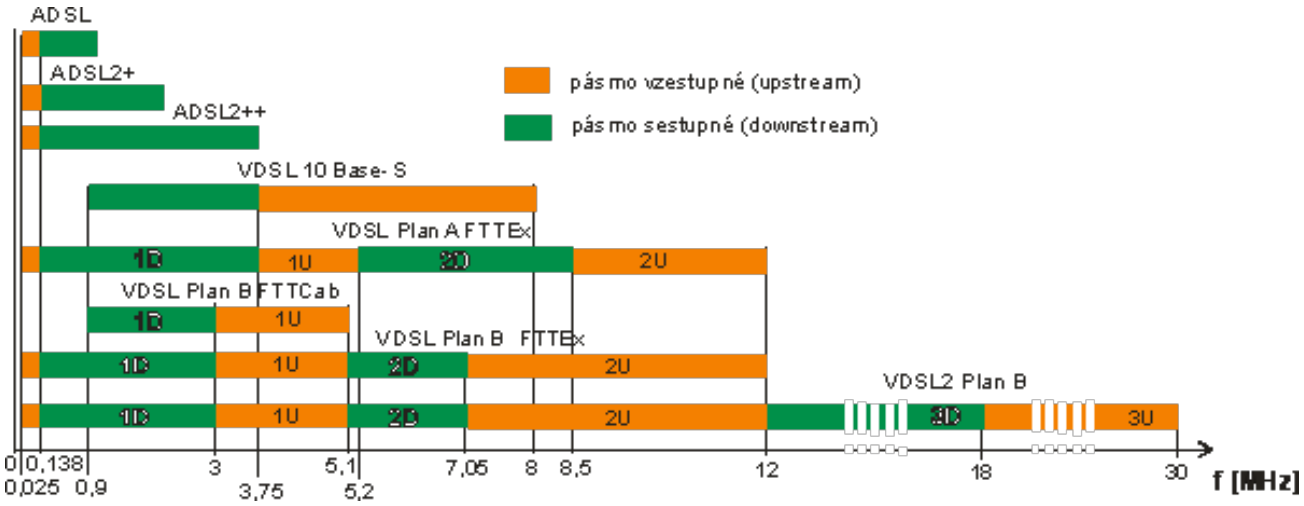
- *PTA* je širokopásmový server, provádí se zde konfigurace IP adres, autentifikace uživatelů, autorizace, účtování (RADIUS),
- *AP (Access Point)* je přístupový bod pro konkrétního ISP, přes který lze přistupovat z virtuální sítě tohoto ISP k směrovači na Internet.



Obrázek 4.16: ADSL – na straně ISP

4.6.2 VDSL

VDSL (Very High Bit-Rate Digital Subscriber Line) je považována za následníka ADSL. Je také asymetrická (může být nastavena na symetrický přenos), používá širší pásmo než ADSL (protáhnutí na vyšší frekvence), a z toho důvodu je rychlejší. Rychlost na *kvalitní* telefonní UTP dosahuje až 52 Mb/s (asymetrická, downstream) nebo 36 Mb/s (symetrická varianta). Teoreticky až 200 Mb/s, reálně opravdu jen kolem 50 Mb/s.



Obrázek 4.17: Porovnání frekvencí ADSL a VDSL[4]

Z obrázku 4.17 je patrné, že oproti ADSL technologie VDSL využívá širší frekvenční pásmo, tedy je k dispozici více komunikačních kanálů. Zatímco u ADSL je na nižších frekvencích upstream a na vyšších downstream, u VDSL jsou kanály pro oba směry rozloženy celkem rovnoměrně v celém spektru (důsledkem je, že zatímco u ADSL při přechodu na novou verzi využívající vyšší frekvence byl navýšen jen downstream, protože upstream z důvodů kompatibility měl pásmo pevně stanoveno, u VDSL mohou být navyšovány rychlosti v obou směrech).

Zatímco ADSL provádí přenos na vzdálenost až 5 km (při vyšších rychlostech méně), VDSL pouze něco přes 1 km, což je daň za rozšíření pásma. Ovšem pokud chceme opravdu využít vysoké rychlosti nabízené VDSL, tak bychom měli být maximálně 500 m od DSLAMu. Upstream a downstream jsou odděleny frekvenčním multiplexem podobně jako u ADSL.

Rozšíření VDSL závisí na telekomunikačních organizacích, tato technologie musí být podporována v telekomunikačních ústřednách.

4.6.3 Další xDSL technologie

High bit-rate Digital Subscriber Line (HDSL) není určena pro účastnické přípojky, používají ji ISP pro propojení pobočkových ústředen, případně ji mohou využít velké firmy pro své vnitřní potřeby. Vyžaduje dva nebo tři páry telefonních vodičů, což je považováno za jednu z nevýhod.

Je to symetrická technologie (stejný rozsah pásem pro downstream a upstream), ale může pracovat i asymetricky. Podporuje pouze přenos dat, neobsahuje podporu telefonních hovorů. Pro oddělení downstreamu a upstreamu používá pouze Echo Cancellation (EC), ne frekvenční multiplex.

[4]Zdroj: http://access.feld.cvut.cz/view.php?cisloclanku=2004120302

HDSL-2 (také SHDSL – SinglePair HDSL) je varianta HDSL, která používá pouze jeden pár telefonních vodičů, tedy při zavádění technologie HDSL-2 na samotném vedení telefonních linek není třeba nic měnit.

Symetric Digital Subscriber Line (SDSL) je rozsahem funkcí podobná technologii ADSL, ale je symetrická (stejný rozsah pásem – počet kanálů – pro upstream a downstream), neobsahuje podporu telefonních hovorů (pouze pro data, nelze připojit analogový telefon). Princip přenosu je podobný HDSL.

Používá se pro účastnické přípojky, u kterých se preferuje symetričnost přenosu. Obvyklá rychlost je až 2,3 Mb/s, dosah až 5 km.

IDSL (ISDN Digital Subscriber Line) je hybrid ISDN a xDSL technologií. IDSL je poskytována tam, kde ještě nelze zprovoznit ADSL (v ústředně není funkční DSLAM), a nebo ISP. Využívá rozhraní „U“ pro ISDN, přenáší pouze data, připojení přes ISDN BRI. Narozdíl od ISDN není vytáčená a je rychlejší (pomalejší než ADSL), používá stejnou modulaci jako ISDN.

Další informace:

- http://www.svetsiti.cz/view_list.asp?rubrika=Tutorialy&temaID=166
- http://access.feld.cvut.cz/view.php?cisloclanku=2004120302
- http://www.earchiv.cz/i_potsadsl.php3
- http://access.feld.cvut.cz/search.php?rsvelikost=sab&rstext=all-phpRS-all&rstema=14&stromhlmenu=14

4.7 Pobočkové ústředny

Již víme, že telefonní ústředny jsou uspořádány hierarchicky – na nejvyšších uzlech hierarchie jsou TTO (tranzitní telefonní ústředny), níže jsou UTO (uzlové telefonní ústředny), pod nimi MTO (místní telefonní ústředny), ke kterým se již připojují účastnické přípojky a nebo v případě větších firem pobočkové ústředny (PBX).

Pobočkové ústředny se používají k zajišťování vnitřních hovorů uvnitř firem a také ke spojování hovorů mezi vnitřní a vnější telekomunikační sítí. Jsou vlastně přípojným bodem do sítě poskytovatele této služby (DTE).

Telefonní ústředny dělíme na analogové a digitální, ale v současné době se většinou již setkáme jen s digitálními. Analogové pobočkové ústředny lze připojit na běžné telefonní linky PSTN sítě (*HTS* – hlavní telefonní stanice, dřív se říkalo „státní linka“), digitální ústředny také na

- běžná telefonní síť,
- E1 u starších digitálních ústředen, BRI a PRI ISDN,
- datovou síť VoIP,
- datovou bezdrátovou síť (DECT bezdrátové pobočky rozmístěné po areálu), atd.

VoIP je ve své podstatě datovou službou, tedy nevyžaduje připojení do telekomunikační sítě (pokud máme jen čistě VoIP ústřednu), stačí připojení do datové sítě. Tomuto typu služeb se budeme věnovat v následující podkapitole.

Ústředna může být buď přímo hardwarové zařízení, a nebo software nainstalovaný na vhodně připojeném počítači. Podíváme se na několik softwarových řešení (jedná se o serverové aplikace). Kromě softwaru potřebujeme také hardware, kterým stanici s tímto softwarem připojíme ke zvolenému rozhraní do sítě poskytovatele služby (telekomunikačního operátora).

Asterisk je populární open-source software pro vytváření VoIP ústředen. Konfiguruje se přes webové rozhraní, existuje také několik GUI (nutno zvlášť doinstalovat, např. FreePBX), dále lze k aplikaci přistupovat programově přes API z aplikací napsaných v běžných programovacích jazycích. Asterisk najdeme také v mnohých prodávaných hardwarových ústřednách.

Cisco Unified Communications Manager (dříve Cisco CallManager) lze instalovat na zařízeních, která tento produkt podporují (většinou na serverech Cisco). Jedná se o komerční produkt s obecně dobrou vybaveností funkcemi.

Microsoft Response Point je systém s jednoduchou konfigurací pro menší telefonní sítě (do 50 stanic, spíše méně, záleží na konkrétním hardwaru). Tento produkt však už oficiálně není podporován.

Další informace:

- http://www.asterisk.org/, http://www.asteriskguru.com/tutorials/
- http://www.abclinuxu.cz/serialy/asterisk-voip-ustredna
- http://www.root.cz/serialy/pobockova-voip-ustredna-asterisk/
- http://www.freepbx.org/
- http://www.cisco.com/en/US/products/sw/voicesw/ps556/index.html
- http://www.microsoft.com/responsepoint/
- http://www.ustredny.cz/

Kapitola 5

Bezdrátové a mobilní sítě

5.1 Bezdrátové technologie

Přenosové technologie dělíme podle přenosového prostředku do dvou kategorií:

- *kabelové* (wired, „drátové“) – přenos probíhá přes kabel (optický nebo metalický),
- *bezdrátové* (wireless) – přenos probíhá bez použití kabelu.

V této kapitole se budeme věnovat bezdrátovým technologiím.

Jaké přenosové médium tedy u bezdrátových technologií používáme? Mohli bychom říci, že vzduch, víceméně je to pravda, ale skutečným nosným médiem pro signál je to, na co signál modulujeme. Jsou tyto možnosti:

- *rádiové vlny* o určité frekvenci – Wi-fi, WiMAX, Bluetooth, NFC, ZigBee apod.,
- *zvukové vlny* (sonická bezdrátová technologie) – například ultrazvuk,
- *světlo* (optická bezdrátová technologie) – laser, infračervené záření (IR), mávání vlajkou...

Nás bude zajímat především první možnost. Sítě založené na přenosu po rádiových vlnách mohou být různě velké – od osobních (PAN, například Bluetooth nebo NFC) přes lokální (LAN, například Wi-fi) a metropolitní (MAN, například WiMAX) až k rozlehlým (WAN, mobilní sítě).

WLAN (Wireless LAN) – touto zkratkou označujeme bezdrátové lokální sítě. Pozor, nepleťte si tuto zkratku s VLAN, jsou to dva naprosto rozdílné pojmy.

Bezdrátové sítě můžeme také rozdělit podle míry mobility:

- *mobilní* – připojená klientská zařízení i při relativně rychlém pohybu neztratí signál, sem řadíme především mobilní WAN sítě typu GPRS, EDGE, LTE apod.,
- *fixní* – připojená klientská zařízení mohou při rychlejším pohybu ztratit signál, což se stává například u Wi-fi.

Nic není jen černé nebo jen bílé. I pro Wi-fi existují standardy, které tuto technologii přibližují k těm mobilním, ale v základu se pořád jedná o fixní bezdrátovou technologii.

Co se týče frekvenčních pásem, ve kterých bezdrátové rádiové sítě vysílají, většina využívá bezlicenční pásmo, ale některé vysílají v licencovaném frekvenčním pásmu.

Bezlicenční frekvenční pásmo je frekvenční pásmo takové, že pokud zařízení vysílá v tomto pásmu, nemusí jeho provozovatel žádat o povolení, pokud dodržuje určitá pravidla stanovená v generální licenci – tzv. *všeobecné oprávnění*.

Přesto je třeba určité limity dodržovat i v bezlicenčních pásmech, jejich použití je podmíněno dodržením podmínek *Všeobecného oprávnění k využívání rádiových kmitočtů* (celý název je delší, viz zdroj dále). Omezení jsou především ve smyslu dodržení maximálního vyzářeného výkonu antén, což má zmenšit pravděpodobnost vzájemného rušení zařízení a také jsou zde určité medicínské důvody (vysokofrekvenční záření je ve velkých dávkách nebezpečné pro lidský organismus).

Pokud dojde k vzájemnému rušení zařízení, obvykle platí pravidlo „kdo dřív přijde, může zůstat“. V reálu bohužel často platí „moudřejší ustoupí“, nicméně možnost stížnosti na ČTÚ je.

Další informace:

Text *Všeobecného oprávnění k využívání rádiových kmitočtů a k provozování zařízení pro širokopásmový přenos dat na principu rozprostřeného spektra nebo OFDM v pásmech 2,4 GHz a 5 GHz* najdeme například na http://www.ctu.cz/1/download/Opatreni%20obecne%20povahy/VO_R_12_08_2005_34.pdf.

Licencované frekvenční pásmo – uživatel takového zařízení si musí zajistit *individuální oprávnění* k vysílání v daném frekvenčním pásmu v dotyčné oblasti, tato služba může být placená. Má také právo na ochranu v případě, že v jeho přiděleném frekvenčním pásmu vysílá někdo jiný a tedy ruší signál.

Licencované frekvenční pásmo	**Bezlicenční frekvenční pásmo**
individuální oprávnění pro jednoho žadatele	všeobecné oprávnění (nepodáváme žádost)
obvykle zpoplatněno	obvykle bez poplatku za frekvenci
ochrana před zneužitím, rušením	bez právní ochrany před zneužitím, rušením
obvykle je možné poskytovat i garanci kvality služeb	obvykle se neposkytuje garance kvality služeb

Tabulka 5.1: Srovnání licencovaného a bezlicenčního frekvenčního pásma

Určení bezlicenčních a licencovaných frekvenčních pásem je v různých zemích různé, u nás je stanovuje a dozoruje Český Telekomunikační úřad (ČTÚ).

5.2 Wi-fi

Technologie Wi-fi (Wireless Fidelity) je bezdrátovou technologií pracující v bezlicenčním pásmu, a to kolem 2,4 GHz a 5 GHz (podle konkrétního podstandardu).

Základním standardem pro Wi-fi je IEEE 802.11, podstandardy (dodatky) pak k tomuto označení přidávají jedno- či dvoupísmenné zkratky (abeceda je krátká, jednopísmenné už nestačily). O standard a kompatibilitu zařízení (včetně certifikace) se stará Wi-fi Alliance.

Wi-fi síť může mít jednu ze dvou podob, resp. pracuje v jednom ze dvou režimů:

- *ad-hoc* – propojíme přímo dvě koncová zařízení, nepoužijeme žádný aktivní síťový prvek (DCE),
- *infrastruktura* – použijeme (minimálně jeden) aktivní síťový prvek (DCE), žádná dvě koncová zařízení nekomunikují přímo, veškerá komunikace jde přes DCE.

IEEE 802.11 implementuje jen spodní podvrstvu L2, tedy MAC podvrstvu. Horní podvrstvu nechává na protokolu LLC (IEEE 802.2), což znamená, že používáme LLC rámce (jak bylo naznačeno v druhé kapitole od strany 32), které zapouzdřujeme do MAC rámce podle IEEE 802.11.

Dále si popíšeme chování zařízení podle standardu IEEE 802.11 a dodatků, nejdřív na vrstvě L1, a následně na vrstvě L2.

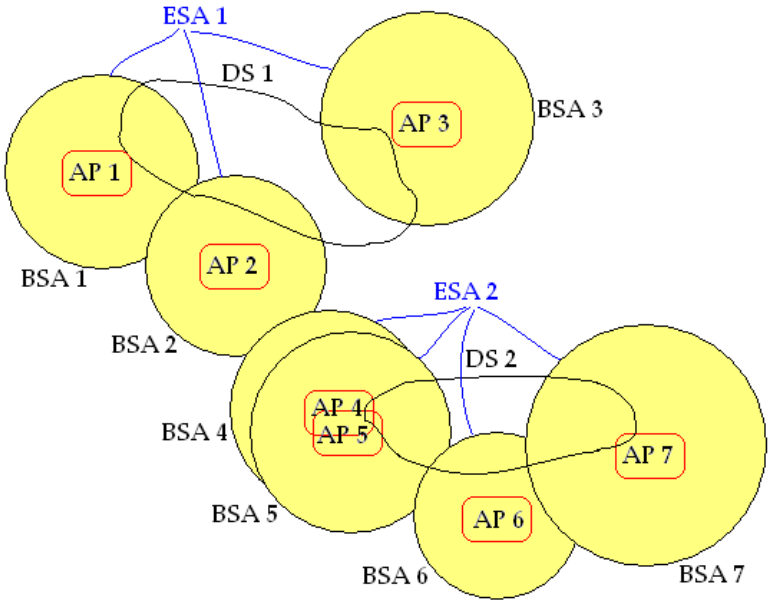
5.2.1 Access point

✎ Aktivním síťovým prvkem je *access point* (AP, přístupový bod). V základu toto zařízení pracuje na vrstvě L2 (a taky samozřejmě L1), tedy protokol IEEE 802.11 se svými podstandardy implementuje vrstvy L1 a L2 (jako Ethernet). Počítá se s podsunutím pod TCP/IP či jiný síťový zásobník pro vrstvy L3–L7.

✎ Ovšem DCE může podporovat i protokoly vyšších vrstev než jen L2, pak samozřejmě poskytuje další funkce související právě s vyššími vrstvami, nebo naopak může implementovat jen vrstvu L1. Takže DCE ve Wi-fi síti může pracovat v těchto *režimech*:

- *Access Point* (AP) – základní varianta, jednoduše implementuje vrstvy L1 a L2. Je to obdoba switche z ethernetových sítí.
- *Wi-fi router* – je to AP s přidanou funkcionalitou vrstvy L3. Díky tomu, že „vidí“ i do záhlaví IP paketů, dokáže směrovat a nabízí funkce typu překladu adres (NAT), DHCP serveru, hardwarového firewallu (filtruje podle IP záhlaví) apod.
- *Wi-fi repeater, extender* – pracuje jen na vrstvě L1, tedy žádné rámce. Přijme signál a zesílený ho znovu odvysílá, případně znovu vygeneruje. Tento typ zařízení má předně problémy s kompatibilitou (zejména při komunikaci mezi zařízeními různých výrobců), a navíc snižuje propustnost sítě (přijímá a vysílá obvykle v tomtéž pásmu, takže propustnost se může snížit i na polovinu). V domácnostech je to celkem užitečný prvek pro zvýšení dosahu signálu.
- *AP klient* – komunikuje bezdrátově s plnohodnotným AP, k němu jsou připojeni Wi-fi klienti obvykle kabelem. Na rozdíl od předchozího řešení zde tedy máme kabely, vzhledem k AP se tváří jako klientské zařízení a tedy jen zprostředkovává komunikaci „pravých“ klientů s AP. Nedochází k tak velkému snížení propustnosti sítě.
- *WDS* (Wireless Distribution System) – propojíme více AP do sítě, abychom pokryli větší prostor, pohybující se klient je „předáván“ mezi propojenými AP. Není to standardizované řešení, jsou problémy s kompatibilitou zařízení od různých výrobců. Navíc se podstatně snižuje propustnost sítě, podobně jako u repeateru.
- *Brána* – zprostředkovává komunikaci s jinou sítí pracující s odlišnými protokoly. Ve formě modulu bývá tato funkcionalita často přítomna v ADSL/VDSL Wi-fi routerech, přičemž modul propojuje místní síť (WLAN) s přístupovou sítí (ADSL nebo VDSL). Zařízení má tedy WAN port (tj. port vedoucí do přístupové sítě) daného typu, například ADSL (to by byl port s rozhraním RJ-11, kabel by vedl do zásuvky na pevnou linku).
- *WISP AP* (Wireless Internet Service Provider AP) – bezdrátový je WAN port (tj. se svým poskytovatelem internetu komunikujeme bezdrátově), ktežto porty pro lokální síť jsou například ethernetové.

✎ AP (ať už jakéhokoliv typu) vysílá signál, oblast pokrytou tímto signálem nazýváme *buňka* nebo také *základní oblast služeb* (BSA – Basic Service Area). Pokud propojíme víc AP do distribučního systému (WDS), pak oblast pokrytou signálem zapojených AP nazýváme *rozšířená oblast služeb* (ESA – Extended Service Area). Schéma vidíme na obrázku 5.1.



Obrázek 5.1: Schéma bezdrátové sítě

Distribuční systém (zde WDS, Wireless Distribution System) tedy funguje jako páteřní síť propojující přístupové body. Obecně lze použít distribuční systém nejen pro propojení (bezdrátových) přístupových bodů, ale také třeba pro propojení dvou metalických lokálních sítí. Nejjednodušší distribuční systém lze vytvořit tak, že přístupový bod jedné BSA pracuje jako běžná stanice v BSA druhého přístupového bodu (opakovač), ale ne každý přístupový bod podporuje režim opakovače.

5.2.2 Fyzická vrstva

Na fyzické vrstvě se rozlišuje více různých podstandardů s hodně odlišnými vlastnostmi, přičemž čas od času přibude nový podstandard.

Specifikace	**Frekvence**	**Max. propustnost**	**Ozn.**
IEEE 802.11	2,4 GHz	2 Mb/s	
IEEE 802.11a	5 GHz	54 Mb/s	Wi-Fi 1
IEEE 802.11b	2,4 GHz	11 Mb/s	Wi-Fi 2
IEEE 802.11g	2,4 GHz	54 Mb/s	Wi-Fi 3
IEEE 802.11n	2,4 GHz, 5 GHz	250 Mb/s na anténu, celkem až 600 Mb/s	Wi-Fi 4
IEEE 802.11ac	5 GHz	433 Mb/s na anténu a stream	Wi-Fi 5
IEEE 802.11ad	60 MHz	cca 7 Gb/s, podle použitých kanálů	WiGig
IEEE 802.11ax	2,4 GHz, 5 GHz	jednotky Gb/s	Wi-Fi 6

Tabulka 5.2: Podstandardy pro fyzickou vrstvu Wi-fi

V tabulce 5.2 je přehled existujících standardů pro fyzickou vrstvu (ve skutečnosti je jich mnohem více, ale zbývající jsou spíše doplňkové podstandardy).

V reálu se rychlost snižuje například tím, že je nutné komunikovat v obou směrech, což je při využití téže frekvence problém, může být asociováno větší množství stanic, svůj vliv má také vzdálenost mezi komunikujícími uzly a případné překážky.

Poznámka:

Zatímco u Ethernetu by znamenal údaj 54 Mb/s u portu to, že propojená zařízení opravdu mohou touto rychlostí komunikovat, u Wi-fi tomu tak není. Daná propustnost platí pro všechna zařízení v buňce a oba směry komunikace v souhrnu, takže čím víc asociovaných zařízení, tím menší propustnost na každé z nich vyjde.

Problém sdílení (bezdrátového) portu částečně řeší použití více antén, přičemž každá z nich vysílá a přijímá na jiné frekvenci.

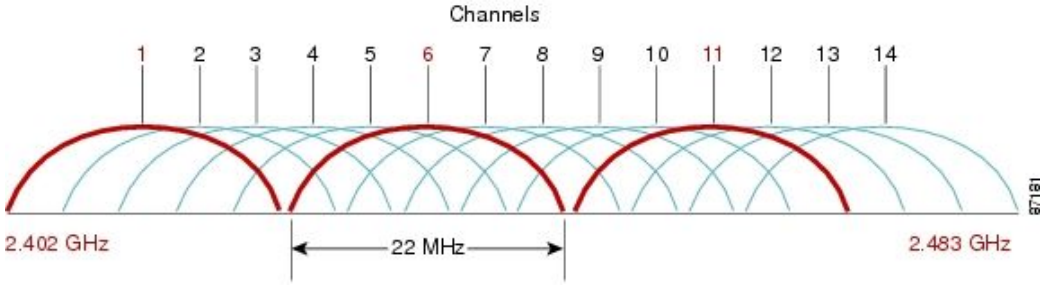
Z tabulky 5.2 se dnes používají jen některé specifikace. První uvedená byla velmi brzy aktualizována dalšími podstandardy, IEEE 802.11ad je zase natolik specifická, že se s ní moc nesetkáváme (navíc při tak vysoké frekvenci má dosah pouze cca 10 m bez překážek).

Zařízení podporující standard pro Wi-fi mívají v označení určeno, které frekvence a rychlosti podporují. Například IEEE 802.11b/g/n znamená, že zařízení má implementovány podstandardy b, g, n z výše uvedených, kdežto IEEE 802.11a/n/ac má implementovány podstandardy a, n, ac.

IEEE 802.11. Původní znění standardu z roku 1997 bylo již po dvou letech aktualizováno, na trhu se objevilo jen málo těchto produktů.

IEEE 802.11b. Tento předpis stanovuje komunikaci na frekvencích kolem 2,4 GHz, přičemž propustnost může být až 11 Mb/s (z toho 30–40 % je režie). Dnes se v zařízeních objevuje jeho implementace jen z důvodů kompatibility s IEEE 802.11g/n.

Na fyzické vrstvě se používá multiplexovací metoda *DSSS* (Direct Sequence Spread Spectrum) určující úpravu posloupnosti bitů s modulací *QPSK* (Quadrature Phase Shift Keying), třebaže se objevila zařízení podporující modulaci 64-QAM. Metoda je robustní, data se přenášejí redundantně, aby při jejich poškození bylo co nejjednodušší zjistit chybu.



Obrázek 5.2: Frekvenční spektrum pro IEEE 802.11b[1]

Na obrázku 5.2 vidíme frekvenční rozsah využívaný podle IEEE 802.11b. Rozhodně nejde pouze o frekvenci 2,4 GHz, ale o interval frekvencí v blízkém okolí.

Celé spektrum se dělí na *kanály*, šířka kanálu je 22 MHz. Zařízení (DTE i AP) má určen konkrétní kanál, na kterém komunikuje. AP si svůj kanál obvykle volí sám podle analýzy zarušení spektra, ale v některých případech je třeba tuto hodnotu určit ručně (AP totiž nemůže „tušit“, které kanály jsou rušeny na straně klientů).

Ovšem pozor, v různých zemích mohou být stanoveny odlišné „povolené“ kanály. Zařízení homologované pro provoz v ČR by mělo komunikovat pouze na kanálech povolených ČTÚ (s čísly 1–13).

[1] Zdroj: http://www.cisco.com/en/US/docs/solutions/Enterprise/Mobility/emob30dg/RFDesign.html

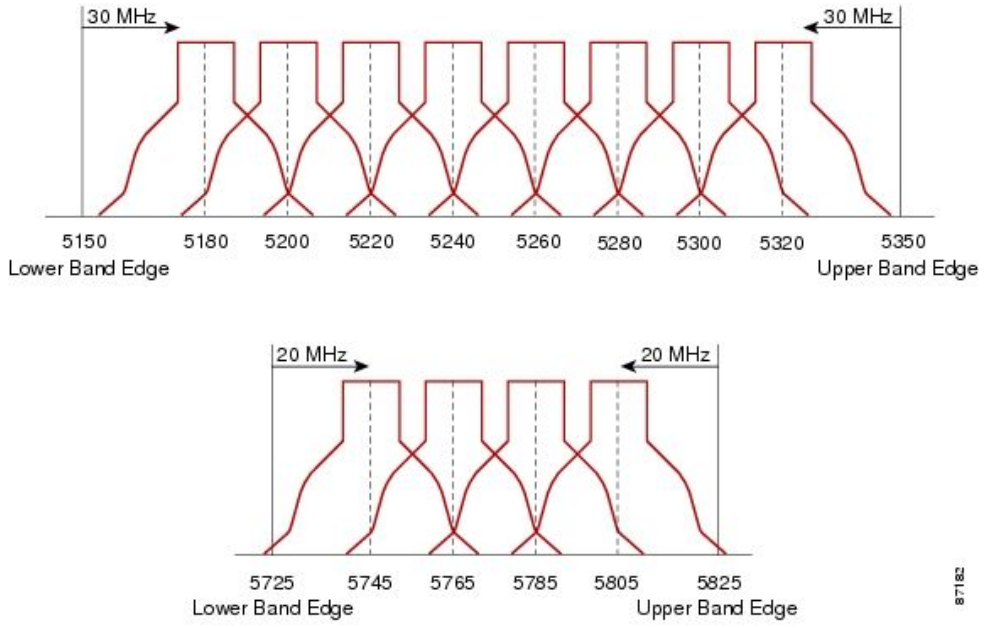
Pokud zařízení komunikuje na konkrétním kanálu, ve skutečnosti je signál i na okolních kanálech. Teoreticky je zabráno celkem pět kanálů (například pokud zařízení komunikuje na kanálu 6, pak zabírá kanály 4–8, podle „obloučku“ na obrázku), ale praxe bývá trochu drsnější – na vzdálenějších frekvencích síla signálu rozhodně neklesá tak prudce jak oblouček předepisuje, slabé rušení je ještě i o několik kanálů dál. Kanály se překrývají v praxi poněkud víc než jak říká teorie.

Poznámka:

Takže kolik zařízení může ve vzájemném dosahu komunikovat v pásmu 2,4 GHz bez vzájemného rušení? Teoreticky tři (kanály 1, 6 a 11). Prakticky se v tomto poměrně úzkém spektru mohou navzájem rušit dokonce i pouhá dvě zařízení, záleží na síle jejich signálu a dalších vlastnostech antény.

IEEE 802.11a. Přesuňme se nyní do pásma 5 GHz. Zařízení s anténami pracujícími v tomto frekvenčním pásmu mají výhodu v tom, že toto pásmo není tak zarušeno jako 2,4 GHz. Zařízení podle standardu IEEE 802.11a mohou komunikovat teoreticky rychlostí až 54 Mb/s (opět jde o relativní údaj, to bude pro všechny specifikace stejné).

Nevýhodou je, že přidání podpory 5 GHz znamená navýšení ceny zařízení. Specifikace je z roku 1999, ale první zařízení s její podporou se objevila až na konci roku 2001.



Obrázek 5.3: Frekvenční spektrum pro IEEE 802.11a[2]

Na obrázku 5.3 je naznačeno rozdělení frekvenčního spektra pro IEEE 802.11a do kanálů. Kanály o šířce 20 MHz se překrývají trochu jiným způsobem (méně) než u IEEE 802.11b a spektrum je rozděleno do dvou oddělených částí.

Používá se multiplexovací metoda OFDM, která je efektivnější než DSSS. Vyšších rychlostí oproti IEEE 802.11b je dosaženo především díky efektivnější modulaci a přechodu na vyšší frekvence (při dvojnásobné frekvenci dokážeme za stejnou časovou jednotku přenést dvakrát víc dat, pokud jsou

[2]Zdroj: http://www.cisco.com/en/US/docs/solutions/Enterprise/Mobility/emob30dg/RFDesign.html

všechny ostatní parametry shodné). Na subnosných se moduluje některou z více různých metod (podle toho, jestli se upřednostňuje robustnost nebo rychlost) – BPSK, QPSK nebo QAM, pro QAM se dají použít různé varianty (robustnější 16-QAM, rychlejší 64-QAM).

IEEE 802.11g. Tato specifikace je z června 2003 a je zpětně kompatibilní s IEEE 802.11b v tom smyslu, že se používá stejné frekvenční spektrum (viz obrázek 5.2). Šířka kanálů je 20 MHz (pozor, změna). Pro multiplexování se používá metoda OFDM, subnosné se také modulují stejně jako u IEEE 802.11a, většinou 64-QAM nebo v zarušeném prostoru některou její robustnější variantou.

Jak vidíme, mezi IEEE 802.11a a IEEE 802.11g je rozdíl především v použitém frekvenčním pásmu, jinak rychlost a modulace jsou podobné. Je tady však ještě jeden podstatný rozdíl – podpora u výrobců a cena. Géčko se stalo cenově lépe dosažitelným a v minulosti jej implementovala téměř všechna bezdrátová síťová zařízení.

Co se stane, když se do buňky pracující podle IEEE 802.11g asociuje zařízení zvládající pouze IEEE 802.11b? To záleží na konfiguraci AP v centru buňky.

- Výchozí bývá *Legacy Mode* fungující takto:
  - zařízení, která mají implementovány oba podstandardy, přejdou na IEEE 802.11b, čímž se sníží propustnost v celé buňce,
  - zařízení, která mají implementován IEEE 802.11g, ale ne 802.b, budou odpojena.
- Pokud máme nastaven *Mixed Mode*, všechna zařízení pracují na nejvyšším možném podstandardu, který je na nich implementován. Komunikovat tedy mohou všichni tak rychle, jak dokážou (mínus vyšší režie), ale pro AP je tento režim náročnější.
- Třetí možnost je, že starým zařízením bez podpory „géčka“ nebude asociace povolena.

Toto chování se nastavuje v konfiguraci AP, a není řečeno, že dotyčný AP podporuje opravdu všechny tyto možnosti.

IEEE 802.11n (Wi-Fi 4). Tato specifikace jako první umožnila pracovat v obou frekvenčních pásmech, navíc bylo možné použít více antén a celkově až 4 komunikační streamy. Maximální propustnost se zvýšila až na 600 Mb/s (součet přes všechny antény), čehož bylo dosaženo takto:

- typicky se používá více než jedna anténa,
- používáme obě frekvenční pásma, a to i paralelně,
- obvyklá šířka kanálu je 40 MHz, tedy do jednoho kanálu se „vejde“ více dat za časovou jednotku,
- kvalitnější modulace (multiplexování taky OFDM, modulace taky 64-QAM, ale s jinými parametry a méně často se padá na pomalejší modulaci),
- změny jsou i na vrstvě L2.

Čím víc antén, tím větší propustnosti teoreticky dosahujeme (prakticky však nemůžeme brát jen součet přes všechny antény).

Poznámka:

Implementace na straně výrobců hlavně ze začátku poněkud pokulhávala – například existují zařízení, která sice implementují IEEE 802.11n, ale pouze v pásmu 2,4 GHz, případně zařízení s pouze jednou anténou, nebo taková zařízení, která implementují IEEE 802.11n v obou pásmech, ale nedokážou je používat paralelně v jednom okamžiku (tj. pracují buď na 2,4 GHz nebo na 5 GHz, ale ne v obou pásmech zároveň).

Zatímco specifikace IEEE 802.11a/b/g se liší opravdu jen na fyzické vrstvě (podvrstvu MAC vrstvy L2 mají stejnou), specifikace IEEE 802.11n mění i podvrstvu MAC.

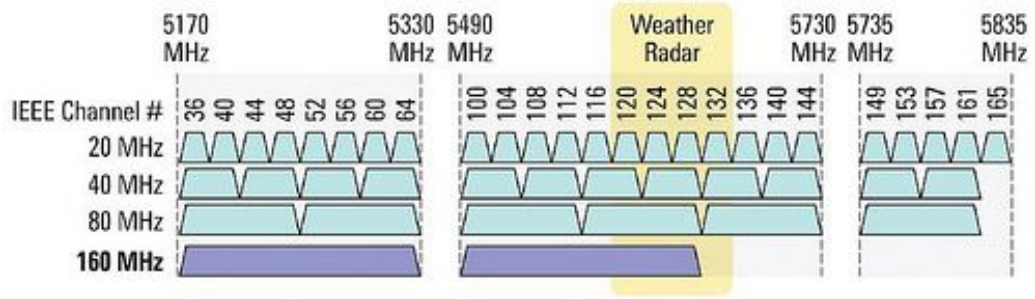
Při souběhu zařízení podporujících různé podstandardy (v tomtéž frekvenčním pásmu) může AP běžet v jednom z těchto tří módů:

- *Legacy Mode* – používá se jen starší standard (z průniku podporovaných podstandardů přes všechna zařízení v buňce). Obvykle rychlost spadne na tu nejnižší společnou.
- *Mixed Mode* – každé zařízení v buňce funguje podle svého nejlepšího podstandardu, pokud ovšem to dovoluje množství a nastavení antén AP.
- *Greenfield* – opak Legacy módu, používá se jen IEEE 802.11n a všechna zařízení, která ho nepodporují, mají smůlu.

Záleží samozřejmě, co vše dotyčný AP dovoluje nastavit. Z hlediska propustnosti sítě je nejlepší třetí možnost.

IEEE 802.11ac (Wi-Fi 5). Tento standard již plně přešel do pásma 5 GHz. Používá multiplexování OFDM s vnitřní modulací QAM, ale opět s jinak nastavenými parametry, aby se co nejvíc zvýšilo množství dat přenesených za časovou jednotku. Vliv na propustnost samozřejmě má i výhradní použití vyšších frekvencí.

Frekvenční pásmo je ve skutečnosti o něco širší než u IEEE 802.11a. Šířka kanálů může být 20 MHz, 40 MHz, 80 MHz nebo 160 MHz, konkrétní hodnotu volíme podle očekávaného množství klientů.



Obrázek 5.4: Frekvenční spektrum pro IEEE 802.11ac[3]

Podstandard IEEE 802.11ac navyšuje rychlost oproti starším následovně:

- používá lepší modulaci,
- dovoluje více antén (až 8) a zachází s nimi mnohem optimálněji (víc v další sekci),
- pracuje ve frekvenčním pásmu 5 GHz, přičemž šířka kanálu může být určena adaptivně podle vytížení sítě.

Používá se multiplexování OFDM, subnosné jsou pak zpracovány modulací QAM, a to až 256-QAM. V jednotlivých generacích se tento parametr měnil následovně:

- 802.11g: 16-QAM (jeden symbol = 4 bity, $2^4 = 16$ možností pro hodnotu symbolu),
- 802.11n: 64-QAM (jeden symbol = 8 bitů),
- 802.11ac: 256-QAM (16 bitů).

Důsledkem bylo postupné zvyšování efektivity přenosu (víc přenesených dat za jednotku času).

[3] Zdroj: http://www.dailywireless.org/2012/04/page/3/

Uvedené platí pro víceméně ideální podmínky. Při rušení nebo větší vzdálenosti, kdy se zhoršuje kvalita doručeného signálu a dochází k častějšímu ztrácení rámců, se přechází na robustnější modulaci (symbol je kódován do více bitů).

IEEE 802.11ad (WiGig) je tak trochu úkrok stranou, není to ani tak další generace, jako spíše specialita pro specifické použití. Hlavním účelem bylo vytvořit bezdrátovou náhradu HDMI kabelu, tedy vysokorychlostní přenos multimediálních dat na krátkou vzdálenost v řádech metrů. Typický dosah je do 10 metrů, a to bez překážek. Podle novější specifikace je k dispozici až 6 kanálů (každá země to má jinak, například Čína pouze 2 kanály, USA všech 6, v Evropě používáme 4 kanály) o šířce 2,16 GHz.

Použití frekvenčního rozsahu kolem 60 GHz totiž má jak kladné, tak i záporné následky:

+ teoretická propustnost až v jednotkách GHz, což postačuje pro přenos nekomprimovaného UHD videa,
− vyšší frekvence jsou náchylnější na rušení a navíc špatně procházejí překážkami, dokonce i vzduch má na signál tlumící účinek.

Typické použití je bezdrátové propojení výpočetního zařízení (počítač, notebook, HTPC apod.) se zobrazovacím zařízením (televize, projektor apod.)., ovšem často narážíme na to, že zařízení tento standard nepodporují.

IEEE 802.11ax (Wi-Fi 6) používá frekvenční spektrum kolem 2,4 GHz a 5 GHz, stejně jako IEEE 802.1n. Kanály jsou široké až 160 MHz (taky záleží, ve které části frekvenčního spektra), modulace je až 1024-QAM s multiplexem OFDMA (stejně jako u LTE), počet paralelních streamů se navýšil na 8 (to vše bude diskutováno dále). Pro zabezpečení se počítá s WPA3.

Maximální teoretická propustnost je oproti Wi-Fi 5 několikanásobně vyšší, ale samozřejmě záleží na počtu antén, šířce a využití kanálů, atd. Zatímco maximum pro Wi-Fi 5 je 3,5 Gb/s (160MHz kanály, 4 streamy), u Wi-Fi 6 to je 9,6 Gb/s (souhrn za všechny antény, 160MHz kanály, 8 streamů).

W-Fi 6 má ještě jednu vlastnost navíc – počítá se se zapojením IoT zařízení (Internet věcí), čemuž je přizpůsobeno mnohé, včetně nové funkce *TWT* (Target Wake Time) – inteligentní plánování uspávání a probouzení IoT zařízení.

Další informace:

- https://www.wi-fi.org/discover-wi-fi (v menu vpravo jsou různé standardy a další témata)
- https://www.cisco.com/c/en/us/products/collateral/wireless/white-paper-c11-740788.html (srovnání IEEE 802.11ax a IEEE 802.11ac)

5.2.3 Interference

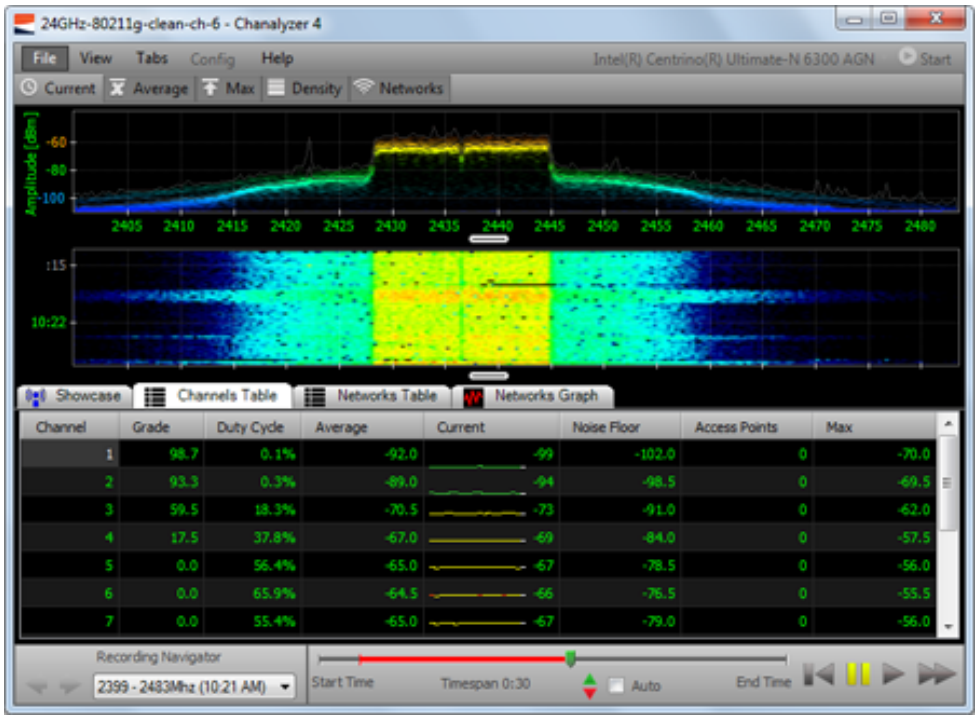
Interference (rušení) je problém hlavně v pásmu kolem 2,4 GHz. Ve stejném frekvenčním spektru pracují také bluetooth zařízení, mikrovlnné trouby (pokud si myslíte, že při provozu je mikrovlnka tak důkladně izolovaná, že ven se žádný signál nedostane, pak vězte, že jste na omylu) a většinou i dětské chůvičky a různá další zařízení (je nelicencované, a zrovna oblíbená frekvence), takže vzájemné rušení se ani zdaleka netýká pouze Wi-fi zařízení.

Teoreticky by bylo řešením navolit na blízkých AP kanály tak, aby se dotyčné pětice co nejméně překrývaly (tj. 2 až 3 komunikující AP, které se „vidí“ a přitom se navzájem neruší), ale to je opravdu

jen teorie. Ve skutečnosti (díky fyzikálním zákonům) běžná komunikace na kanálu „zamořuje“ nejen dva okolní kanály na každé straně, ale bohužel i vzdálenější, i když v menší míře. Na obrázku 5.2 vidíme tři zvýrazněné obloučky, které se nepřekrývají (1., 6. a 11. kanál). V reálu je spodní část obloučků značně roztáhnuta, šum zasahuje mnohem dále („do ztracena“), jak vidíme na obrázku 5.5 (výstup z aplikace Wi-Spy Chanalyzer 4 pro jeden zdroj signálu podle IEEE 802.11g, v horní části si všimněte amplitudy v různých kanálech, kanály jsou označeny frekvencí v MHz).

Tj. „vhodné“ rozdělení kanálů prakticky neexistuje, dva AP se budou navzájem rušit, i když budou pracovat na „dostatečně vzdálených“ kanálech. Pokud tedy nemáme jinou možnost, lze dva AP umístit tak, aby se jejich dosahy prolínaly, kanály zvolíme co nejdál od sebe (nicméně automaticky se to tak obvykle také děje), ovšem musíme počítat s jistým snížením rychlosti. Je zajímavé, že za určitých okolností je paradoxně lepší pro více AP zvolit tentýž nebo hodně blízký kanál, pro další informace viz diplomovou práci

Jedlička, T. *Diagnostika bezdrátových sítí.* Diplomová práce na ÚI FPF SU. Opava, 2012.



Obrázek 5.5: Wi-Spy Chanalyzer[4]

Výše popsané rušení je také jedním z důvodů, proč při použití WDS musíme počítat s nižší propustností sítě. Wi-fi síť může být rušena také z jiných zdrojů (čímkoliv na blízké frekvenci), například mikrovlnnou troubou, naštěstí tento zdroj rušení nebývá používán dlouhodobě.

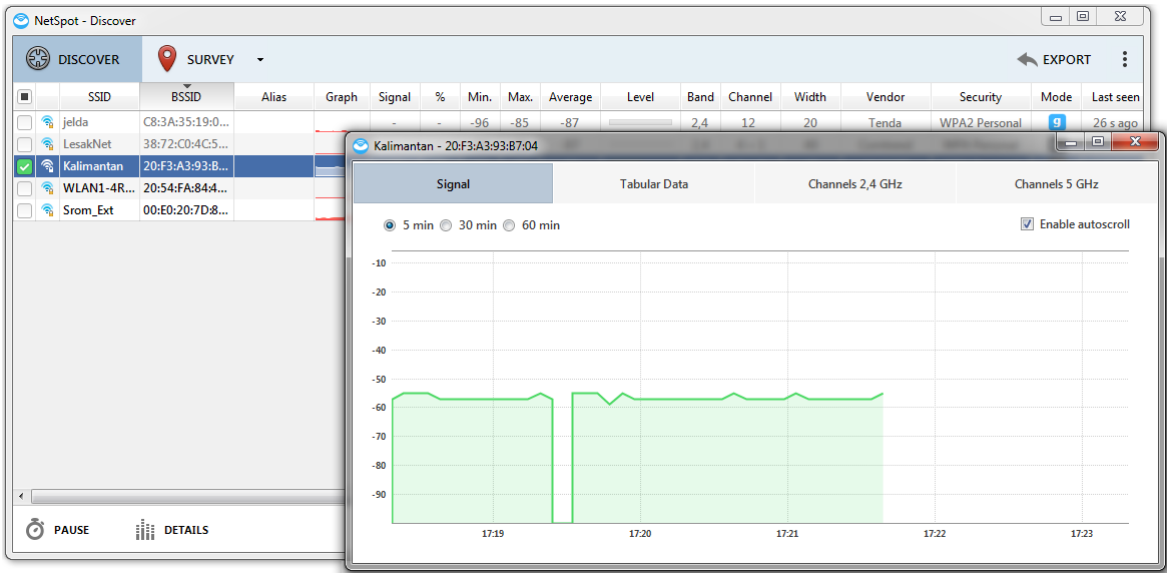
5.2.4 Diagnostické nástroje pro fyzickou vrstvu

Aplikace *inSSIDer* pro Windows a MacOSX od společnosti MetaGeek byla původně volně ke stažení, teď už se bohužel jedná o shareware.

[4] Zdroj: http://www.wi-spy.com.au/

Velice se InSSIDeru podobá nástroj *Acrylic WiFi*, který ve volně stažitelné verzi Home (pouze pro domácí použití) umí vpodstatě totéž a také jeho rozhraní je podobné.

Velmi nadějně vypadá volně šiřitelná aplikace *NetSpot* pro Windows a MacOS X, která má dokonce mnohem více funkcí než InSSIDer (například mapování pokrytí signálem v oblasti či hledání problémů v síti).



Obrázek 5.6: Netspot

Program *WifiInfoView* od Nirsoftu je volně šiřitelný nástroj pro zjišťování informací o dostupných bezdrátových sítích. Dá se spustit buď jako aplikace s oknem (vidíme seznam dostupných sítí, jejich vlastnosti a ve spodním podokně podrobnosti), tak i s různými dodatečnými parametry (na webu projektu jsou podrobné informace).

Pro Linux existují obdobné nástroje jako InSSIDer, například LinSSIDer a iwScanner. Pro Android lze stáhnout aplikaci *Wifi Analyzer*. Sice se nedostaneme k tolika informacím jako obdobné aplikace pro „nemobilní“ operační systémy (už z principu, to by ten Android musel být rootnutý), ale pro představu o frekvenčním spektru to bohatě stačí.

Existují i další nástroje, které umožňují zkoumat frekvenční spektrum. Většinou jde o komerční nástroje anebo sice volně šiřitelné, ale vyžadující kompatibilní hardware (tj. záleží na štěstí, jestli takový máme). Z nejznámějších:

- *NetStumbler* – volně šiřitelný analyzátor spektra,
- *Wi-Spy* – komerční nástroj (analyzátor spektra včetně rozdělení na jednotlivé kanály, set se skládá z upravené Wi-fi síťové karty do USB konektoru a speciálního softwaru),
- *Chanalyzer* je sice komerční nástroj (jako součást Wi-Spy), ale existuje volně šiřitelná varianta; je to výkonný spektrální analyzátor (viz obrázek 5.5),
- *AirMagnet WiFi Analyzer* je komerční nástroj funkčností podobný nástroji Wi-Spy, taktéž spektrální analyzátor (také vyžaduje speciální hardware),
- *Xirrus Wi-fi Inspector* – volně šiřitelný nástroj na zjišťování dostupných AP a jejich vlastností (firma Xirrus se stala součástí jiné firmy a tento nástroj už není na jejich stránkách, nicméně na

download serverech je ještě dostupný),

- *Iperf* (taktéž volně šiřitelný, ovládá se v textovém shellu) slouží k ladění parametrů sítě a je to užitečný také jako pomocný nástroj při diagnostice sítě, dokáže odchytávat pakety (sniffer) a generovat „přirozený provoz“ na síti (užitečné při testování),
- *Jperf* je volně šiřitelná grafická nástavba programu Iperf (tj. většina lidí si s Iperfem instaluje Jperf).

Existují nástroje určené vysloveně pro mapování signálu, například *Ekahau Heatmapper*, *Ekahau Site Survey*, *Meraki Wi-fi Mapper*, *Aerohive Free Wi-fi Planner*, *Cisco Clean Air*, *VisiWave Site Survey*, *RF3D WifiPlanner*.

Další informace:

- https://www.acrylicwifi.com/en/wlan-software/wlan-scanner-acrylic-wifi-free/
- https://www.netspotapp.com/
- http://www.nirsoft.net/utils/wifi_information_view.html
- http://alternativeto.net/software/inssider/
- http://www.stumbler.net
- https://shop.metageek.com/products/chanalyzer-essential/
- Airmagnet Wifi analyzer: http://www.nextlan.cz/?page_id=1498
- http://sourceforge.net/projects/iperf/, http://code.google.com/p/xjperf/

5.2.5 Signál a antény

Existují různé druhy antén – všesměrové s různými způsoby formování signálu či směrové určené pro point-to-point spoje. Antény mohou být buď *interní* (například v notebooku jsou antény obvykle vedeny podél displeje) nebo *externí* (vyměnitelné jsou obvykle připojeny koaxiálovým konektorem typu N nebo BNC).

U běžných malých všesměrových antén je signál nejsilnější v rovině kolmé na osu antény, tedy nakloněním antény můžeme například ovlivnit to, zda bude signál dostatečně silný v okolních podlažích budovy. Pokud chceme mít signál jen v rámci jednoho patra, necháme anténu směřovat nahoru nebo dolů, kdežto když má signál dosáhnout do horního nebo spodního patra, nakloníme anténu do úhlu 45 stupňů. Pozor, ve směru, do kterého anténa míří, je signál nejslabší!

Zařízení s jedinou anténou může fungovat jen v *poloviční duplexu*, tedy buď vysílat nebo přijímat, ale nikoliv obojí zároveň. Pokud ale má alespoň dvě antény, může pro každý směr použít jednu z nich, přičemž budou pracovat na různých kanálech, čímž implementujeme *plný duplex*.

Ve specifikaci IEEE 802.11a/b/g se setkáváme s *diverzitou*, tedy možností využití více antén pro frekvenční pásmo 2,4 GHz. Diverzita funguje tak, že při komunikaci s konkrétním zařízením se nejdřív přijímá všemi, a po vyhodnocení kvality přijatého signálu ze všech antén je pak určena jedna, která bude pro dané zařízení využívána.

Ve specifikaci pro IEEE 802.11n se objevuje nástupce diverzity – *MIMO* (Multiple Input, Multiple Output) [maimo] – určující využití více antén a algoritmus pro kombinování signálu z těchto antén. Na rozdíl od diverzity mohou (téměř) všechny antény fungovat zároveň.

Při použití MIMO jsou antény rozděleny na Tx (odesílající) a Rx (přijímající), abychom zajistili komunikaci v plném duplexu, a dalším parametrem je počet paralelních streamů (tedy kolik nezávislých kanálů se nám vejde do spektra). Zapisuje se takto:

Tx		Rx		streamy
3	×	3	:	2

což znamená 3 odesílající antény, 3 přijímající antény a 2 streamy.

Technologie MIMO má jednu slabinu – v jednom okamžiku může AP komunikovat maximálně s jediným klientem, třebaže více anténami.

To řeší technologie *MU-MIMO* (Multi-User MIMO) používaná ve specifikaci IEEE 802.11ac, a také například u metropolitních sítí WiMAX. V jednom okamžiku může AP paralelně komunikovat s více různými klienty (maximum je 8 streamů, pro každého klienta 2, tedy maximálně 4 klienti).

V implementaci je bohužel trochu zpoždění, tedy když kupujeme zařízení podle IEEE 802.11ac podporující MU-MIMO, může jít o jednu ze tří fází, obvykle druhou:

- Wave 1 – SU-MIMO (Single-User MIMO), tedy žádná paralelní komunikace, max. 3 streamy,
- Wave 2 – MU-MIMO (více klientů paralelně), max. 3–4 streamy,
- Wave 3 – max. 8 streamů (s tím se obvykle u Wi-Fi 5 nesetkáme).

V IEEE 802.11ax (Wi-Fi 6) se přechází z multiplexování OFDM na OFDMA. Co to znamená? Ve skutečnosti se liší i význam „konců“ těchto zkratek: OFDM je Orthogonal Frequency Division Multiplexing, OFDMA je Orthogonal Frequency-Division Multiple Access. Při OFDM je sice možné používat MU-MIMO, ale pouze ve formě, kdy různé kanály jsou přiřazeny různým koncovým zařízením. OFDMA jde dál: kanály dělí na subkanály (nazývají se RU = Resource Unit). Také v rámci subkanálů se používá ortogonální dělení, aby se jednotlivé subkanály daly jednoduše oddělit.

Zatímco IEEE 802.11ac (a také starší IEEE 802.11n) reálně používá max. čtyři streamy (Wave 2), u IEEE 802.11ax se setkáme s až osmi streamy. Počet streamů udává maximální počet (sub)kanálů, které lze používat paralelně (počítají se zvlášť pro různé směry). Například pokud jsou u Wi-fi routeru pracujícího podle IEEE 802.11ac k dispozici 4 streamy a v síti máme dva aktivní klienty, pak ke každému z nich mohou vést dva streamy (jeden pro každý směr). Ovšem pozor, klienti by v tom případě taky museli podporovat alespoň dva streamy.

Další informace:

- http://www.cisco.com/c/en/us/solutions/collateral/enterprise-networks/802-11ac-solution/q-and-a-c67-734152.html
- https://www.extremenetworks.com/extreme-networks-blog/ofdm-and-ofdma-subcarriers-what-are-the-differences/

5.2.6 Identifikátory a služby

Klientská zařízení potřebují identifikovat síť, do které se připojují nebo v ní pracují. Od toho máme tyto *identifikátory*:

- *SSID* (Service Set ID) je identifikátor celé sítě, název sítě. Je to řetězec o délce maximálně 32 znaků, který musí znát každé zařízení, které se do sítě přihlašuje.

- *BSSID* (Basic Service Set ID) je identifikátor access pointu. Obvykle se jedná o MAC adresu tohoto AP, tedy délka je 6 oktetů.
- *ESSID* (Extended Service Set ID) je identifikátor WDS, pokud máme víc AP propojených do distribučního systému.

AP může poskytovat více SSID, přičemž každé z nich představuje jednu bezdrátovou síť. Například můžeme mít zvlášť SSID pro hosty, to se hodí nejen ve firmách, ale i v domácnosti. Ve firmách se na různá SSID napojují různé VLAN (čímž se také určují přístupová oprávnění těch, kdo v dané bezdrátové síti s konkrétním SSID komunikují), v domácnosti se na routeru obvykle také dá nastavit alespoň stupeň důvěryhodnosti.

Definice (Beacon rámec)

Každý AP v pravidelných intervalech vysílá *Beacon rámec* („majákový“ rámec), ve kterém informuje o svých parametrech. Kromě jiného je v Beacon rámci řetězec SSID potřebný pro komunikaci s AP.

AP poskytuje v bezdrátové síti tyto služby:

- *autentizace* – rozhoduje, zda do své buňky a tím i do sítě pustí konkrétní koncové zařízení,
- *asociace* (přidružení) – pokud je koncovému zařízení povolen přístup, je třeba vytvořit vazbu mezi AP a tímto zařízením, tedy technicky zajistit zařízení možnost komunikace v buňce,
- *de-asociace* – zrušení této vazby.

5.2.7 Přístupová metoda

Standard IEEE 802.11 definuje také přístupovou (kolizní) metodu, která je používána na podvrstvě MAC vrstvy L2. Na rozdíl od Ethernetu je v bezdrátových sítích používána neustále, protože při sdíleném médiu se nikdy nedá zcela vyhnout kolizím. Navíc se používají mechanismy snižující na minimum riziko kolizí a řešící problém skrytých stanic.

Problém skrytých stanic nastává tehdy, když je buňka rozsáhlejší a dvě stanice patřící do téže buňky se navzájem „nevidí“ (ke každé z nich sice dosáhne signál z centrálního AP, ale nikoliv jejich signály navzájem). To je vcelku běžné, stanice mohou být od sebe příliš vzdáleny nebo navzájem zastíněny nábytkem či zdí, signál koncových zařízení taky bývá typicky slabší než signál AP.

Definice (Přístupová metoda CSMA/CA a mechanismus RTS/CTS)

Podle IEEE 802.11 se používá přístupová metoda *CSMA/CA*:

- CS (Carrier Sense) – naslouchá na nosné,
- MA (Multiple Access) – vícenásobný přístup (sdílené přenosové médium),
- CA (Collision Avoidance) – kolizím se vyhýbáme, tedy jim předcházíme (nestačí je jen detekovat).

Každé zařízení, které chce vysílat, musí nejdřív požádat AP o povolení k vysílání (to je krátký správní rámec) a až ho získá, může odesílat data. CSMA/CA se dá implementovat více různými způsoby, jeden z mechanismů je například *RTS/CTS* (Request to Send, Clear to Send):

- DTE vyšle signál RTS (v krátkém řídícím rámci), čímž požádá o povolení k vysílání dat,
- pokud je možné vysílání povolit, AP odešle žádajícímu DTE signál CTS (v řídícím rámci),

- DTE může vyslat datový rámec (MAC podle IEEE 802.11 s LLC rámcem, v něm jsou data),
- zpět je doručeno potvrzení (ACK), také v řídícím rámci.

Jak vidíme, ve Wi-fi se v takovém případě používá potvrzovaný přenos bez navázání spojení (na spoji).

I přes takto určenou přístupovou metodu ke kolizím může docházet, protože přenosové médium je sdílené. Dochází k nim například tehdy, pokud dvě zařízení zároveň vyšlou signál RTS nebo se takto potkají rámce s RTS a začátkem vysílání datového rámce.

Poznámka:

Mechanismus RTS/CTS víceméně řeší problém skrytých stanic – počet kolizí snižuje na minimum tím, že u krátkých řídících rámců je mnohem menší pravděpodobnost kolize než u velkých datových, a AP taky nepošle signál CTS s povolením vysílání, pokud zrovna na dané frekvenci komunikuje jiný (žadateli skrytý) uzel v síti.

5.2.8 Wi-fi rámce

Rozlišujeme tři typy MAC rámců podle IEEE 802.11:

- *datové rámce* (Data Frame) – obsahují payload v LLC rámci,
- *řídící rámce* (Control Frame) – podtypy:
  - RTS/CTS rámce,
  - ACK – potvrzení přijatých rámců,
- *rámce pro správu* (Management Frame) – sem řadíme
  - beacon rámec (AP se ohlašuje),
  - probe, probe response (zjišťování přítomnosti uzlů sítě a jejich možností),
  - association request, association response (žádost o asociaci od stanice, odpověď),
  - re-association request, response (přechod k jinému AP ve stejné ESS),
  - diassociation (ukončení spojení k AP),
  - authentication (žádost o autentizaci uzlu), de-authentication.

Uvnitř datového rámce najdeme LLC rámec (IEEE 802.2), ve kterém je obvykle zapouzdřen IP paket. Nicméně například ve Wiresharku nemáme obvykle šanci tento fakt zjistit, protože celý datový prostor je šifrován (včetně LLC záhlaví).

Než se dostaneme k formě samotného záhlaví rámce, podíváme se na práci s adresami uvedenými v rámci. Údaje o adresách se v záhlaví mění podle toho, kterou částí Wi-fi sítě právě rámec prochází. Rozlišujeme tyto případy:

1. Jsme v ad-hoc síti, tedy spoje vedou vždy právě mezi dvěma klientskými zařízeními, žádný AP na cestě není (nebo se jedná o rámec, který v principu míří jen k sousedovi a ne dále).
2. Jsme v síti typu infrastruktura, rámec je na spoji mezi zdrojem rámce (klientským zařízením) a některým AP.
3. Jsme v síti typu infrastruktura, rámec je na spoji mezi dvěma AP.
4. Jsme v síti typu infrastruktura, rámec je na spoji mezi některým AP a cílem rámce.

Takže záleží, na jakém typu spoje se právě rámec nachází, resp. mezi jakými dvěma typy uzlů v síti právě prochází (kdo je momentálním odesílatelem a příjemcem).

V záhlaví rámce máme kromě samotných adres tyto dva příznaky (tj. jednobitové hodnoty):

- `FromDS` – je nastaven na 1, pokud na daném spoji je momentálním odesílatelem AP (resp. DCE), a je nastaven na 0, pokud je odesílatelem klientské zařízení (DTE).
- `ToDS` – je nastaven na 1, pokud na daném spoji je momentálním příjemcem AP (DCE), a je nastaven na 0, pokud je příjemcem DTE.

Zkratka „DS“ v názvech příznaků je *distribuční systém*.

Z toho vyplývá, že v nastíněných čtyřech případech jsou tyto příznaky nastaveny takto:

1. V ad-hoc síti jsou oba příznaky vždy nastaveny na 0, protože žádné AP na cestě ani být nemohou.
2. Na spoji mezi zdrojem rámce (DTE) a nejbližším AP je `FromDS=0` (odesílatel je DTE), `ToDS=1` (momentální příjemce je AP).
3. Na spoji mezi dvěma AP jsou oba příznaky nastaveny na 1: `FromDS=1`, `ToDS=1`. Na obou koncích spoje jsou AP, a tedy jsme uvnitř distribučního systému.
4. Na spoji mezi některým AP a cílem rámce (DTE), tedy v závěru cesty, je `FromDS=1`, `ToDS=0`.

Nyní se zaměřme na *adresy*. V ethernetovém rámci máme vždy dvě adresy – MAC adresu cíle a MAC adresu zdroje. Ve Wi-fi rámci sice také používáme MAC adresy, ale jejich počet se dynamicky mění podle momentální pozice v síti, podobně jako zmíněné dva příznaky. Takže postupně:

1. V ad-hoc síti je to stejně jako u Ethernetu, máme MAC adresu cíle a MAC adresu zdroje, nic jiného nepotřebujeme.
2. Na spoji mezi zdrojem rámce (DTE) a nejbližším AP (tedy v první fázi cesty) jsou v záhlaví tři MAC adresy:
   - adresa momentálního příjemce na spoji (AP),
   - adresa momentálního odesílatele na spoji (zdroje rámce, DTE),
   - adresa cíle rámce (DTE).

   Všimněte si, že první dvě adresy se vztahují k právě aktuálnímu spoji, přičemž pořadí je zachováváno – nejdřív příjemce, pak odesílatel.
3. Na spoji mezi dvěma AP jsou v záhlaví dokonce čtyři adresy:
   - adresa momentálního příjemce na spoji (zdrojového AP),
   - adresa momentálního odesílatele na spoji (cílového AP),
   - adresa cíle rámce (DTE),
   - adresa zdroje rámce (DTE).

   Opět se první dvě adresy vztahují k aktuálnímu spoji, ale samozřejmě musí být v záhlaví uloženo i to, kdo je zdrojem a cílem rámce. V první i druhé dvojici je zachováváno dané pořadí.
4. Na spoji mezi některým AP a cílem rámce (DTE), tedy v závěru cesty, jsou v záhlaví opět „pouze“ tři adresy:
   - adresa momentálního příjemce na spoji (cíle rámce, DTE),
   - adresa momentálního odesílatele na spoji (AP),
   - adresa zdroje rámce (DTE).

Poznámka:

BSSID konkrétního AP je MAC adresa tohoto AP. Každé zařízení v buňce musí znát BSSID svého AP, protože ji umísťuje do záhlaví odesílaných rámců. Koncové zařízení ve skutečnosti přímo komunikuje pouze s jediným zařízením – AP v buňce, ke kterému je asociováno.

Struktura rámce se liší podle jeho typu.

 *Záhlaví datového rámce* je dlouhé 30 oktetů, což je poněkud více než u Ethernetu (navíc se také přidává zápatí s kontrolním součtem, stejně jako u Ethernetu). Najdeme tam následující položky (seznam je mírně zjednodušen):

- Frame Control (1 oktet, pole pro řízení) – bity tohoto pole se dělí do několika skupin:
  - version – verze,
  - Type – typ rámce (Data, Control, Management),
  - Subtype – podtyp, například u typu Management máme podtypy Beacon, Probe, atd.,
- Flags (1 oktet, příznaky), například:
  - příznaky ToDS, FromDS,
  - příznak More Fragments – Wi-fi rámce mohou být fragmentovány podobně jako IP pakety, tento příznak slouží stejně jako příznak MF v záhlaví IP paketu (viz kapitolu o síťové vrstvě),
  - Retry – tento příznak se nastaví, pokud se jedná o opakovaný přenos téhož rámce, atd.
- 4 adresy ve vhodném pořadí (Receiver, Destination, Transmitter, Source),
- BSSID platné v dané buňce (tj. MAC adresa AP),
- Sequence Number, Fragment Number – význam je podobný jako u obdobného pole v TCP segmentu, resp. IP paketu.

V záhlaví datového rámce toho ve skutečnosti najdeme více, toto byl jen „promazaný“ souhrn.

 *Řídící rámec* (Control Frame) víceméně zachovává strukturu, jakou jsme viděli u datového rámce, ale implementuje jen některá pole.

První pole samozřejmě používá (z něj se dá zjistit, že to je řídící rámec a jeho podtyp – RTS, CTS nebo ACK), druhé pole se v něm sice také nachází, ale obvykle je celé nastavené na 0 (včetně příznaků ToDS a FromDS, protože je určen jen pro zařízení v buňce odesílajícího AP, přes žádné další zařízení nejde, nanejvýš může být nastaven příznak Retry).

Třebaže se to může zdát zvláštní, z adres tam najdeme buď dvě nebo pouze jedinou.

- U RTS rámce (žádost o povolení vysílání) jsou dvě adresy – cílový AP a zdrojové zařízení v jeho buňce.
- U CTS rámce (povolení vysílání) je jen jedna adresa – Receiver Address (příjemce, tj. zařízení, které žádalo o povolení), protože odesílající AP je „jasný“ (zařízení patří vždy jen do jedné buňky, buňka má jen jeden AP) a je důležité jen to, aby byl rámec zachycen a přijat „tím správným cílem“. Takže pokud klient odeslal žádost RTS, čeká na CTS obsahující svou adresu.
- U ACK rámce (AP potvrzuje koncové stanici doručení rámce) taktéž najdeme jen adresu klienta (Receiver Address). Pokud tedy po „povolovacím kolečku“ RTS/CTS klient odeslal datový rámec, čeká na ACK rámec se svou adresou. Jestliže čeká marně, bude nutné datový rámec přenést znovu s příznakem Retry, třeba i několikrát.

Dále už nic nenásleduje (kromě kontrolního součtu), ani BSSID.

Správní rámec (Management Frame) má celkovou strukturu rámce (vlastně rámců) naopak mnohem složitější. Ve skutečnosti se jedná o dva do sebe vnořené rámce:

- na podvrstvě LLC je to rámec IEEE 802.11 Wireless LAN Management Frame (místo 802.2),
- na podvrstvě MAC je to rámec IEEE 802.11 typu Management Frame, který v sobě zapouzdřuje ten z podvrstvy LLC.

Ve „vnějším“ rámci je struktura jako v datovém – nejdřív pole pro verzi, typ a podtyp tak, jak bylo dříve uvedeno, následuje pole příznaků (obvykle nulové, případně může být nastaven příznak Retry, pokud je nutné rámec posílat opakovaně). Další pole jsou stejná jako u datového – čtyři adresy, BSSID, Sequence Number, Fragment Number. V některých podtypech mohou být některé adresy broadcastové, dokonce i obě první adresy (to by u Ethernetu nešlo).

Za záhlavím vnějšího rámce následuje vnitřní rámec z podvrstvy LLC. Rámec odesílaný access pointem obsahuje:

- časové razítko (aby bylo jasné, kdy byl rámec odeslán, také kvůli synchronizaci),
- hodnotu beacon interval (jak často se posílají beacon rámce),
- pole příznaků (jednotlivé bity jsou nastaveny podle toho, co AP podporuje – typy multiplexování, zabezpečení apod.),
- sekvenci tagů (položek, kde u každé je typ položky, délka a konkrétní hodnota), podle toho, o jaký podtyp rámce se jedná.

Například u Beacon rámce nebo Probe Response (kde je odesílatelem AP) jsou tagy pro SSID, podporované rychlosti, seznam podporovaných šifrovacích algoritmů, apod.

V rámci Probe Request (odesílaném stanicí žádající o přidružení do buňky) je pouze pole tagů.

Poznámka:

Ke zkoušce nebude nutné znát podrobně všechna pole, je jen dobré mít hrubou představu o tom, co se v záhlavích rámců nachází. Například – pokud „skryjeme SSID“ (tedy zakážeme vysílání Beacon rámců), bude SSID snadno odposlechnutelné z rámců Probe Request a Probe Response.

Úkol

Ve Wiresharku zkuste odposlechnout provoz Wi-fi sítě (jiné než Eduroam) v době, kdy se do buňky bude asociovat některý (jiný) klient. Prozkoumejte Beacon rámec, rámce Probe Request a Probe Response, některý datový rámec. Všimněte si rozdílů v záhlavích a také umístění SSID.

Diagnostické nástroje pro MAC podvrstvu jsou především sniffery, tj. programy, které odchytávají pakety, umožňují je analyzovat či vizualizovat a případně nabízejí další doprovodné funkce. Vpodstatě se tyto nástroje používají souhrnně pro drátové i bezdrátové sítě. Síťové rozhraní, které je takto analyzováno, by mělo běžet v *promiskuitním módu*, to znamená, že by mělo přijímat všechny rámce včetně těch, ve kterých je cílová adresa jiná (určených jinému uzlu v síti). V případě nástrojů pro bezdrátové sítě se také setkáváme s nástroji pro „prolamování“ zapomenutých hesel/WEP, apod.

Nejznámějším a nejpopulárnějším volně šiřitelným snifferem je *Wireshark* (původně Ethereal). Taktéž oblíbený volně šiřitelný je *Kismet*[5] – detektor sítí, sniffer a potenciálně IDS (budeme se učit později). Další jsou komerční *Eye P.A.* a *Cascade Pilot*.

5.3 Zabezpečení bezdrátové komunikace

5.3.1 AAA

AAA (Authentication, Authorization, Accounting) je termín označující řízení přístupu, v našem případě přístupu do bezdrátové sítě. Jak zkratka napovídá, zahrnuje tři fáze:

- *Autentizace* (Authentication) – klient (a v některých případech i AP) má prokázat svou totožnost, jde tedy o kontrolu identity. Přístupové údaje se také označují anglickým termínem *credentials*.
- *Autorizace* (Authorization) – když je identita klienta zjištěna, je třeba mu přidělit přístupová oprávnění, vymezit povolenou činnost.
- *Účtování* (Accounting) – evidujeme stanovené činnosti klienta v síti (podle konfigurace), případně reagujeme při pokusech o překročení oprávnění přidělených ve fázi autorizace.

Každá z následujících možností zajišťuje šifrování jak v první fázi komunikace (autentizace a asociace), tak i při běžném provozu. Ovšem používají se různé postupy a různé (i vzhledem k bezpečnosti) šifrovací algoritmy.

Krátká poznámka k šifrování. U jakéhokoliv šifrování potřebujeme *šifrovací klíč*, který mají mít obě strany komunikace (odesílatel pro šifrování a příjemce pro dešifrování).

Existují dva základní druhy šifer – *symetrické a asymetrické*. Vzhledem k šifrovacím klíčům:

- Symetrické šifry používají tentýž klíč pro šifrování i dešifrování. Hlavním problémem (a taky nevýhodou) je, jak *bezpečně* dostat klíč druhému komunikujícímu.
- Asymetrické šifry používají pro každý směr komunikace pár klíčů – soukromý a veřejný, přičemž co zašifruji veřejným klíčem, musí být dešifrováno soukromým (a naopak – ve funkčnosti jsou zaměnitelné). Uživatel vygeneruje svůj pár klíčů, soukromý si uloží a veřejný předá (jakkoliv) druhé straně. Druhá strana udělá totéž se svým párem klíčů. Když chci něco odeslat, zašifruji data veřejným klíčem adresáta, pošlu a adresát si to dešifruje svým soukromým klíčem.

Nevýhodou symetrických šifer je tedy problém předání klíče protistraně, ale výhodou velká rychlost šifrování a dešifrování. U asymetrických šifer je to přesně naopak – výhodou je možnost transportu veřejného klíče i nezabezpečeným kanálem, nevýhodou velká výpočetní náročnost (hodně zdržuje přenos).

Proto se v praxi oba přístupy některým způsobem kombinují – používáme *hybridní šifrování*. Po navázání spojení se přenášená data šifrují symetricky, aby nebyla komunikace příliš zdržována, ale předem se symetrický klíč předá pomocí asymetrického šifrování (místo dat prostě zašifrujeme symetrický klíč, čímž ho dokážeme relativně bezpečně předat protistraně).

Další informace:

Pro ty, kteří cítí potřebu navýšit své znalosti o šifrování – různé šifrovací algoritmy jsou popsány například ve skriptech jiného předmětu: http://vavreckova.zam.slu.cz/obsahy/analyzadat/analyzadat.pdf.

Princip šifrovacích algoritmů je vcelku podrobně popsán v knize [11].

[5] http://www.kismetwireless.net/

Pro zvýšení zabezpečení se kromě (symetrického) statického šifrovacího klíče pro každý odeslaný rámec používá také dynamicky měnící se dodatek – *IV vektor* (inicializační vektor). Účelem je maximálně ztížit odposlech komunikace. Jenže když se něco neustále mění, musíme mít mechanismus, jak dát adresátovi vědět, jakou podobu má IV vektor pro ten konkrétní rámec (je pěkné, že hacker náš rámec nedešifruje, ale adresát by toho měl být schopen). Pro to jsou dva přístupy: buď mají obě strany algoritmus, kterým pro daný rámec IV vektor určí, nebo momentální IV vektor prostě adresátovi pošleme. První možnost je samozřejmě mnohem bezpečnější.

WEP (Wired Equivalent Privacy). Jedná se o mechanismus zabezpečení, který už rozhodně bezpečným nazývat nelze. Jedinou výhodou je plná kompatibilita se vším, co má implementováno IEEE 802.11.

Používá se stejné šifrování pro první fázi komunikace i pro běžný provoz (pokud vůbec nějaké). Šifruje se pouze symetrickým algoritmem, tedy klíč musí být nějak předán adresátovi (klientovi). IV vektor se v naprosté většině případů přenáší jako součást (nešifrovaného) záhlaví.

WEP nabízí tři režimy:

- otevřená síť (bez šifrování),
- symetrická šifra RC4 s 64bitovým klíčem, přičemž IV vektor tvoří jeho 24 bitů (takže 40 bitů je statická část klíče),
- symetrická šifra RC4 se 128bitovým klíčem, přičemž IV vektor taky tvoří jeho 24 bitů (104 bitů je statická část klíče).

V každém případě pro cracknutí klíče stačí několik minut (s použitím programu AirCrack), takže WEP je považován za naprosto nevyhovující způsob zabezpečení.

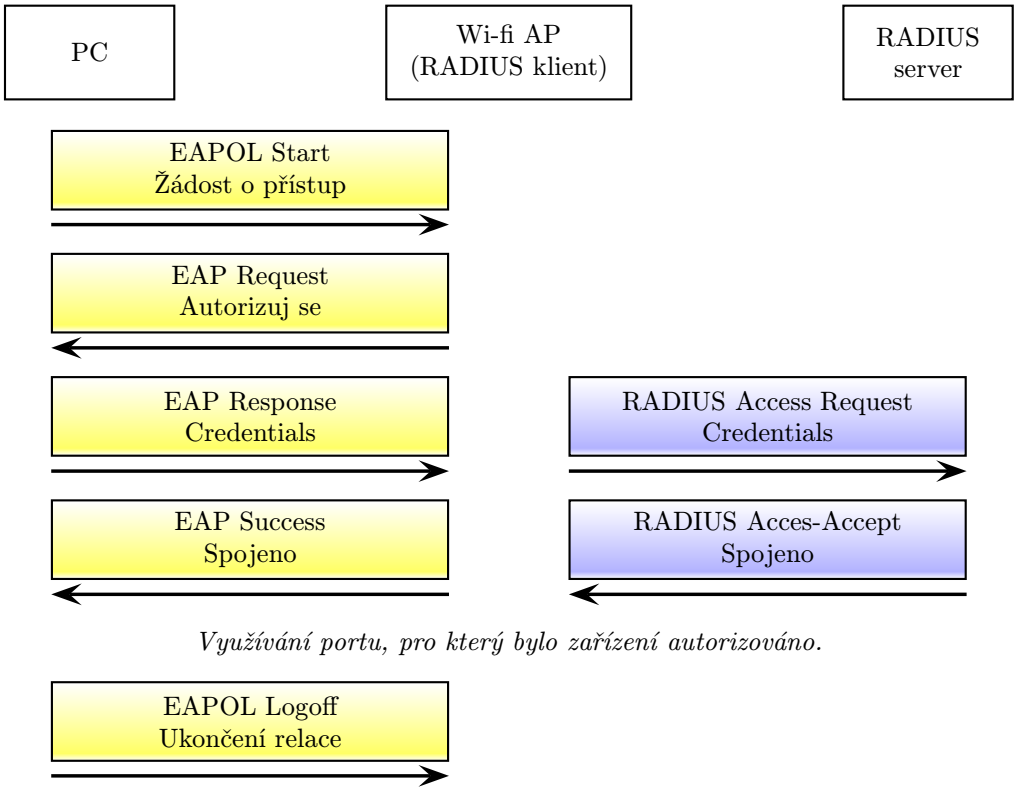
WEP a IEEE 802.1X. Mechanismus WEP byl brzy shledán nedostatečným, ale jiná možnost dlouho neexistovala. Proto se ve firemních sítích začala používat dodatečná možnost zabezpečení přihlašování ve formě implementace protokolu IEEE 802.1X (pozor, ne 11). Spočívá v zajištění autentizace a autorizace pomocí autentizačního serveru, který si vede databázi „povolených“ uživatelů s přístupovými údaji pro autentizaci (credentials) a údaji pro autorizaci.

Existují dva používané typy autentizačních serverů – RADIUS a TACACS. Pro RADIUS existují volně dostupné implementace (nejoblíbenější je open-source projekt FreeRADIUS), kdežto TACACS je proprietární řešení (Cisco).

Princip je takový, že uživateli nestačí jen vlastnit klíč, musí mít i své přihlašovací údaje (například jméno a heslo nebo třeba hardwarový token). Na obrázku 5.7 je naznačena komunikace s RADIUS serverem – klient se sice přihlašuje do některé buňky k příslušnému AP, ale místo toho, aby AP o připojení rozhodoval přímo, pouze přihlašovací údaje přepošle RADIUS serveru a při autentizaci a autorizaci vlastně funguje jen jako zprostředkovatel. Kromě přihlašovacích údajů (credentials) si RADIUS server může navíc vyžádat další údaje, podle kterých pak rozhodne o autentizaci a konkrétních parametrech autorizace (přístupových oprávněních).

RADIUS síť obsahuje jen dva uzly – RADIUS server a RADIUS klienta, kterým je dotyčný AP, spojení je point-to-point. Toto spojení by mělo být dobře zabezpečeno (žádná Wi-fi, zde by měl být kabel, a taky šifrování). Samotný klient (žadatel, suplikant) není součást RADIUS sítě.

WPA (Wi-fi Protected Access). Mechanismus WPA byl původně dočasným řešením, vznikl při čekání na standard WPA2. Dnes je považován za *něco mezi* nezabezpečeným WEP a bezpečným WPA2, přičemž na rozdíl od WPA2 je vhodný i pro starší zařízení, na kterých nelze WPA2 použít. Na



Obrázek 5.7: Zjednodušená komunikace v síti při použití RADIUS serveru

takovém zařízení stačilo upgradovat firmware (tedy softwarová záležitost), WPA nevyžadoval úpravu hardwaru. Zjednodušeně můžeme WPA brát jako WPA2 osekaný o hardwarově závislé součásti.

Základ je podobný jako u WEP – šifra RC4 se 128bitovým klíčem, jehož IV vektor je 48bitový. Ovšem správa dynamických IV vektorů je prováděna protokolem TKIP (Temporal Key Integrity Protocol), takže IV vektor se už nedá odposlechnout z nešifrovaného záhlaví rámce. Navíc protokol TKIP přidává k rámci 64bitový kontrolní součet (spíše digitální podpis rámce), který nazýváme MIC (Message Integrity Check, lidově „Michael“), podle kterého se pozná, jestli nebyl rámec cestou poškozen či záměrně upraven.

WPA pracuje ve dvou variantách:

- WPA-Enterprise – pro korporátní sféru, vpodstatě pokračovatel WEP+RADIUS, v síti musí být autentizační server,
- WPA-Personal (WPA-PSK) pro domácnosti a malé firmy, pro autentizaci se používá sdílený klíč (PSK = Pre-Shared Key), nemáme autentizační server, ale uživatel i tak potřebuje autentizační informace (heslo 8–64 znaků nebo 64 hexadecimálních číslic), ze kterých se dynamicky určuje klíč.

WPA2 (IEEE 802.11i). Pro tento mechanismus existuje vlastní standard. Oproti WPA se místo protokolu TKIP používá protokol CCMP (Counter-Mode CBC MAC Protocol) s šifrováním AES, a právě použití šifrování AES vyžaduje hardwarovou podporu na zařízeních (dnes již většina procesorů obsahuje AES instrukce, takže obvykle nebývá problém se zprovozněním).

Stejně jako u WPA, i zde potřebuje uživatel přihlašovací údaje, které jsou využity buď pro autentizační server nebo pro protokol PSK, tedy opět máme dvě možnosti:

- WPA2-Enterprise – pro korporátní sféru, s autentizačním serverem,

- WPA2-Personal (WPA2-PSK) pro domácnosti a malé firmy.

V roce 2017 se objevila zpráva, že mechanismus WPA2 je náchylný na útok KRACK (Key Reinstallation Attacks), pomocí kterého lze obejít zabezpečení pomocí šifrování spojení.

WPA3 byl uveden roku 2018. Opět existují dva režimy – WPA3-Enterprise a WPA3-Personal. Pro Enterprise použití se předepisuje šifrování AES-256 v módu GCM s hashováním HMAC SHA-384. V Personal režimu lze použít AES-128 s protokolem CCMP jako u WPA2, ale výměna klíčů probíhá s využitím algoritmu SAE (Simultaneous Authentication of Equals), což je varianta výměny klíčů Dragonfly Key Exchange (funguje podobně jako Diffie-Hellman).

Příchod WPA3 byl uspíšen objevením závažných bezpečnostních chyb ve WPA2, nicméně bezpečnostním chybám se nevyhnul ani samotný mechanismus WPA3. Například výše zmíněný algoritmus Dragonfly (ve WPA3 se používá při navázání spojení pro bezpečnou výměnu klíčů) má bezpečnostní slabinu DragonBlood, která je zneužitelná rovnou několika různými způsoby.

Další informace:

- https://www.root.cz/clanky/sifrovani-wpa2-bylo-prolomeno-wi-fi-site-je-mozne-odposlouchavat/
- https://www.root.cz/zpravicky/dragonblood-bezpecnostni-chyba-ve-wpa3-umoznuje-ziskat-heslo-k-wi-fi/
- https://www.wi-fi.org/discover-wi-fi/security
- https://medium.com/@reliancegcs/wpa3-explained-wi-fi-is-getting-major-security-update-2b6dca8f3aff

5.3.2 WPS

Mechanismus WPS (Wi-fi Protected Setup) se objevil s tím, jak přibývalo nejrůznějších typů zařízení, která se měla připojovat do Wi-fi sítě. Mnohá zařízení nemají klávesnici ani pořádný display, takže případné zadávání hesel a klíčů by bylo problematické. WPS právě slouží ke zjednodušené autentizaci a asociaci takovýchto zařízení do Wi-fi sítě.

Existují dvě možnosti použití mechanismus WPS – WPS tlačítko nebo WPS PIN.

WPS tlačítko (PBC – Push Button). Na AP zmáčkneme tlačítko označené WPS (nebo QSS či PBC) nebo v administraci AP klepneme na „softwarové“ tlačítko. V průběhu několika desítkem sekund AP skenuje svou buňku a hledá nová zařízení. Pokud se v tomto časovém intervalu (kterékoliv) zařízení v buňce začne hlásit (čehož docílíme tak, že na na dotyčném zařízení taky zmáčkneme tlačítko nebo provedeme něco podobného podle návodu), je považováno za důvěryhodné a AP s ním dojedná asociaci (takže nemusíme zadávat žádná hesla).

Problémy jsou především tyto:

- V uvedeném časovém intervalu je naše síť prakticky nechráněná, cokoliv se může přihlásit a dostat se dovnitř.
- Musíme stihnout na asociovaném zařízení včas zmáčknout tlačítko, takže v některých případech to je spíše práce pro dva.

WPS PIN. Pro zařízení, které chceme takto dostat do sítě, musíme zjistit jeho PIN – 8místné číslo, které najdeme buď na nálepce přímo na zařízení nebo někde v dokumentaci. PIN pak naťukáme

v administraci svého AP, načež si opět AP se zařízením vyjednají veškeré parametry a provede se asociace.

Ovšem problémy jsou i s touto metodou:

- AP s podporou WPS PIN *naslouchá neustále*.
- PIN je 8místné číslo, přičemž poslední číslice se vypočítává z předchozích, je obdobou kontrolního součtu. Pokud PIN do administrace zadáme špatně, AP vrací informaci, ze které se dá odvodit, která polovina je chybná – pokud je chybná první polovina, stačí uhodnout tři číslice, což je pro průměrného hackera časově nenáročná hra.

Pokud náš Wi-fi AP má tuto funkci, je lepší ji vypnout a zapínat opravdu jen tehdy, když chceme takto asociovat nové zařízení.

5.3.3 Jak tedy zabezpečit bezdrátovou síť

Takže co udělat, aby naše Wi-fi síť byla co nejlépe zabezpečená? Předně nastavit šifrování WPA2, pokud to všichni naši klienti zvládají. A co dál?

✎ **Credentials do administrace.** Přístupové údaje do administrace AP jsou z výroby nastaveny na určité standardní hodnoty (typicky admin–admin nebo něco takového). V žádném případě bychom to tak neměli nechat! Takže hned po zakoupení během počáteční konfigurace bychom jako první měli nastavit vlastní přístupové údaje do administrace AP.

✎ **Skrytí SSID.** AP vysílá v pravidelných intervalech Beacon rámec, kde je mimo jiné SSID (tedy název sítě). SSID je potřeba pro přihlášení do sítě, a tedy když AP nebude vysílat Beacon rámce s SSID, teoreticky bychom tím zabránili pokusům o přihlášení osob, které v síti nemají co dělat. Prakticky se však SSID dá odposlechnout v autentizační komunikaci „povolených“ zařízení, takže stačí donutit některé takové zařízení k znovupřipojení (shodíme jeho komunikaci).

Navíc skrytí SSID může způsobovat problémy při hledání zdroje rušení jiné sítě – může nastat situace, kdy vidíme, že nám něco ruší síť, ale nevidíme, co to je, protože dotyčný má skryté SSID.

✎ **Filtrování MAC adres.** V administraci AP si můžeme vytvořit black list (seznam zakázaných) nebo white list (seznam povolených, nikdo jiný nebude asociován) MAC adres. Ale stačí si uvědomit, že MAC adresa se dá pozměnit (v Linuxu na to stačí jeden příkaz, ve Windows úprava registru nebo speciální program), takže opět bezzubé opatření.

5.4 Centrálně řízená bezdrátová síť

✎ V domácnostech a malých firmách se čím dál více objevují tzv. *Wi-fi mesh sítě*, což je ve skutečnosti jednoduchý WDS propojující několik AP v sadě. Jeden prvek ze sady bývá řídící, jeden z jeho portů slouží jako WAN port, ostatní komunikují přes tento řící prvek. Mezi AP probíhá komunikace buď bezdrátově po vyhrazeném kanálu, nebo přes kabel v ethernetové síti, záleží na konkrétním řešení. Obvykle se jedná o proprierární řešení, tedy koupíme sadu od jednoho výrobce.

✎ Ve firemním prostředí se už mnohem déle používají *centrálně řízené sítě* (původně se nazývaly sítěmi čtvrté generace), kde je účelem spolehlivé pokrytí rozlehlých prostor signálem tak, aby se přes odlišnost v konceptu sítě mohli připojovat klienti s běžnými Wi-fi zařízeními. U různých výrobců se používá mírně odlišná terminologie, ale princip je podobný.

Centrálním prvkem sítě je prvek, který se nazývá *Wireless controller* (Wireless LAN Controller, WLC). Tento prvek obsahuje konfiguraci sítě, zde se zajišťuje synchronizace celé sítě, zabezpečení, management, napojení „ven“, atd. Může se jednat o fyzické zařízení, ale existují i virtuální implementace. Další prvky v síti jsou *Lightweight AP* (LAP), „odlehčené“ přístupové body, které si můžeme představit jako „prodlouženou ruku“ controlleru, obdobu hodně dlouhých antén. LAP jsou ke controlleru připojeny pomocí ethernetových kabelů, a přes tyto kabely jsou i napájeny s použitím PoE.

Typické použití této technologie je ve velkých halách či jiných rozlehlých místnostech (továrny, sklady, hypermarkety, ale třeba i nemocnice nebo open-space prostory). Předpokládejme, že bychom použili klasické AP propojené do běžného WDS. Pokud bychom v takovém případě v rozlehlé místnosti s vysokým stropem umístili access pointy na strop, museli bychom řešit dilema: aby signál dosáhl až na zem, musel by být dostatečně silný, ale potom by AP musely být hodně daleko od sebe, aby se navzájem nerušily. Jenže pak by u země měl signál velmi proměnlivou dostupnost, vznikaly by „mapy“ míst bez signálu.

Řešení využívající controller a odlehčené AP umožňuje vytvořit hustou síť těchto AP, které se nebudou navzájem rušit, protože všechny mohou vysílat v tomtéž pásmu (resp. na více stejných kanálech) – je úplně jedno, se kterým z nich klient komunikuje. Handshake (předávání mezi AP) není nutné komplikovat, veškerá správa je čistě na straně controlleru.

LAP komunikují s WLC pomocí určitého protokolu – buď některého proprietárního (například Cisco u starších WLC používalo svůj protokol LWAPP), nebo se používá protokol CAPWAP (Control and Provisioning of Wireless Access Points Protocol) standardizovaný v RFC 5415.

V síti se obvykle používají VLAN sítě, pro každou na síťové vrstvě existuje vlastní podsíť. Vzhledem k tomu, kde se takové sítě využívají, bývají součástí infrastruktury také autentizační servery (například Radius, TACACS+, Diameter). Mezi autentizačními servery a centrálním řídícím prvkem musí existovat fyzicky zabezpečené spoje.

Jako koncová zařízení se do sítě mohou připojit jakákoliv zařízení podporující IEEE 802.11(něco) – notebooky, mobily, IoT zařízení apod. Jako řešení pro pokrytí rozsáhlých prostor to je výborné (čím vyšší hustota LAP, tím lepší signál), jedinou nevýhodou je cena, která rozhodně není nízká.

Další informace:

Další informace o Wi-fi:

- http://www.marigold.cz/item/wds-pro-pohodlnou-stavbu-vetsi-wifi-site
- http://www.cs.vsb.cz/grygarek/SPS/projekty0607/RADIUS-Stankus.pdf
- http://www.security-portal.cz/clanky/wifi-sítě-jejich-slabiny
- http://www.svethardware.cz/art_doc-121AB59EC9127B44C12570A60045C902.html
- http://www.intelek.cz/art_doc-D7A489A18B634F84C12575550053ECAE.html
- http://www.intelek.cz/db/media.nsf/w/E4A2BCABAE379135C125732300589FD3/$file/extricom.pdf

5.5 WiMAX

Zatímco Wi-fi je spíše síť typu LAN (WLAN – Wireless LAN), *WiMAX* (IEEE 802.16, Worldwide Interoperability for Microwave Access) je bezdrátová metropolitní síť (WMAN – Wireless MAN). Jejím

účelem je především poskytnout rychlý a spolehlivý přístup k Internetu, tedy technologie poslední míle. Důležitou vlastností je také přímá podpora QoS (to je jedna z odlišností oproti Wi-Fi, kde se setkáváme pouze s „best effort“, QoS je možné zavést až dodatečně pomocí IEEE 802.11e).

Na standardizaci a certifikaci produktů se podílí *WiMAX Forum*, které je sdružením výrobců WiMAX zařízení (je to obdoba Wi-Fi Alliance).

Původní specifikace IEEE 802.16 z roku 2001 stanovila použití kmitočtového pásma 10–66 GHz, které však vyžaduje přímou viditelnost komunikujících uzlů. Proto byla v roce 2003 tato specifikace zcela nahrazena novou, kde se využívá kmitočtové pásmo 2–11 GHz (u nás nejčastěji 3.5 GHz) nevyžadující přímou viditelnost komunikujících uzlů (NLOS, Non Line of Sight), využívá odrazů od okolních objektů. Pásmo je *licencované*, což se projevuje i na cenách zařízení a připojení. Roku 2005 se objevila možnost mobility (handover, roaming – při pohybu do rychlosti max. 150 km/h) ve standardu IEEE 802.16e-2005, do té doby nebylo možné zajistit plynulé předávání pohybujících se uzlů mezi základnovými stanicemi.

Teoretický dosah signálu je 50 km, v běžné zástavbě se počítá s dosahem 3–5 km. To je pořád víc než Wi-Fi, kde je obvyklý dosah maximálně v desítkách metrů. Přenosová kapacita je 30–75 Mb/s (podle konkrétního standardu), tu však sdílejí všechny připojené stanice.

Na fyzické vrstvě se používá OFDM nebo OFDMA (OFDM Access). První zmíněné modulační schéma už známe (stovky nosných frekvencí – subkanálů, vzájemně ortogonálních, aby je bylo možné oddělit). OFDMA je podobné, jen zatímco v OFDM jsou všechny subkanály jediného kanálu na frekvencích „za sebou“ a celý kanál je vyhrazen jedinému uživateli, v OFDMA jsou kanály jednoho subkanálu rozprostřeny v celém spektru a je možné různé subkanály přiřazovat různým uživatelům.

Další vylepšení, S-OFDMA (Scalable OFDMA), má zlepšit kvalitu signálu při NLOS (bez přímé viditelnosti).

Hlavním článkem sítě je *základnová stanice* umístěná obvykle na vyvýšeném místě, což je většinou vysoká budova. Další úrovní v hierarchii jsou *vnější zařízení* zahrnující anténu, ta se obvykle umísťují na střechy nebo venkovní zdi domů. Poslední úrovní jsou *vnitřní koncová zařízení*. Existují také zařízení, která v sobě sdružují obě posledně jmenovaná, tedy vnitřní koncová zařízení s (integrovanou) anténou, ta však vyžadují lepší pokrytí.

V komunikaci lze použít časový (TDD) i frekvenční (FDD) duplex (u nás jsou povolena zařízení používající FDD) – nemluvíme zde o multiplexu, protože komunikace je sice obousměrná, ale v jednom okamžiku na dané frekvenci (ve skutečnosti ve dvou kanálech, jednom pro každý směr) jen mezi dvěma uzly. Způsob využití signálu je poměrně volný, záleží na poskytovateli připojení, jak se se svými zákazníky dohodne. Spojení může být symetrické nebo asymetrické, je smluvně garantována konkrétní kvalita služby (QoS).

Jedním z nejznámějších dodavatelů WiMAX zařízení je firma Alvarion, z dalších například RedMAX nebo AirSpan.

V současné době je WiMAX provozován na více místech v ČR, ale spíše lokálně v menších oblastech (spíše jde o oblasti ve velkých městech, kde se očekává firemní klientela). Kanály 1–14 (o šířce 3,5 MHz) jsou určeny pro lokální operátory, kanály 15–20 pro celoplošné operátory.

Pro WiMAX je typická *centralizovanost*, jejímž cílem je především odstranit nebezpečí kolizí. Celá komunikace s koncovými uzly je vždy řízena základnovou stanicí, ta určuje, kdy kdo bude vysílat, přiděluje kanály pro komunikaci.

Další informace:

- http://www.intelek.cz/art_doc-29B0470BE0F689D6C125740C002F6883.html
- http://access.feld.cvut.cz/view.php?cisloclanku=2008050005

Poznámka:

Od WiMAXu se původně očekávalo, že se bude hodně využívat jako metropolitní přístupová síť hlavně ve velkých městech, ale nějak se nepovedlo... Tato technologie totiž není zrovna levná a menší poskytovatelé internetu spíše volí některou variantu Wi-fi.

5.6 Mobilní technologie

Mobilní sítě jsou především charakteristické svou mobilitou, tedy stanice, která je součástí mobilní sítě, se může pohybovat se zachováním signálu, a to i mezi různými buňkami (oblastmi pokrytí signálu). V této sekci se budeme zabývat především *mobilními WAN*, třebaže existují řešení i pro menší sítě.

Proces přecházení mezi buňkami bez přerušení spojení se zachováním všech parametrů a probíhajících datových přenosů (relací) se nazývá *handover* (předávání, také handoff). U mobilních sítí je tato funkcionalita velmi důležitá.

5.6.1 Struktura mobilní sítě

V mobilní síti potřebujeme určitou infrastrukturu sloužící jako pro přenos hlasu a dat, tak i pro řídící a správní účely. Konkrétní názvy jednotlivých typů zařízení jsou v různých typech a generacích mobilních sítí odlišné, zde jen nastiňujeme základní princip a strukturu.

Klientská zařízení přímo komunikují se *základnovou stanicí* (Base Station), oblast pokrytá signálem určité základnové stanice je její buňka (tj. klientská zařízení se nacházejí v buňce některé základnové stanice). Označení může být například BTS (Base Transceiver Station), Node B, atd.

Základnové stanice zvládají spíše jen samotnou komunikaci, logika sítě je trochu jinde. Vždy skupina základnových stanic je napojena na *řadič pro hlasovou síť* a *řadič pro datovou síť* (podle toho, co má být přenášeno), zde se dělí trasa hlasového signálu a dat.

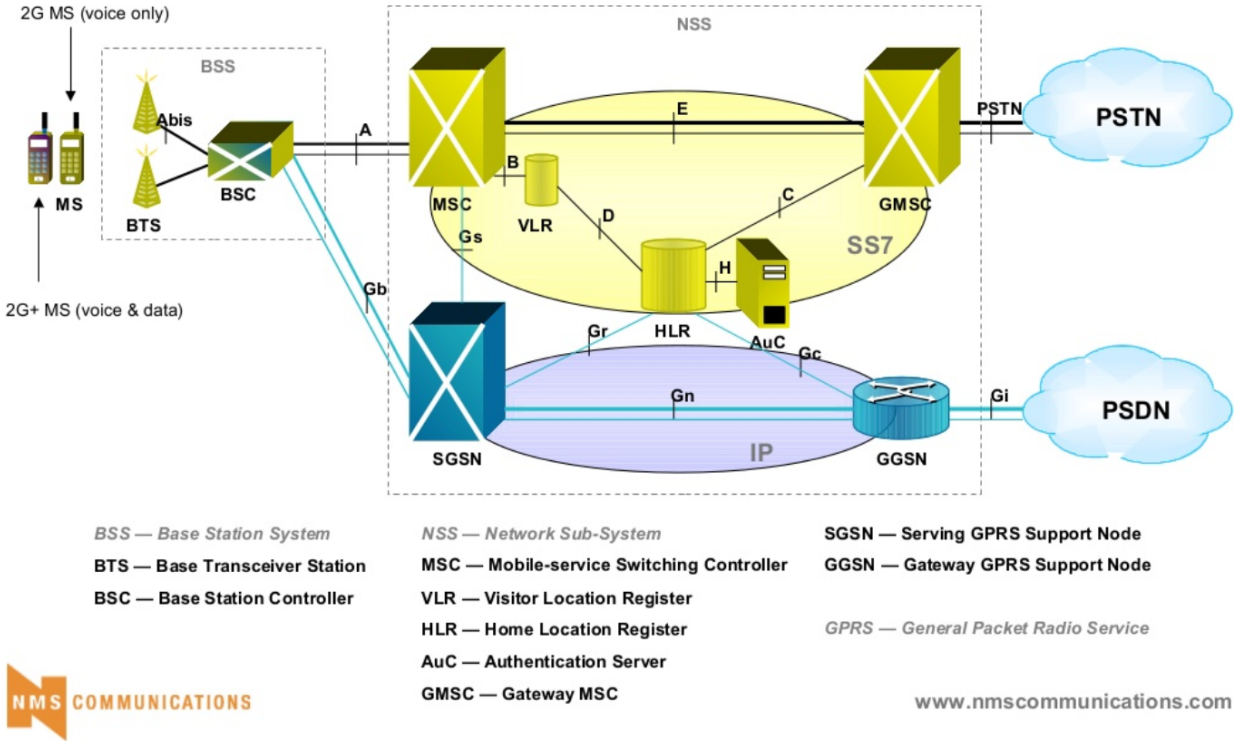
To by byla *vrstva základnových stanic* a jejich správy.

Trasy pro hlas a data mají společné body v dalších dvou vrstvách celé struktury:

- *administrativní podsystém* (operační) – zde najdeme například registr stanic a s ním související autentizační servery, tarifikaci apod.,
- *podsystém správy sítě* – zajišťuje fungování sítě včetně provozu bran do jiných sítí a jiných typů sítí (brány do sítí jiných poskytovatelů, brány na Internet, atd.).

To je jen hrubý a osekaný nástin, u každé technologie se používá jiná terminologie a také mohou být přidány další typy uzlů. Na obrázku 5.8 je ukázka struktury pro síť GSM kombinovanou s GPRS.

https://github.com/jreisinger/blog/blob/master/posts/gsm.md



Obrázek 5.8: Struktura mobilní sítě[6]

5.6.2 První generace (1G)

První generace mobilních datových sítí byla ještě analogová. Z toho důvodu se také využíval frekvenční multiplex (FDMA), každý připojený uživatel má přidělen jeden kanál, a to výhradně.

První komerční mobilní systém byl *NMT* (Nordic Mobile Telephone) fungující nejdřív v Saudské Arábii a následně v severských zemích (Norsko, Švédsko) od roku 1981. Do první generace také patří americký AMPS (Analog Mobile Phone System) z roku 1982 a TACS (Total Access Communications System) vytvořený původně pro Velkou Británii, ale později se rozšířil především v Asii.

5.6.3 Druhá generace (2G)

Na začátku 90. let (1992) vznikla druhá generace, která je dodnes funkční (vedle dalších generací). Byl zaveden digitální přenos dat, ale síť je optimalizována především pro hlas, tedy propustnost je na velmi nízké úrovni.

Sítě druhé generace pořád pracují na principu *přepojování okruhů*, a to i pro posílání dat. Důsledkem je obvyklá tarifikace odvozená od doby připojení (jde o vytáčené připojení).

GSM (Global System for Mobile Communications) je digitální síť, ale pořád využívá přepojování okruhů, a to s časovým (TDMA) i frekvenčním (FDMA) multiplexem – kombinace TDMA over FDMA (časové sloty se přenášejí vždy v rámci určité frekvence). Hovoříme o *kanálech* (to jsou vlastně SVC okruhy), uživateli je vždy po určitou dobu vyhrazen určitý kanál. GSM pracuje na třech různých frekvencích, hovoříme o GSM900 (frekvence 900 MHz), GSM1800 (1800 MHz) a GSM1900 (1900 MHz).

[6] Zdroj: https://www.slideshare.net/tbutomo/3-g-tutorial-32431714

Struktura víceméně odpovídá nákresu výše, jen zde není „datová odbočka“ – v GSM se data přenášela přímo po okruzích.

Základnové stanice jsou rozmístěny tak, aby se buňky (přibližně kruhové, i když jsou zobrazovány jako šestiúhelníky – „plástve“) překrývaly. To však znamená, že dvojice buněk, které jsou přímo vedle sebe, nesmí používat tytéž komunikační kanály, tytéž kanály může používat až buňka natolik vzdálená, aby nedocházelo k inferencím. Rozdělení kanálů se provádí metodou podobnou obarvování, kterou známe z teorie grafů.

Při plném pokrytí signálem je možná plná mobilita stanice, *handover* – předávání stanice mezi jednotlivými buňkami.

Oproti první generaci je zde především *přechod na digitální technologii*, dále větší odolnost proti rušení a odposlechu, snadné napojení na pozemní telekomunikační systémy, a také vyšší rychlost. Teoretická rychlost je 13 kb/s pro každý směr, reálně 9.6 kb/s.

 **CDMA** (Code Division Multiple Access) má sice ve specifikaci přenos dat i hlasu, ale u nás je CDMA poskytována jen jako datová síť. Oproti GSM má výhodu nižších frekvencí (450 MHz), základnové stanice CDMA mohou být od sebe více vzdáleny. Teoretická rychlost uploadu je 1024 kb/s, té však lze dosáhnout jen v místech s výkonnějšími základnovými stanicemi (například v Praze nebo v Brně). Download je o něco rychlejší (až 3.1 Mb/s, v reálu také méně, několik set kb/s), jde o asymetrický přenos.

CDMA používá jiný typ multiplexu – kódový. Spočívá v tom, že v rámci jednoho sdíleného média se přenáší více digitálních signálů, které jsou rozlišovány pomocí kódování (tedy ne rozdělením do časových slotů ani pouhým rozvržením do frekvencí).

Poznámka:

U jednotlivých technologií jsou uváděny (teoretické) rychlosti. Tyto údaje berte s rezervou (může se na nich podepsat poměrně hodně parametrů), a taky nebudou vyžadovány na zkoušce. Jen je dobré mít představu o tom, které technologie jsou rychlejší a které pomalejší.

5.6.4 Přechodová generace (2.5G)

Zatímco mobilní sítě druhé generace byly založeny na přepínání okruhů, v přechodové generaci se již používá přepínání paketů, které je pro přenos dat výhodnější. Typická tarifikace je buď za přenesená data nebo paušál, netarifikuje se připojená doba (tj. nejedná se o vytáčené připojení).

 **GPRS** (General Packet Radio Service) je nástavba GSM (technicky ne nutně GSM, může být vázána na jiné typy sítí), která umožňuje jedné stanici využívat až 8 GSM kanálů (pokud jsou zrovna volné). Nejedná se ještě o širokopásmovou technologii, ale díky rozšíření pásma až na 8násobek jsou přenosové rychlosti vyšší. Jsou využívány ty z osmi GSM kanálů, které jsou právě volné, tedy přenosová kapacita sítě je využívána účelněji, kanály nejsou blokovány po celou dobu „připojení“.

Aby fungovalo napojení na GSM, bylo nutné do sítě GSM přidat ještě jednu vrstvu. Vrstva základnových stanic GSM zůstává, ale byla přidána dodatečná vrstva podpůrných uzlů sítě GPRS a další uzly byly přidány do vrstvy řízení sítě i do administrativní vrstvy (tam také, protože mechanismus tarifikace je odlišný). Přidané uzly mezi sebou komunikují jiným protokolem než ty původní – GTP

(GPRS Tunelling Protocol), což je aplikační protokol nad protokoly TCP a UDP.

Přenosové rychlosti se odvozují od použitého kódování. Čím silnější kódování, tím nižší rychlost. Celková možná dosažená rychlost je 22.8 kb/s na jeden kanál, ale část rychlosti je pohlcena režií kódování. Při nejsilnějším kódování (CS-1) dosahujeme rychlosti maximálně kolem 9 kb/s na kanál, při nejslabším kódování (CS-4) maximálně kolem 21 kb/s na kanál. Volba kódování je zcela automatická, závisí na vzdálenosti stanice od základnové stanice (čím dál je stanice, tím silnější musí být kódování).

Dále je rychlost přenosu ovlivněna množstvím GSM kanálů, které stanice může využívat (maximálně 8 pro každý směr). V základním nastavení musí uživatel GPRS vzít zavděk tím, co „zbude“ po uživatelích GSM hlasových přenosů, ale v některých typech tarifů se můžeme setkat s garancí určitého počtu kanálů (vyhrazením pouze pro data).

GPRS zavádí základní možnost řízení kvality služeb (QoS) založenou na prioritě, propustnosti, zpoždění nebo spolehlivosti. Je samozřejmě na operátorovi, zda bude QoS nabízet svým zákazníkům.

✎ **EDGE.** Technologie EDGE (Enhanced Data rates for GSM Evolution) je sice také nástavbou sítě GSM a základní princip je podobný, ale navíc signál moduluje metodou 8-PSK, což v reálu znamená až trojnásobný nárůst rychlostí oproti GPRS.

5.6.5 Třetí generace (3G)

Třetí generace přináší možnost *souběžného využívání několika služeb* (u předchozích generací nebylo možné kombinovat hlas a data). Technologie jsou založeny na variantách metody CDMA, tedy se používá kódový multiplex. Na rozdíl od předchozích generací, třetí generace již není striktně optimalizována pro přenos hlasu, více se počítá s datovými přenosy.

✎ **UMTS** (Universal Mobile Telecommunication System) není pouhá nástavba, zavedení UMTS znamenalo poměrně velké zásahy do infrastruktury sítě. V tomto typu sítě je také nabízena vylepšená QoS.

Základem je WCDMA (Wideband CDMA), což je širokopásmová metoda přenosu. V metodě WCDMA má každý uživatel přiděleno frekvenční pásmo, které používá. V kódovém multiplexu může totéž pásmo využívat více uživatelů, jsou odlišeni jednoznačným binárním kódem.

Architektura sítě je podobná síti GPRS a zčásti na ní staví. Uzel s názvem *Node B* je obdobou základnové stanice. Zprostředkovává komunikaci mezi koncovou stanicí a samotnou UMTS sítí, zajišťuje modulaci, kódování, ochranu proti chybám, tak jako původní základnové stanice. Také do dalších vrstev původní sítě byla přidána řada nových uzlů.

UMTS není u nás, ale ani v jiných zemích, moc rozšířená. Vzhledem k nutným úpravám infrastruktury se s UMTS typicky počítá pouze ve velkých městech. Teoretická rychlost je až 2 Mb/s, ve skutečnosti se však dosahuje pouze rychlosti několika desítek (místy i stovek) kb/s.

✎ **HSDPA** (High-Speed Downlink Packet Access) přináší další zvýšení rychlosti a snížení latence, jedná se tedy o jakési vylepšení UMTS pro download (slovo „Downlink“ v názvu). Změny je docíleno zvolenou modulací – QPSK (Quadrature Phase Shift Keying, používá se i v UMTS) nebo 16QAM, a dále jinými mechanismy plánování vysílání (plánování založené na QoS) a řízení chyb.

Stejně jako UMTS, také HSDPA používá WCDMA, ale navíc přidává nové mechanismy a nový typ čistě datového kanálu. Zatímco v UMTS jsou pakety odesílány každých 10 ms, v HSDPA jsou odesílány každé 2 ms. Pro zvýšení propustnosti je také možné využít technologii MIMO.

Přenosové rychlosti jsou obecně vyšší než v případě UMTS, v současné době kolem 1 Mb/s, ovšem stejně jako u jiných mobilních sítí s velkými odchylkami v obou směrech. Existují varianty HSDPA, které dosáhnou až na 14 Mb/s, ale ty u nás nejsou implementovány.

HSUPA (High-Speed Uplink Packet Access) je obdobou předchozí technologie pro upload. Jejím účelem je tedy zvýšit rychlost a celkovou kvalitu (vč. latence) u uploadu.

Technologie HSDPA a HSUPA se souhrnně označují *HSPA*, ale operátor nemusí nutně použít obě (u nás O2 implementuje pouze UMTS/HSDPA).

LTE (Long Term Evolution) je sice prezentována jako mobilní síť čtvrté generace, ale ve skutečnosti se jedná o třetí generaci (čtvrtá je až LTE Advanced). Je to čistě datová technologie, s přenosem hlasu se zde vůbec nepočítá. To znamená, že pokud chce uživatel použít hlasové služby, buď zvolí VoIP nebo obdobnou technologii (existuje také VoLTE), nebo bude LTE dočasně deaktivováno (což znamená, že když se přes LTE zrovna stahují data, bude jejich stahování přerušeno).

Poznámka:

Někteří uživatelé si kupují mobilní zařízení „v Číně“. To není až takový problém, až na to, že standard LTE umožňuje vysílat ve více různých frekvenčních pásmech, přičemž jiná pásma se používají v Asii a jiná v Evropě. Nejběžnějším pásmem pro LTE u nás je pásmo kolem frekvence 800 MHz, ve větších městech mohou (nemusejí) být dostupné i některé další frekvence (1800 MHz a 2100 MHz) – s posunem se do budoucna moc nepočítá. V každém případě: pokud si koupíme LTE zařízení, které nepodporuje frekvenci 800 MHz, nebude nám toto zařízení tak užitečné jak bychom čekali.

LTE je první technologií 3G sítí, která je plně implementovaná jako síť postavená na IP (přepínání paketů), předchozí alespoň částečně stojí na přepínání okruhů. Na downlinku používá multiplex OFDMA, na uplinku SC-FDMA (speciální odnož OFDM optimalizovaná pro použití na uplinku).

Cílem je dosáhnout velmi vysokých rychlostí (až 100 Mb/s na downstreamu, 50 Mb/s na upstreamu na kanál) při velmi nízké latenci (10 ms, i méně). V současné době LTE reálně dosahuje na downstreamu téměř 80 Mb/s.

5.6.6 Čtvrtá generace

Technologie FLASH-OFDM (Fast Low-latency Access with Seamless Handoff-Orthogonal Frequency-Division Multiplexing) je již řazena do čtvrté generace (4G), zákazník je často seznámen pouze se zkratkou 4G.

Ortogonální multiplex (OFDM) je znám již velmi dlouho. Ve variantě FLASH-OFDM znamená propojení OFDM s metodami CDMA a TDMA. Výsledkem jsou vyšší přenosové rychlosti (relativně, opět s rozptylem) a nízká latence. Rychlost se pohybuje i ve stovkách kb/s (download je většinou v rozmezí 200–300 kb/s, třebaže se teoreticky udává až 1,5 Mb/s). Další výhodou je pokročilá možnost řízení kvality služeb (QoS). Díky nízkým latencím a využití QoS je na FLASH-OFDM možné používat IP telefonii.

LTE Advanced (také IMT Advanced – International Mobile Telecommunications) vypadá pro uživatele velmi výhodně – přenosová rychlost může být teoreticky až 1 Gb/s (downlink), resp. zhruba poloviční rychlost na uplinku. Latence je přibližně 5 ms, což je polovina latence původního LTE (takže

síť má lepší odezvu). Používá se stejné modulační schéma jako u původního LTE (OFDMA a SC-FDMA) v kombinaci s MU-MIMO (využívání více antén), rozdíl je v efektivitě a rozsahu využívání spektra a dále v metodách správy celé sítě.

☎ http://www.radio-electronics.com/info/cellulartelecomms/lte-long-term-evolution/4g-lte-advanced-carrier-channel-aggregation.php

☎ **Další informace:**

Další informace o mobilních sítích:

- http://tomas.richtr.cz/mobil/index.htm
- http://www.earchiv.cz/a008s200/a008s200.php3
- http://rychlost.cz/pripojeni-internetu/
- http://coverage.o2.position.cz/
- http://www.t-mobile.cz/web/cz/residential/tarifysluzby/mapapokryticr
- http://home.pf.jcu.cz/~pepe/Diplomky/velicky.pdf
- http://access.feld.cvut.cz/search.php?rsvelikost=sab&rstext=all-phpRS-all&rstema=10&stromhlmenu=10

☎

Kapitola 6

Síťová vrstva

V této kapitole se budeme zabývat tím, co se obvykle děje na síťové vrstvě. Předpokládáme, že čtenář již některé znalosti má, nicméně je jasné, že každý do tohoto předmětu přichází s trochu jinými počátečními znalostmi (nebo jejich zbytky), takže pro mnohé bude část textu spíše opakováním.

6.1 Síťová vrstva a logické adresy

Síťová vrstva (v terminologii podle RM ISO/OSI, vrstva L3), resp. internetová vrstva (podle síťového modelu TCP/IP), pracuje s logickou topologií sítě. Úkolem vrstvy L3 je zajištění jednotného síťového rozhraní pro nadřízené vrstvy (vyšší vrstvy se již nezabývají samotným procesem komunikace s konkrétními zařízeními), a taky směrování, kterému se budeme věnovat v jiné kapitole.

Zatímco protokoly podřízené vrstvy (L2) komunikují v rámci jedné sítě, protokoly síťové vrstvy zajišťují komunikaci i za hranice jedné sítě, tedy zařízení této vrstvy dokážou propojovat různé sítě (dokonce i takové, které na nižších vrstvách používají navzájem odlišné protokoly). Typickými zařízeními síťové vrstvy jsou routery a switche s funkcionalitou vrstvy L3.

O adresách síťové vrstvy hovoříme jako o *logických (softwarových) adresách*, na rozdíl od fyzických (hardwarových) adres linkové vrstvy.

Vrstva L2 s fyzickými (hardwarovými, MAC) adresami se zabývá adresováním a doručováním (zde přepínáním) v rámci místní sítě, kdežto vrstva L3 s logickými (softwarovými, IP) adresami se zabývá adresováním a doručováním (zde směrováním) mezi sítěmi.

Na síťové vrstvě je nejdůležitějším protokolem protokol IP. V současné době se setkáváme s protokolem IP ve dvou verzích – 4 a 6. Protože zastoupení v síťovém provozu je velké u obou, budeme se zabývat oběma verzemi.

Z dalších protokolů nás bude zajímat především protokol ICMP určený k posílání krátkých servisních zpráv mezi zařízeními a protokol ARP (třebaže u něj je diskutabilní, na které vrstvě vlastně pracuje) s jeho následníkem NDP pro evidování vztahu mezi IP a MAC adresou nejbližších sousedů v síti. Dále zde pracují směrovací protokoly.

6.2 Protokol IPv4

Hlavním úkolem protokolu IP (Internet Protocol) je přijímat z nadřízené vrstvy segmenty s daty, zapouzdřovat do IP paketu (tj. přidat IP záhlaví) a předat podřízené vrstvě, a naopak.

6.2.1 Adresy IPv4

IPv4 adresa je dlouhá 32 bitů, tedy 4 oktety. V zápisu se jednotlivé oktety oddělují tečkou a zapisují se buď v desítkové soustavě nebo binárně.

Příklad

IPv4 adresa může vypadat třeba takto:

- `10.6.29.181`
- `169.251.220.5`
- `169.251.255.255`
- `255.255.255.255`

Všechny tyto adresy jsou zapsány v desítkové soustavě. První uvedená adresa by v binárním tvaru byla: `1010.110.11101.10110101`. Poslední uvedená: `11111111.11111111.11111111.11111111`.

IPv4 adresou může být označeno jak konkrétní zařízení, tak i například síť nebo podsíť (to vše jsou unicast adresy), dále existují IP adresy skupinové (multicast) a všesměrové (broadcast).

 IP adresy jsou *hierarchické* – to znamená, že zohledňují určité členění zařízení s těmito adresami do skupin a podskupin (sítí a podsítí), přičemž „příbuznost“ dvou zařízení v rámci sítě či podsítě (tedy příslušnost do stejné sítě či podsítě) znamená, že tato dvě zařízení budou mít část adresy stejnou. Takže hierarchie se na adrese projevuje následovně:

- *síťová část adresy (prefix)* – všechna zařízení patřící do stejné sítě mají tuto část shodnou,
- *část adresy hostitele* – v zbývající části adresy se tato zařízení budou lišit.

 Pokud v dané adrese nastavíme všechny bity v hostitelské části na 0, získáme *adresu sítě*. Pokud naopak nastavíme všechny bity v hostitelské části na 1, získáme *broadcast adresu pro danou síť*.

 Jak bylo napsáno výše, někde uvnitř adresy je hranice mezi síťovou a hostitelskou částí adresy. Pro určení této hranice se používá jedna z těchto dvou metod:

1. *Maska sítě nebo podsítě* – v binárním zápisu jsou v (pod)síťové části adresy jedničky a v hostitelské části nuly.
2. *Zápis délkou prefixu* – za adresou je lomítko a číslo určující počet bitů síťové části.

 Na vrstvě L3 se provádí směrování mezi sítěmi, tedy musí být možnost propojení různých sítí. K tomuto propojení potřebujeme zařízení, přes které půjde veškerá komunikace mezi propojenými sítěmi – router nebo switch s funkcionalitou L3. V terminologii IP se tomuto zařízení říká *brána*, a z pohledu jakéhokoliv zařízení v síti je brána takový aktivní síťový prvek, na který směrujeme pakety určené pro zařízení v jiné síti – tj. brána je zařízení pro „cestu ven ze sítě“.

Abychom vůbec byli schopni komunikovat s někým mimo vlastní síť, potřebujeme znát adresu brány. Takže z pohledu vlastní identity a možnosti komunikace na vrstvě L3 by každé zařízení mělo obdržet tyto informace:

- IP adresa,
- maska (pod)sítě nebo délka prefixu,
- adresa brány.

6.2.2 Speciální adresy podle IPv4

✎ *Broadcastová (všesměrová) adresa pro danou síť je* taková adresa, kde jsou všechny bity v hostitelské části nastaveny na 1. Cílem jsou všechna zařízení v dané síti. *Univerzální broadcastová adresa* má všechny bity jak v hostitelské, tak i v síťové části nastaveny na 1, tj. je to adresa `255.255.255.255`. Cílem jsou všechna zařízení v propojených sítích.

✎ *Multicast (skupinová) adresa* se používá jako cílová v těch paketech, které mají být doručeny více než jednomu cílovému zařízení, ale ne nutně všem v síti. Pro tento typ adres je vyhrazen rozsah `224.0.0.0` až `239.255.255.255`
(možná to není vidět, ale jsou to všechny adresy takové, kde jsou první tři bity nastaveny na 1 a čtvrtý na 0). Některé skupinové adresy jsou vyhrazeny pro konkrétní účely:

- `224.0.0.1` označuje skupinu všech zařízení v lokální síti, která „rozumí“ protokolu IPv4, tedy je to obdoba broadcast adresy v lokální síti,
- `224.0.0.2` je skupina všech routerů v lokální síti (takže když chceme poslat paket routerům, pošleme ho právě na tuto adresu),
- `224.0.1.1` je skupina pro NTP servery (časové servery) sloužící k synchronizaci času v síti, atd.

Adresy ve tvaru `224.0.0.x` (tedy včetně prvních dvou uvedených) jsou určeny pouze pro použití v rámci lokální sítě. Mohou být použity jako cílové v paketech pouze s nastaveným TTL = 1, to znamená, že se nedostanou přes router do jiné sítě.

✎ *Loopback adresa* (adresa zpětné smyčky) je adresa začínající oktetem `127`, v naprosté většině případů je to adresa `127.0.0.1`. Loopback adresu má každé zařízení.

Loopback je uvnitř zařízení implementován jako virtuální zařízení, ze síťového pohledu je to vlastně „pohled na sebe sama“ ve směru ze sítě, jakési zrcadlo. Takže když chceme otestovat, zda je naše síťové rozhraní funkční (bez ohledu na to, zda má či nemá přiřazenu IPv4 adresu), otestujeme právě loopback.

✎ Rozlišujeme *veřejné* a *soukromé adresy*. Veřejné adresy jsou celosvětově unikátní a lze je používat bez jakýchkoliv omezení. Soukromé adresy jsou unikátní jen v rámci dané sítě a mimo tuto síť se nesmí dostat (tj. nejdou za router). Pro soukromé adresy se používají tyto adresní rozsahy:

- `10.x.x.x`
- `172.16.x.x` až `172.31.x.x`
- `192.168.0.x` až `192.168.255.x`

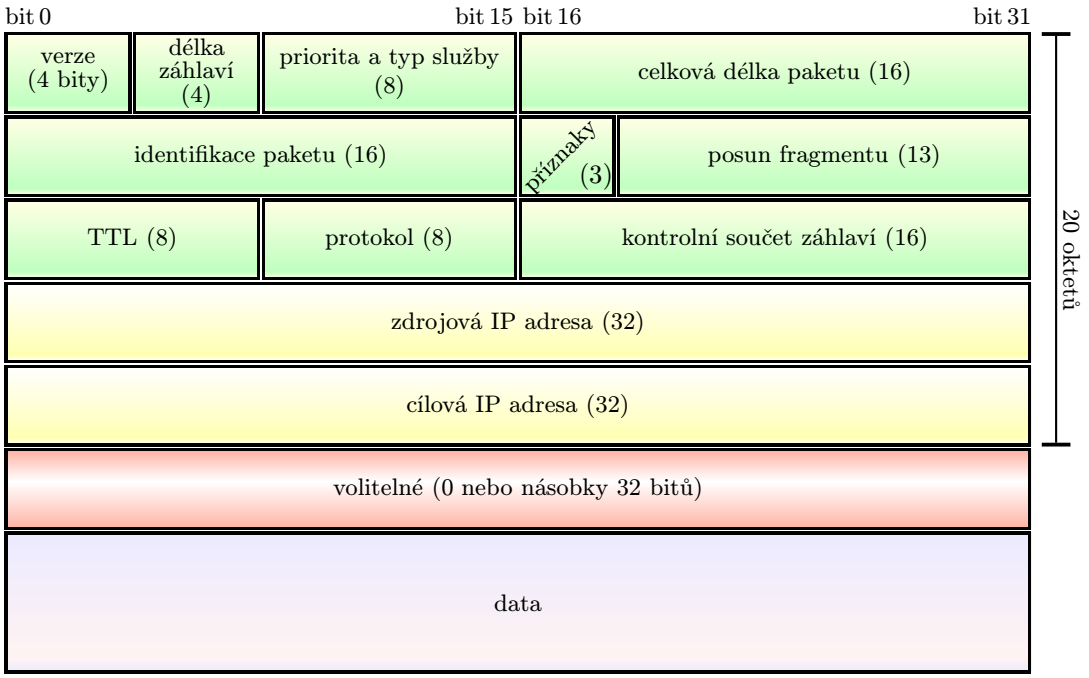
Takže například pokud je prvním oktetem adresy číslo `10`, jde o soukromou adresu. Soukromým adresám se budeme věnovat dále v sekci o NAT.

✎ *Nedefinovaná adresa* je adresa, kterou zařízení používá, když ve skutečnosti ještě žádnou adresu přidělenu nemá. Číselně to je `0.0.0.0`.

6.2.3 IPv4 pakety

✎ Formát IPv4 paketu je naznačen na obrázku 6.1. Velikost polí je udána v bitech. Jednotlivá pole v záhlaví mají tento význam:

- *Verze* (Version, 4 bity) – verze protokolu IP, zde bude číslo 4, binárně 0100.
- *Délka záhlaví* (Header Length, 4 bity) – velikost záhlaví v 32bitových slovech, tedy počet řádků záhlaví (většinou tu najdeme číslo 5, binárně 0101).



Obrázek 6.1: Paket podle IPv4

- *Priorita a typ služby* (Priority and Type of Service, 8 bitů) – určuje, jak má být s tímto paketem zacházeno na cestě. První tři bity určují prioritu podle IEEE 802.1p, zbývající bity určují, zda má být při směrování vybrána spíše cesta s lepší hodnotou pro menší zpoždění zpracování, propustnost, spolehlivost apod. Obvykle je celé pole nastaveno na 0.
- *Celková délka paketu* (Total Length, 16 bitů) – délka paketu včetně záhlaví a zapouzdřených dat v oktetech.
- *Identifikace paketu* (Identification, 16 bitů) – číslo přidělené paketu; odesílatel toto číslo přiděluje během odesílání a mělo by být pro každý odeslaný paket odlišné.
- *Příznaky* (Flags, 3 bity) – používají se pouze dva bity (třetí je rezervován), jejich význam si vysvětlíme později v sekci o fragmentaci.
- *Posun fragmentu* (Fragment Offset, 13 bitů) – v případě, že bylo třeba paket fragmentovat, je do tohoto pole uloženo číslo pro výpočet pozice části dat přenášené v tomto fragmentu vzhledem k původním nerozděleným datům.
- *TTL* (Time to Live, 8 bitů) – životnost paketu. Na každém aktivním síťovém prvku s funkčností L3 (třeba routeru) se číslo v tomto poli vždy sníží o 1. V případě, že klesne na 0, je paket považován za bloudící a zahozen, přičemž je obvykle odesílatel o zahození informován.
- *Protokol* (Protocol/Type, 8 bitů) – informace o tom, co je v paketu zapouzdřeno.
- *Kontrolní součet záhlaví* (Header Checksum, 16 bitů) – počítá se přes předchozí 2oktetové sekvence.
- *Zdrojová a cílová adresa* (Destination and Source Address, každá 32 bitů) – zdrojová adresa bývá vždy unicast.
- *Volitelné* (Options, 0 nebo násobky 32 bitů) – slouží k servisním účelům, například k ovlivnění směrování paketu sítěmi. Není povinné a obvykle toto pole nepoužíváme.
- *Data* (Payload) – přenášená data podle konkrétního protokolu.

Protože je pole pro celkovou délku paketu (čtvrté pole) dlouhé jen 16 bitů, je maximálním možným číslem $2^{16} - 1 = 65\,535$, z čehož vyplývá omezení pro maximální délku celého paketu. Omezení pro délku zapouzdřených dat získáme odečtením délky záhlaví.

V poli *Protokol* najdeme identifikátor protokolu, jehož datová jednotka je v IP paketu zapouzdřena. Svůj identifikátor pro toto pole mají například protokoly TCP (6) a UDP (17) z transportní vrstvy, ale také ty protokoly ze síťové vrstvy, které jsou zapouzdřovány do IP paketu, například ICMP (1) a směrovací protokoly, například OSPF (89).

Další informace:

Seznam všech hodnot pro pole Protokol najdeme na
http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml.

6.2.4 Pole TTL a životnost paketu

Jak bylo napsáno výše, v poli TTL (Time to Live) je číslo, které se na každém aktivním síťovém prvku pracujícím na vrstvě L3 snižuje většinou o 1. Účelem je omezit existenci nedoručitelných paketů, které by jinak do nekonečna bloudily sítí a zahlcovaly spoje (na vrstvě L2 máme k tomu účelu protokol STP, tady je třeba použít jiný mechanismus, protože smyčkám a redundantním spojům na L3 se prostě vyhnout nedá).

U některých protokolů, jejichž PDU se zapouzdřují do IP paketů, je přímo stanoveno, jaké TTL se má použít (například hodnotu 1 dáváme v případě, že paket nemá opustit síť). U jiných existují určité zvyklosti, jakou hodnotu konkrétně použít. Maximální hodnota je samozřejmě 255 (protože máme 8 bitů a $2^8 - 1 = 255$), ale většinou se používá číslo 64 nebo 128. Je to záležitost operačního systému (například Windows používají hodnotu 128).

Další informace:

Typické hodnoty pro pole TTL:

- http://subinsb.com/default-device-ttl-values
- http://www.binbert.com/blog/2009/12/default-time-to-live-ttl-values/

Pole TTL se také nazývá „hop limit“, maximální počet přeskoků přes sítě. Tento druhý název se dokonce používá jako výchozí u následující verze protokolu – IPv6.

6.2.5 MTU a fragmentace IPv4 paketu

Definice (MTU)

MTU (Maximum Transmission Unit) je hodnota definovaná pro určitý spoj, která stanovuje maximální velikost paketu pro odeslání přes tento spoj. MTU konfigurované na konkrétním zařízení znamená maximální velikost paketu, který toto zařízení dokáže přijmout a zpracovat.

MTU je především ovlivněno nastavením na aktivních síťových prvcích, přes které je paket směrován, ale také záleží na konkrétním protokolu vrstvy L2, do kterého je IP paket zapouzdřen.

V případě IP paketu zapouzdřovaného na vrstvě L2 do ethernetového rámce máme jako maximum číslo 1500 (ale například pro Token Ring je toto číslo zhruba dvojnásobné). Protože naprostá většina (především lokálních) sítí používá na vrstvě L2 Ethernet, je podobná hodnota MTU nastavena i na mnoha aktivních síťových prvcích. Vzhledem k tomu, že data v IP paketu mohou být až 65 535 oktetů dlouhá (a k tomu musíme přičíst i IP záhlaví), je jasné, že tu máme značný nepoměr.

Definice (Jumbo rámec a jumbogram)

Na vrstvě L2: *Jumbo rámec* (jumbo frame) je ethernetový rámec, jehož velikost překračuje stanovenou maximální hodnotu. Zatímco maximum pro payload v běžném rámci je 1500 oktetů, u jumbo rámce to je obvykle 9000 oktetů. Může být odeslán pouze po takové přenosové cestě, kde jsou všechna zařízení na cestě nakonfigurována na přijímání jumbo rámců. U síťových rozhraní pro Gigabit Ethernet to obvykle není až takový problém, u nižších rychlostí už méně pravděpodobně.

Na vrstvě L3: *Jumbogram* je IP paket větší než 65 535 oktetů.

Do jumbo rámce sice naskládáme více než do obyčejného rámce, ale pro uložení opravdu velkého IP paketu to taky nestačí. Z toho důvodu potřebujeme jiný mechanismus: *fragmentaci IP paketu*, tedy možnost IP paket rozdělit na menší části (fragmenty) podle velikosti MTU a zajistit, aby cílové zařízení dokázalo tyto fragmenty zkompletovat do původního paketu.

Každý fragment se samozřejmě taky musí stát paketem, tedy ke každému fragmentu připojíme IP záhlaví. Většinu polí fragment „zdědí“ z původního paketu, ale některá pole budou jiná. Pro fragmentaci a především následné sestavení v cíli potřebujeme určité konkrétní hodnoty v polích druhého řádku paketu podle obrázku 6.1 na straně 140:

- *Identifikace paketu* (16 bitů) – všechny fragmenty mají toto pole stejné (zdědí po původním paketu), v cíli slouží k určení, které fragmenty patří k sobě,
- *Příznaky* (3 bity) – zajímají nás dva jednobitové příznaky:
  - DF (Do not Fragment) – abychom mohli fragmentovat, musí být tento příznak nastaven na 0; pokud odesílatel tento příznak nastaví na 1, zakáže fragmentaci po cestě,
  - MF (More Fragments) – nastavíme na 1 ve všech fragmentech kromě posledního,
- *Posun fragmentu* (13 bitů) – u každého fragmentu zjistíme, na kterém oktetu původních dat začíná, toto číslo vydělíme 8 a uložíme sem; z toho vyplývá, že velikost dat ve fragmentu je vždy násobek 8 oktetů (abychom se vždy mohli „strefit“ na použitelnou adresu), kromě posledního.

Postup (Fragmentace IPv4 paketu)

Pokud je IP paket větší než kolik dovoluje MTU na cestě, musíme předně zkontrolovat, jestli vůbec můžeme fragmentovat. Jestliže je v záhlaví v poli *Příznaky* bit DF nastaven na 1, ani se s fragmentací neobtěžujeme a rovnou paket zahodíme. V opačném případě postupujeme takto:

1. Vypočteme délku dat pro fragment a počet fragmentů:
   - vezmeme hodnotu MTU na následné cestě a odečteme velikost záhlaví paketu (bývá obvykle 20 oktetů, ale pozor, může být větší),

- vzniklé číslo vydělíme celočíselně osmi (zaokrouhlíme směrem dolů), toto číslo si označme X,
- maximální možná velikost dat zapouzdřených do paketu je tedy $X * 8$,
- pokud je původní délka dat *po odečtení záhlaví* paketu M, pak počet výsledných fragmentů je $M/(X * 8) + 1$ s použitím celočíselného dělení, přičtená jednička je poslední fragment se „zbytkem“.

2. Vytvoříme první fragment:
   - většinu záhlaví zkopírujeme z původního paketu (včetně *Identifikace paketu*),
   - příznak MF (More Fragments) nastavíme na 1,
   - do pole *Posun fragmentu* uložíme číslo 0, protože tento fragment je první v pořadí a v posloupnosti oktetů z původního paketu se začíná právě adresou 0,
   - jako data vložíme do prvního fragmentu oktety s adresami $0 \ldots (X * 8 - 1)$ (tedy prvních $X * 8$ oktetů).
3. Vytvoříme další fragmenty kromě posledního (n je pořadí paketu, který právě zpracováváme, od $n = 2$):
   - opět přejmeme většinu záhlaví,
   - příznak MF (More Fragments) nastavíme na 1,
   - do pole *Posun fragmentu* uložíme číslo $X * (n - 1)$ (obsah tohoto pole po vynásobení osmi dává adresu začátku tohoto fragmentu v původních datech),
   - jako data vložíme oktety s adresami $X*8*(n-1) \ldots X*8*n-1$ (tedy n-tých $X*8$ oktetů).
4. Vytvoříme poslední fragment ($n = M/(X * 8) + 1$):
   - přejmeme většinu záhlaví,
   - příznak MF (More Fragments) nastavíme na 0, protože další fragmenty už nebudou,
   - do pole *Posun fragmentu* uložíme číslo $X * (n - 1)$,
   - jako data vložíme oktety s adresami $X * 8 * (n - 1) \ldots M - 1$.

Skládání fragmentů se provádí vždy až v cíli, přičemž se může stát, že některé fragmenty budou po cestě znovu fragmentovány. Cílové zařízení

- shromáždí všechny pakety s tímtéž identifikátorem paketu (první pole druhého řádku),
- seřadí je podle pole *Posun fragmentu*, přičemž poslední v pořadí by měl být fragment s příznakem MF nastaveným na 0,
- zkontroluje, zda některý fragment nechybí (u každého fragmentu porovná jeho délku s polem Posun fragmentu z následujícího fragmentu v pořadí).

Pokud vše souhlasí, seskládá fragmenty dohromady, pokud však najde chybu, všechny fragmenty zahodí a přenos se musí opakovat (v režii nadřízené vrstvy).

http://www.root.cz/clanky/velke-trable-s-malym-mtu/

Poznámka:

Všimněte si, že tady nikde není zmínka o žádosti o opětovné poslání. Protokol IP je v principu „nespolehlivý“ a ničím takovým se nezabývá (poskytuje službu typu „best effort“). Znovuposlání si dojednají nadřízené vrstvy, pokud je to zapotřebí.

Na síťové vrstvě taktéž není navazováno spojení (alespoň to neprovádí protokol IP). Právě proto se v literatuře často píše o *IP datagramech* (místo IP paketů) – připomeňme si, že datagramová služba je služba bez navázání spojení.

6.2.6 NAT a soukromé adresy

 O *soukromých adresách* už něco víme – jsou pro ně v každé třídě vyhrazeny určité rozsahy a pakety s touto adresou nesmí „za router“.

Jejich hlavním účelem je šetření rozsahů veřejných IP adres. Druhým účelem je mírné zvýšení zabezpečení, protože soukromé adresy je třeba na hranici sítě překládat na veřejné (nebo na soukromé fungující v nadřízené síti), což může být chápáno jako dodatečný ochranný mechanismus.

Soukromé adresy je třeba překládat na hranici sítě, ale ve skutečnosti tento překlad můžeme využívat i u jiných typů adres. Překlad v základu spočívá v tom, že v paketu zaměníme jednu IP adresu za jinou.

Poznámka:

Abychom si nepletli pojmy:

- veřejná adresa je globálně platná, může do Internetu (resp. za router),
- soukromá adresa je pouze lokálně platná, nemůže za router.

Dále rozlišujeme staticky a dynamicky přidělenou adresu:

- statická adresa je pevně určena danému zařízení, jiné zařízení ji nemůže získat,
- dynamická adresa je momentálně danému zařízení přidělena, ale zítra ji může mít úplně jiné zařízení.

Adresa zařízení může být soukromá a zároveň dynamická nebo soukromá a zároveň statická. Veřejné adresy bývají statické, ale ne každá statická adresa je veřejná.

 *NAT (Network Address Translation)* je mechanismus překladu IP adres na hranici sítě. V paketu můžeme překládat jen zdrojovou adresu nebo jen cílovou adresu nebo obojí, obvykle se kombinuje překlad zdrojové adresy v jednom směru a překlad cílové adresy v druhém směru.

Rozlišujeme tyto druhy NAT:

- Statický NAT (bezstavový) – je pevně určeno, kterou adresu z vnitřní sítě překládáme na kterou adresu platnou ve vnější síti a naopak.
- Dynamický NAT (stavový, Masquerade) – dynamicky se určuje dvojice adres pro překlad, vyžaduje uložení záznamu o překladu, který se využije pro zpětný překlad.
- PAT (Port Address Translation, přetížení NAT) – přidává k dynamickému NAT dodatečný mechanismus rozlišení jednotlivých konverzací.

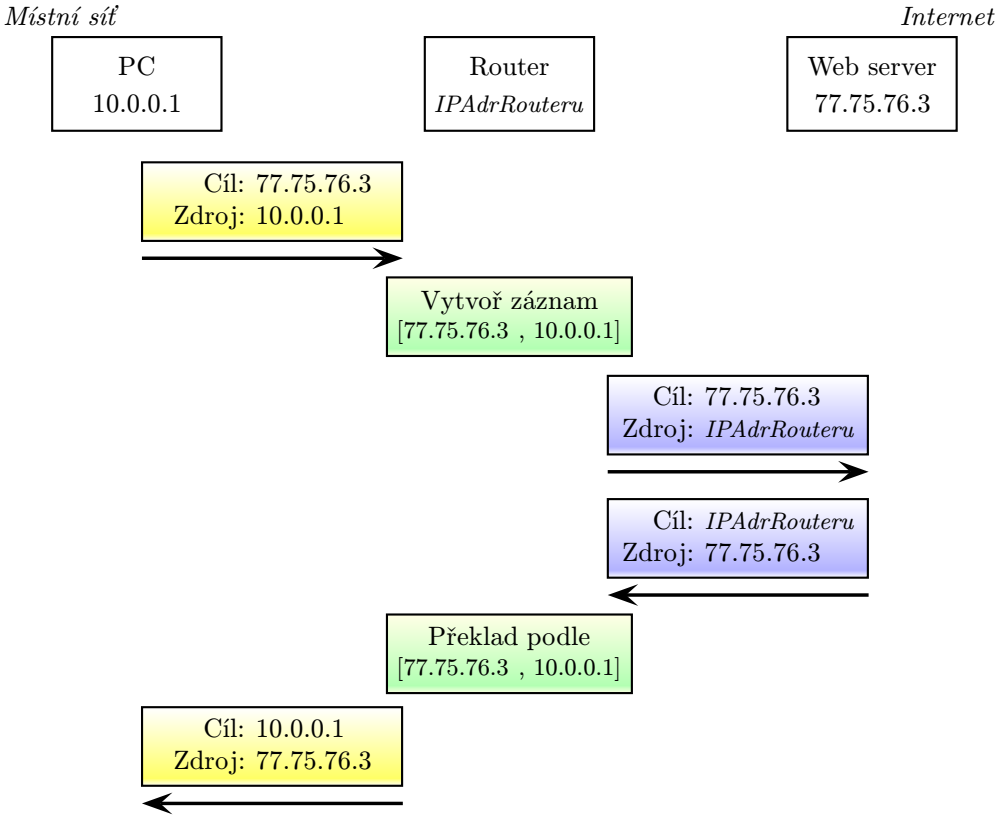
 *Statický NAT* znamená, že v odchozím provozu zaměníme v paketu zdrojovou IP adresu našeho zařízení za jinou, pod kterou má být toto zařízení viditelné „venku“, a naopak v příchozím provozu provedeme na (tentokrát cílovou) IP adresu tohoto zařízení zpětný překlad. Statický NAT vyžaduje,

abychom měli pro dané zařízení staticky vyhrazeny dvě adresy – jednu pro vnitřní síť a druhou pro vnější síť.

Typicky se používá pro skrývání IP adres našich serverů. Server je „venku“ viditelný pod jinou IP adresou než jakou má ve skutečnosti. Dalším důvodem využití statického NATu je dočasná změna v síti třeba z důvodu údržby.

Dynamický NAT (Masquerade, českým patvarem „maškaráda“) je již stavová konfigurace. Používá se v případě, že vnitřní adresy, které překládáme, jsou dynamické, a tedy si je nelze „staticky zapamatovat“. Abychom ušetřili adresní rozsah, máme ve vnitřní síti dynamicky přidělované adresy a směrem ven je překládáme na jedinou adresu (společnou pro všechno dynamické z vnitřní sítě).

NAT server (to bývá router provádějící NAT překlad) si vede tabulku spojení, ve které si pro každé spojení mezi vnitřní a vnější sítí zaznamenává adresu zařízení ve vnitřní síti a adresu zařízení z vnější sítě (typicky některého serveru na Internetu, se kterým je spojení navazováno). Princip je naznačen na obrázku 6.2.



Obrázek 6.2: Zjednodušené schéma dynamického NAT

Jednoznačnost je zajištěna v případě, že každé zařízení z vnitřní sítě komunikuje s jiným serverem. Problém nastává tehdy, když víc zařízení z vnitřní sítě chce komunikovat s tímtéž „vnějším“ serverem, pak by záznamy v tabulce spojení přestaly být jednoznačnými. Ve směru od klienta k serveru by se paket poslat dal, ale odpověď od serveru bychom nemohli doručit příslušnému klientovi, protože bychom nevěděli, kterému.

PAT (překlad portů, přetížený NAT) řeší právě tento problém. Do IP paketů se obvykle zapouzdřují TCP nebo UDP segmenty, ve kterých používáme zdrojové a cílové číslo portu. Klientská čísla portů

bývají dynamická, tj. je velká pravděpodobnost, že dva různí klienti z naší sítě komunikující s tímtéž serverem budou mít tato čísla odlišná.

Takže místo abychom do tabulky ukládali jen IP adresy, přidáme i čísla portů (připomeňme si, že například socket je identifikován dvojicí IP adresa + číslo portu). Pokud by přesto nastala situace, že by dva klienti z vnitřní sítě při komunikaci s tímtéž serverem použili stejné zdrojové číslo portu, na NAT serveru by byla přeložena nejen IP adresa, ale i číslo portu tak, abychom zajistili jednoznačnost.

Další informace:

Mechanismus NAT je obvykle chápán jako berlička pro IPv4, ale i v IPv6 pro něj mohou existovat důvody: http://www.lupa.cz/clanky/10-duvodu-proc-mit-nat-na-ipv6/

6.2.7 Jak získat IPv4 adresu

IPv4 adresu můžeme získat

- dynamickou alokací podle protokolu DHCP,
- staticky, přičemž buď adresu zadáváme do konfigurace ručně nebo použijeme automatickou statickou alokaci (také pomocí DHCP).

Statická IP adresa je tedy adresa pevně určená pro dané zařízení, zařízení jinou ani nedostává. Typicky se statické IP adresy používají pro servery, které mají být dostupné vždy pod stejnou adresou, aby se nemusely průběžně měnit směrovací tabulky. Naproti tomu *dynamická IP adresa* je vždy propůjčena (leased) jen na určitou dobu (třeba jeden den) a po této době může zařízení získat znovu tutéž adresu nebo úplně jinou.

DHCP server je server zajišťující přidělování a evidování dynamických IP adres a taky adres přidělovaných statickou alokací. Má přidělen rozsah adres, které může přidělovat (Address Pool, Address Stack) a v tomto rozsahu si poznamenává, komu (které MAC adrese) zatím přidělil kterou IP adresu a na jak dlouho (leased time). Tato tabulka je vlastně stavovou informací DHCP serveru (určuje stav přidělování adres), a tedy využití DHCP serveru pro přidělování adres označujeme jako *stavový režim DHCP*.

Co se týče *statické alokace*, probíhá tak, že na DHCP serveru máme nadefinováno přiřazení konkrétní IP adresy ke konkrétní MAC adrese. Jestliže zařízení s takovou MAC adresou žádá DHCP server o IP adresu, je mu přidělena vždy tatáž.

6.3 Protokol IPv6

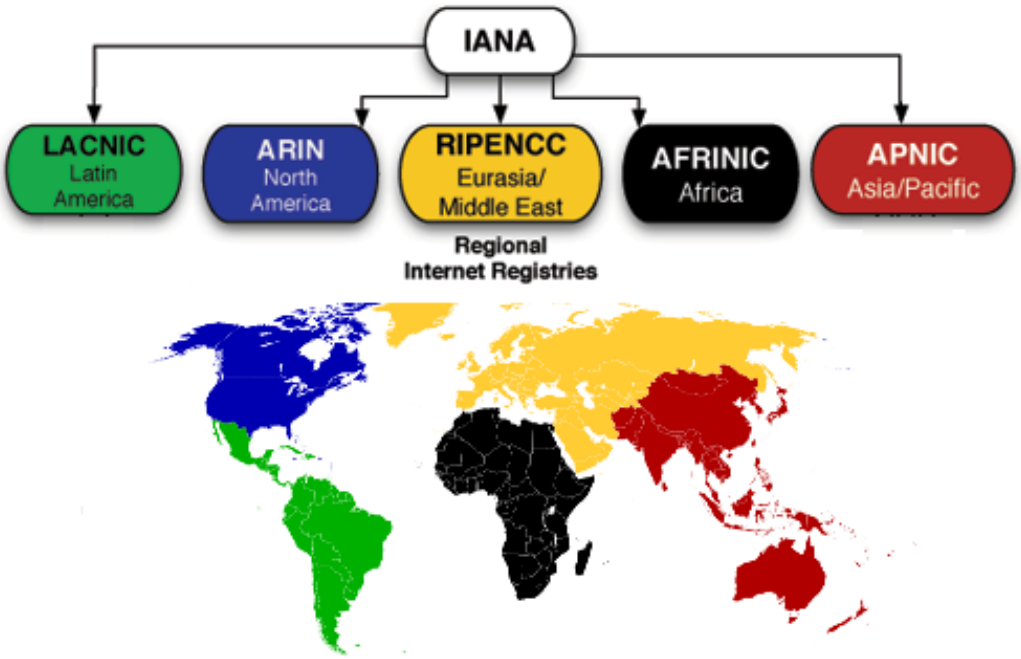
Protokol IP verze 6 (IPv6, také IPng – next generation) má vyřešit zoufalou situaci s akutním nedostatkem adres protokolu IPv4. Zatímco adresy IPv4 zabírají 32 bitů, adresy IPv6 jsou dlouhé 128 bitů, což by bohatě mělo stačit pro jakákoliv zařízení na světě (teoreticky jde o počet 10^{38} adres).

Ve skutečnosti je důvodů přechodu z IPv4 na IPv6 více než jen rozsah adres. Důležitá je například rozsáhlejší podpora zabezpečení – IPv4 tuto možnost vůbec nenabízí, ale v současné době je bezpečnost mnohem důležitější než v minulých desetiletích. Další důvody jsou zvýšená podpora pro mobilní zařízení, možnost zjednodušeného získání IP adresy pro koncová zařízení a další.

Protokoly IPv4 a IPv6 mohou být používány zároveň, dokonce i na tomtéž uzlu v síti.

Hlavním garantem přidělování IPv6 adres je *ICANN* (Internet Corporation for Assigned Network Numbers, http://www.icann.org), kdežto organizace *IANA* (Internet Assigned Numbers Authority, http://www.iana.org/) toto přidělování fyzicky provádí.

Celková struktura přidělování adres je hierarchická. IANA přiděluje bloky adres *regionálním registrátorům RIR* (Regional Internet Registery), což jsou RIPE (Evropa a část Asie), ARIN (Severní Amerika), AfriNIC (Afrika), LACNIC (Latinská Amerika), APNIC (Asie a Pacifik). Další patro hierarchie tvoří *lokální registrátoři* (LIR), kteří své bloky získávají od regionálních registrátorů. Od lokálních registrátorů pak své rozsahy adres získávají zákazníci nebo další subjekty, které mohou své rozsahy dále distribuovat.



Obrázek 6.3: Základ struktury přidělování IP adres[1]

Na obrázku 6.3 jsou naznačena horní dvě patra této struktury. Níže jsou registrátoři pro jednotlivé země LIR, což mohou být třeba ISP (poskytovatelé Internetu), následují jejich zákazníci a drobní poskytovatelé Internetu.

Další informace:

Seznam LIR pro Evropu je na https://www.ripe.net/participate/member-support/info/list-of-members/europe, seznam pro ČR získáte tak, že v mapě klepnete na „Czech“.

6.3.1 Adresy IPv6

Adresa podle IPv6 je 128 bitů dlouhá (tj. 16 oktetů, čtyřnásobek IPv4 adresy) a skládá se ze dvou částí – *prefixu* a *identifikátoru síťového rozhraní*.

[1] Zdroj: http://whitengreen.com/blog-1124-how-to-trace-and-locate-ip-addresses

Protože jsou IPv6 adresy hodně dlouhé, zapisujeme je hexadecimálními číslicemi ve skupinách po čtyřech číslicích (tj. po dvou oktetech) oddělené dvojtečkou. Z toho vyplývá, že adresa bude mít osm částí oddělených dvojtečkami.

Příklad

IPv6 adresa může vypadat takto:

- `a4cb:57b1:60aa:000E:113a:b201:042a:02b1`
- `a4cb:57b1:60aa:E:113a:b201:42a:2b1`
- `2001:0db8:3c4d:0000:0000:a010:0000:0000`
- `2001:db8:3c4d:0:0:a010:0:0`

Všimněte si, že ve dvojicích adres pod sebou je vlastně spodní adresa stejná jako horní, jen jsme v jednotlivých částech odstranili nuly zleva. Totéž jsme samozřejmě mohli udělat i u IPv4.

Definice

Kanonický tvar IPv6 adresy je standardizován jako RFC 5952 a předepisuje tyto podmínky:

- hexadecimální číslice se mají zapisovat malými písmeny,
- vynechávání počátečních nul ve skupině je povinné (takže v příkladu by byl správně zkrácený tvar na druhém řádku každého sloupce),
- mechanismus zkrácení počtu skupin pomocí `::` musí mít co největší efekt, což znamená, že pokud máme víc řad nulových skupin, vybírá se ta delší, když je víc stejně dlouhých, vybereme tu víc vlevo, a musí pohltit všechny dosažitelné nulové skupiny; pokud je nulová skupina v adrese jen jedna, konstrukce `::` se nepoužije.

Kanonický tvar adresy je jednoznačný.

Oproti IPv4 lze zápis zjednodušit *odstraněním posloupnosti nulových skupin oktetů*. V jedné adrese tak můžeme odstranit pouze jednu posloupnost nulových skupin a místo odstranění musí být označeno dvojitou dvojtečkou.

Příklad

Například adresu

`2001:0db8:3c4d:0000:0000:a011:0000:0000`

krátíme takto:

`2001:db8:3c4d::a011:0:0`

V nenulových skupinách jsme umazali nuly zleva (například místo `0db8` je `db8`), jsou tam dvě posloupnosti nulových skupin stejně dlouhé, takže pro zlikvidování si vybereme tu víc vlevo.

Jiná možnost by byla `2001:db8:3c4d:0:0:a011::`, ale zde nejde o kanonický tvar. Sice z této zkrácené verze dokážeme rekonstruovat „plnou“ adresu, ale postup krácení neodpovídal požadavku pro kanonický tvar (v informatice je třeba používat jednoznačné postupy).

špatně by bylo `2001:0db8:3c4d::a011::`, protože po tomto krácení již nedokážeme jednoznačně rekonstruovat „plnou“ adresu. Mohla by to být například i chybná možnost

`2001:0db8:3c4d:0000:a011:0000:0000:0000`

6.3.2 Speciální adresy podle IPv6

Podle IPv6 rozlišujeme tyto typy adres:

- unicast (konkrétní uzel v síti),
- multicast (skupinové),
- anycast (adresace komukoliv ze zadané skupiny).

Broadcast adresy již nejsou podporovány.

Co se týče konkrétních adres zařízení, existují tyto možnosti:

- *Unique Local Address* (ULA adresy) – slouží k posílání unicast dat v rámci lokální sítě (organizace apod.), je to obdoba soukromé IP adresy v IPv4, nesmí být viditelná mimo lokální síť, ale je zde víceméně zaručena unikátnost (na rozdíl od následujícího typu adresy). Unikátnost ULA adresy se zajišťuje odvozením z data (času) generování adresy a z MAC adresy stanice. ULA adresy mají prefix `fd00::/8`, takže hexadecimálně vždy začínají oktetem `fd`.
- *Link Local Address* (lokální na segmentu) – významem ještě více odpovídají soukromým adresám podle IPv4, není zaručena unikátnost (v jiné síti může existovat zařízení s naprosto stejnou linkovou lokální adresou), ale je unikátní alespoň v rámci segmentu. Pakety s touto adresou jako cílovou nepřecházejí přes router. Tyto adresy mají vždy prefix `fe80::/10`, takže prvních deset bitů je `1111 1110 10` (pozor, první oktet je jasný – `fe`, ale dál už je to složitější, třetí hexadecimální číslice může být `8`, `9`, `a` nebo `b`).
- *Globální adresy* – jsou vždy unikátní v rámci celého Internetu, jejich prefix je vždy `2000::/3`, tedy v binárním zápisu začínají třemi bity `001` a hexadecimálně je první nibble (čtveřice bitů) na hodnotě `2` nebo `3`.

Původně se počítalo ještě s tzv. *site local* adresou, která by fungovala přesně jako lokální adresy v IPv4 (včetně překladu adres), její prefix je `feC0::/10`. Nicméně tento typ adres byl velmi brzy ze standardu vyňat a jediný, kdo s ním počítá, je společnost Microsoft v některých svých technologiích.

Loopback (zpětná či lokální smyčka) má tentýž význam jako u IPv4, tedy jde o testovací adresu znamenající „pohled zvenčí na sebe sama“. Adresa loopbacku v IPv6 je `::1/128`, což bychom mohli přepsat jako `0:0:0:0:0:0:0:1`.

Nedefinovaná adresa (tedy informace, že stanice nemá přiřazenu adresu) je `::/128`, což v přepisu znamená `0:0:0:0:0:0:0:0`.

Také v IPv6 se používají *skupinové adresy* (pro multicast), jejich prefix je vždy `ff00::/8` (to znamená, že prvních osm bitů je nastaveno na 1). Některé multicast adresy jsou vyhrazeny, například:

- `ff02::1` – skupinová adresa všech uzlů sítě podporujících IPv6,
- `ff02::2` – skupinová adresa všech dostupných routerů (směrovačů),
- `ff02::1:2` – skupinová adresa představující všechny dostupné DHCP servery (od nich lze získat dynamickou IP adresu).

Další informace:

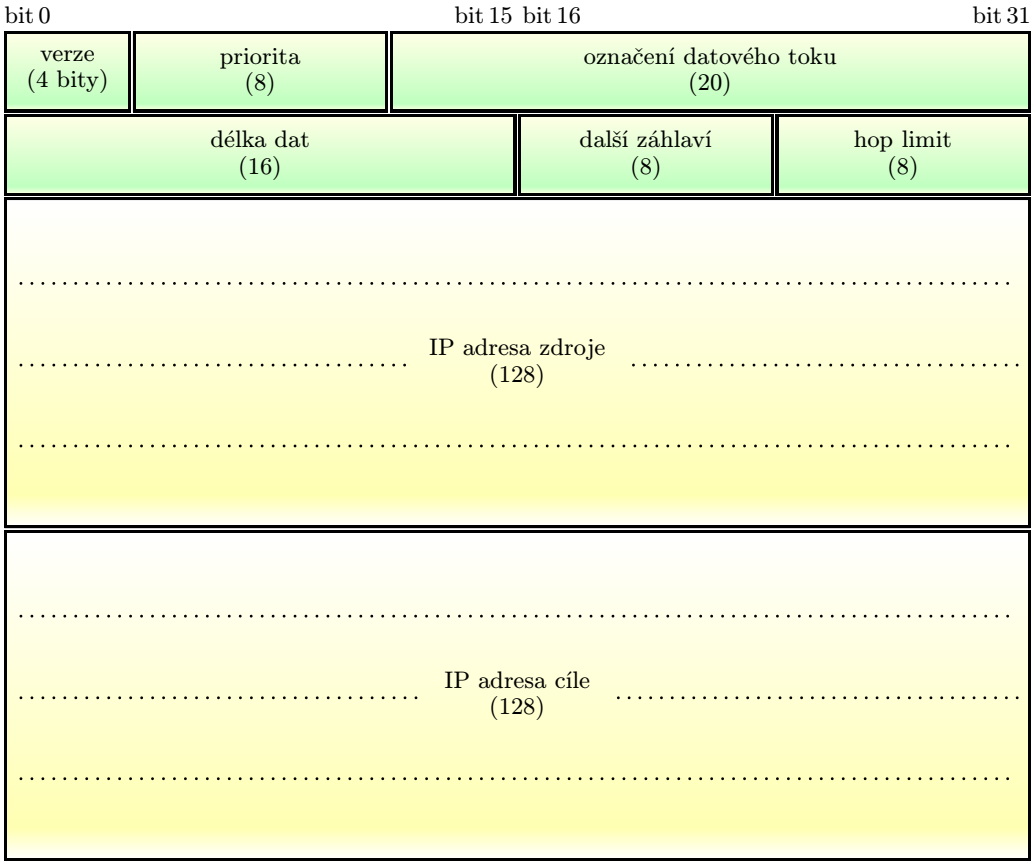
Seznam všech dobře známých a registrovaných multicast adres je na
http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml.

Anycast adresy jsou v principu sice také skupinové (s tím rozdílem, že adresátem není každý člen skupiny, ale jen jeden z nich), ale ve skutečnosti se s nimi pracuje trochu jinak a nemohou označovat koncová zařízení (hosty). Nemají vyhrazený prefix, na první pohled je nijak nerozeznáme od unicast adresy.

Anycast adresy se globálně prakticky nepoužívají (pro routery by to bylo zbytečně složité), spíše se s nimi setkáme v lokálních sítích. Odlišné zacházení než s unicast adresami se zajistí konfigurací v směrovacích tabulkách na síťových zařízeních (routerech apod.).

6.3.3 IPv6 pakety

Struktura IP paketu verze 6 je oproti verzi 4 značně pozměněná. Hlavní odlišnost je struktura záhlaví. Zatímco záhlaví IPv4 paketu je jen jedno a může mít různou délku (podle pole *Volitelné*), v paketu podle IPv6 máme jedno povinné záhlaví o pevné délce (několik nejdůležitějších polí) a dále můžeme přidat volitelná záhlaví, z nichž každé má svůj stanovený účel.



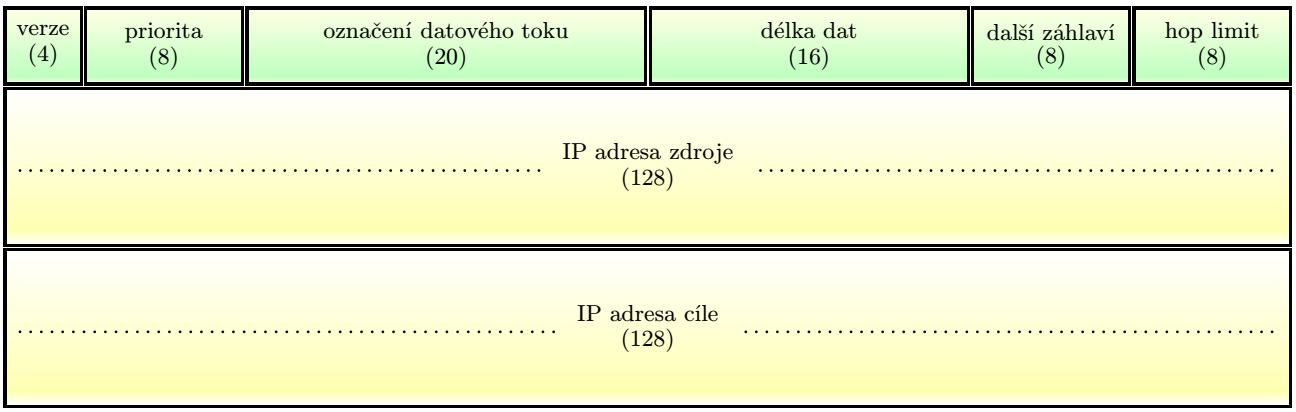
Obrázek 6.4: Paket podle IPv6 – povinné záhlaví

Povinné záhlaví pevné délky je naznačeno na obrázku 6.4. Jak vidíme, první pole je stejné jako u předchozí verze (číslo verze), jen jeho obsah bude samozřejmě jiný – u paketu podle IPv4 tam bude uložena hodnota 4 (binárně 0100), kdežto u paketu podle IPv6 tam bude hodnota 6 (binárně 0110). Další pole jsou už úplně jiná, ovšem přijímající zařízení už podle toho prvního pole rozhodne, o jakou verzi jde a jaká pole má očekávat dále.

Význam jednotlivých polí:

- *Verze* (Version, 4 bity) – verze protokolu (tedy 6).
- *Priorita* (Priority, 8 bitů) – podobně jako pole *Typ služby* u IPv4, jednotlivé bity umožňují optimalizaci priorit.
- *Označení datového toku* (Flow Label, 20 bitů) – stanovuje způsob „speciálního zacházení“ na směrovačích pro některé typy protokolů, s pakety patřícími do téhož datového toku se má zacházet stejně (žádný datový tok: = 0). Pokud se používá, pak zároveň s předchozím polem (pro všechny pakety téhož datového toku platí stejná priorita).
- *Délka dat* (Payload Length, 16 bitů) – délka zbytku paketu (bez povinného záhlaví), tj. všech volitelných záhlaví a vlastních dat; pokud je zde hodnota 0, jedná se o *jumbogram*.
- *Další záhlaví* (Next Header, 8 bitů) – určuje typ následujícího (volitelného) záhlaví téhož paketu nebo typ zapouzdřených dat.
- *Hop limit* (8 bitů) – obdoba TTL u IPv4, označuje maximální počet směrovačů či jiných zařízení vrstvy L3 na cestě, na každém síťovém prvku se snižuje o 1.
- *Zdrojová a cílová IP adresa* (obě 128 bitů, tj. pro každou adresu čtyři řádky podle obrázku 6.4).

Na obrázku 6.5 je vlastně totéž jako na předchozím, jen je zdvojnásobena velikost řádku z 32 bitů na 64. Zatímco před lety byly výpočetní systémy (počítače, servery apod.) 32bitové, dnes jsou již téměř výhradně 64bitové a tedy se do procesoru (a jinam) načítá vždy najednou 64 bitů. IPv6 je optimalizován právě pro 64bitové zpracování dat, a jak vidíme, téměř všechna pole povinného záhlaví (všechna zeleně podbarvená) se načtou při jediném přístupu, každá z adres pak po dvou přístupech.



Obrázek 6.5: Paket podle IPv6 – povinné záhlaví pro řádek 64 bitů

Volitelná záhlaví následují za povinným záhlavím v předem daném pořadí (tedy pořadí je důležité, RFC 2460), některá (nebo i všechna) mohou být vynechána. Každý typ volitelného záhlaví má své číslo, toto číslo najdeme v poli *další záhlaví* předchozího záhlaví (tj. v povinném záhlaví zjistíme, jakého typu je první volitelné záhlaví, v prvním zjistíme typ druhého volitelného, atd.).

Některá z používaných volitelných záhlaví:

- *Hop-by-hop Options Header* (0, informace pro směrovače na cestě, směrovače čtou z volitelných jen toto záhlaví),
- *Routing Header* (číslo 43, použijeme, pokud chceme „natvrdo“ předepsat cestu, zde mohou být určeny směrovače, přes které má cesta vést, v obráceném pořadí je závazné i pro odpověď),

- *Fragment Header* (44, pokud je paket na zdrojové stanici fragmentován, je použito toto záhlaví s informací o fragmentaci, podobné údaje jako v záhlaví IPv4; stanice musí znát hodnoty MTU na cestě),
- *Encapsulating Security Header* (50, údaje o šifrování),
- *Authentication Header* (číslo 51, obsahuje autentizační informaci),
- TCP segment (6), UDP segment (17), ICMP paket (58), ...

Poznámka:

Všimněte si, že v poslední odrážce předchozího seznamu máme kódy pro TCP segment, UDP segment, ICMP paket atd. Z toho vyplývá, že pole *další záhlaví* plní v IPv6 paketu také roli pole protokol z IPv4 paketu – říká, co je zapouzdřeno uvnitř (je zajímavé, že pro ICMPv6 je jiný identifikátor než pro ICMPv4).

Takže skutečný význam pole *další záhlaví* je: určuje, co konkrétně následuje za tím záhlavím, ve kterém toto pole právě čteme. Může to být buď některé volitelné záhlaví nebo přímo data předaná konkrétním protokolem z transportní nebo síťové vrstvy.

Fragmentace IPv6 paketu může být provedena pouze odesílajícím zařízením, žádné zařízení na cestě ji nesmí provést. Pokud některý mezilehlý síťový prvek (router nebo switch s funkcionalitou L3) zjistí, že daný IPv6 paket je větší než kolik dovoluje MTU na cestě, pak tento paket zahodí, jinou alternativu nemá. Ovšem fragmentovat může odesílatel, a tehdy použije volitelné záhlaví číslo 44, které je pro tento účel určeno a obsahuje podobné informace jako IPv4 paket v druhém řádku podle obrázku 6.1.

6.3.4 Jak získat IPv6 adresu

V síti se zprovozněným IPv6 každý router v pravidelných intervalech rozesílá ICMPv6 zprávu *Router Advertisement – RA* (oznámení směrovače). Router (nebo switch s funkcionalitou L3) do těchto zpráv vkládá všechny důležité informace o sobě a o síti, například síťový prefix (tedy adresa sítě), délka prefixu, adresa brány, hodnota MTU apod.

Pokud zařízení chce doprovodné údaje k IPv6 adrese, buď naslouchá na síti nebo se aktivně zeptá ICMPv6 zprávou *Router Solicitation – RS* (dotaz na oznámení směrovače), kterou donutí router vyslat RA.

Poznámka:

Zprávy RA (ICMP zpráva 134) a RS (ICMP zpráva 133) jsou součástí mechanismu protokolu NDP (objevování sousedů).

Podle IPv6 máme tyto možnosti získání IP adresy:

- dynamickou alokací pomocí DHCP (stavový režim DHCP),
- autokonfigurací stanice s využitím EUI-64 (SLAAC – Stateless Address Autoconfiguration),
- autokonfigurace s Privacy Extensions,
- statická konfigurace.

První možnost – *dynamická alokace pomocí DHCP* – je prakticky stejná jako u IPv4 (rozdíl je především v typu a formátu zasílaných zpráv). Jako jediná je *stavová*, tedy DHCP server si ukládá stavové (proměnlivé, dynamické) informace o přidělování adres, ostatní možnosti jsou bezstavové (nikam se takové záznamy neukládají).

Autokonfigurace s využitím EUI-64 (SLAAC) má ulehčit DHCP serverům a zejména síti od nadbytečného provozu. Hostitelskou část adresy si zařízení sestaví samo, jen musí získat ostatní informace (síťovou část adresy apod.). Jak to probíhá:

- Hostitelskou část adresy (zde přesně polovinu, 64 bitů) tvoří EUI-64, které se vyrobí z MAC adresy. MAC adresa má 48 bitů, takže ještě musíme 16 bitů přidat (2 oktety). Upravíme:
  - přesně do poloviny MAC adresy vnoříme dva oktety `FF-FE`,
  - nastavíme U/L bit na 1 (v prvním oktetu druhý bit zprava).
- Síťovou část adresy (zbývajících 64 bitů) musíme získat jinak, například z RA zprávy routeru.

Příklad

Ukážeme si vytvoření EUI-64 a jeho začlenění do IPv6 adresy. Předpokládejme tyto údaje:

- MAC adresa je `50-E5-49-C2-AC-27`,
- přes RA jsme zjistili, že prefix (síťová část adresy) je `2001:718:2601:265`.

Nejdřív vytvoříme EUI-64. MAC adresa se skládá z šesti oktetů, tedy mezi třetí a čtvrtý (tj. doprostřed) vnoříme dva oktety `FF-FE`:

```
50-E5-49-FF-FE-C2-AC-27
```

Pak v prvním oktetu nastavíme U/L bit – binárně je první oktet původně `01010000`, po změně `01010010`, což je hexadecimálně `52`. Takže výsledné EUI-64 je:

```
52-E5-49-FF-FE-C2-AC-27
```

Zkompletujeme celou adresu, tedy EUI-64 upravíme, aby „opticky“ odpovídalo části IPv6 adresy (skupiny, dvojtečky) a předsadíme prefix.

```
2001:718:2601:265:52E5:49FF:FEC2:AC27/64
```

Jenže to ještě není všechno. Velkou výhodou sice je, že stanice nemusí kvůli získání adresy komunikovat s DHCP serverem (jen musíme mít v síti takový router, který dokáže v RA odesílat síťovou část adresy), ale jak se stanice dozví například IP adresy DNS serverů? Teoreticky je víc možností:

- *Přidat info o adresách DNS serverů do RA.* Tato možnost je standardizována od roku 2010 v RFC 6106 a podporují ji skoro všechny systémy (u klientských zařízení prakticky všechny běžné UNIXové systémy včetně Linuxu, Android, systémy od Applu), kromě těch od Microsoftu (podpora musí být jak na straně routeru, tak na straně žadatele). Očekávalo se, že RFC 6106 bude podporováno ve Windows 10 a Windows Server 2016, ale není.
- *Speciální anycast adresy pro DNS servery.* Standardizace nebyla nikdy dokončena, protože se počítalo se site-local adresami, které byly ze standardů o IPv6 vypuštěny. Implementaci můžeme najít v některých systémech od Microsoftu.

- *Použití DHCPv6.* Adresu si sice stanice zkompletuje přes EUI-64 a pomocí RA, ale adresy DNS serverů si zjistí od DHCP serveru. Toto řešení vyžaduje, aby v síti byl alespoň jeden DHCP server, třebaže nebude fungovat stavově a provoz na síti nebude tak velký jako při dynamické alokaci.

Autokonfigurace s Privacy Extensions je třetí možností získání IPv6 adresy. Funguje to vpodstatě podobně jako autokonfigurace s EUI-64, jen hostitelskou část adresy vytváříme jinak.

- Hostitelská část adresy (taky přesně polovina, 64 bitů) je dynamicky generována z různých hardwarových a softwarových charakteristik systému a každých několik dnů se mění.
- Síťová část adresy je získávána například z RA zprávy routeru.

Pro získání adres DNS serverů platí totéž co u EUI-64. Zatímco EUI-64 se používá především v UNIXových systémech, Privacy Extensions jsou typické pro Windows.

Poznámka:

Účelem je co nejvíc ztížit identifikaci koncových zařízení v síti, ale ve skutečnosti tento postup ztěžuje práci i správcům. Správce sítě potřebuje mít přehled o tom, co se děje v síti, jaká zařízení má momentálně připojená a kam se obvykle připojují, IP adresa je důležitým identifikátorem v monitorování a evidenci. Pokud se tento identifikátor často mění, způsobuje to chaos v síti.

Statická konfigurace je poslední možností získání adresy, znamená situaci, kdy zařízení dostane adresu předem přidělenou. Stejně jako u IPv4, i zde ji lze provést ručně nebo zajistit pomocí statické alokace (zařízení se dotáže DHCP serveru, ale získává vždy tutéž IP adresu).

Staticky lze konfigurovat buď celou adresu nebo jen její hostitelskou část, přičemž síťovou část si zařízení doplní stejně jak bylo popsáno u předchozích možností.

Poznámka:

Statická alokace je *bezstavové využití DHCP*. Sice musí být uložena informace o tom, kterou IP adresu je třeba dotyčnému zařízení s určitou MAC adresou přidělit, ale nejedná se o dynamickou (proměnlivou) informaci.

6.4 Adresní rozsah IPv4

6.4.1 Třídy

Adresa protokolu IPv4 zabírá 32 bitů (4 oktety), což by teoreticky znamenalo $2^{32} = 4\,294\,967\,296$ možných adres, ale ve skutečnosti je to mnohem méně – kromě jiného z „organizačních“ důvodů. Adresní rozsah je hierarchicky členěn na sítě a podsítě, aby se zjednodušilo směrování.

Původně se právě z důvodu snadnějšího škálování této hierarchie rozlišovalo *pět tříd adres* podle pozice hranice mezi síťovou a hostitelskou částí adresy:

- *Třída A* používá první oktet pro adresu sítě, zbytek je hostitelská část, adresa s prvním oktetem nulovým není platná,
- *Třída B* používá pro adresu sítě první dva oktety, zbytek je hostitelská část,

- *Třída C* používá pro adresu sítě první tři oktety, zbytek je hostitelská část,
- *Třída D* slouží pro skupinovou adresaci,
- *Třída E* slouží pro experimentální účely.

Struktura adres těchto tříd je naznačena v tabulce 6.1.

Třída	Struktura adresy	První nibble adresy	První oktet		Max. zařízení v síti
A	síť.uzel.uzel.uzel	`0xxx...`	1–127	1–7F	$2^{24} - 2$
B	síť.síť.uzel.uzel	`10xx...`	128–191	80–BF	$2^{16} - 2$
C	síť.síť.síť.uzel	`110x...`	192–223	C0–DF	$2^8 - 2 = 254$
D	skupinové	`1110...`	224–239	E0–EF	–
E	experimentální	`1111...`	240–255	F0–FF	–

Tabulka 6.1: Třídy IPv4 adres

Poznámka:

Takže když vidíme adresu, jejíž první oktet je menší než 128, pak by nám mělo být jasné, že jde o adresu třídy A. Jestliže je první oktet menší než 192 (a větší nebo roven 128), jde o adresu třídy B a u prvního oktetu menšího než 224 adresu třídy C. Číslo 224 bychom si měli pamatovat obzvlášť dobře, protože tímto oktetem začíná většina skupinových adres.

Vraťme se ke speciálním adresám pro IPv4:

- Loopback (tj. `127.0.0.x`) je zjevně adresou třídy A.
- V každé třídě máme část rozsahu vyhrazenu pro soukromé adresy:
  - pro třídu A to jsou adresy `10.x.x.x`,
  - pro třídu B to jsou adresy `172.16.x.x` až `172.31.x.x`,
  - pro třídu C to jsou adresy `192.168.0.x` až `192.168.255.x`.
- Broadcast pro síť s adresami dané třídy (A, B nebo C) má síťovou část nastavenou běžným způsobem, v hostitelské části (tj. poslední tři, dva nebo jeden oktet) jsou všechny bity nastaveny na 1.

  Takže pokud máme adresu sítě například `10.0.0.0`, bude broadcastovou adresou v dané síti adresa `10.255.255.255`.

Poznámka:

Všimněte si, že pokud bychom zůstali u tříd a třídního směrování, pak vlastně nepotřebujeme masku sítě ani délku prefixu. To, která část adresy je síťová a která hostitelská, nám zcela jednoznačně určuje prvních pár bitů prvního oktetu. Jenže my jsme u tříd nezůstali, a tedy budeme tyto „dodatečné údaje“ potřebovat.

6.4.2 Podsíťování

Jak jsme výše naznačili, třídy postupně přestaly stačit. Správci větších sítí prostě potřebovali vytvářet hierarchii, a to jen se třídami nešlo. Proto se začalo používat *podsíťování* (subnetting). Jednu síť rozdělili na několik dílů – podsítí (subnetů), přičemž příslušnost k určité podsíti se dá poznat podle konkrétních bitů adresy.

Podsíťování není nic jiného než administrativní zásah do hostitelské části adresy (s adresou sítě a jejím rozsahem se u subnettingu nic neprovádí). Takže bity v hostitelské části nejsou všechny vyhrazeny pro adresování stanice, ale část (zleva, hned za síťovou částí) vyhradíme pro adresu podsítě.

Takže adresa se při podsíťování skládá ze tří částí:

- síťová část adresy,
- podsíťová část adresy,
- hostitelská část adresy.

Součet bitů všech tří částí samozřejmě musí dávat 32. Zatímco počet bitů síťové části je jednoznačný (zatím jsme pořád u tříd), počet bitů podsíťové části už jednoznačný není, a tedy je nutné k adrese přidat masku podsítě (nebo délku prefixu).

Příklad

Pokud máme adresu sítě třídy C ve tvaru `192.168.48.0` (tj. tři oktety jsou adresou sítě) a chceme tuto síť rozdělit na podsítě, potřebujeme vědět především:

- kolik má být podsítí,
- kolik maximálně stanic v každé podsíti chceme mít.

Oba údaje navzájem souvisejí – když chceme víc podsítí, budeme muset vyhradit víc bitů pro podsíťovou část adresy a zůstane nám méně bitů v hostitelské části, a tedy méně stanic v každé podsíti.

Takže předpokládejme, že podsítí nechceme nijak moc – vystačíme si se dvěma bity (tj. maximálně $2^2 = 4$ podsítě).

Zatím jsme vypotřebovali $3*8+2 = 26$ bitů pro síťovou a podsíťovou část adresy, tedy pro hostitele nám zbývá $32-26 = 6$ bitů. Protože $2^6-2 = 62$, v každé podsíti může být 62 zařízení (pozor, jakýchkoliv zařízení včetně routeru). Rozvržení bitů v adrese – část pro síť, podsíť a hostitele:

```
ssssssss.ssssssss.ssssssss.ppHHHHHH
```

Jak tedy budou vypadat adresy jednotlivých podsítí: část adresy pro podsíť bude u různých podsítí nabývat hodnot binárně `00`, `01`, `10` a `11`. Pro každou podsíť stanovíme adresu sítě (bity v hostitelské části = 0) a broadcast adresu (bity v hostitelské části = 1), adresy mezi nimi pak budou patřit zařízením. V reálu:

- Podsíť 1: první dva bity posledního oktetu budou 00, tedy:
  - adresa podsítě je `192.168.48.0/26` (poslední oktet binárně `00000000`),
  - broadcast adresa pro tuto podsíť je `192.168.48.63/26` (poslední oktet `00111111`),
  - hostitelé mají adresy z rozsahu `192.168.48.1/26` až `192.168.48.62/26`.
- Podsíť 2: první dva bity posledního oktetu budou 01, tedy:
  - adresa podsítě je `192.168.48.64/26` (poslední oktet binárně `01000000`),

  - broadcast adresa pro tuto podsíť je `192.168.48.127/26` (poslední oktet `01111111`),
  - hostitelé mají adresy z rozsahu `192.168.48.65/26` až `192.168.48.126/26`.
- Podsíť 3: první dva bity posledního oktetu budou 10, tedy:
  - adresa podsítě je `192.168.48.128/26` (poslední oktet binárně `10000000`),
  - broadcast adresa pro tuto podsíť je `192.168.48.191/26` (poslední oktet `10111111`),
  - hostitelé mají adresy z rozsahu `192.168.48.129/26` až `192.168.48.190/26`.
- Podsíť 4: první dva bity posledního oktetu budou 11, tedy:
  - adresa podsítě je `192.168.48.192/26` (poslední oktet binárně `11000000`),
  - broadcast adresa pro tuto podsíť je `192.168.48.255/26` (poslední oktet `11111111`),
  - hostitelé mají adresy z rozsahu `192.168.48.193/26` až `192.168.48.254/26`.

Tento postup je takový „upovídaný“. V reálu je dobré vytvořit si tabulku s těmito sloupci:

1. označení podsítě, příp. VLAN nebo WAN,
2. adresa podsítě (tj. všechny bity v druhé části adresy jsou 0),
3. broadcastová adresa v podsíti (tj. všechny bity v druhé části adresy jsou 1),
4. rozsah adres pro klienty (tj. mezi předchozími dvěma hodnotami).

Podle příkladu to bude:

Název	**Adresa podsítě**	**Broadcast podsítě**	**Rozsah pro klienty**
LAN1	`192.168.48.0/26` poslední oktet: `.00000000`	`192.168.48.63/26` poslední oktet: `.00111111`	od `192.168.48.1/26` do `192.168.48.62/26`
LAN2	`192.168.48.64/26` poslední oktet: `.01000000`	`192.168.48.127/26` poslední oktet: `.01111111`	od `192.168.48.65/26` do `192.168.48.126/26`
LAN3	`192.168.48.128/26` poslední oktet: `.10000000`	`192.168.48.191/26` poslední oktet: `.10111111`	od `192.168.48.129/26` do `192.168.48.190/26`
LAN4	`192.168.48.192/26` poslední oktet: `.11000000`	`192.168.48.255/26` poslední oktet: `.11111111`	od `192.168.48.193/26` do `192.168.48.254/26`

Názvy podsítí jsou tu jen tak „nadhozeny“, v reálu ve firmách máme metodiku, jak podsítě nazývat, případně při používání VLAN sem dáme názvy VLAN. Délka prefixu je u všech adres stejná, vpodstatě bychom ji nemuseli všude psát. Binární forma posledního oktetu, která je připsaná v druhém a třetím sloupci, tam taky nemusí být, pokud to umíme zpaměti.

Jak vidíte, sloupce pro adresu podsítě a broadcast v podsíti není těžké vyplnit, když zvládáme binární soustavu, poslední sloupec obsahuje to, co patří mezi adresy v předchozích dvou sloupcích.

Všimněte si, že když k broadcastu podsítě přičtete jedničku, dostanete adresu podsítě pro další řádek. Takže vlastně stačí nejdřív vytvořit sloupec s adresami podsítí, v dalším sloupci vždy dát adresu o 1 menší než je podsíťová z následujícího řádku (až na poslední řádek samozřejmě), a pak doplníme rozsahy pro klienty.

Proč to všechno?

- Podsítě nám zjednodušují směrování v rámci velké sítě (ve směrovacích tabulkách stačí mít pro celou jednu podsíť jediný řádek).
- Optimalizujeme tím síťový provoz, což je důsledkem zjednodušení směrování.

- Broadcastové domény můžeme omezovat i na podsíť, takže broadcasty nám nebloudí v celé síti, což opět znamená snížení provozu v síti.

Poznámka:

Již jsme se seznámili s VLAN (virtuálními lokálními sítěmi), což je obvykle chápáno jako mechanismus vrstvy L2. Ve skutečnosti se členění sítě na VLANy distribuuje i na vrstvu L3, a to právě pomocí podsítí – zařízení patřící do stejné VLANy budou v téže podsíti, zařízení z různých VLAN budou v různých podsítích.

6.4.3 VLSM

Masky podsítí s proměnnou délkou – VLSM (Variable Length Subnet Mask) – jsou mechanismem rozšiřujícím možnost podsíťování. Zatímco u podsíťování jsme si pevně stanovili, kolik bitů bude pro adresu podsítě, u VLSM se tento počet bitů může u různých podsítí měnit. Ještě pořád zůstáváme u třídního směrování, ale zvyšujeme pružnost návrhu.

K čemu je to dobré? Vpodstatě navyšujeme výhody, které nám přinesla možnost podsíťování. Při podsíťování máme pro každou podsíť tentýž maximální počet hostitelů, při použití VLSM můžeme mít pro různé podsítě různý počet hostitelů, podle potřeby. Ale pozor, jednoznačnost je důležitá i zde, takže jednotlivé podsítě (jejich adresní rozsahy) se nám v žádném případě nesmí prolínat.

Poznámka:

Můžeme si to představit tak, že si vytvoříme vícestupňovou hierarchii podsítí, přičemž v některých „větvích“ zastavíme členění dřív (méně bitů pro podsíť, více bitů pro hostitele) a v jiných „větvích“ zastavíme členění později (více bitů pro podsíť a tedy více podsítí, méně bitů pro hostitele).

Příklad

Opět máme adresu sítě třídy C ve tvaru `192.168.48.0` a chceme tuto síť rozdělit na podsítě. Víme, že budeme potřebovat jednu velkou podsíť a dvě menší. Pro první podsíť si zvolíme délku prefixu 25 (tedy pro podsíť vyhradíme jeden bit), pro druhou a třetí podsíť pak 26 (dva bity pro podsíť).

- Podsíť 1: první bit posledního oktetu bude 0, tedy:
  - adresa podsítě je `192.168.48.0/25` (poslední oktet binárně `0`0000000),
  - broadcast adresa pro tuto podsíť je `192.168.48.127/25` (poslední oktet `0`1111111),
  - hostitelé mají adresy z rozsahu `192.168.48.1/25` až `192.168.48.126/25`.
- Podsíť 2: první dva bity posledního oktetu budou 10, tedy:
  - adresa podsítě je `192.168.48.128/26` (poslední oktet binárně `10`000000),
  - broadcast adresa pro tuto podsíť je `192.168.48.191/26` (poslední oktet `10`111111),
  - hostitelé mají adresy z rozsahu `192.168.48.129/26` až `192.168.48.190/26`.
- Podsíť 3: první dva bity posledního oktetu budou 11, tedy:
  - adresa podsítě je `192.168.48.192/26` (poslední oktet binárně `11`000000),
  - broadcast adresa pro tuto podsíť je `192.168.48.255/26` (poslední oktet `11`111111),

– hostitelé mají adresy z rozsahu `192.168.48.193/26` až `192.168.48.254/26`.

Srovnejte s předchozím příkladem. Všimněte si, že v tomto příkladu nám první podsíť vlastně „spolkla“ první a druhou podsíť z předchozího příkladu (tj. obě původní podsítě začínající bitem `0` se staly jedinou podsítí). Jaký je důsledek? Zatímco v „menších“ podsítích máme k dispozici 62 adres pro zařízení ($2^5 - 2$), ve „velké“ podsíti je 126 adres pro zařízení ($2^6 - 2$). Kdybychom používali adresy třídy A, měli bychom ještě větší možnosti rozlišení.

V praxi bude častější jiný typ zadání, například:

Příklad

Adresa sítě je `172.16.0.0/20`. Rozdělte tuto síť na podsítě, přičemž adresy budou potřeba pro:

- několik poboček, přičemž se používají VLAN, v jednotlivých VLAN skupinách počítáme s těmito maximálními počty klientů:
  VLAN10: 400 klientů, VLAN20: 45, VLAN30: 72, VLAN40: 220, VLAN50: 10 klientů,
- tři WAN spoje typu point-to-point (tj. v každém dva klienti) propojující lokální sítě.

Každá podsíť má jiný maximální počet klientů, takže VLSM je jedinou vhodnou volbou (nehledě na WAN spoje). Pro WAN spoje typu point-to-point se nechávají podsítě s délkou prefixu 30 (přesně pro dva klienty). Postupujeme *vždy od největší podsítě k nejmenší.*

Pro danou podsíť volíme délku prefixu tak, aby se tam klienti „vešli“, zaokouhlujeme na číslo $2^n - 2$ směrem nahoru, délka prefixu pak bude $32 - n$. Pro pobočky:

- 400 klientů $\Rightarrow$ $254 < 400 \leq 510$, tj. $2^8 - 2 < 400 \leq 2^9 - 2$, délka prefixu bude $32 - 9 = 23$,
- 220 klientů $\Rightarrow$ $126 < 220 \leq 254$, tj. $2^7 - 2 < 220 \leq 2^8 - 2$, délka prefixu bude $32 - 8 = 24$,
- 72 klientů $\Rightarrow$ $62 < 72 \leq 126$, tj. $2^6 - 2 < 45 \leq 2^7 - 2$, délka prefixu bude $32 - 7 = 25$,
- 45 klientů $\Rightarrow$ $30 < 45 \leq 62$, tj. $2^5 - 2 < 45 \leq 2^6 - 2$, délka prefixu bude $32 - 6 = 26$,
- 10 klientů $\Rightarrow$ $8 < 10 \leq 14$, tj. $2^3 - 2 < 10 \leq 2^4 - 2$, délka prefixu bude $32 - 4 = 28$,
- pro WAN 2 klienti $\Rightarrow$ $1 < 2 \leq 2$, tj. $2^1 - 2 < 2 \leq 2^2 - 2$, délka prefixu bude $32 - 2 = 30$.

Sestavíme tabulku (je na další stránce). Aby se vešla na šířku stránky, nebudeme v adresách vypisovat první dva oktety, které jsou vždy `172.16`, u masky podsítě `255.255`. Modře je u binárního zápisu vyznačena podsíťová část adresy. Postup:

- Nejdřív vyplníme celé první tři sloupce, a to podle množství klientů v podsíti: v druhém sloupci máme vypsán očekávaný počet klientů a přepočet podle seznamu výše. V třetím slouci taktéž doplníme hodnotu podle seznamu, souvisí s maximálním počtem klientů v podsíti.
- Poslední sloupec vpodstatě můžeme také doplnit hned, nebo ho nechat na konec. Je to údaj ekvivalentní údaji o délce prefixu.
- Zbytek tabulky už budeme doplňovat po řádcích.
- Adresa první podsítě bude mít všechny bity za hranicí sítě (za 20 bity ze zadání) nulové, v našem případě `172.16.0.0`.
- Adresa prvního klienta v podsíti bude o 1 vyšší.
- Adresa posledního klienta v podsíti bude vyšší o maximální počet klientů v podsíti (tj. v prvním řádku přičteme 510, resp. dojde k „přetečení“ do levějšího oktetu, jako když v desítkové soustavě přecházíme do vyššího řádu, takže rozdělíme $510 = 256 + 254$ a poslední oktet bude 254, a předposlední 1).

- Jak vypočíst broadcast podsítě – jsou dvě možnosti:
  - druhou polovinu adresy podsítě si přepíšeme binárně, vyznačíme si hranici prefixu podsítě (23 bitů, takže od konce $32 - 23 = 9$ bitů), všechny bity za hranicí nastavíme na 1,
  - k adrese posledního klienta v podsíti přičteme jedničku.

  Pro kontrolu můžeme vypočíst obojí. Když vyjdou dvě různé hodnoty, víme, že je někde chyba.
- Adresa podsítě pro další řádek je o 1 vyšší než broadcast podsítě z předchozího řádku (při přičtení jedničky může dojít k „přetečení“ do vyššího oktetu).
- atd. až k poslednímu řádku.

	K, K_{max}		**S**	**S+K_{max}+1**	**S+1**	**S+K_{max}**	
Název	**Klientů /max**	**Délka prefixu**	**Adresa podsítě**	**Broadcast podsítě**	**První klient**	**Poslední klient**	**Maska podsítě**
VLAN10	400 510	23	.0.0 ···0000.0000 0000	.1.255 ···0001.1111 1111	.0.1	.1.254	.254.0 ···1110.0000 0000
VLAN40	220 254	24	.2.0 ···0010.0000 0000	+1 .2.255 ···0010.1111 1111	.2.1	.2.254	.255.0 ···1111.0000 0000
VLAN30	72 126	25	.3.0 ···0011.0000 0000	.3.127 ···0011.0111 1111	.3.1	.3.126	.255.128 ···1111.1000 0000
VLAN20	45 62	26	.3.128 ···0011.1000 0000	.3.191 ···0011.1011 1111	.3.129	.3.190	.255.192 ···1111.1100 0000
VLAN50	10 14	28	.3.192 ···0011.11000000	.3.207 ···0011.11001111	.3.193	.3.206	.255.240 ···1111.11110000
WAN1	2 2	30	.3.208 ···0011.1101 0000	.3.211 ···0011.1101 0011	.3.209	.3.210	.255.252 ···1111.1111 1100
WAN2	2 2	30	.3.212 ···0011.1101 0000	.3.215 ···0011.1101 0011	.3.213	.3.214	.255.252 ···1111.1111 1100
WAN3	2 2	30	.3.216 ···0011.1101 1000	.3.219 ···0011.1101 1011	.3.217	.3.218	.255.252 ···1111.1111 1100

Pokud si nepamatujete mocniny čísla 2, je dobré je mít někde napsané:

n	10	9	8	7	6	5	4	3	2	1	0
$\mathbf{2^n}$	1024	512	256	128	64	32	16	8	4	2	1
$\mathbf{2^n - 2}$	1022	510	254	126	62	30	14	6	2	0	-

6.4.4 CIDR

Beztřídní směrování – CIDR (Classless Inter-Domain Routing) již třídy plně odbourává v tom smyslu, že ruší závislost mezi tvarem prvního oktetu a nejlevější hranicí pro síťovou část adresy (což právě stanovují třídy). Hlavním motivem bylo zpřehlednění a zjednodušení směrování ve vyšších úrovních hierarchie, tedy u poskytovatelů síťového připojení na úrovni RIR a LIR. Dále budeme používat terminologii, se kterou jsme se seznámili v sekci 6.3 od strany 147.

Jak to funguje s CIDR: IANA přiděluje základní rozsahy jednotlivým RIR, tedy určitý počet bitů v adrese zleva. Tím je dána určitá základní délka prefixu. Každý RIR svým zákazníkům (tedy jednotlivým LIR) přidělí rozsah adres takový, že LIR „zdědí“ to, co má přiděleno RIR (tedy tuto část

mají všichni zákazníci společnou), a k tomu jsou přidány další bity specifické pro dotyčného LIR – provede se podsíťování, každý LIR bude mít (vzhledem k RIR) jednu podsíť. Takže LIR má adresu s delším prefixem. LIR má také své zákazníky, kterým provede totéž – každý zákazník „zdědí“ prefix od LIR plus část specifickou pro daného zákazníka.

Z toho vyplývá, že CIDR vlastně buduje hierarchii sítí a podsítí navzájem vnořených do takové míry, jak je zapotřebí. Hlavním účelem CIDR je právě sladit hierarchii IP adres s hierarchií (fyzických) sítí a především routerů.

Poznámka:

Jak se CIDR projevuje na routerech: předpokládejme, že (pro zjednodušení) máme kromě jiných tyto podsítě, které se případně ještě dělí na další podsítě:

Adresa podsítě	binárně druhý oktet
172.16.0.0/20	···.0001 0000.···.···
172.20.0.0/20	···.0001 0100.···.···
172.24.0.0/20	···.0010 0100.···.···
172.28.0.0/20	···.0010 1100.···.···

Pokud máme na některém routeru například první dvě podsítě na stejném portu (cesty k nim se dělí až později), pak můžeme mít ve směrovací tabulce pro obě podsítě jediný záznam, přičemž u nadsíťové adresy v tomto záznamu budeme brát v úvahu pouze ty bity (ve směru zleva), ve kterých jsou shodné. Jak vidíme, prvních 13 bitů (8 + 5) je u prvních dvou podsítí stejných, přičemž právě ve třináctém bitu se liší od následujících dvou sítí, takže bychom mohli souhrnně první dvě podsítě reprezentovat adresou `172.16.0.0/13`. Adresa vypadá skoro stejně jako adresa první podsítě, rozdíl je jen v délce prefixu – díky tomu sem spadá i rozsah pro druhou podsíť.

Podobně bychom mohli adresou `172.24.0.0/13` reprezentovat souhrnně třetí a čtvrtou síť, kdyby na daném routeru byly za stejným portem.

Sumarizace (také agregace) cest, resp. *nadsíťování* (supernetting) je právě to, co je popsáno v poznámce. Pokud do více podsítí vede cesta přes tentýž port routeru, není důvod mít pro každou zvlášť ve směrovací tabulce samostatný řádek. Všechny budou shrnuty do jednoho řádku, přičemž se zkrátí prefix (posune se hranice síťové části adresy směrem doleva) tak, aby v sobě zahrnoval všechny tyto podsítě. Klidně i před původní hranici tříd A/B/C. Tomuto souhrnnému řádku reprezentujícímu více (pod)sítí se říká *CIDR blok*. Ovšem podmínkou je, aby doopravdy fyzické propojení sítí odpovídalo hierarchii IP adres.

Poznámka:

Rozdíl mezi podsíťováním (včetně VLSM) a nadsíťováním (CIDR) je kromě jiného v tom, že u podsíťování posouváme hranici mezi (pod)síťovou a hostitelskou částí směrem doprava, kdežto u nadsíťování ji posouváme doleva. Podsíťování se týká každého prvku hierarchie (firma dostane od ISP adresní rozsah a ten si může libovolně rozčlenit), nadsíťování se týká spíše ISP provozujících hodně vytížené směrovače (potřebují redukovat směrovací tabulky právě pomocí CIDR).

Právě systém CIDR přišel se zjednodušeným označováním hranice mezi síťovou a hostitelskou částí pomocí lomítka a délky prefixu, tedy v třídním směrování bychom teoreticky měli používat zápis pomocí masky. Zápisu s lomítkem a délkou prefixu se říká *CIDR notace*.

Systém adresních rozsahů IPv6 je již přirozeně typu CIDR. Právě proto jsme s popisem hierarchie pro IANA, RIR, LIR apod. začali právě u popisu principu IPv6 adresování.

Poznámka:

Také v IPv6 existují výše popsané možnosti včetně subnettingu a VLSM, jen jsou o něco náročnější vzhledem k počtu bitů v adresách, se kterými je třeba pracovat.

6.5 Protokol ICMP a zprávy

6.5.1 Účel protokolu ICMP

Protokol ICMP (Internet Control Message Protocol, tedy protokol pro posílání řídících zpráv) je také protokolem vrstvy L3. Jeho účelem je zajišťovat posílání krátkých zpráv, obvykle reprezentovaných jedním nebo dvěma čísly (každé takové číslo má určitý význam), výjimečně ještě s dodatečnou datovou informací. Typicky se takto posílají informace o chybách, problémech na cestě nebo jiná upozornění.

Každá zpráva posílaná pomocí ICMP má své *číslo*. V následující tabulce jsou některé běžné hodnoty pro ICMPv4 uvedeny:

Číslo	Význam
8	*Echo Request* – žádost o odezvu (posíláme, když příkazem `ping` zjišťujeme, zda je konkrétní zařízení dostupné)
0	*Echo Reply* – odpověď na *Echo Request* (8) ve smyslu „ano, jsem tady“
3	*Destination Unreachable* – zpráva o nedostupnosti cíle (pokud router obdrží IP paket, který nedokáže doručit, pošle ICMP paket se zprávou 3 zdrojovému zařízení)
11	*Time Exceeded* – vypršel čas čekání; buď hodnota TTL v IP paketu klesla na 0, vypršel časový limit při čekání na zbývající fragmenty IP paketu apod. (posílá router zdroji IP paketu)
12	*Parameter problem* – v IP paketu (většinou v jeho záhlaví) je nějaký problém, který nelze oznámit jinou ICMP zprávou
17	*Address Mask Request* – žádost o informaci o masce (pod)sítě
18	*Address Mask Reply* – odpověď na *Address Mask Request* (17)

Tabulka 6.2: Některé zprávy protokolu ICMPv4

Samotná zpráva je v některých případech příliš obecná, tedy ji může upřesnit kód zprávy. Například pro zprávu typu 3 (*Destination Unreachable*) existuje několik kódů, které upřesňují, proč je cílové zařízení nedostupné:

- kód 0 – cílová síť je nedostupná (router tuto síť nezná a nedokáže ji směrovat),
- kód 1 – cílová stanice je nedostupná (v zadané síti nelze najít dotyčnou stanici),
- kód 2 – cílový protokol je nedostupný (v poli *Protokol* je neznámá hodnota),

- kód 3 – cílový port je nedostupný (tím je myšleno číslo portu uvedené v zapouzdřeném segmentu transportní vrstvy),
- kód 4 – paket je větší než MTU na cestě, ale nelze fragmentovat, protože je nastaven příznak DF (Do not Fragment),
- atd. (pro zprávu 3 je celkem 15 různých kódů).

Pro zprávy typu 11 (*Time Exceeded*) se obvykle používají kódy 0 (hodnota pole TTL je 0) nebo 1 (překročen čas čekání na zbývající fragmenty).

Takže zatím máme dvojici číslo zprávy + kód. Zpráva ale může být upřesněna ještě jinak, například *jako data* může být přiložena část IP paketu, na který je takto reagováno (přiloží se IP záhlaví a prvních 8 oktetů datové části).

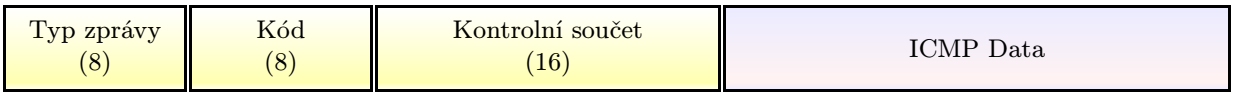
Poznámka:

Ve skutečnosti mají datovou část i ty ICMP pakety, které ji vlastně ani nepotřebují – přidávají tam výplň, protože na nižších vrstvách bývá stanovena určitá minimální délka pro zapouzdřená data (payload). S tím se setkáváme například u ICMP zprávy *Echo Request* (8).

ICMP paket má velmi jednodouchou strukturu – žádné adresy, délka paketu apod., záhlaví má jenom tři pole:

- *Typ zprávy* (ICMP Type, 8 bitů) – určení typu zprávy, například číslo 8 pro Echo Request,
- *Kód* (Code, 8 bitů) – upřesňující kód,
- *Kontrolní součet* (Checksum, 16 bitů) – kontrolní součet přes předchozí dvě pole záhlaví.

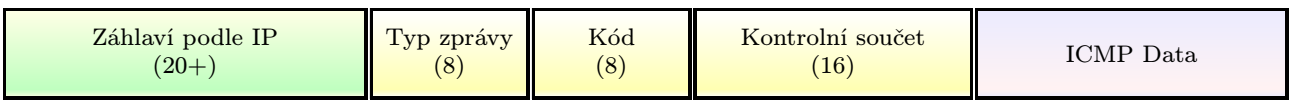
Formát celého ICMP paketu vidíme na obrázku 6.6.



Obrázek 6.6: Formát ICMP paketu

Dále v této kapitole se podíváme na konkrétní případy použití ICMP včetně toho, co je v těchto případech použito jako data.

ICMP paket se sice při odesílání předává na vrstvu L2, ale ještě předtím se zabalí do IP paketu – samotný ICMP paket nemá ve svém záhlaví ani adresy ani jiné důležité položky. Na obrázku 6.7 vidíme formát paketu do zapouzdření do IP.



Obrázek 6.7: Formát ICMP paketu zapouzdřeného v IPv4 paketu

Některé typy zpráv používají TTL = 1, ale většina využívá hodnotu TTL nastavenou v odesílajícím systému (tj. obvykle 128 nebo 64).

Po zapouzdření do IP je paket zpracován na L2 jako každý jiný paket, tj. například je zapouzdřen do paketu typu Ethernet II a poslán po ethernetové síti.

6.5.2 ICMP verze 6

Pokud používáme protokol IPv6, pak je třeba také používat ICMPv6. Oproti ICMPv4 byly některé zprávy přidány a taky se rozšířily oblasti využití tohoto protokolu (což souvisí) – v protokolu ICMPv6 se shrnuly role tří původních protokolů: ICMPv4, ARP a IGMP.

Jenže zdaleka nešlo jen o přidání typů zpráv pro nové způsoby využití protokolu, mnohé původní typy byly přečíslovány. Už u ICMPv4 platilo, že některé zprávy jsou chybové a jiné informační, ICMPv6 toto rozlišení přenesl i do číslování typů zpráv – pole pro typ zprávy je taky 8bitové, ale první (nejlevější, nejvíce významný) bit je nastaven na 0 u chybových zpráv a na 1 u informačních zpráv.

Číslo	Význam
128	*Echo Request* – žádost o odezvu, původně číslo 8
129	*Echo Reply* – odpověď na *Echo Request* (128), původně číslo 0
1	*Destination Unreachable* – zpráva o nedostupnosti cíle, původně číslo 3
3	*Time Exceeded* – vypršel čas čekání, původně číslo 11
4	*Parameter problem* – v IP paketu je nějaký problém, který nelze oznámit jinou ICMP zprávou, původně číslo 12

Tabulka 6.3: Některé zprávy protokolu ICMPv6, které byly i u ICMPv4

V tabulce 6.3 je několik typů zpráv, které najdeme v obou verzích. Všimněte si, že tyto zprávy mají v ICMPv6 jiná čísla než v ICMPv4, a že při novém číslování zachovávají pravidlo, kdy nejvíce významný bit z 8 bitů prvního pole je nastaven na 0 u chybových zpráv (tj. číslo typu je v rozmezí 0–127, poslední tři řádky tabulky) a na 1 u informačních zpráv (rozmezí 128–255, první dva řádky tabulky). Také v kódech pro některé konkrétní typy zpráv jsou změny (ale třeba kódy pro zprávu Time Exceeded jsou stejné, jen číslo typu se změnilo).

Nově přidané typy zpráv souvisejí obvykle buď s mapováním IP adres na MAC adresy (což u starší verze byla práce protokolu ARP) nebo se správou skupin (původně v protokolu IGMP), a taky pár typů zpráv pro běžné nebo mobilní sítě. Nejdůležitější z nich jsou v tabulce 6.4.

Zajímavá je například zpráva *Packet Too Big* (2), která v předchozí verzi nebyla, třebaže by se vpodstatě hodila i tam. Pokud router dostane k přeposlání paket větší než je hodnota MTU na následující cestě, chová se u různých verzí IP odlišně.

- Je to paket protokolu IPv4: buď se ho pokusí fragmentovat, nebo (když to nejde, příznak DF = 1) paket zahodí a jeho původci pošle ICMPv4 zprávu *Destination Unreachable* (3).
- Je to paket protokolu IPv6: v každém případě ho zahodí a jeho původci pošle ICMPv6 zprávu *Packet Too Big* (2).

Další informace:

Seznam všech typů ICMP zpráv a k nim příslušných kódů najdete na

- IPv4: http://www.iana.org/assignments/icmp-parameters/icmp-parameters.xhtml
- IPv6: http://www.iana.org/assignments/icmpv6-parameters/icmpv6-parameters.xhtml

Číslo	Význam
2	*Packet Too Big* – v případě, že router dostane k přeposlání příliš velký paket (větší než hodnota MTU na portu pro odeslání), paket zahodí a původnímu odesílateli pošle tuto ICMP zprávu
134	*Router Advertisement* (vizitka routeru) – routery v pravidelných intervalech posílají tuto ICMP zprávu s informacemi o sobě a síti, například adrese sítě a délce prefixu
133	*Router Solicitation* (žádost o vizitku routeru) – stanice si touto zprávou může na routeru vynutit vyslání zprávy Router Advertisement (134) mimo interval
135	*Neighbor Solicitation* (žádost o oznámení souseda) – máme IP adresu některého zařízení v sousedství, touto zprávou se ptáme svého okolí na jeho MAC adresu (tj. „Kdo má tuto IP adresu?“, soused, který pozná svou IP adresu, odpoví následující zprávou:
136	*Neighbor Advertisement* (oznámení souseda) – v reakci na předchozí zprávu (pokud poznám svou iP adresu) oznámím svou MAC adresu
130	*Multicast Listener Query* (dotaz na členy skupiny) – posílá router při ověřování, kdo z jeho sítě patří do konkrétní skupiny a tedy odebírá pakety adresované na danou multicast adresu
143	*Multicast Listener Report* (oznámení členství ve skupině) – posílá zařízení, když se přihlašuje do skupiny (informuje router, že chce dostávat příslušné pakety) nebo jako odpověď na předchozí zprávu

Tabulka 6.4: Některé zprávy protokolu ICMPv6, které nejsou v ICMPv4

6.5.3 Testování dosažitelnosti

Protokol ICMP se často používá pro posílání zpráv *ICMP Echo Request* a *ICMP Echo Reply*.

Pokud potřebujeme otestovat dostupnost některého zařízení v síti, obvykle k tomu účelu používáme příkaz `ping` odesílající právě ICMP zprávu *Echo Request* (číslo 8 nebo 128 podle verze). Příkaz `ping` posílá vždy několik paketů s touto zprávou, na každý očekává odpověď a následně podle toho, jak dlouho trvala odpověď na jednotlivé pakety, vypočte statistiky. Tento příkaz najdeme v jakémkoliv operačním systému; do Windows se dostal z UNIXu, což je vidět na jeho syntaxi.

Příkaz `ping` posílá ICMP zprávu *Echo Request* v tomto tvaru:

Typ = 8	Kód = 0	Kontrolní součet
Identifikátor		Pořadové číslo
volitelná data		

Tabulka 6.5: ICMPv4 paket zprávy *Echo Request*

Každý řádek má délku 32 bitů. Pole na prvním řádku jsou součástí ICMP záhlaví, od druhého řádku dále jde o data specifická pro tento typ zprávy. U ICMP *Echo Request* posíláme

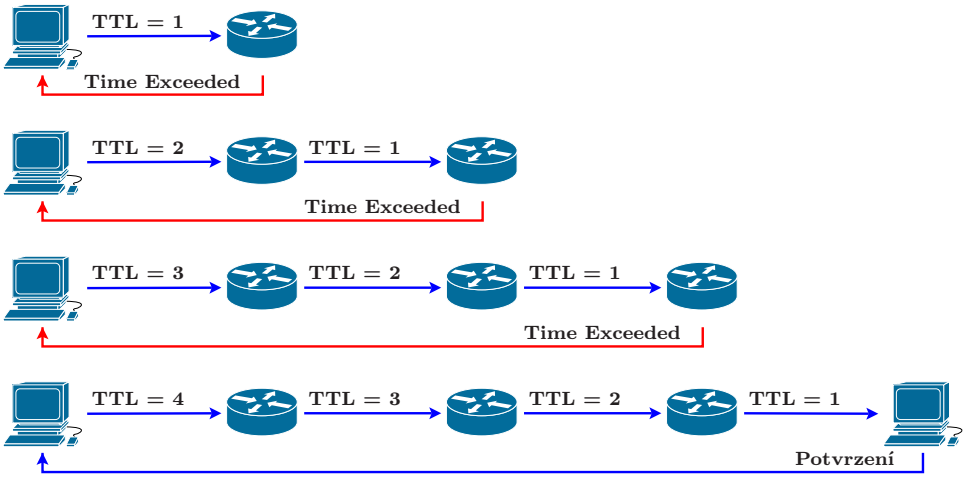
- *identifikátor* (16 bitů) – většinou je zde nějaké velmi malé číslo, obvykle 1 nebo 3,
- *pořadové číslo* (Sequence Number, 16 bitů) – pořadové číslo paketu posílaného příkazem `ping`, takže když pošleme čtyři pakety, budou v nich postupně čísla zvyšující se o 1, pokud příkaz spustíme znovu, pokračuje se v posloupnosti (nezačíná se číslovat znovu),
- *volitelná data* (ICMP Data) – datová výplň bez významu, obvykle o délce 32 nebo 48 oktetů.

Volitelná data si zařízení určuje samo. Ve Windows je to abeceda (písmena anglické abecedy v rozsahu `a...wa...i`, všimněte si, že podle Microsoftu je posledním písmenem abecedy písmeno `w`), v Linuxu sekvence speciálních znaků a číslic.

Testované zařízení odpoví ICMP zprávou *Echo Reply*, která víceméně kopíruje pole dotazu (změní obsah pole *Typ zprávy* a samozřejmě kontrolní součet). Tato zpráva je doručena zpět tázajícímu se zařízení, které podle hodnoty pole *pořadové číslo* v jednotlivých paketech odpovědí zkompletuje časy odeslání dotazu a přijetí odpovědi a vypočte příslušnou statistickou informaci.

Mechanismus `traceroute` (ve Windows `tracert`) také pracuje s ICMP pakety. Na rozdíl od předchozího příkazu zde ani tak nejde o zjištění dostupnosti dotyčného zařízení, ale spíše o zmapování cesty (trasování, proto ten název). Pokud je cíl dostupný, vypíše se seznam všech routerů na cestě včetně doby trvání přenosu od předchozího uzlu sítě, pokud je cíl nedostupný, vypíše se seznam routerů na té části cesty, která je „v pořádku“. Takže pomocí `traceroute` můžeme zjistit:

- kterou cestou (přes které routery) jdou pakety k určitému cíli,
- který úsek této cesty je nejpomalejší (v síti to může znamenat problém s propustností),
- od kterého routeru začíná být cesta problematická.



Obrázek 6.8: Obecně k mechanismu traceroute

Obecně tento mechanismus funguje takto (viz obrázek 6.8):

- z našeho zařízení se odesílají IP pakety s cílovou adresou „zájmového“ zařízení s TTL nastaveným na 1, 2, 3, atd.,
- na každém zařízení vrstvy L3 na cestě se hodnota TTL snižuje o 1,
- na tom zařízení vrstvy L3, kde po odečtení 1 je TTL = 0, už nedojde k přeposlání, náš paket je zahozen,
- zpět nám přijde ICMP zpráva Time Exceeded (v IPv4 číslo 11, v IPv6 číslo 3),
- až se konečně některý paket dostane k cíli, získáme potvrzující odpověď a přestaneme posílat testovací pakety.

Pro totéž TTL se obvykle posílá více než jeden paket, většinou tři pakety pro každé TTL.

Ve Windows se posílají testovací pakety ve formě ICMP zpráv *Echo Request* zabalených do IP paketu s daným zvyšujícím se TTL, potvrzení od cílové stanice je ICMP zpráva *Echo Reply*. Takže podle této zprávy poznáme, že jde o poslední uzel na cestě, tedy cíl.

V Linuxu a většině dalších UNIXových systémů si můžeme vybrat – ve výchozím nastavení to jsou UDP segmenty zabalené do IP paketu s daným zvyšujícím se TTL, přičemž jako číslo portu v UDP segmentu je použito „nereálné“ číslo. Proto je potvrzením od cílové stanice ICMP zpráva *Destination Unrecheable* s kódem *Port Unreachable* (nedostupný port). Podle tohoto paketu poznáme, že náš dotaz dorazil do cíle. V parametrech příkazu můžeme zvolit jiný druh testovacích paketů – buď ICMP pakety jako ve Windows nebo TCP segmenty.

Poznámka:

Mnohé servery a jiná veřejně dostupná síťová zařízení na příkazy `ping` a `traceroute` (`tracert`) nereagují. Je to proto, že buď samotné zařízení nebo firewall na cestě má v konfiguraci nastaveno nereagovat na zprávy ICMP Echo Request, případně na UDP segmenty s neexistujícím číslem portu. Důvodem je bezpečnost, protože tyto mechanismy bývají často zneužívány hackery k mapování „zájmové“ sítě.

Většinou je dobré nechat reakci na tyto pakety alespoň tehdy, když pocházejí z vnitřní (důvěryhodné) sítě, ovšem to je na administrátorovi.

Ovšem v Linuxu můžeme zvolit TCP segmenty používající známý port 80 (protokol HTTP), ten obvykle přes firewally projde.

Tyto testovací mechanismy fungují na vrstvě L3, takže pokud je problém s odezvou a dostupností na jiné vrstvě (ať už nadřízené nebo na kterémkoliv zařízení na cestě v podřízené vrstvě), nemáme šanci to zjistit.

ftp://ftp.hp.com/pub/hpcp/UDP-ICMP-Traceroutes.pdf

6.6 Správa skupin

Jak už víme, také v adresním prostoru IPv4 a IPv6 existují skupinové adresy, a tedy musí existovat protokol na jejich správu. Skupina v prostoru IP je především dána konkrétní skupinovou IP adresou.

Správa skupin obnáší:

- evidování portů s členy dané skupiny na aktivních síťových prvcích (například routerech),
- možnost ohlásit příslušnost ke konkrétní skupině (zajistit si členství ve skupině),
- ověřování trvání členství ve skupině.

Na aktivních síťových prvcích vrstvy L3 je třeba evidovat, za kterými porty jsou členové patřící do konkrétní skupiny (nikoliv samotné členy, to je zbytečné), protože na tyto prvky je třeba rozesílat pakety, které mají danou skupinovou adresu jako cílovou. Router (nebo obdobné zařízení vrstvy L3) se může pravidelně na portech dotazovat, zda se za nimi nachází nějaké zařízení patřící do dané skupiny, a pokud na daném portu žádné zařízení nereaguje, port ze skupiny odstraní (tj. nebude na něj zasílat pakety s danou skupinovou adresou jako cílovou).

Pokud sítí prochází multicast paket, na jednotlivých routerech, přes které jde, se podle cílové (multicastové) adresy ověří, na které porty má být tento paket replikován. Pole TTL se na každém routeru dekrementuje, zachází se s ním stejně jako v jakémkoliv jiném paketu.

Ať už jde o IPv4 nebo IPv6, vždy máme vyhrazen určitý rozsah pro skupinové adresy, a dále existují některé rezervované skupinové adresy – typicky adresa pro skupinu všech zařízení podporujících protokol IP dané verze, adresa pro skupinu všech routerů, atd. Multicasty se také využívají pro distribuci multimediálních dat (v komunikaci s více cíli – multipoint, např. IPTV nebo Video-on-Demand), zpráv, bezpečnostních záplat, atd.

V mechanismu IPv4 je za správu skupin odpovědný *protokol IGMP* (Internet Group Membership Protocol). Struktura IGMP paketu je vcelku podobná ICMP, taky se podobným způsobem používá, včetně zapouzdřování do IP paketu. Tak jako ICMP má zprávy (ICMP Echo Request apod.), IGMP má zprávy týkající se skupin, například *Membership Query* (17) a *Membership Report* (22).

Skupinové adresy mají vyhrazen adresní rozsah `224.0.0.0` až `239.255.255.255`, přičemž místní skupiny mají adresy `224.0.0.x` (v obalovém IP paketu je TTL při odesílání nastaveno na 1). S některými vyhrazenými skupinovými adresami jsme se seznámili v předchozím textu.

Další informace:

Seznam registrovaných multicast IPv4 adres je dostupný na
http://www.iana.org/assignments/multicast-addresses/multicast-addresses.xhtml

V mechanismu IPv6 roli správy skupin přebírá *protokol ICMP*, přičemž pro účely správy skupiny jsou do něj přidány nové typy zpráv (nahrazující to, co původně bylo v IGMP):

- *Multicast Listener Query* (130), kterou posílá router na downlink porty pro ověřování, zda na těchto portech existují zařízení patřící do konkrétní skupiny,
- *Multicast Listener Report* (143), kterou posílá zařízení, když chce informovat router (nebo routery na cestě), že chce odebírat pakety pro konkrétní skupinu (přihlášení do skupiny).

U IPv6 jsou pro skupiny rezervované adresy začínající prefixem `ff00::/8`.

Poznámka:

V IPv6 se multicastové adresy používají více než v IPv4, kromě jiného taky tam, kde se v IPv4 používal broadcast. Například v komunikaci s DHCP serverem (protokol DHCPv6).

Ve správě skupin se ve skutečnosti používají i další protokoly, například obdoby směrovacích protokolů pro distribuci obsahu multicastových tabulek mezi routery.

Další informace:

- http://www.samuraj-cz.com/clanek/tcpip-skupinove-vysilani-ip-multicast-a-cisco/
- https://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xhtml
- https://www.ipv6.cz/Skupinov%C3%A9_adresy_(multicast) http://www.lupa.cz/clanky/ipv6-myty-a-skutecnost-dil-vii-podpora-multicast-a-anycast-provozu/
- https://networklessons.com/ip-routing/icmp-internet-control-message-protocol/

6.7 Objevování sousedů

6.7.1 Tabulky sousedů

Představme si tuto situaci: známe svou IP adresu, masku (či prefix) a IP adresu brány. Potřebujeme poslat paket do cizí sítě, a tedy přes bránu. Jenže „kterým směrem“ je brána? Přes které síťové rozhraní je dostupná a jaká je její MAC adresa?

Vlastně podobný problém musíme řešit i tehdy, když chceme poslat IP paket někomu do vnitřní sítě. Třeba znám IP adresu dotyčného zařízení, ale jakou tedy má MAC adresu a přes který port je toto zařízení dostupné?

Co se týče „směru“, na ten máme jiné mechanismy, teď se zaměříme na mapování IP adresy na korespondující MAC adresu.

Mapování IP adres na MAC adresy provádí v případě IPv4 *protokol ARP* (Address Resolution Protocol) a v případě IPv6 *protokol NDP* (Neighbor Discovery Protocol). Jejich hlavní funkce:

- vést tabulku sousedů,
- pomocí ARP/NDP dotazů tuto tabulku doplňovat.

V obou případech si zařízení vede *tabulku sousedů* s informacemi o nejbližším síťovém okolí (sousedech). V tabulce sousedů máme pro každého „souseda“ tyto informace:

- IP adresa (IPv4 nebo IPv6),
- MAC adresa,
- typ (statická/trvalá nebo dynamická/zjištěná).

První dva údaje nám mapují IP adresu souseda na jeho MAC adresu, třetí údaj určuje, jak se dotyčný údaj dostal do tabulky (zda byl staticky vložen nebo zjištěn dynamicky dotazováním).

Pro každé síťové rozhraní s vlastní IP adresou se vede jedna tabulka (včetně virtuálních rozhraní, pokud máme nainstalován virtualizační software).

Poznámka:

Koncová zařízení v síti mají v této tabulce především jeden důležitý záznam – adresu brány, na kterou směřují většinu svého provozu.

V případě Windows vyšších verzí máme ARP tabulku i NDP tabulku poněkud přeplácané – najdeme tam kromě brány taky multicast adresy a v případě ARP taky broadcast adresy včetně mapování na MAC adresy. Obsah tabulek zobrazíme takto:

- `arp -a` pro ARP tabulku v IPv4,
- `netsh interface ipv6 show neighbors` pro NDP tabulku v IPv6
- v PowerShellu: `get-NetNeighbor` (nefunguje ve Windows 7)

(příkaz `netsh` se dá použít i pro zobrazení ARP tabulky v IPv4, pokud místo `ipv6` napíšete `ipv4`).

Tyto příkazy se také dají použít k ovlivňování tabulky sousedů (přidávání nových záznamů nebo jejich odstraňování), ve většině případů však tyto úlohy zvládají dotyčné protokoly dynamicky.

V Linuxu a jiných UNIXových systémech se pro zobrazení ARP/NDP tabulky používá příkaz

```
ip neigh show
```

(tedy můžeme napsat `ip neighbor show` nebo `ip n s` nebo něco mezi tím, nejlepší je volit kompromis – nepsat příliš a zároveň psát tolik, aby to bylo srozumitelné). Na výstupu máme obvykle pro každé síťové rozhraní právě jeden údaj – záznam pro bránu, nejsme zahlcováni dalšími informacemi. Také tento příkaz (`ip neigh` s některým podpříkazem) je možné použít pro ovlivňování obsahu tabulky.

Také v UNIXových systémech existuje příkaz `arp` (pro zobrazení tabulky stačí zadat bez parametrů), ten můžeme použít tehdy, když nám jde opravdu jen o IPv4 adresy.

Postup

Mechanismus objevování sousedů podle RFC 4861 si ukážeme na příkladu zjištění MAC adresy souseda (v případě, že známe IPv6 adresu) – je to obdoba ARP dotazu. Tento příklad je převzat z webu https://www.ipv6.cz/.

Postup:

- posílám datagram na skupinovou IPv6 adresu s prefixem `ff02:0:0:0:0:1:ff00:0/104`
- určení celé skupinové IPv6 adresy:
  - znám IPv6 adresu souseda, např. `2001:db8:1:1:22a:fff:fe32:5ed1`
  - z ní vezmu posledních 24 bitů (3 oktety) a dodám do skupinové IPv6 adresy z prvního bodu: `ff02::1:ff32:5ed1`
- na tuto adresu pošlu ICMPv6 zprávu *Neighbor Solicitation* – výzvu sousedovi
- soused odpoví zprávou *Neighbor Advertisement* obsahující jeho MAC adresu

6.7.2 K protokolům

Úkolem protokolů ARP a NDP není jen vést tabulku sousedů, ale také v případě potřeby se dynamicky poptávat na to, co by mělo v této tabulce být. Existují tedy pakety posílané dotyčným protokolem.

ARP pakety (tedy pro IPv4) se zapouzdřují přímo do rámců Ethernet II nebo rámců bezdrátové sítě Wi-fi. Rozlišujeme ARP dotaz a ARP odpověď.

Pokud zařízení potřebuje zjistit MAC adresu k určité IP adrese, vyšle ARP paket s dotazem „Who has xxx? Tell yyy.“ (Kdo má MAC adresu xxx? Sdělte to na MAC adresu yyy). Protože MAC adresáta neznáme (teprve ji zjišťujeme), dotaz se odesílá jako univerzální broadcast (tj. na adresu `FF-FF-FF-FF-FF-FF`). Reaguje pouze to zařízení, které pozná svou MAC adresu, zpět odešle odpověď, tentokrát unicast (protože z dotazu zná MAC adresu tazatele).

Každý ARP paket je stejně dlouhý, ať už jde v kterémkoliv směru. V dotazu i odpovědi jsou všechna pole (vlastní IP a MAC adresa, dotazovaná IP a MAC adresa), ale v dotazu je to, co odesílatel neví, vyplněno nulami.

NDP pakety (mechanismus pro IPv6) nemají vlastní specifikaci, posílají se jako ICMPv6 pakety se speciálními typy zpráv a zapouzdřují se do IPv6 paketů (jako každá ICMP zpráva). NDP je komplexnější protokol než ARP, plní více úloh. Z ICMPv6 zpráv se sousedy souvisejí především

- *Neighbor Solicitation* (ICMP zpráva číslo 135) – žádost o oznámení souseda, významem odpovídá ARP dotazu,
- *Neighbor Advertisement* (ICMP zpráva číslo 136) – odpověď na předchozí žádost.

Na rozdíl od ARP se formát a velikost dotazu a odpovědi liší, v dotazu najdeme opravdu jen žádanou adresu. V dotazu je jako adresa cíle použita multicast adresa odvozená od skupinové adresy pro směrovače. Odpověď je poslána buď jako unicast (pokud byla v dotazu IP adresa tazatele) nebo jako multicast na všechny uzly v síti zvládající protokol IPv6 (tj. na `ff02::1`).

Mechanismus NDP slouží i k jiným účelům – komunikaci s routery při zjišťování konfigurace sítě (adresa sítě, délka prefixu apod.), zjišťování duplicitních adres a dalším. Pro každý účel existují zvlášť čísla ICMPv6 zpráv.

Poznámka:

Otázkou je, na které vrstvě RM ISO/OSI, resp. síťového modelu TCP/IP, protokoly ARP a NDP vlastně pracují. V případě ARP není v standardu o nějaké vrstvě ani slovo. ARP pakety se zapouzdřují do rámců vrstvy L2, takže spíše patří na vrstvu L2 nebo na rozhraní mezi vrstvami L2 a L3, ale rozhodně ne na vrstvu L3.

Protokol NDP využívá ICMP zprávy, takže jednoznačně patří na vrstvu L3.

6.7.3 Bezpečnost

Objevování sousedů má jedno závažné úskalí: pokud obdržíme NS paket (paket s informací o mapování MAC a IP adres souseda), předpokládá se, že údajům v tomto paketu budeme věřit. Jenže tento paket může být podvržený a důsledkem používání této IP adresy je odesílání dat na jiný počítač než předpokládáme, což zvláště ve firemním prostředí může způsobit odcizení dat.

Pro tento problém existuje několik řešení, z nichž je zajímavý například *protokol SEND* (Secure Neighbour Discovery – RFC 3971), který umožňuje digitálně podepisovat oznámení protokolu NDP. Používá se pro zajištění „objevovací“ a keep-alive komunikace se sousedy.

Možnosti zajištění bezpečnosti v SEND:

- *kryptograficky generované adresy* (CGA) – RFC 3972: koncept soukromého a veřejného klíče, ICMPv6 zprávy jsou digitálně podepsány,
- *certifikační cesty*: ochrana proti falešným směrovačům – řetězce navazujících certifikátů dokazující, že určitá *důvěryhodná* certifikační autorita schválila toto zařízení jako směrovač poskytující dané informace (certifikační autority tvoří hierarchii).

Další informace:

O objevování sousedů najdeme informace například na

- http://www.abclinuxu.cz/clanky/architektura-ipv6-konfigurace-adres-a-objevovani-sousedu
- http://www-public.it-sudparis.eu/~lauren_m/articles/M-CGA-Tony-Cheneau-SAR-SSI-2011.pdf

6.8 Jak funguje router

Routery implementují funkcionalitu vrstvy L3, a to softwarově, což znamená, že potřebují velký výpočetní výkon – zpracování IP záhlaví (plus případné související činnosti) je totiž výpočetně mnohem

náročnější než zpracování záhlaví ethernetových rámce. V IP záhlaví (kterékoliv verze) totiž máme mnohem více polí, tato pole jsou poměrně dlouhá a v některých je třeba jít na úroveň bitů (což znamená použití bitových operací). Routery také obvykle plní různé další funkce, včetně směrování, filtrování provozu, překladu adres, atd.

Proto průměrný router potřebuje dostatečně výkonný procesor (úměrně provozu, který má reálně zvládat) a dostatek paměti (pakety jsou totiž nejdřív uloženy, pak zpracovány a pak teprve odeslány). Typicky mívají routery mnohem méně portů než switche – každý port navíc znamená navýšení množství paketů ke zpracování a tedy navýšení potřeby výkonu.

Protokol IP poskytuje službu typu best-effort, tj. negarantuje doručení. Takže IP paket se může cestou ztratit nebo dokonce zahozen (z různých důvodů: je poškozený, nebo se nevejde do MTU a nelze ho fragmentovat, nelze ho doručit – viz ICMP, router je zahlcen a nestíhá přijímat/odesílat, atd.

Z předchozích odstavců vyplývá, že se opravdu může stát, že provoz je nad možnosti routeru a některé pakety bude nutné zahodit, třebaže by v jiné situaci mohly být doručeny. Jak tedy řešit problém zahlcení routeru?

Na směrovačích se může používat jednoduchá metoda řízení propustnosti sítě nazvaná *Token Bucket*. Pracuje na principu virtuálního koše (bucket), do kterého jsou konstantní rychlostí neustále doplňovány tokeny („povolenky“). Pokud přeteče, tokeny jsou zahazovány. Jeden token představuje určité množství odesílaných dat (třeba jeden oktet).

Pokud má být zpracován (směrován, přeposlán) paket, nejdřív je zjištěna jeho velikost. Velikost paketu je porovnána s množstvím odpovídajících tokenů v Token Bucketu. Pokud dostačuje, odstraní se z Bucketu potřebné množství tokenů a paket může být zpracován. Jestliže však nedostačuje, paket bude zahozen.

Kapitola 7

Transportní vrstva

Transportní vrstva je vyrovnávací vrstvou mezi vrstvami orientovanými na přenos (L1–L3) a vrstvami orientovanými na aplikace (L5–L7). Sice se podílí na vyjednávání přenosu dat (čímž je blízká nižším vrstvám), ale žádným způsobem neadresuje zařízení ani sítě. Na druhou stranu se sice podílí na určení konkrétní komunikující aplikace (čímž je blízká vyšším vrstvám), ale je jí naprosto lhostejný formát dat a se všemi druhy dat zachází vpodstatě stejně, aplikace je pro protokoly této vrstvy jednoduše jen číslo, nic víc.

7.1 Komunikace typu klient-server

Základní schéma komunikace mezi dvěma uzly může být buď typu klient-server nebo typu peer-to-peer.

V *komunikaci typu klient-server* je pevně určeno, kdo službu poskytuje (uzel typu server) a kdo službu využívá (uzel typu klient). Komunikaci zahajuje klient svou *žádostí* (dotazem), server následně klientovi zašle *odpověď*. Obvykle není přípustné, aby v tomto modelu komunikaci zahajoval server.

Komunikace typu peer-to-peer probíhá mezi rovnocennými uzly, které v síti mohou plnit roli klienta i serveru. Komunikaci může zahájit kterýkoliv uzel.

7.2 Čísla portů

Na transportní vrstvě se neadresují počítače ani sítě (směrem dolů), adresují se aplikace (směrem nahoru v ISO/OSI). Každá aplikace musí být jednoznačně určena *číslem portu*, a protože komunikující strany jsou dvě, potřebujeme číslo portu zdroje a číslo portu cíle. Znovu připomínáme, že pojem port na vrstvě L4 není totéž co pojem port na nižších vrstvách.

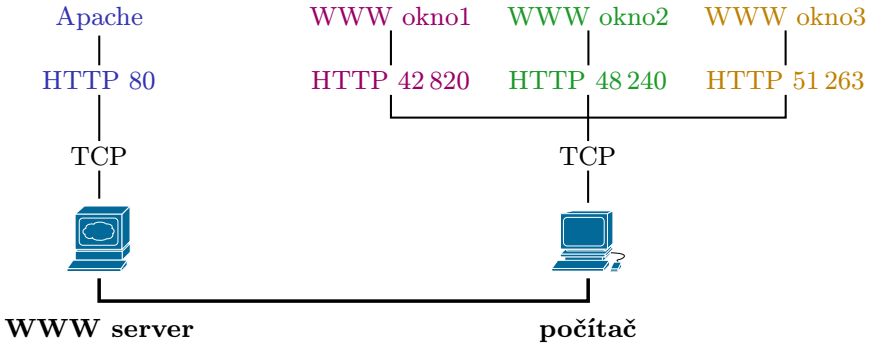
Číslo portu zabírá dva oktety, což nám dává rozsah 0 – 65 535. Z toho jsou

- *porty dobře známé* (well-known) v rozsahu 0–1023, tato čísla se používají pro konkrétní běžně známé služby na serverech (WWW/HTTP, FTP, RPC, SQL, Syslog, Kerberos, atd.),
- *porty registrované* (registered, official) v rozsahu 1024 – 49 151, které si mohou různé společnosti registrovat u organizace IANA (například své registrované porty má Microsoft, IBM, Citrix, Novell, Cisco, Oracle, Eset), sem také spadají porty pro RADIUS, Nessus, nebo BOINC,
- *dynamické* (soukromé) ve zbývajícím rozsahu 49 152–65 535 jsou používány na straně klienta.

Poznámka:

Proč toto rozlišení? První dvě skupiny portů jsou určeny pro serverovou stranu komunikace, přičemž na straně serveru obvykle běží jediná instance dotyčné služby (aplikace) – například na web serveru běží jediná aplikace poskytující webové služby (například Apache nebo IIS), tedy nám stačí jedno číslo. Na straně klienta však může být více komunikujících aplikací stejného typu – například s webovým serverem může komunikovat víc oken webového prohlížeče (vlastně i víc webových prohlížečů), navíc v každém okně můžeme mít několik záložek se zobrazenými stránkami od různých webových serverů.

Kdybychom i na straně klienta chtěli mít jen jedno jediné číslo, nebylo by možné nijak určit, kterému procesu/oknu/záložce konkrétně je určena zrovna ta webová stránka, která právě došla od některého webového serveru.



Obrázek 7.1: Význam portů na transportní vrstvě

Na obrázku 7.1 je naznačen význam portů pro rozlišení jednotlivých navázaných spojení, kterých může být pro daný protokol v rámci běžného počítače více než jen jedno. Pokud se jedná o komunikaci s webovým serverem (což může být například server Apache), na straně serveru běží služba Apache komunikující na TCP portu 80 (jeden z *dobře známých portů*), kdežto na straně počítače máme více oken či záložek webového prohlížeče, pro něž potřebujeme různá čísla portů, a to z rozsahu *dynamických (soukromých) portů*.

Další informace:

Oficiální seznam dobře známých a registrovaných čísel portů je na
http://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

7.3 Protokoly transportní vrstvy

Na transportní vrstvě pracují především protokoly TCP a UDP, ale pro určité konkrétní typy úloh zde najdeme i další.

TCP (Transmission Control Protocol) poskytuje potvrzovanou službu se spojením, garantuje řazení, tedy doručení více úseků dat ve správném pořadí. Používá se pouze pro spojení typu point-to-point, komunikuje vždy jen jedno koncové zařízení s jiným koncovým zařízením.

UDP (User Datagram Protocol) poskytuje nepotvrzovanou službu bez spojení, tedy datagramovou službu. Může se používat i pro spojení typu point-to-multipoint, tedy cílem může být i skupina koncových zařízení.

Z dalších protokolů najdeme na transportní vrstvě například *SCTP* (Stream Control Transmission Protocol), který je podobný TCP, ale dovoluje navázat více nezávislých paralelních spojení (streamů), což snižuje pravděpodobnost ztráty PDU po cestě – každý stream může jít jinou cestou, a pokud má zařízení víc IP adres, lze mezi nimi přecházet. Tento protokol je implementován v jádru mnoha unixových systémů (FreeBSD, Linux, atd.), pro Windows existují implementace třetích stran.

Transportní protokol *RTCP* slouží k rezervaci zdrojů převážně v komunikaci vyžadující upřednostňování při přidělování zdrojů (řízení kvality služby), setkáváme se s ním v některých VoIP technologiích.

Protokol *DCCP* poskytuje službu se spojením (jako TCP), ale negarantuje doručení ve správném pořadí ani nepotvrzuje (jako UDP), jeho hlavním účelem je zajistit rychlé dodání dat. Za tím účelem poskytuje mechanismus řízení zahlcení (dokáže adaptivně měnit přenosové cesty, aby se vyhnuly přetíženým místům). Používají ho například internetová rádia, některé technologie pro internetovou telefonii, on-line videa.

PDU na transportní vrstvě se většinou nazývají *segmenty*, třebaže v případě protokolů poskytujících datagramovou službu (třeba UDP) se setkáváme také s názvem *datagram*.

7.4 Protokol TCP

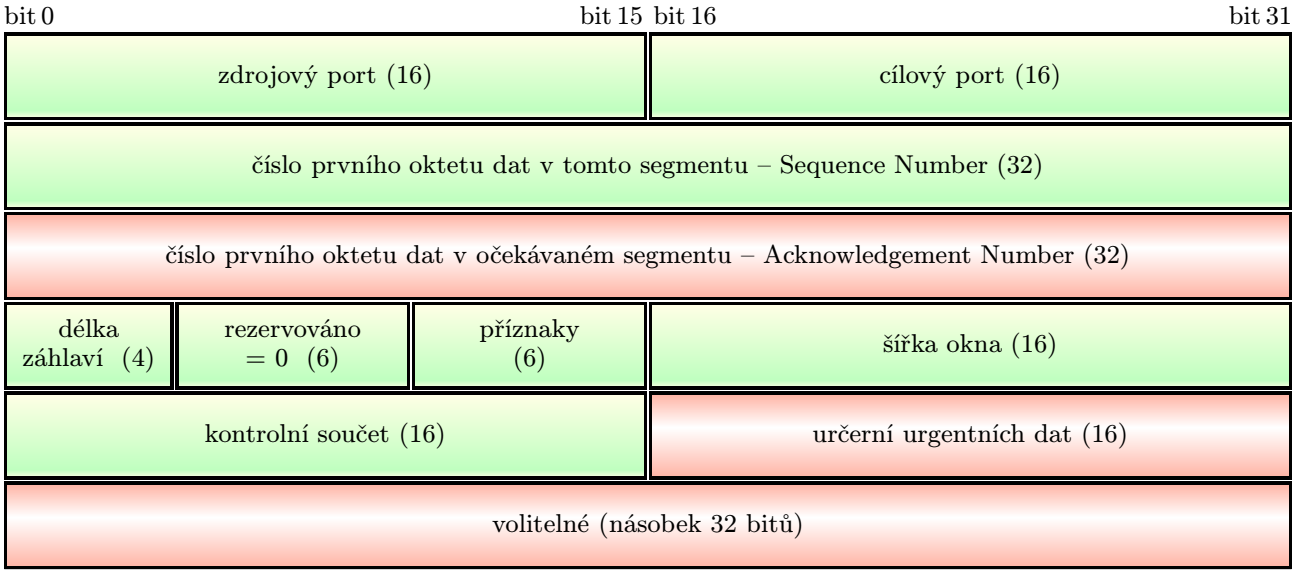
7.4.1 TCP segment

Úkoly protokolu TCP (Transmission Control Protocol) jsou:

- navázat spojení s druhou stranou (vytvořit virtuální kanál),
- udržovat spojení, potvrzovat přijaté segmenty, řídit datový tok,
- převzít SDU z nadřízené vrstvy (obvykle od některého aplikačního protokolu, třeba HTTP) a takto zpracovat:
  - zkontroluje délku tohoto bloku dat, a pokud je větší než povolená délka segmentu, podle potřeby rozdělí na menší části – *segmentuje*,
  - ke každé z částí vzniklých v předchozím bodu přidá TCP záhlaví, přičemž určí čísla portů pro obě komunikující strany, tak vzniknou TCP segmenty,
  - TCP segmenty postupně předá podřízené vrstvě (obvykle protokolu IPv4 nebo IPv6),
- na konci komunikace ukončit spojení.

Na obrázku 7.2 vidíme záhlaví TCP segmentu. Jednotlivá pole mají tento význam:

- *Zdrojový a cílový port* (Source Port, Destination Port, každý 16 bitů) – čísla portů na zdrojovém a cílovém zařízení.
- *Číslo prvního oktetu v segmentu* (*Sequence Number*, 32 bitů) – posloupnost dat z vyšší vrstvy může být rozdělena do několika segmentů, zde je číslo oktetu z původní posloupnosti dat, kterým začíná úsek zapouzdřený v tomto konkrétním segmentu (oktety jsou číslovány od 0).



Obrázek 7.2: Záhlaví TCP segmentu

- *Číslo prvního oktetu v očekávaném segmentu* (*Acknowledgement Number*, 32 bitů) – číslo oktetu začátku části původních dat, který očekáváme (může náležet buď následujícímu segmentu v pořadí nebo takovému předchozímu segmentu, který nebyl řádně doručen). Toto pole je v každém segmentu kromě prvního segmentu při navazování spojení.
- *Délka záhlaví* (Header Length/Data Offset, 4 bity) – délka celého záhlaví v násobcích 32 bitů (takže „počet řádků“ tabulky podle obrázku 7.2).
- *Příznaky* (Flags, Control Bits/Řídící bity, funkce řízení, 6 bitů) – pole bitů (Wireshark je sdružuje s polem rezervovaných bitů), z nichž každý má svůj význam:
  - URG (Urgent) – příznak urgentních dat (v segmentu jsou data, která mají v cíli při zpracování přednost), pole *Určení urgentních dat* má být bráno v úvahu,
  - ACK (Acknowledgement, potvrzení) – tento segment je (kromě jiného) potvrzením předchozího segmentu, pole *Acknowledgement Number* má být bráno v úvahu,
  - SYN (Synchronize bit) – synchronizační bit, používá se při navazování spojení,
  - FIN (Finish bit) – používá se při ukončování spojení,
  - RST (Reset) – žádost o opětovné navázání spojení,
  - PSH (Push) – segment obsahuje aplikační data, která mají být odeslána bez čekání v bufferu.
- *Šířka okna* (Window Size, 16 bitů) – určuje, kolik oktetů maximálně chce odesílatel tohoto segmentu přijmout od svého protějšku bez potvrzování (tj. po odeslání tolika oktetů odešle potvrzující segment).
- *Kontrolní součet* (Checksum, 16 bitů) – počítá se přes celý segment včetně záhlaví plus *pseudozáhlaví*; pseudozáhlaví se vytváří jen pro tento účel (na zdrojovém i cílovém zařízení) a nepřenáší se, obsahuje nejdůležitější údaje z PDU, do které je segment zapouzdřen – většinou protokolu IP (IP adresy zdroje a cíle, protokol – 6 pro TCP, a délku TCP segmentu).
- *Určení urgentních dat* (Urgent Pointer, 16 bitů) – pokud jsou v segmentu také urgentní data a tedy je nastaven bit URG, zde je číslo oktetu, kterým *končí* urgentní data (tj. vlastně určuje, kolik urgentních dat v segmentu máme).

- *Volitelné* (Options, násobek 32 bitů) – volitelné možnosti, které si zařízení navzájem potřebují předat (většinou na začátku spojení nebo když je třeba v průběhu spojení pozměnit jeho parametry), například maximální velikost segmentu, časová razítka (kvůli synchronizaci) nebo parametry pro potvrzování.

 Protokol TCP předepisuje potvrzování doručení, k čemuž nám slouží pole *Sequence Number* a *Acknowledgement Number*. Tato čísla předepisují určitou návaznost mezi tím, co jeden odesílá, a tím, co druhý přijímá, a stejně to funguje i v opačném směru (takže pozor – *Sequence number* jdoucí v jednom směru nemá nic společného se *Sequence number* v opačném směru, obě strany mohou posílat data).

Šířka okna určuje, po jakém kvantu dat je třeba odeslat potvrzení, a obvykle zabírá velikost několika segmentů – to znamená, že se nepotvrzuje každý segment zvlášť, ale odešle se potvrzující segment až po několika doručených segmentech. Tato hodnota obvykle odpovídá velikosti bufferu (vyrovnávací paměti), ale v případě, že se po cestě ztrácí hodně segmentů, bývá nižší.

Poznámka:

Komunikace obvykle nebývá zcela symetrická (tj. většinou jedna strana posílá „krátké“ žádosti a druhá strana „dlouhé“ odpovědi), takže v jednom směru typicky roste *Sequence number* rychleji než v druhém.

Jak tedy potvrzování probíhá? Použití *Sequence Number* je zřejmé – pokud posílám data, v tomto poli určím, kde konkrétně je posílaná část lokalizovaná v původním streamu dat. Pole *Acknowledge Number* určuje, kterou část dat očekávám v opačném směru.

 Jaká tedy má být posloupnost hodnot *Sequence Number* v posílaných (resp. přijímaných) segmentech? Když vezmeme hodnotu *Sequence Number* z jednoho segmentu a přičteme délku dat v tomto segmentu zapouzdřených (tj. délka segmentu mínus velikost záhlaví, obojí v oktetech), získáme hodnotu *Sequence Number* pro následující segment. Délku segmentu zjistíme v IP záhlaví. Wireshark hodnotu pro následující segment taky počítá a zobrazuje ji v hranatých závorkách jako *Next Sequence Number*.

Cílové zařízení přijme tolik segmentů, kolik se vejde do šířky okna, a zkontroluje hodnoty *Sequence Number*, jestli jdou ve správné posloupnosti. Pokud jde všechno tak jak má (žádné segmenty se neztrácejí), odešle druhé straně potvrzující segment, kde je v poli *Acknowledgement Number* číslo vypočtené podle předchozího postupu z posledního získaného segmentu (vezme *Sequence Number* a přičte délku zapouzdřených dat). To odpovídá následujícímu segmentu, který má být přijat.

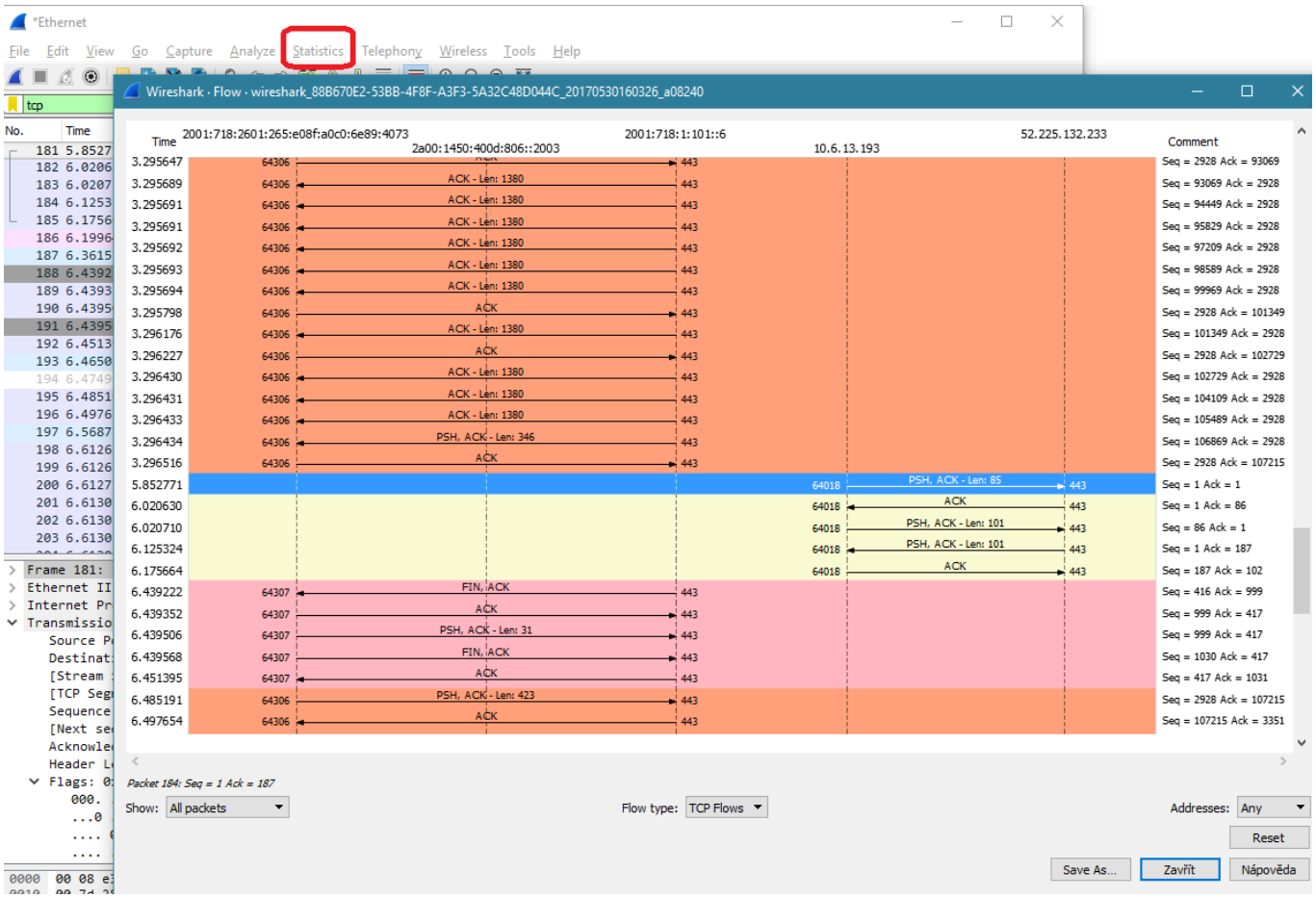
Jak zjistíme, že se některý segment cestou ztratil? V řadě doručených (zatím nepotvrzených) segmentů provedeme výše naznačený výpočet. Pokud při výpočtu z jednoho segmentu nesedí výsledek na *Sequence Number* následujícího v pořadí (v segmentu je větší číslo než nám vyšlo), pak to znamená, že na tom místě chybí minimálně jeden segment. Proto odešleme potvrzující segment s hodnotou *Acknowledgement Number* takovou, která nám vyšla pro chybějící segment.

Poznámka:

Wireshark ve výchozím nastavení zobrazuje pouze *relativní* Sequence a Acknowledge Number (menší než ve skutečnosti, aby pro daný stream začínalo číslem 0) – účelem je zjednodušit optické srovnávání těchto hodnot. Pokud nám to vadí, můžeme si nastavit zobrazování skutečných hodnot těchto čísel (v menu zvolíme *Edit* ; *Preferences* ; *Protocols*, najdeme *TCP*, zrušíme zatržení u *Relative sequence*

numbers). Problém je, že komunikace tím ztratí na přehlednosti.

Hodnoty Sequence Number jsou pěkně vidět například v nástroji Flow Graph, který ve Wiresharku zobrazíme přes menu *Statistics* ; *Flow Graph*, viz obrázek 7.3 (na obrázku jsou relativní čísla, výše uvedené nastavení nebylo provedeno).



Obrázek 7.3: FlowGraph zobrazený ve Wiresharku, filtr na protokol TCP

V okně Flow Graph můžeme používat několik jednoduchých filtrů, a pokud v hlavním okně Wiresharku máme nastavený některý filtr (třeba na konkrétní protokol), v okně Flow Graph (vlevo dole) můžeme zapnout zobrazování jen těch profiltrovaných paketů.

Poznámka:

Jaký je rozdíl mezi příznaky PSH (Push) a URG (Urgent)?

TCP posílá data v segmentech, a segmenty kompletuje ve speciální paměti, které říkáme buffer. Pokud na tentýž socket (tj. tytéž adresy + porty pro oba směry) má být zasláno více úseků dat, může je protokol TCP zatím ukládat do bufferu a vše odešle, až se nashromáždí dostatečně velký segment. Pokud však „nechceme čekat“ (potřebujeme, aby třeba i malý úsek dat byl poslán okamžitě), nastavíme příznak PSH (tj. data mají být hned vyndána – pushed – z bufferu). Takže příznak PSH je ve skutečnosti určen implementaci TCP na odesílající stanici.

Příznak URG je naproti tomu určen cílové stanici – pokud je nastaven, znamená to, že příchozí data se nemají „zbytečně“ zdržovat v bufferu (i při přijímání se používá buffer), ale mají být přednostně přijata a zpracována. Přesněji – ne všechna data z tohoto segmentu, ale jen ta od začátku segmentu do adresy označené v poli Urgent Pointer (určení urgentních dat). Protože se adresuje od 0, je v tomto poli vlastně počet oktetů urgentních dat.

http://packetlife.net/blog/2011/mar/2/tcp-flags-psh-and-urg/

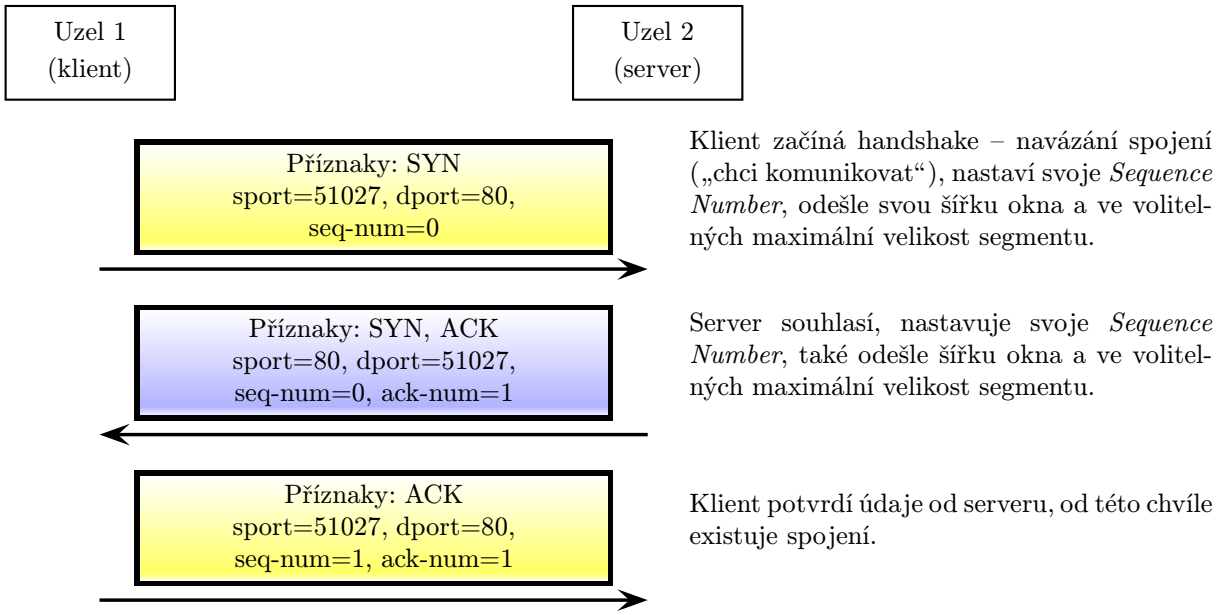
7.4.2 Průběh TCP spojení

TCP spojení musí být nejdřív navázáno, čemuž se říká *Three-way Handshake* (třícestné zahájení). Nazývá se tak proto, že se během zahájení spojení postupně posílají tři segmenty s dojednáváním parametrů spojení (bez dat).

Postup navázání spojení pro komunikaci s webovým serverem je znázorněn na obrázku 7.4:

1. První zařízení (klient) odešle první segment ve významu „Chci navázat spojení“. Tento segment má nastaven příznak SYN a obsahuje prvotní parametry v poli *Volitelné*, například časové razítko kvůli synchronizaci a maximální velikost segmentu, který toto zařízení může přijmout.
2. Druhé zařízení odešle v odpověď svůj první segment (proto má taky nastaven příznak SYN), který je tedy odpovědí, a tedy má nastaven taky příznak ACK. V poli *Volitelné* má podobný typ informací jako byly v předchozím paketu.
3. Druhý segment potvrzoval přijetí toho prvního, tedy je ještě nutné potvrdit přijetí druhého segmentu. Třetí segment má právě tuto funkci. Už není inicializační a neobsahuje synchronizační informace, takže má nastaven pouze příznak ACK. Pole *Volitelné* obvykle není využito.

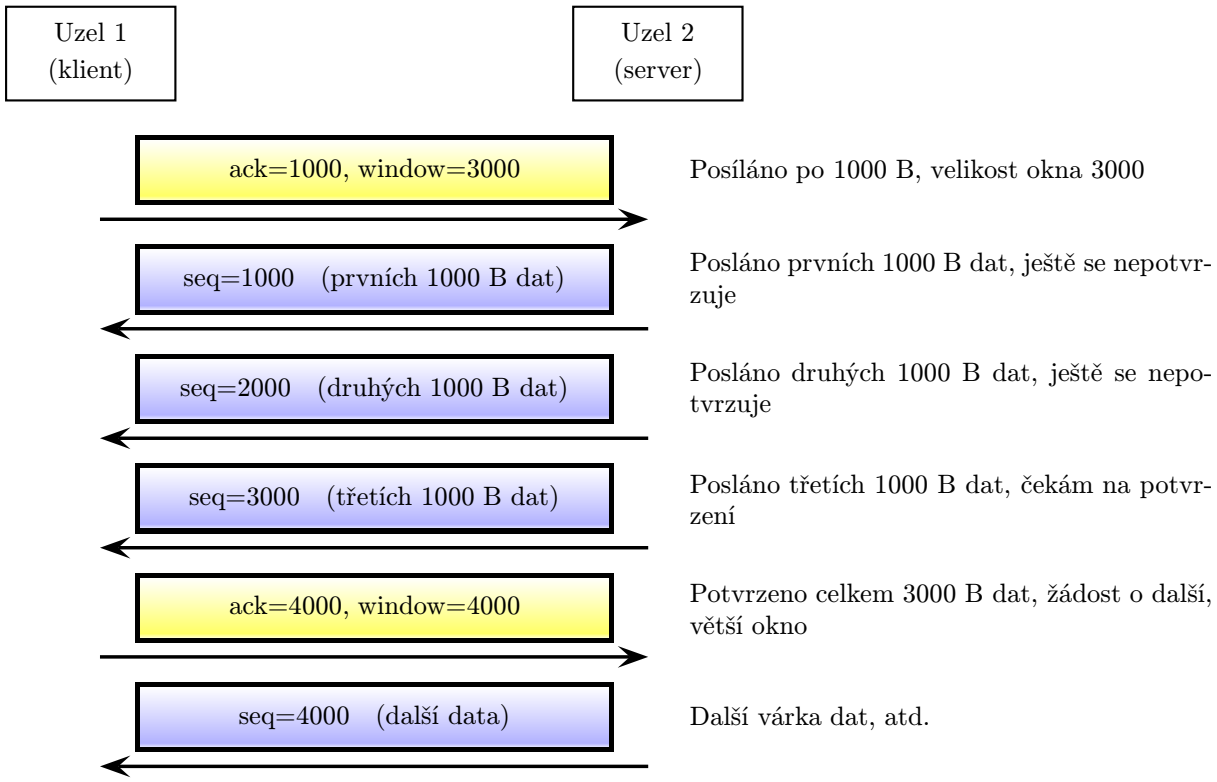
Čísla portů jsou zřejmá – server používá číslo portu 80, klient má dynamický port.



Obrázek 7.4: TCP handshake – navázání spojení s webovým serverem

V prvních dvou segmentech navolí odesílatel svou výchozí hodnotu pro *Sequence Number* (Wireshark místo toho zobrazí 0 jako relativní číslo), v třetím segmentu je *Sequence Number* o 1 vyšší než

v prvním segmentu. *Acknowledgement Number* se používá až od druhého segmentu (v prvním není co potvrzovat) a v druhém a třetím segmentu se použije o 1 vyšší než je *Sequence Number* v předchozím (obdrženém) segmentu. Po navázání spojení (tedy od čtvrtého segmentu dále) se již tato čísla používají tak, jak bylo výše popsáno.



Obrázek 7.5: Běžná komunikace v rámci TCP spojení

Pokud nedochází ke ztrátám segmentů, posílá se potvrzení (acknowledgement) vždy po tolika segmentech, kolik se jich vejde do šířky okna. Na obrázku 7.5 vidíme průběh takové komunikace pro případ, že velikost segmentu je nastavena na 1000 a velikost okna na 3000. V potvrzujícím segmentu je *Acknowledgement Number* nastaven na 4000, protože žádáme o poslání segmentu s tímto *Sequence Number* (tím dáváme na vědomí, že všechny předchozí byly doručeny).

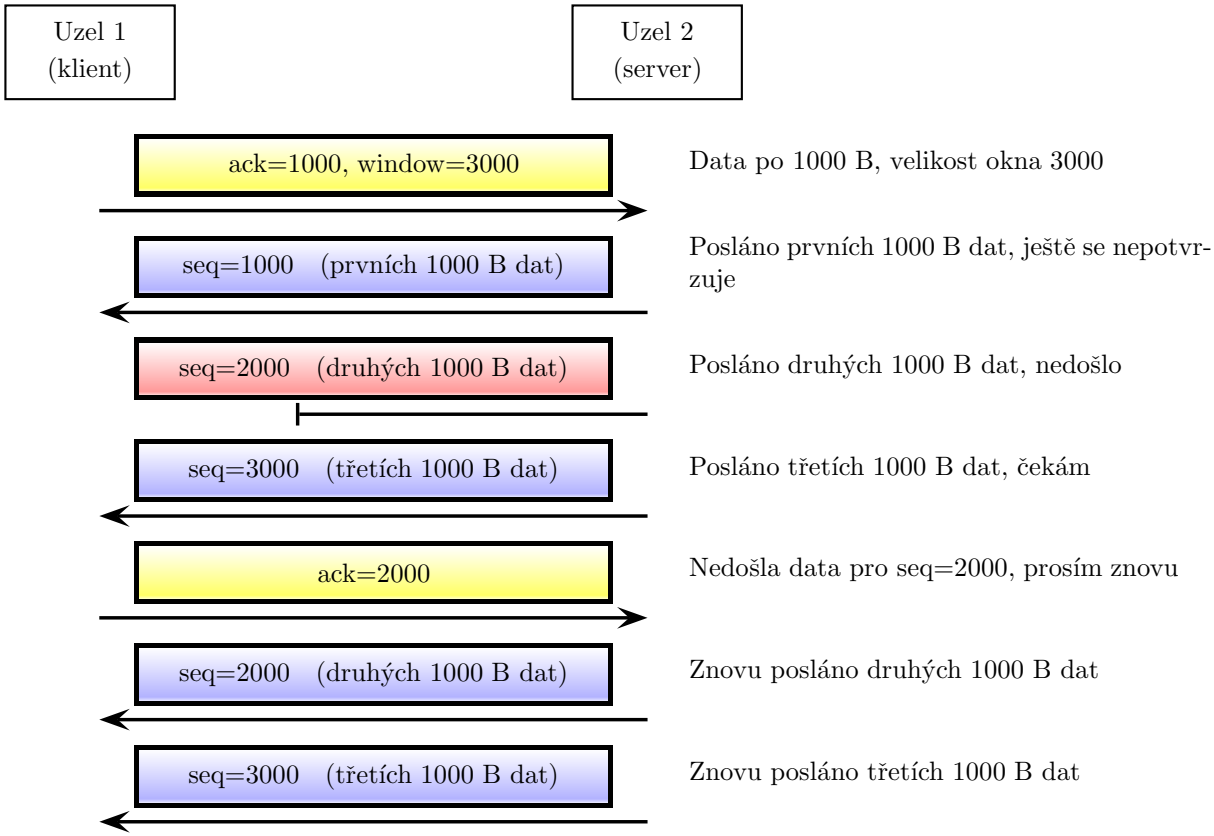
Na obrázku 7.6 je znázorněna situace, kdy došlo ke ztrátě segmentu. Příjemce dat podle *Sequence Number* v záhlaví segmentů zjistil, že chybí segment pro *Sequence Number* 2000, tedy do potvrzujícího segmentu zadá jako *Acknowledgement Number* právě toto číslo. Pokud by se ztratilo víc segmentů, příjemce dat odešle nejdřív žádost o první ztracený, po jeho doručení požádá o druhý ztracený apod. V tom případě se opravdu potvrzuje každý (znovuposílaný) segment.

Poznámka:

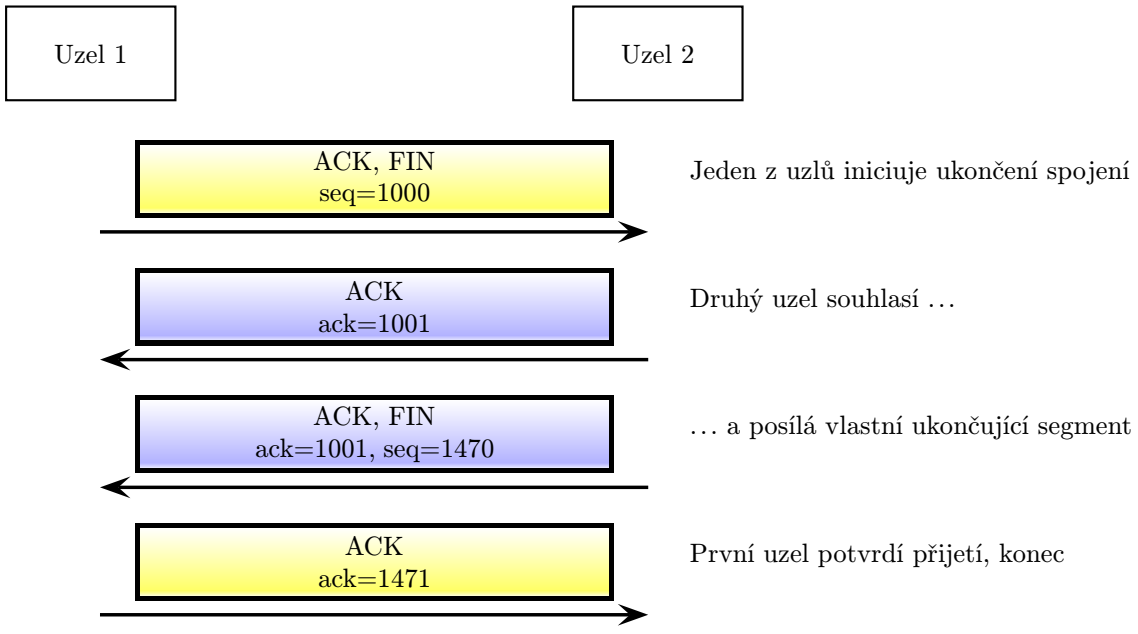
Důležité je taky to, kdy konkrétně začne příjemce dat zjišťovat, zda má všechny segmenty patřící do probíhajícího okna. Je stanovena maximální doba čekání (timeout) na doručení segmentů, a pokud tato doba uplyne, jsou dosud nedoručené segmenty považovány za ztracené.

Ukončení spojení může být provedeno kterýmkoliv z komunikujících zařízení, a proběhne pomocí

čtyř segmentů, jak je naznačeno na obrázku 7.7. Nejdřív ukončení oznámí jedna strana (je nastaven příznak FIN), pak druhá potvrdí a pošle vlastní ukončující segment (taky s příznakem FIN), následuje potvrzení přijetí. Oba potvrzující segmenty mají nastaven pouze příznak ACK.



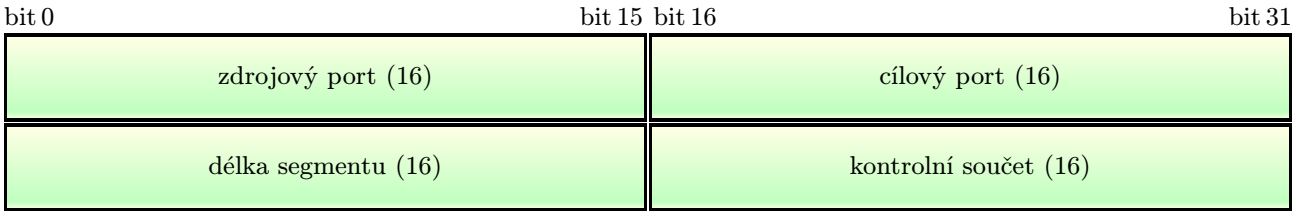
Obrázek 7.6: Komunikace v rámci TCP spojení, jeden segment nedorazil



Obrázek 7.7: Ukončení TCP spojení

7.5 Protokol UDP

Jak bylo napsáno výše, protokol UDP (User Datagram Protocol) poskytuje nepotvrzovanou službu bez navázání spojení, tedy jednoduchou datagramovou službu. Jeho úkoly jsou tedy mnohem jednodušší než u protokolu TCP – jednoduše převezme SDU z nadřízené vrstvy, přidá záhlaví a vytvořený segment (datagram) předá podřízené vrstvě. Nenavazuje se spojení, nepotvrzuje se, neřeší se velikost okna, nepočítají se čísla pro chybějící segmenty (vlastně se ani žádná taková čísla nepoužívají).



Obrázek 7.8: Záhlaví UDP segmentu

Proto je záhlaví UDP segmentu mnohem jednodušší, část polí tam prostě nepotřebujeme. Na obrázku 7.8 vidíme záhlaví UDP segmentu – první řádek je stejný jako u TCP, z druhého řádku je stejný kontrolní součet, „navíc“ máme délku celého segmentu (včetně záhlaví) – to u TCP není, a naopak všechna zbylá TCP pole chybí.

Protokol UDP se používá tehdy, když

- je zbytečné navazovat spojení (nebo to z nějakého důvodu není možné či žádoucí),
- posíláme jen málo dat, která není nutné segmentovat do více segmentů,
- chceme, aby byl přenos rychlý a moc nezahlcoval linku,
- chceme poslat data na multicast nebo broadcast adresu (TCP podporuje pouze unicast přenosy).

	TCP	**UDP**
Spojení	navazuje	nenavazuje
Potvrzování	ano	ne
Spolehlivost	ano	ne
Segmentace do více segmentů	ano	ne
Rychlost	pomalý	rychlý
Typ cíle	unicast	unicast, multicast, broadcast

Tabulka 7.1: Srovnání

Kapitola 8

Některé aplikační protokoly

V této kapitole se zaměříme na vrstvu L7, tedy aplikační. Budeme se zabývat službami poskytovanými na této vrstvě, a protokoly, které určují technologii implementující tyto služby (tedy aplikačními protokoly).

Nicméně zde nejsou ani zdaleka všechny běžné aplikační protokoly. Některé další najdeme v následujících kapitolách, ostatní se nám do semestru bohužel nevejdou.

8.1 Služba WWW a protokol HTTP

Protokol HTTP (HyperText Transfer Protocol) známe především z adresního řádku webového prohlížeče, ale obecně je tento protokol používán pro přenos strukturovaných informací; přenos webových stránek ve formátu HTML či XML je jen jednou z mnoha možností jeho využití. Nicméně, nejčastěji pomocí HTTP komunikujeme právě s webovým serverem.

8.1.1 Adresace

URI (Uniform Resource Identifier) je sekvence znaků identifikující konkrétní zdroj. Může to být lokátor (stanovuje umístění zdroje) nebo název (bez lokace). Existují tedy dva druhy URI:

- *URL* (Uniform Resource Locator) je URI typu lokátoru,
- *URN* (Uniform Resource Name) je URI typu názvu.

Odlišnost je tedy v tom, zda v identifikátoru máme nebo nemáme zakódováno umístění zdroje. Například telefonní číslo je URI typu název (URN), protože neurčuje konkrétní lokaci.

Protokol HTTP používá URL kromě jiného pro určení webového serveru. Je to řetězec začínající názvem protokolu (v tomto případě `http://` následovaným jmennou adresou serveru a podle potřeby určením konkrétní stránky v adresářové struktuře na serveru.

Plná specifikace URI je `protokol://server:port/cesta`, kde

- `protokol` určuje, přes který protokol se komunikuje (http, ftp apod.),
- `server` je adresa serveru, obvykle jmenná (hostname) nebo IP adresa,
- `port` je číslo portu TCP/UDP, přes který se má komunikovat (zadáváme, pokud se má použít jiný než je obvyklé),
- `cesta` určuje cestu k žádanému zdroji v rámci zadaného serveru (v adresářové struktuře).

Některé části jsou nepovinné. Pokud je nezadáme, je zvolena výchozí možnost. Například pokud je zadána adresa webového serveru a nepřipíšeme číslo portu, zvolí se port 80.

Příklad

V následujícím řetězci, který je příkladem URL, máme tři části:

http://www.firma.cz/dokumenty/kontakty.html

První část určuje protokol, druhá jmennou adresu serveru (která může být nahrazena jeho IP adresou, ať už ručně nebo překladem přes DNS), třetí cestu k dokumentu ve formátu HTML uloženém na serveru.

Kdybychom jmennou adresu serveru nahradili jeho IP adresou, taky se bude jednat o URL, a bude lokalizovat stejný zdroj.

8.1.2 HTTP zprávy

PDU protokolu HTTP se nazývají zprávy a zapouzdřují se do TCP segmentů (tj. spojová komunikace s potvrzováním). V naprosté většině případů se na straně serveru používá port 80 (můžeme se setkat také s portem 8080), na straně klienta máme opět pro každého tazatele nad aplikační vrstvou samostatné číslo portu z rozsahu pro dynamické porty. Celá komunikace se označuje jako *HTTP relace* (session) a patří do ní všechny zprávy v rámci daného spojení.

HTTP předepisuje komunikaci typu dotaz–odpověď. HTTP zpráva může být tedy buď dotazem (request) nebo odpovědí (response). Dotaz může být proveden jednou z těchto *metod*:

- HTTP GET – nejčastěji používaná metoda. Součástí URL je i specifikace konkrétních dat.
- HTTP POST – používá se pro odesílání vyplněných webových formulářů, posílaná data nedává do URL, ale do těla zprávy.
- Další metody – například PUT přenese data na server, DELETE umožňuje mazat objekty na webovém serveru, HEAD funguje jako GET, ale neposílá data (jenom poskytuje metadata v záhlaví), CONNECT přidává do adresy informaci o číslu portu.

První informace ze záhlaví je použitá metoda, zbytek záhlaví je posloupnost dvojic ve tvaru „položka: hodnota“. Položky takto předávané jsou například typ obsahu, označení serveru, konkrétní adresa objektu na serveru, použité kódování, v položce User-Agent zase najdeme co nejúplnější specifikaci programu, který na straně klienta žádá o webovou stránku.

Následuje tělo HTTP zprávy. Pokud je co přenášet, při odeslání od serveru klientovi tam většinou najdeme HTML kód žádané stránky.

8.1.3 Komunikace podle HTTP

Komunikace podle HTTP probíhá následovně:

- Proběhne TCP Three-Way Handshake. V poli pro číslo portu na straně serveru je číslo 80, třebaže uvnitř těchto TCP segmentů nemáme zapouzdřenou žádnou HTTP zprávu.
- Následně klient odešle žádost serveru v TCP segmentu, který již obsahuje zapouzdřenou HTTP zprávu se záhlavím typu request.

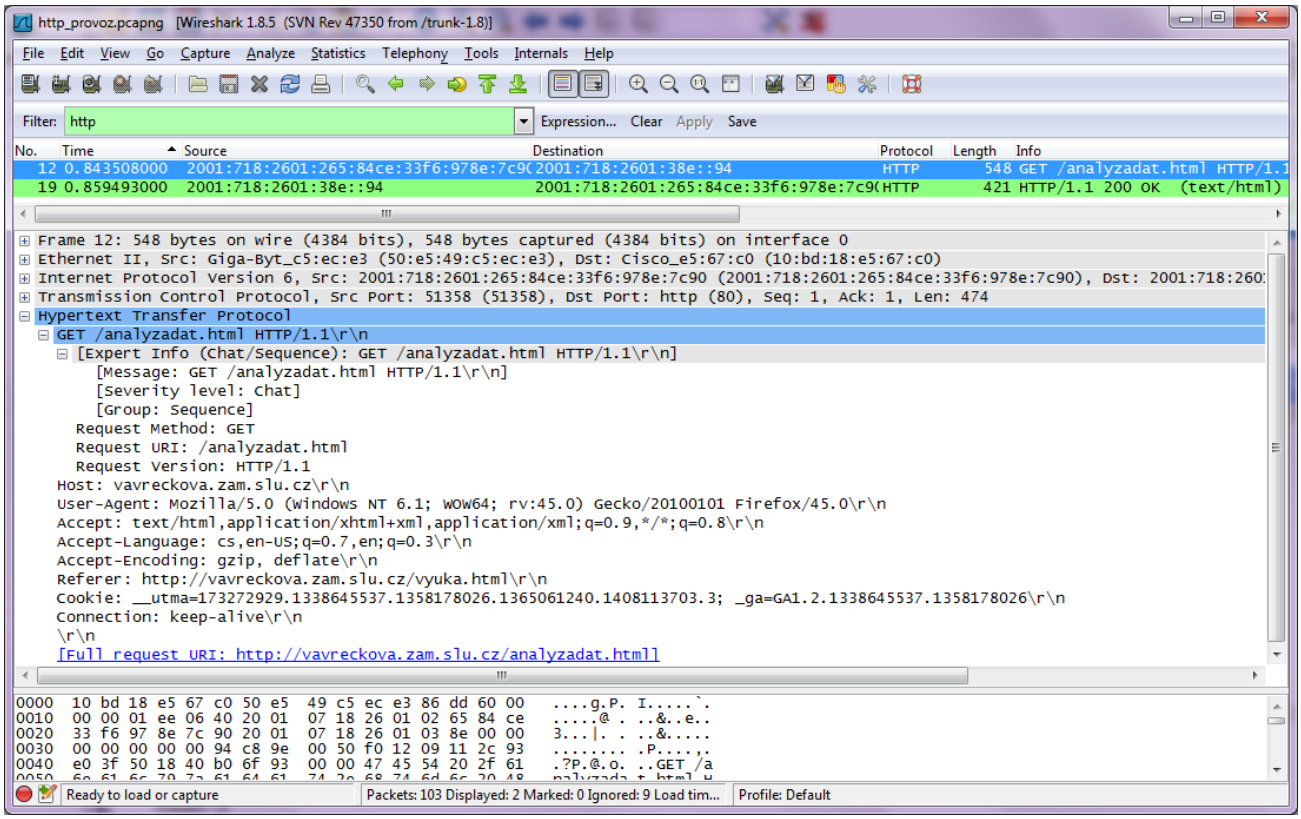
- Server odpoví TCP segmentem se zapouzdřenou HTTP zprávou se záhlavím typu response.
- Podle potřeby se dotazy a odpovědi opakují.
- Následuje konec HTTP relace a ukončení TCP spojení (bez zapouzdřených HTTP zpráv).

Jestliže klient z předchozí odpovědi serveru zjistil, že na stránce, jejíž text již obdržel, se nachází adresa některého vnořeného objektu k načtení (obrázku, rámu apod.), pošle serveru nový HTTP dotaz, tentokrát na vnořený objekt. To se může opakovat i rekurzívně, také ve vnořeném objektu může být jiný vnořený objekt.

Samozřejmě kromě výše uvedených kroků je třeba udělat kroky „doprovodné“, například protokolem DNS zjistit IP adresu zadaného serveru (a až pak se provádí Three-Way Handshake). Ještě předtím může být třeba pomocí protokolu ARP nebo NDP zjistit MAC adresu brány.

Příklad

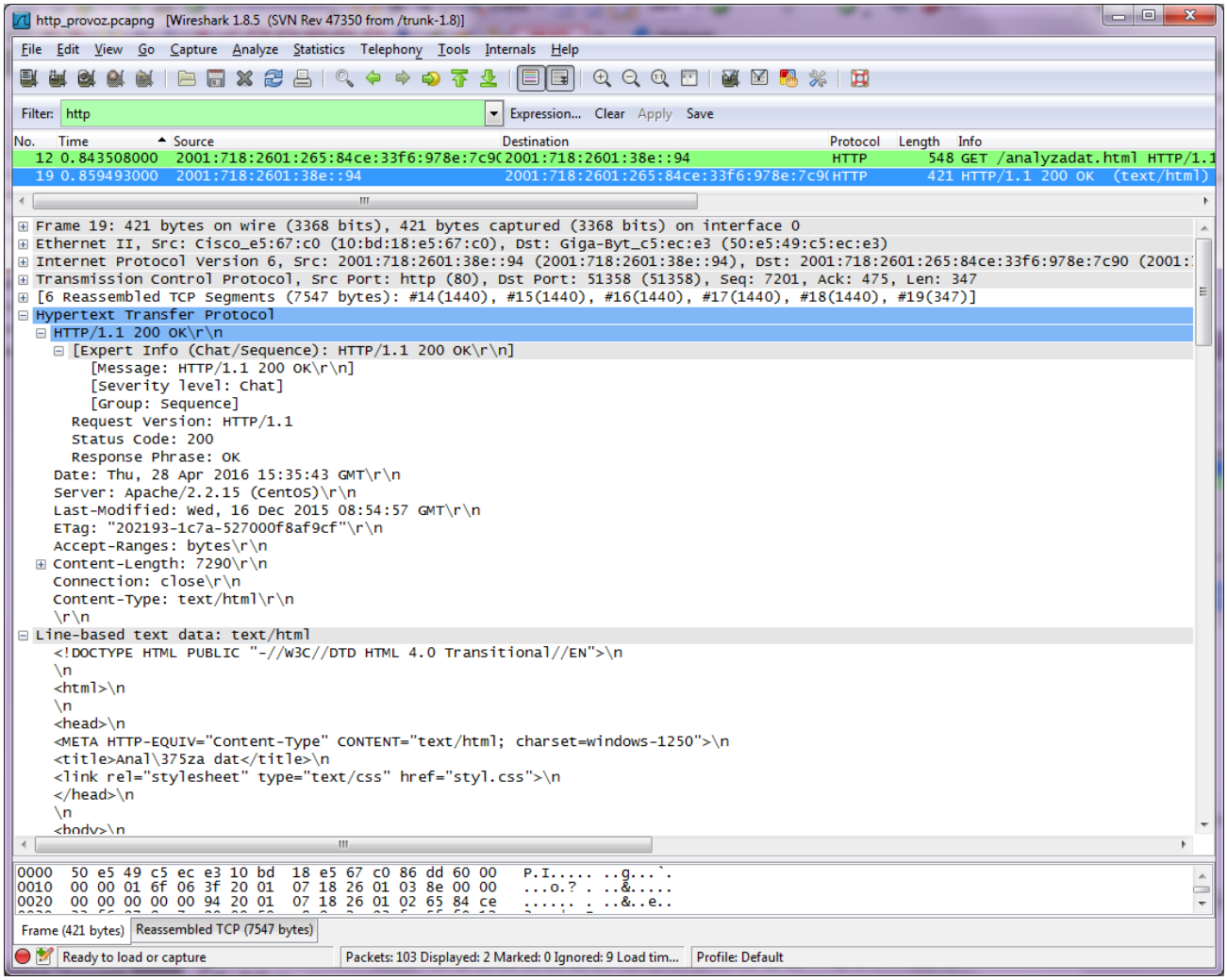
Ukážeme si na konkrétním příkladu (jen HTTP zprávy, žádný „kontext“ v podobě TCP či DNS), jak vypadá vyžádání stránky a její doručení.



Obrázek 8.1: Zpráva HTTP GET

Na obrázku 8.1 je HTTP zpráva s dotazem (žádostí) typu GET. Všimněte si, že za `GET` je vlastně poslední část URL, tedy konkrétní soubor (kdyby byl tento soubor zanořen v některém adresáři, byla by tu celá cesta). Až za údaji o dotazu GET je řádek `Host: ...`, kde je část URL odpovídající doménovému názvu. Jednotlivé záznamy v záhlaví jsou ukončeny dvojicí znaků `\r\n`, což je zařádkování. Na konci celého záhlaví (v případě dotazu to odpovídá i konci celé HTTP zprávy) je toto ukončení zdvojeno, čímž cíl pozná konec záhlaví.

Všimněte si také, že náš prohlížeč poněkud „bonzuje“ – nejen cíl, ale také kdokoliv na cestě se dozví, jaký máme webový prohlížeč a v jakém operačním systému pracujeme, včetně verzí (zde Firefox na 64bitových Windows verze 6.1, což jsou Windows 7).



Obrázek 8.2: Potvrzení zprávy HTTP GET

A co jsme se dozvěděli z odpovědi? Například to, že na serveru běží webový server Apache, a operační systém je CentOS. Záhlaví opět končí dvojitým odřádkováním, ale tentokrát máme i datovou část – `Line-based text data`, kde je nejdřív uveden typ podle MIME (o MIME víc v následující sekci). Následuje kód webové stránky ve formátu HTML.

Nejznámější webové (HTTP) servery jsou Apache, MS IIS a nginx, většina webových sídel používá Apache. Obvykle jde o službu nebo démona (démon je v UNIXových systémech obdoba služby) – proces běžící na pozadí a neustále vyhodnocující požadavky, které přicházejí ze sítě.

Další informace:

Podle *Netcraftu* (známá britská analytická společnost) podíl Apache a IIS mírně klesá, podíl nginxu mírně roste. Analýza z března 2016 je na http://news.netcraft.com/archives/2016/04/21/april-

2016-web-server-survey.html. U aktivních webových stránek má Apache podíl 49 %, nginx 17 %, IIS 10 %. U „nejrušnějších“ (busiest) stránek má Apache 45 %, nginx 26 % a IIS 11 %.

Poznámka:

Jak vlastně webový server (třeba Apache) pozná, že mu právě byla doručena HTTP zpráva? O to se musí postarat sám démon či služba. Říkáme tomu, že *naslouchá* na portu (obvykle na portu číslo 80), a kdykoliv přijde na tento port požadavek, naslouchající proces je informován.

Dnes často komunikujeme s webovým serverem zabezpečeně. To znamená, že mezi protokoly HTTP a TCP se „nasune“ jeden z protokolů SSL nebo TLS a zajistí šifrování komunikace. Kombinace HTTP a jednoho z těchto zabezpečujících protokolů se označuje *HTTPS*. Hned po navázání TCP spojení se dojednají parametry zabezpečeného připojení a od té chvíle je komunikace šifrovaná.

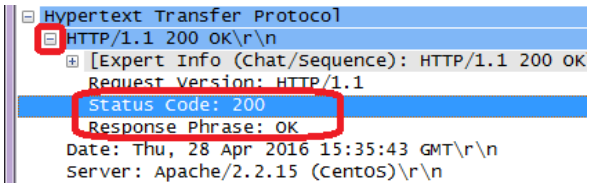
Komunikace podle HTTPS probíhá na jiném portu než 80, na straně serveru se používá port 443. Takže webový server vlastně naslouchá nejen na portu 80, ale také na portu 443 (a případně jiných portech podle konfigurace).

8.1.4 Informační kódy HTTP

Ne vždy se povede dostat webovou stránku ke klientovi tak, jak by to mělo být. Může nastat mnoho různých chyb, na straně klienta, serveru i po cestě. Uživatel na straně klienta pak může (nemusí) být informován webovým prohlížečem, že něco není v pořádku. Kromě chyb je samozřejmě klient informován obecně o stavu plnění požadavku.

Protokolem HTTP je klient informován tříciferným kódem. Kódy začínající určitou číslicí mají tento význam:

- 1xx – informační, server zpracovává požadavek, například kód 100 znamená, že server úspěšně přijal žádost a začal pracovat na jejím řešení,
- 2xx – oznámení o úspěšném zpracování požadavku nebo jeho části, například kód 200 znamená úspěšné dokončení operace,
- 3xx – pro úspěšné zpracování požadavku je třeba provést některou nestandardní operaci (například přesměrování na jinou adresu),
- 4xx – chyba na straně klienta,
- 5xx – chyba na straně serveru.



První a druhý typ kódu jsou informace o průběhu, od třetího typu dále již jde o informace o chybách.

Z chyb na straně klienta je asi nejběžnější chyba 404 (Not Found – požadovaný zdroj nebyl na zadaném serveru nalezen), což většinou znamená, že jsme se překlepli v poslední části URL (název souboru nebo některý adresář na cestě k němu), nebo tento soubor byl na serveru přesunut, přejmenován či smazán.

Chyba 401 je poslána klientovi v případě, že žádá přístup ke zdroji, ke kterému je požadována autentizace (tj. musíme zadat přihlašovací informace a pak bude zdroj zpřístupněn.

Ze serverových chyb se můžeme setkat třeba s chybou 500 (Internal Server Error), která nastává v případě, že proces běžící na straně serveru nereaguje tak, jak by měl (například zamrzl), nebo 503 (Service Unavailable), pokud je server odstaven (například pro účely údržby, asi na něm běží Windows).

Tento kód je součástí záhlaví HTTP zprávy. Všimněte si na obrázku 8.2 na straně 186 (také na obrázku na předchozí straně) řádku `Status Code: 200`, to je přesně ono, server sděluje klientovi, že je vše hotovo a v této zprávě posílá požadovanou stránku (obecně objekt).

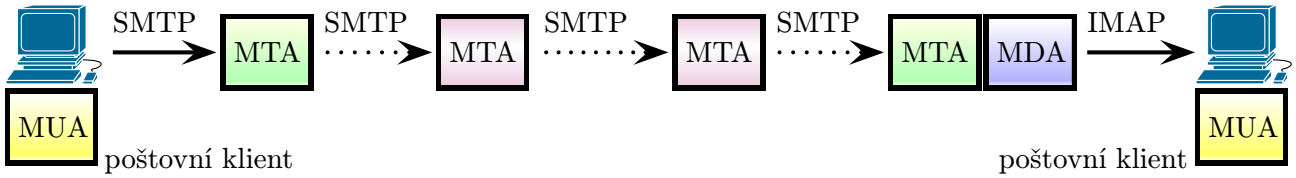
Další informace:

Na stránce https://www.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html je přehledně popsáno, jak protokol HTTP funguje, včetně záhlaví s informačními a chybovými kódy.

8.2 Služby elektronické pošty

8.2.1 Infrastruktura

Jak jistě každý ví, svou e-mailovou schránku máme na poštovním (e-mail) serveru. Když odesíláme e-mail, pošleme ho právě tomuto serveru (ten zajistí přeposlání do schránky příjemce), na příchozí e-maily (jejichž jsme adresátem) se díváme do naší schránky. Takže tento server je pro nás poskytovatelem poštovní služby. Celá infrastruktura je však složitější než jen poštovní servery se schránkami, jak vidíme na obrázku 8.3.



Obrázek 8.3: Infrastruktura poštovních serverů

Součástí infrastruktury jsou následující prvky:

- *MTA* (Mail Transfer Agent, agent pro transfer zpráv) – jeho úkolem je přijmout e-mailovou zprávu, zkontrolovat a podle adresy určit, kam ji případně přeposlat. Roli MTA mohou plnit mail servery jako je MS Exchange, postfix, qmail, sendmail, atd.
- *MDA* (Mail Delivery Agent, agent doručení zpráv) – vede poštovní schránky uživatelů, přijímá od MTA zprávy určené do těchto schránek. Funkce MDA bývá obvykle zahrnuta v nástrojích pro MTA.
- *MUA* (Mail User Agent) – aplikace, která komunikuje přímo s uživatelem (obvykle lokálně na počítači) na jedné straně a s jeho schránkou (vedenou MDA) na druhé straně. Obvykle se jedná buď o klientský program (Outlook, Thunderbird, Evolution apod.) nebo o webový prohlížeč se spuštěným webovým rozhraním schránky.

Takže pokud posíláme e-mail, nejdřív naše MUA aplikace naváže spojení s naším mail serverem plnícím roli MTA. Odešle mu e-mail, ten je pak přeposlán dalšímu MTA na cestě, atd. (zpráva postupně prochází do nadřízených domén a sítí a pak v opačném směru v hierarchii sítě příjemce směrem dolů),

v doménách, kterými prochází, se v DNS záznamech hledají záznamy typu MX. Poslední MTA na cestě již zprávu předá cílovému MDA, který zajistí umístění do schránky příjemce.

Poznámka:

Pokud to umíme, můžeme se obejít i bez MUA. Spojení s MTA se totiž dá navázat i přímo přes Telnet nebo SSH, přičemž postupně odesíláme ty informace, které by jinak odesílal MUA agent.

8.2.2 Protokoly

Pro práci s elektronickou poštou potřebujeme dva druhy protokolů:

- protokol zajišťující odeslání zprávy a její transfer až do schánky adresáta, dnes se prakticky výhradně používá protokol SMTP,
- protokol pro přístup ke zprávám ve vlastní schránce, kde je na výběr mezi protokoly POP a IMAP.

Protokol SMTP (Simple Mail Transfer Protocol) tedy slouží k transferu (přeposílání) zprávy směrem k poštovním serverům. Právě podle tohoto protokolu také hovoříme o SMTP serverech (to jsou servery poskytující službu MTA). Pomocí SMTP komunikujeme se svým MTA serverem, když odesíláme e-mail, a stejně komunikují mezi sebou jednotlivé MTA servery.

SMTP komunikuje na portu 25, SMTP zprávy jsou zapouzdřovány do TCP nebo UDP segmentů (většinou TCP).

U protokolu HTTP jste si určitě všimli, že záhlaví je poměrně variabilní (podle směru a fáze komunikace). Stejně je to i SMTP. Mezi MUA a MTA je nejdřív navázáno TCP spojení (s číslem portu 25), dojednáno zabezpečení (abychom nepřenášeli přihlašovací údaje jako obyčejný text) a následně se odesílají „tematické“ SMTP zprávy s adresami odesílatele a příjemce, předmětem a dalšími částmi. Zprávu pak zkompletuje náš lokální MTA.

Na další cestě (když už na obou stranách spoje najdeme MTA) je už komunikace jednodušší. Zpráva se přenáší „v celku“, a na každém MTA se na její začátek přidá nový záznam – údaj o dotyčném MTA. Z toho vyplývá, že velikost SMTP zprávy po cestě postupně narůstá, a příjemce si pak v záhlaví může ověřit, přes které MTA (SMTP) servery zpráva šla.

Protokol POP (Post Office Protocol) ve verzi 3, tedy POP3, je již poněkud zastaralý. Jeho účelem je přístup do e-mailové schránky za účelem stažení zpráv do lokálního úložiště. Ve verzi 3 komunikuje na portu 110, používá protokol TCP.

Protokol IMAP (Internet Message Access Protocol) je již modernější náhradou protokolu POP. Umožňuje přistupovat do schránky, ale nejen pro stažení zpráv – nabízí službu on-line správy se vším všudy, tedy například:

- číst, kopírovat a mazat vybrané zprávy přímo ve schránce na MDA,
- třídit zprávy, označovat, volněji používat vyhrazený paměťový prostor ve schránce,
- prohlížení či stažení zpráv nebo jen záhlaví zpráv, čímž se šetří přenosová kapacita sítě,
- ke schránce a jednotlivým zprávám lze přistupovat z různých klientských zařízení a na každém mít vlastní kopii zpráv ve schránce,
- pokročilé možnosti automatizace, například automatické zálohování zpráv.

IMAP komunikuje na TCP portu 143. Dnes se téměř výhradně používá protokol IMAP, s protokolem POP3 se setkáváme výjimečně.

8.2.3 Komunikace a nastavení

V případě elektronické pošty jde také o spojovanou komunikaci, tedy protokol TCP, do kterého jsou zapouzdřovány SMTP a IMAP zprávy. To vždy znamená komunikaci typu klient-server.

U SMTP je klientem to zařízení, které odesílá SMTP zprávu. SMTP server (tedy MTA) naslouchá na portu 25 a očekává příchozí zprávy.

IMAP klient je příjemcem zprávy, a musí se (obvykle ve vhodných intervalech) dotazovat IMAP serveru na změny. IMAP server naslouchá na portu 143 a očekává právě tyto dotazy.

Když konfigurujeme MUA (poštovního klienta), musíme mu sdělit, kdo je jeho protějškem – zadáváme jmennou adresu SMTP serveru a číslo portu pro komunikaci s ním, a dále adresu IMAP serveru s číslem portu.

Pokud se jedná o pouhé webové rozhraní ke schránce, pak něco takového nemusíme dělat, protože konfigurace je součástí webové aplikace a souvisí s adresou, kterou píšeme do adresního řádku. Navíc je otázkou, zda opravdu jde o MUA, protože s SMTP/IMAP serverem komunikuje protokolem HTTPS.

Poznámka:

Dnes se již běžně s poštovními servery komunikuje zabezpečeně. Pozor, to neznamená, že by zprávy byly šifrovány bez přerušení po celé cestě nebo digitálně podepisovány. Pouze se šifruje spojení mezi dvěma komunikujícími zařízeními, aby například nebylo možné odposlechnout přihlašovací údaje nebo samotnou zprávu, SMTP server „vidí“, co píšeme.

Zabezpečená komunikace probíhá trochu jinak než jednoduchá nezabezpečená. Mezi protokol SMTP nebo IMAP a protokol TCP se vsouvá bezpečnostní protokol SSL nebo TLS, což mimo jiné znamená použití jiného čísla portu. Takže pokud komunikaci SMTP zabezpečujeme, jde o port 465, v případě protokolu IMAP to je port 993 a u POP3 port 995.

Poštovní protokoly používají adresaci založenou na podobném principu jako HTTP, tedy URL. Jako protokol se uvádí `mail:` a ve zbytku lokátoru je název schránky a jmenná adresa MDA.

Příklad

URL pro elektronickou poštu vypadá takto:

`mail:jan.novak@firma.cz`

Oproti HTTP URL je přehozen význam druhé a třetí části – v druhé části máme název schránky a až třetí část je jmenná adresa. Symbol zavináče `@` se obvykle čte „at“ (jan.novak at firma.cz).

Další informace:

- https://www.siteground.com/tutorials/email/pop3-imap-smtp-ports.htm
- https://www.port25.com/how-to-check-an-smtp-connection-with-a-manual-telnet-session-2/

Poznámka:

Relay servery (Open Mail Relay) se nazývají SMTP servery, které při komunikaci s uživatelem či MDA odesílajícím e-mail neověřují, zda souhlasí jméno a adresa odesílatele. Tyto servery jsou často zneužívány k rozesílání spamu, tedy síť s takovým serverem bývá považována za nedůvěryhodnou.

8.2.4 MIME

MIME (Multipurpose Internet Mail Extensions) je standard pro definování formátů posílaných dat. Nepoužívá vlastní PDU, ale určuje, jakým způsobem mají být reprezentovány různé typy dat v PDU jiných aplikačních protokolů. Tento standard používáme velmi často právě v e-mailech, kde bývá využit pro reprezentaci téže informace v různých formátech (čistý text, HTML apod.) a také pro reprezentaci vložených příloh.

Některé nejznámější MIME typy jsou:

- *text* – pro textové formáty (například čistý text/plain, html, rtf),
- *image* – pro obrázky (bmp, gif, png, jpeg apod.),
- *audio* – pro zvukové soubory (audio/mpeg, audio/mp4, audio/ogg, atd.),
- *video* – pro videosoubory (napříkad h263, h264, audio/mp4),
- *application* – pro soubory aplikací, moduly a datové soubory aplikací, jednoduše binární soubory (například java-vm, json, msword, octet-stream, pdf),
- *multipart* – „multiformát“, který může například indikovat, že následují tatáž data postupně v několika různých formátech (multipart/alternative), případně postupně za sebou více různých objektů, například že je přiložen digitální podpis, atd.

Pokud se použije MIME, je ve zprávě místo jednoduchého textu nejdřív (jedno či více) MIME záhlaví následované MIME obsahem. V MIME záhlaví najdeme verzi MIME, typ obsahu (Content-type), případně kódovací metoda pro následující data a další podle potřeby.

Další informace:

http://www.iana.org/assignments/media-types/media-types.xhtml

8.3 Souborové služby

8.3.1 Protokol FTP

Souborový server (file server) poskytuje službu úložiště dat a unifikovaného přístupu k těmto datům. Klient tedy na souborový server ukládá data (upload), stahuje si data (download), popřípadě data aktualizuje.

Protokol FTP (File Transfer Protocol) slouží ke komunikaci se souborovými servery a má vyhrazené porty 20 a 21, přičemž:

- port 21 je používán pro zasílání řídících informací (příkazy),
- port 20 je používán pro zasílání dat.

Takže FTP server naslouchá především na portech 20 a 21, přesněji naslouchá vždy jen na jednom z nich – standardně na 21 (pro příkazy), a pokud je přes port 21 vyjednán přenos dat, pracuje se pro změnu jen s portem 20. Co se transportních protokolů týče, lze sice používat TCP i UDP, ale prakticky je použitelná jen TCP komunikace, protože mnohé souborové servery vyžadují autentizaci, pro kterou potřebujeme navázat spojení.

Příklad

URL pro souborový server vypadá podobně jako u HTTP: `ftp://fileserver.firma.cz`

Zabezpečená komunikace s FTP serverem probíhá obdobně jako u předchozích protokolů, tedy mezi FTP a TCP nasuneme protokol SSL nebo TLS (označujeme jako FTPS), ale lepší alternativou je přímo použití zabezpečeného protokolu SSH (port 22). SFTP se označuje přenos FTP přes SSH, což je poněkud komplikovanější komunikace.

Protokol FTP se typicky používá ke stahování dat z webu (především multimediálních), protože pro protokol HTTP (který by to taky zvládl) bývá práce s velkými soubory zbytečně obtížná. Dalším typickým použitím je správa vlastních webových stránek – práce se soubory, které tyto webové stránky na serveru tvoří (na serveru se spuštěným HTTP serverem bývá i FTP server).

Zatímco na straně serveru musí běžet FTP server (coby proces, služba, démon), na klientském zařízení potřebujeme *FTP klienta*. FTP klienti bývají součástí webových prohlížečů a správců souborů (například Total Commander nebo Free Commander), nebo můžeme zvolit specializovaný program (FileZilla, český WinSCP apod.).

Abychom mohli daného FTP klienta používat pro přístup ke konkrétnímu serveru (resp. nám vyhrazené části), musíme si obvykle pro tuto komunikaci vytvořit profil (to není nutné, pokud se uživatelé nemusejí autentizovat). V profilu je třeba zadat název serveru (tj. doménové jméno), umístění (plná URL serveru včetně protokolu), číslo protokolu (většinou 21) a autentizační informace (jméno a heslo), případně lze tyto informace zadávat dynamicky vždy při přístupu na daný server.

S FTP serverem komunikujeme pomocí FTP příkazů. Pár nejzákladnějších:

- `open server` – otevře relaci k danému serveru, parametr je název souborového serveru,
- `get soubor` – chceme si stáhnout zadaný soubor (ze serveru k sobě),
- `send soubor` – odesíláme zadaný soubor od sebe na server,
- `!` – tímto příkazem se přepínáme mezi datovým prostorem na našem počítači a datovým prostorem serveru, platí pro pohyb v adresářové struktuře na našem počítači nebo serveru,
- `lcd adresář` – přesun do jiného adresáře buď na našem počítači nebo na serveru, místo působení se přepíná příkazem `!`, používají se stejné způsoby zadávání adresáře jako v příkazovém řádku (včetně `..` pro přesun o úroveň výše).

Další informace:

FTP příkazů je samozřejmě mnohem více. Stručný návod a přehled najdete například na http://www.computerhope.com/issues/ch001246.htm.

Poznámka:

Jednoduchého FTP klienta ve skutečnosti máme k dispozici v každém operačním systému, dokonce i ve Windows. Když na příkazovém řádku zadáme `ftp`, dostaneme se do příkazového prostředí tohoto klienta. Pak lze otevřít relaci s určitým serverem, přičemž jsme dotázáni na jméno a heslo, a po úspěšném přihlášení můžeme zadávat FTP příkazy. Takže začátek relace vypadá takto:

```
ftp
open fileserver.firma.cz
...
```

Pak jsme požádáni o jméno a heslo, následně už můžeme zadávat FTP příkazy. Relaci ukončíme příkazem `quit` nebo `bye`.

8.3.2 Sdílení prostředků v lokální síti

Pro sdílení prostředků v lokálních sítích (často typu peer-to-peer) se kromě FTP může používat *protokol SMB* (Server Message Block), také nazývaný CIFS (Common Internet File System). Je to aplikační protokol poskytující autorizovaný přístup ke zdrojům typu souborů, tiskáren apod. Zprostředkovává přístup k souborovým a tiskovým serverům.

SMB byl původně vyvinut společností IBM ve spolupráci s Microsoftem a dodnes je oblíbený zejména v sítích s Windows servery a klienty, ale existuje také open-source implementace pro jiné operační systémy nazvaná *Samba*.

Protokol definuje komunikaci typu klient-server – žadatel o prostředek je klient a poskytovatel prostředku je server. SMB zprávy se nazývají *bloky* (taky je to v názvu protokolu), jejich formát je stejný v obou směrech komunikace, jen do odpovědi se mohou přidat požadovaná data.

8.4 Přidělování IP adres a protokol DHCP

Každé zařízení v síti potřebuje nějak získat IP adresu. Můžeme ji buď na každém zařízení zadat ručně nebo nechat DHCP server, aby ji připojovaným zařízením přiděloval dynamicky.

IP adresa může být získána těmito způsoby:

- *dynamická alokace* – uživatel se nemusí o nic starat, při připojení zařízení do sítě je tomuto zařízení automaticky přiřazena IP adresa, může být pokaždé jiná,
- *staticky* – uživatel má předem určenou IP adresu, do konfigurace se dostane takto:
  - uživatel ji *ručně* zadá na patřičné místo v konfiguraci síťového rozhraní,
  - *statická alokace*: tato adresa je při připojení zařízení do sítě tomuto zařízení automaticky přiřazena (jako u dynamické alokace, ale adresa je rezervovaná pro toto zařízení).

Nás bude zajímat první případ a z druhého případu možnost statické alokace, což vše probíhá podle *protokolu DHCP* (Dynamic Host Configuration Protocol). Protokol DHCP tedy nabízí mechanismus distribuce konfigurace síťového rozhraní. Klientské zařízení může od DHCP serveru během dynamické nebo statické alokace získat různé informace:

- IP adresu, masku sítě,
- adresu výchozí brány,
- adresu DNS serveru,
- další informace.

Takže rozhodně nejde jen o IP adresu a masku sítě. Klient se taky dozví, kudy vede cesta ze sítě a kdo v síti umí překládat jmenné adresy na IP adresy. Podle potřeby může server distribuovat i další informace.

S přechodem od IPv4 na IPv6 byly vytvořeny nové verze i pro jiné protokoly (zatím jsme se s tím setkali u ICMPv4/v6), totéž potkalo i protokol DHCP. Změny v IP byly prostě tak rozsáhlé, že musel být značně pozměněn i tento protokol.

Starší DHCPv4 je jednodušší a jako svůj nosný protokol směrem k transportní vrstvě využívá protokol Bootstrap (Bootp). Celá struktura je následující: DHCP zpráva se zapouzdří do Bootstrap zprávy (resp. je její součástí) a výsledná zpráva se zapouzdří do UDP segmentu. Číslo portu je 67 na straně serveru a 68 na straně klienta.

DHCPv6 je již komplexnější a nevyužívá protokol Bootstrap, sám vytváří zprávy na aplikační vrstvě. Naproti tomu úzce spolupracuje s protokolem ICMPv6 (ale nikoliv ve smyslu zapouzdřování). DHCPv6 zprávy se také zapouzdřují do UDP segmentu, používají se čísla portů 547 na straně serveru a 546 na straně klienta.

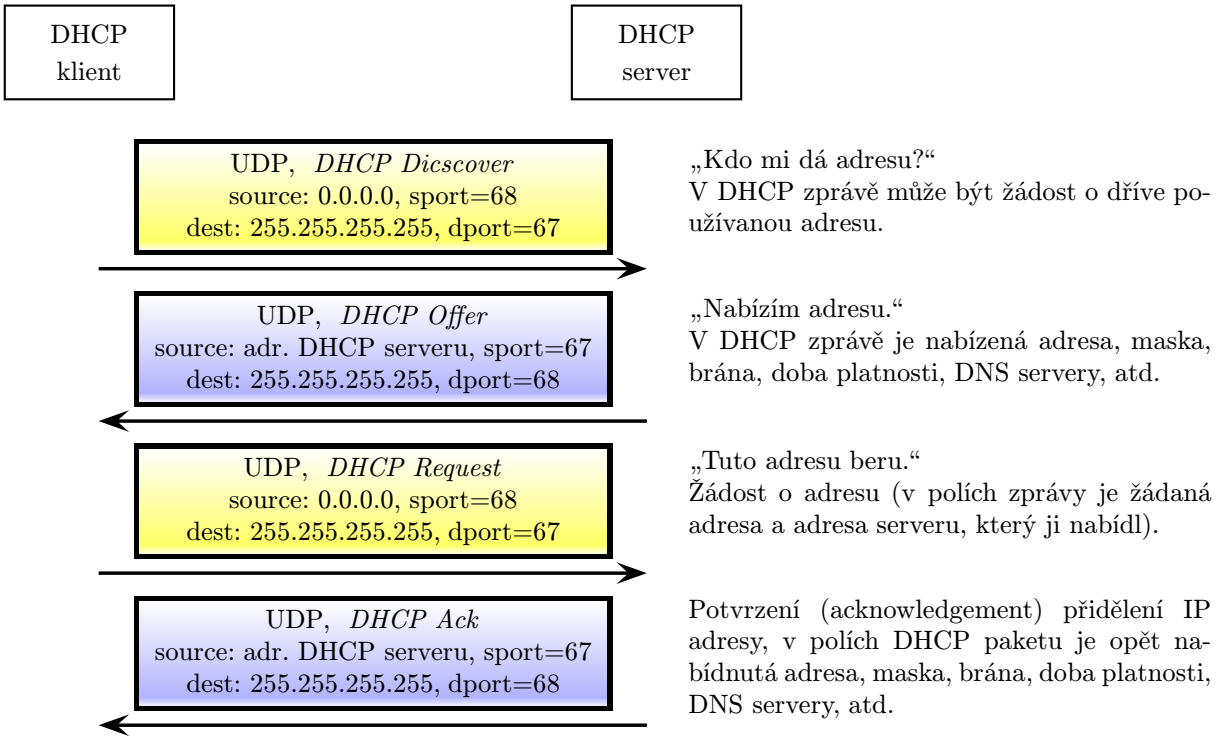
Poznámka:

Všimněte si, že na rozdíl od předchozích aplikačních protokolů používá DHCP klient port z rozsahu „dobře známých“ portů a nikoliv dynamické porty. Uvědomte si, že DHCP sice pracuje na aplikační vrstvě, ale rozhodně nekomunikuje s (různými) aplikacemi a procesy v různých oknech či záložkách – nemůže nastat situace, kdy by dvě různé aplikace chtěly přidělit adresu. Proto nemusí klient využívat dynamické porty, kterými by odlišil jednotlivé aplikace – není co odlišovat.

Na obrázku 8.4 je naznačen postup získání IPv4 adresy. Komunikuje se pouze broadcastově, protože klient ještě nemá přidělenou IP adresu (v obou směrech – klient sice postupně zjistí adresu serveru, ale i tak mu posílá broadcasty). Jednotlivé kroky:

1. *DHCP Discover* (poptávka) – klient zjišťuje, kdo v síti mu může přidělit adresu (nezná adresu DHCP serveru). V žádosti posílá svou MAC adresu a seznam parametrů, které požaduje (Parameter Request List – masku sítě, adresy DNS serverů, adresu routeru/brány apod.).
2. *DHCP Offer* (nabídka) – DHCP server obdržel žádost a v odpovědi posílá nabízenou adresu (nebo jejich rozsah), dobu platnosti adresy (Lease Time), svou vlastní adresu, adresy DNS serverů apod. Pokud je v síti víc DHCP serverů, může klientovi přijít i víc než jedna nabídka. Nabídnutá adresa je na určitou krátkou dobu (například 1 minutu) dočasně rezervována.
3. *DHCP Request* (žádost) – klient přijme nabízenou adresu a zároveň znovu pošle seznam dalších požadovaných parametrů, stejně jako v poptávce. Pokud přišlo víc nabídek, odpoví jen na jednu.
4. *DHCP Ack* (potvrzení) – server klientovi potvrdí přidělení IP adresy a (znovu) pošle hodnoty dalších požadovaných parametrů. Adresa je klientovi definitivně přidělena a DHCP server ji má zaznamenanou v databázi.

   Jestliže server zjistí na síti zprávu DHCP Request, která není určena jemu (ale jinému DHCP serveru), znamená to, že klient si vybral jiného poskytovatele, tedy dočasně rezervovanou adresu uvolní.



Obrázek 8.4: Získání adresy přes DHCPv4

Pokud jde o *znovupřidělení téže adresy* (například tehdy, kdy klient má adresu přidělenou, ale blíží se vypršení doby platnosti – *Lease Time*), probíhají pouze poslední dva kroky, protože klient zná svého poskytovatele adresy (nemusí se poptávat) a nabídka už jednou taky proběhla. Takže jen žádost a potvrzení.

Až po DHCP Ack nemůže klient při prvním přidělování používat skutečnou IP adresu, do té doby má 0.0.0.0 (nedefinovanou). Ovšem při znovupřidělování toto neplatí – klient používá svou dříve přidělenou adresu a komunikace je unicastová (oba uzly se navzájem „znají“).

Při použití DHCPv6 je víc možností jak může komunikace proběhnout, tyto možnosti probereme v následující kapitole. Jedním z rozdílů oproti DHCPv4 je také to, že pro tento protokol je již definován vlastní formát zprávy, nepoužívají se již zprávy protokolu Bootstrap.

8.5 Další typy serverů

Databázový server je server, na kterém běží některý databázový systém (MySQL, PostgreSQL, Oracle, MS SQL apod.).

Pro databázové servery je typické, že uživatel obvykle nekomunikuje přímo s nimi, ale existuje zprostředkovatel – webový server poskytující uživatelské rozhraní k databázovému serveru. Takže uživateli je zobrazena webová stránka, na které zadá, co od databáze vlastně chce, odešle se webovému serveru, ten vše zkontroluje a odešle dotaz do databáze na databázovém serveru. Odpověď jde opět přes webový server, který ji „zformátuje“ do webové stránky, aby byl výstup uživatelsky příjemnější.

Tiskový server je server poskytující tiskové služby (vede tiskové fronty připojených tiskáren, přijímá požadavky od klientů ze sítě a řadí je do front).

Pokud k běžnému počítači máme připojenou tiskárnu, kterou máme nasdílenou do sítě, pak náš počítač taky funguje jako tiskový server. V současné době je tiskový server přímo vestavěn především ve větších síťvých tiskárnách, aby nebylo nutné vedle nich provozovat ještě další „zprostředkující“ zařízení.

Aplikační server je server poskytující do sítě služby určité (síťové) aplikace. Jedná se vlastně o proces (službu, démona) běžící na daném hardwarovém serveru, přičemž tento proces přijímá požadavky ze sítě.

V každém případě je potřeba, aby mezi klientem a aplikačním serverem byl webový server, který bude zprostředkovávat komunikaci. Není to jen kvůli pohodlí uživatelů, ale také z důvodu lepší ochrany aplikačního serveru. Samotné aplikační servery často dokonce ani nemají implementovaný protokol HTTP, ale komunikují (s webovým serverem) pomocí jiných protokolů na jiných než běžných portech.

Za speciální typ aplikačního serveru můžeme považovat například i databázový server, protože na něm běží databázový systém jako speciální proces a mezi klientem a databázovým serverem taky obvykle bývá webový server.

Poznámka:

Stejně funguje taky nám známý systém STAG. Databáze uživatelů, studijních plánů, předmětů, sylabů, hodnocení,... je na databázovém (aplikačním) serveru, ale uživatelé se přímo dostanou jen k webovému rozhraní (portálu), kterému rozhodně neposílají databázové (SQL) dotazy, ale pouze webové formuláře. Až webový server podle formulářů sestaví SQL dotaz, přepošle ho do databáze a v opačném směru pak podle výsledku dotazu sestaví webovou stránku, kterou následně pošle klientovi.

8.6 Vzdálená konfigurace

8.6.1 Telnet

Pokud potřebujeme konfigurovat zařízení, u kterého z nějakého důvodu nemůžeme přímo sedět (je daleko, je hůře dostupné, nemá klávesnici a obrazovku apod.), potřebujeme protokol, který by nám dokázal zprostředkovat vzdálený přístup na dané zařízení v takovém rozsahu, jako když u toho zařízení přímo sedíme. Při správě serveru si obvykle vystačíme s textovým rozhraním (ano, dokonce i Windows server lze z bezpečnostních a výkonových důvodů nainstalovat bez grafického rozhraní – instalace typu „Server Core“), kdežto při správě uživatelských zařízení na dálku se může hodit i přenos grafického rozhraní uživatele.

Protokol Telnet slouží pro interaktivní vzdálený (textový) přístup typu klient-server k zařízením přes síť, a to s možností autentizace (zadáváme jméno a heslo). V reálu provádí *emulaci terminálu* – terminál je specializované zařízení skládající se z obrazovky a klávesnice a připojené například přes síť k serveru. Pokud terminál emulujeme, znamená to, že se náš počítač dokáže chovat jako terminál, třebaže terminálem není.

Vpodstatě se Telnet dá provozovat na jakémkoliv zařízení (klientském i serverovém), ale většinou bývá zablokován či jiným způsobem je znemožněno jeho používání. V čem je problém? Telnet totiž vznikl v době, kdy síť byla bezpečným místem (a taky byla mnohem menší) a nebylo nutné nic šifrovat. Při autentizaci přenáší jméno a heslo v plain textu (čistý text) a kdokoliv, kdo se k síti dostane, si tyto

údaje může odposlechnout.

Dnes se Telnet používá pouze v provozech (vnitřních sítích), o kterých si administrátor myslí, že jsou bezpečné (všimněte si, že tu nepíšu v bezpečných sítích – nic takového jako bezpečná síť totiž ve skutečnosti neexistuje). Logičtější použití je při ladění serverového softwaru, přičemž jsou klient i server odděleni od sítě a technicky se do komunikace nikdo nemůže dostat.

Telnet používá spojovanou komunikaci (protokol TCP) přes port 23 (klient používá dynamické porty). Nejdřív je navázána TCP komunikace na portu serveru 23, dále pokud je vyžadována autentizace, jsou poslány a potvrzeny autentizační údaje, a dál už záleží, co konkrétně chceme dělat.

Poznámka:

Pokud se nám povede zpřístupnit Telnet na našem počítači, můžeme začít Telnet relaci takto:

```
telnet
open adresa.serveru port
...
```

Jako adresu serveru zadáme ten server, na který chceme vzdáleně (i přes Internet) přistupovat, číslo portu zadáváme jen tehdy, když má být jiné než 23. Například pokud chceme přistupovat k webovému serveru tím způsobem, že v příkazech budeme posílat části HTTP zprávy, použijeme port 80.

Následně jsme požádáni o jméno a heslo (pokud je dotyčný server chráněn autentizací) a pak už můžeme zadávat příkazy.

V UNIXových serverech je naprosto běžné, že se k nim dá přistupovat vzdáleně formou emulace terminálu (ať už přes Telnet nebo některý bezpečnější protokol), takto můžeme zadávat prakticky kterýkoliv příkaz tak, jako bychom seděli přímo u dotyčného zařízení. Prostě se nám zobrazí prompt a my zadáváme příkaz. Ve Windows se tento vzdálený přístup poněkud omezenější, protože na rozdíl od UNIXových systémů existuje mnoho úloh, které v textovém režimu prostě nelze provést, a navíc Windows vzdálený přístup poněkud hůře zvládají.

Často (především při testování a ladění) se využívá toho, že mnoho aplikačních protokolů je textově orientovaných (ne binárních) a přes telnet tak můžeme posílat i jejich zprávy. Dá se to tak dělat například s protokolem HTTP, SMTP a dalšími.

Příklad

Pokud chceme přes Telnet otestovat webový server, postupujeme takto:

```
telnet www.server.cz 80
GET /index.html HTTP/1.1
```

Nejdřív jsme navázali spojení se zadaným serverem, a to na portu 80 (protože chceme poslat něco, co má být HTTP zprávou). Webové servery většinou nevyžadují autentizaci, tedy nebudou vyžadovány přístupové údaje. Dalším příkazem odešleme HTTP požadavek typu GET (pozor, musí být velkými písmeny, jinak bude hlášena chyba), žádáme o poslání souboru `index.html` z kořenového adresáře serveru, taky se přidává informace o verzi protokolu HTTP. V odpovědi se nám vypíše HTTP záhlaví a taky tělo zprávy (tedy HTML kód).

Podobně se dá komunikovat například i s SMTP serverem na portu 25, dokonce takto můžeme

odeslat e-mail. Co vše se děje, když chceme odeslat e-mail?

- klient se připojí k SMTP serveru na portu 25 (třeba přes telnet nebo ssh),
- sdělí SMTP serveru, že chce poslat e-mail, dále zadá svou adresu a adresu příjemce,
- zadá případně další části záhlaví, dále text e-mailu.

Následuje ukázka jednoduché komunikace s SMTP serverem (na kterém běží Sendmail), kdy chceme odeslat e-mail obsahující předmět a tělo zprávy ve formě čistého textu.

Příklad

Navážeme spojení se serverem (zde to je pomocí telnetu), a to na portu 25, čímž se napojíme na SMTP. Červeně jsou zbarveny odpovědi serveru.

- `telnet adresa.smtp.serveru.com 25`
  `Trying ip.adresa.serveru.`
  `Escape character is '^]'.`
- `220 adresa.smtp.serveru.com ESMTP Sendmail 8.10.0/8.10.0 ready; datum čas`
- `helo moje.identifikace`
  `250 adresa.smtp.serveru Hello moje.identifikace [moje.ip.adresa], pleased to meet you`
- `mail from: moje@adresa`
  `250 2.1.0 moje@adresa... Sender ok`
- `rcpt to: adresa@prijemce`
  `250 2.1.5 adresa@prijemce... Recipient ok`
- `data`
  `354 Enter mail, end with "."on a line by itself`
- ```
  From: Moje Jmeno <moje@adresa>
  To: Jmeno Adresata <adresa@prijemce>
  Subject: Predmet e-mailu

  Tady napisu telo zpravy, prazdny radek pred textem zpravy je nutny.
  xxxxxx

  .
  ```
  `250 Message accepted for delivery`
- `quit`
  `221 2.0.0 staff.uiuc.edu closing connection`

Jak vidíme, pro odeslání e-mailu vlastně ani není třeba žádný klient nebo webové rozhraní, pokud ovšem dokážeme přímo komunikovat s SMTP serverem. Taky záleží na tom, do jaké míry server ověřuje odesílatele (resp. uživatele, od kterého přijímá zprávu k odeslání).

Další informace:

- http://www.earthinfo.org/example-smtp-conversation/
- http://www.anta.net/misc/telnet-troubleshooting/smtp.shtml
- https://www.cs.cf.ac.uk/Dave/PERL/node175.html


  ```

8.6.2 SSH

SSH (Secure Shell) je bezpečnější variantou protokolu Telnet, ale kromě samotného zabezpečení komunikace má oproti Telnetu i další přídavnou funkčnost. Slouží k zabezpečenému přístupu ke vzdálenému zařízení, a to v různých směrech. Kromě toho, že umožňuje bezpečnou konfiguraci, plní taky podobnou (ale zabezpečenou) úlohu jako FTP (přenos souborů), dovoluje monitorovat procesy a zdroje na vzdáleném systému, vytvářet šifrované VPN tunely (dlouhodobé zabezpečené spojení), atd. V současné době se používá SSH verze 2 vydaná roku 2006 sdružením IETF (RFC 4254).

Jedná se o komunikaci typu klient-server přes TCP (navazuje se spojení), na straně serveru je port 22. Vzhledem k tomu, že zde jde především o bezpečnost, je lepší na serveru nakonfigurovat jiný port než 22. Postup této konfigurace záleží na konkrétním produktu, obvykle jde o změnu v některém konfiguračním souboru. Ovšem pokud na serveru změníme číslo portu pro SSH, klient musí při navazování spojení toto číslo použít (a tedy být o něm informován).

Nejznámější implementací protokolu SSH je open-source projekt *OpenSSH*. Pro tento produkt existuje klientská i serverová varianta a na UNIXových systémech včetně Linuxu a MacOS X již obvykle bývá nainstalován. Pro Windows existuje klientská varianta a slouží pro přístup k UNIXovým serverům, a v poslední době se dokonce objevila i serverová implementace pro Windows.

Pro Windows máme k dispozici i program *PuTTY*, který však existuje pouze v klientské variantě. Opět se používá pro vzdálený přístup k UNIXovým serverům.

SSH konverzace vypadá takto:

- Nejdřív je navázáno TCP spojení, začíná klient.
- SSH klient i server se „představí“ (vzájemně si sdělí svou verzi), začíná server.
- Server sdělí klientovi, které šifrovací algoritmy zvládá, klient si pak v odpovědi mezi nimi vybere.
- Následuje bezpečnostní procedura – dojde k výměně šifrovacího řetězce atd., podle zvoleného algoritmu. Účelem je zajistit, aby se v následující komunikaci nevmísil do spojení někdo, kdo by mohl odposlechnout provoz.
- Po dokončení bezpečnostní procedury je již další provoz šifrován, včetně případné autentizace, pokud je vyžadována.

Když přes SSH s některým serverem navazujeme spojení poprvé (ještě jsme s ním z našeho zařízení nekomunikovali), posílá se (zatím nezabezpečeným spojem) veřejný klíč a zobrazí se na displeji. Pokud veřejný klíč serveru známe (máme „odjinud“), můžeme porovnat a zkontrolovat.

Jestliže jsme už z našeho zařízení s daným serverem komunikovali, máme u sebe veřejný klíč serveru uložen z předchozích relací, SSH tedy od serveru převezme posílaný veřejný klíč a srovná s uloženým. Teď nastane jedna ze dvou situací:

- Veřejný klíč, který nám server poslal, souhlasí s tím, který máme uložený z předchozích relací (tj. jeho veřejný klíč se nezměnil); pak jsme běžně požádáni o přihlašovací údaje a relace se rozběhne.
- Veřejný klíč zaslaný serverem je jiný než ten, který máme k názvu tohoto serveru uložen; to je způsobeno buď tím, že server teď používá jiný pár klíčů než v minulosti, nebo tím, že probíhá útok typu man-in-the-middle. SSH nás informuje, a pokud víme, že se nejedná o útok, můžeme aktualizovat uložený veřejný klíč serveru.

Další informace:

- https://www.ietf.org/rfc/rfc4254.txt
- http://www.dsl.cz/jak-na-to/jak-na-ssh
- http://www.debianhelp.co.uk/ssh.htm

8.7 Přehled protokolů a portů

V této kapitole jsme se dozvěděli, že aplikační protokoly využívají protokol TCP (pro spojovanou potvrzovanou komunikaci) nebo UDP (pro nespojovanou nepotvrzovanou, tedy datagramovou službu), což znamená, že jejich zprávy jsou zapouzdřovány do TCP nebo UDP segmentů a v případě TCP je také navazováno spojení.

V TCP a UDP segmentech se v záhlaví uvádí číslo portu, které na straně serveru určuje konkrétní službu, se kterou se komunikuje (taky může znamenat typ zapouzdřené zprávy, ale ne vždy uvnitř segmentu něco je), na straně klienta to bývá konkrétní aplikace či její část. Ovšem neznamená to, že ve všech segmentech obsahujících SDU od HTTP je vždy serverový port nastaven na 80 apod. – také na serverové straně mohou být používána jiná čísla portů, ovšem klient musí vědět, jaké číslo portu použít.

V tabulce 8.1 najdete přehled některých aplikačních protokolů a pro ně typických portů. K některým z těchto protokolů se dostaneme v následujících kapitolách.

HTTP(S)	SMTP	IMAP	FTP	Telnet	SSH	DNS	DHCP	SNMP
80	**25**	143					**67, 68**	161, 162
443	465	993	**20, 21**	**23**	**22**	**53**	547, 546	10 161, 10 162
TCP							UDP	

Tabulka 8.1: TCP a UDP porty s protokoly

Kapitola 9

Decentralizované a distribuované systémy

V této kapitole se po probrání pojmů budeme podrobněji zabývat některými síťovými technologiemi, které jsou v principu decentralizované nebo distribuované.

9.1 Pojmy

9.1.1 Typy systémů

Centralizovaný systém je systém s jedinou centrální řídicí jednotkou. Komunikace probíhá obvykle ve formě klient-server, kde server je právě ta centrální řídicí jednotka.

V oblasti počítačových sítí můžeme k centralizovaným architekturám zařadit

- *protokol RADIUS*, kde centralizovanou jednotkou je RADIUS server, komunikující se svými klienty (například access pointy – přístupovými body bezdrátové sítě),
- *protokol CMIP* (Common Management Information Protocol), což je protokol z ISO/OSI určený pro správu sítě (obdoba SNMP).

Decentralizovaný systém má více řídicích jednotek (serverových/správních uzlů), víceméně rovnocenných (s podobnými funkcemi). Účelem (a výhodou oproti centralizovaným systémům) je zvýšení robustnosti sítě (tutéž funkci poskytuje více „centrálních“ – serverových – uzlů, klientské uzly mohou komunikovat s kterýmkoliv uzlem poskytujícím žádanou službu). Je možné snadno vyřešit situaci, kdy jeden ze serverových uzlů přestane službu poskytovat (například je odpojen od sítě).

Nevýhodou je náročnější implementace, protože serverové uzly musejí být synchronizovány (především jejich databáze).

V oblasti počítačových sítí se s decentralizovanou architekturou setkáme například

- mnohé *směrovací protokoly* jsou decentralizované,
- *protokol SIP* (Session Initiation Protocol), který se používá například u VoIP (obecně kdekoliv, kde je nutné navazovat relace).

Samotný Internet byl původně decentralizovaný a do značné míry to platí i dnes. Ale plná decentralizovanost by znamenala zachování běžného chodu systému (třeba i s určitým zpomalením nebo neposkytováním některých služeb, které mohou „počkat“, nejsou časově kritické) i po odpojení naprosté většiny serverových uzlů, což Internet nesplňuje.

Distribuovaný systém je takový systém, jehož funkce jsou distribuovány (rozděleny) na různé uzly sítě se zachováním několika důležitých vlastností.

Především jde o vlastnost *transparentnost*. Například přístupová transparentnost znamená, že k lokálním i vzdáleným prostředkům se přistupuje stejným způsobem, zde přes protokoly. Migrační transparentnost znamená možnost přesouvání poskytovaných služeb mezi různými uzly sítě bez výraznějšího vlivu na výkonnost sítě.

Další důležitou vlastností distribuovaného systému je *flexibilita*. Tato vlastnost znamená přizpůsobivost systému změnám prostředí (včetně poruch a výpadků částí systému). To vylučuje jakoukoliv centralizaci rozhodování, každý uzel sítě musí být ve své činnosti co nejvíc samostatný a služby ostatních uzlů vyžaduje se zachováním transparentnosti.

Flexibilní systém může být jakkoliv rozšiřován (téměř – jsou technické hranice, které se jen velmi nesnadno překračují) nebo naopak omezován, funkce odstraňovaného uzlu může plnit jiný uzel.

S distribuovaností se setkáváme u některých směrovacích protokolů a dále například v systému DNS (obecně u doménových služeb).

9.1.2 Distribuované systémy

Distribuovaný systém může být určen architekturou *klient-server*, kdy je jednoznačně určeno, které uzly jsou serverové (poskytují služby, implementují funkce) a které jsou klientské (využívají služeb serverů). Jiný způsob zavedení distribuovaného systému je *integrovaná architektura* (symetrický systém), kde každý uzel může být serverem i klientem, podle požadavků probíhající komunikace.

Podle způsobu využívání paměti můžeme distribuované systémy rozdělit na

- *multiprocesory* (multiprocessors) – uzly mají společnou paměť (každý má svou vlastní paměť a dále existuje paměť sdílená, která může být také distribuována ve více uzlech),
- *multipočítače* (multicomputers) – uzly nesdílejí žádnou paměť, adresové prostory jsou zcela oddělené, tento koncept je u počítačových sítí obecně výhodnější.

Podle způsobu propojení (platí pro oba výše zmíněné typy) rozlišujeme dva typy architektur:

- *sběrnicová architektura* (bus) – využívá jediné sdílené médium (obvykle kabel), signál se šíří po celém médiu, například kabelová televize,
- *přepínačová architektura* (switch) – signál je přenášen vždy mezi dvěma konkrétními uzly (až na výjimky jako je broadcast a multicast), může jít o různé topologie:
  - hierarchická stromová struktura, mesh (varianty), kruh, atd.,
  - mřížka – uzel je propojen se svými nejbližšími sousedy, při ideálním uspořádání jsou ke každému uzlu kromě okrajových 4 sousední uzly,
  - hyperkrychle.

Hyperkrychle je n-rozměrná krychle (v obvyklém případě je $n = 4$) s uzly uspořádanými do více rozměrů (víceméně hierarchicky do linií, rovin, skupin, atd.). Uzel je přímo propojen se dvěma uzly pro každý podporovaný rozměr, tedy je přímo propojen s celkem $n * 2$ sousedy (u čtyřrozměrné krychle s 8 sousedy). Celá struktura je z principu acyklická. Mřížku lze brát jako speciální případ (hyper)krychle pro $n = 2$.

Se sběrnicovou architekturou se už v distribuovaných sítích moc nesetkáváme (až na přenosy probíhající po koaxiálu). V praxi je zřejmě nejběžnější propojení multipočítačů v hierarchické struktuře.

Může dojít trochu ke „zmatení pojmů“, protože v přepínačové architektuře nemusíme jako mezilehlé prvky nutně používat přepínače.

9.2 Bridging, switching, routing

Nejdřív se krátce podíváme na pár pojmů, které souvisejí s distribuovaným zpracováním dat (zde spíše přenosem) v počítačových sítích, přičemž jsme se těmito technologiemi trochu zabývali už v předchozích kapitolách.

✎ **Pojem bridging** označuje technologie související s mosty (bridge). Zahrnuje především to, co najdeme v standardu IEEE 802.1, včetně dříve diskutovaného protokolu STP.

Mosty (obecně jakákoliv zařízení na L2) mohou pracovat v jednom z těchto režimů – co se týče práce s informacemi o cestě rámce:

- *Source-route bridging* – odesílající uzel napevno určí seznam mostů (obecně mezilehlých zařízení L2), přes které má rámec projít (buď se použije pro vytvoření obdoby okruhu, nebo se uloží do záhlaví rámce).
- *Transparent bridging* – odesílatel pouze určí fyzickou adresu cíle, mosty si vedou a dynamicky udržují a dolňují přepínací tabulku, pomocí které stanovují svůj úsek cesty vedoucí k danému cíli.

Je zřejmé, že v Ethernetu se používá transparent bridging. Source-route bridging se používal například u sítě Token-Ring, s jeho obdobou se setekáváme ve WAN sítích (tam jsme obvykle taky na vrstvě L2). Při source-route bridging je tedy nutné dodávat informace pro správné určování adres a portů, kdežto při transparent bridging musí v síti existovat mechanismus transportu fyzických adres na mosty.

Jak víme, u Ethernetu je tento problém řešen velice jednoduše – pokud je na portu detekován rámec, jehož zdrojovou adresu neznáme (což pro samotné přepnutí rámce není nutné), poznamenáme si do tabulky tuto zdrojovou adresu a označení portu, ze kterého rámec přišel, protože někde za tímto portem toto zařízení velice pravděpodobně bude připojeno.

✎ **Switching** se nevztahuje k některému standardu, ale spíše ke konkrétním technologiím, které najdeme na přepínačích, vlastně je to jakási nástavba bridgingu. Rozlišujeme tyto typy přepínání:

- *Store-and-Forward* (ulož a pošli) – příchozí rámec je nejdřív celý přijat a uložen do vyrovnávací paměti přepínače, až po přijetí celého rámce je přečteno záhlaví a určen výstupní port pro odeslání rámce. Dokáže odchytávat chybné rámce, ale pomalý, vhodné pro sítě s vyšší chybovostí nebo switche s výkonnějším hardwarem.
- *Cut-Through* (průběžné zpracování, také on-the-fly) – příchozí rámec se začíná odesílat průběžně ještě během svého příjmu, hned po načtení informací ze záhlaví potřebných k určení výstupního portu. Velmi rychlé přepínání, menší potřeba cache, ale nedokáže odchytávat chybné rámce.
- *Fragment Free* (s omezením malých rámců) – něco mezi prvními dvěma metodami. Zpracovávání rámce začne po načtení jeho prvních 64 oktetů (tj. záhlaví a alespoň část dat). Účelem je odhalení alespoň jednoho typu chybných rámců: nepovolených příliš krátkých rámců.

Co je lepší? To je otázka. Záleží na vytíženosti sítě a výkonu switchů. Na zařízeních určených pro korporátní sféru je obvykle možné zvolený mód nastavit. Některé funkce v síti vyžadují určitý konkrétní mód, například pokud potřebujeme kvalitu služby (QoS), mělo by být na switchích nastaveno store-and-forward.

Příklad

Například na některých switchích Cisco je jako výchozí nastaven mód cut-through. Pokud chceme nastavit store-and-forward, zadáme:

```
switch# switching-mode store-forward
```

Pokud bychom chtěli nastavit zpět cut-through, předřadíme před tento příkaz `no`:

```
switch# no switching-mode store-forward
```

Jiné switche od Cisca mají napevno nastaven mód store-and-forward a nelze to změnit. Pokud v příslušném módu zadáme otazník, získáme seznam podporovaných příkazů. Pokud tam je příkaz `switching-mode`, napíšeme ho a místo parametru dáme otazník. Tak zjistíme, který mód lze nastavit.

Obecně platí, že mosty a přepínače odesílají unicast provoz (se známou cílovou adresou) pouze na jeden port, kdežto broadcast a unicast s neznámou cílovou adresou odesílají na všechny porty kromě toho, ze kterého rámec přišel.

Routing se týká činnosti routerů a dalších zařízení pracujících na vrstvě L3, včetně switchů s funkcionalitou vyšších vrstev. Víme už, jak vypadají směrovací tabulky; v rámci routingu je třeba zajistit distribuci aktuálního obsahu těchto tabulek a tyto tabulky používat při směrování paketů.

Účelem je, aby směrovací tabulky byly vždy aktuální, a proces uvádění tabulek do aktuálního (konzistentního) stavu se nazývá *konvergence sítě*. Pokud se v síti používá dynamické směrování (tj. výměna směrovacích informací pomocí protokolů), je proces konvergence sítě jedním z nejdůležitějších parametrů daného protokolu – konvergence by měla být rychlá a zároveň by neměla moc vytěžovat síť.

Routingem se budeme zabývat podrobněji hned v další sekci.

9.3 Směrování

9.3.1 Jak směrujeme

Směrování (routing) probíhá na vrstvě L3 a vlastně znamená proces určení cesty do konkrétních IP sítí. Tato informace je dále využívána k určení cesty pro konkrétní IP paket.

Definice (Směrovací tabulka)

Na routeru a všech dalších zařízeních vrstvy L3 je vedena minimálně jedna *směrovací tabulka* (routing table), ve které máme pro každý záznam především tyto informace:

- adresa (pod)sítě a maska nebo délka prefixu,
- určení síťového rozhraní pro odeslání,
- metrika (kvalita určené cesty), případně další informace.

Součástí směrovací tabulky také bývá informace o bráně (také se říká „gateway of the last resort“), tedy určení směru, na který se má posílat vše, pro co nemáme určenou cestu jinde ve směrovací tabulce.

Tuto tabulku je třeba nějak naplnit a dále aktualizovat podle momentální situace (některá cesta přestane být platná, nebo naopak jiná cesta je zpřístupněna, mění se metriky či přidáváme novou podsíť). To se buď provádí ručně (staticky) nebo to můžeme nechat na *směrovacích protokolech* (dynamická údržba).

Z toho vyplývá, že směrování může být

- *statické* (static routing) – správce sítě ručně vkládá záznamy do směrovacích tabulek,
- *dynamické* (dynamic routing) – na směrovačích v síti běží směrovací protokoly, které si udržují přehled o topologii sítě (jak je co propojeno) a aktualizují směrovací tabulky.

V praxi se obojí kombinuje, tedy některé (blízké či nějakým způsobem citlivé) cesty vkládáme ručně a zbylé necháme doplňovat a aktualizovat dynamicky.

Údaje ze statického směrování jsou považovány za důvěryhodnější, proto jsou obvykle upřednostňovány před údaji získanými dynamicky pomocí protokolů.

Metrika (kvalita cesty) je číslo, které bylo vypočteno podle určitých kritérií. Každý směrovací protokol používá trochu jiná kritéria a taky trochu jiným způsobem z nich metriku počítá. Jako kritéria se například mohou použít:

- počet routerů na cestě,
- propustnost cesty (například zda jde o Fast Ethernet nebo Gigabit Ethernet),
- latence (zpoždění v ms),
- spolehlivost cesty (nízká chybovost, málo zahozených paketů apod.),
- vytíženost cesty,
- maximální hodnota MTU po cestě (tj. maximální velikost paketu, který lze poslat),
- důvěryhodnost zdroje, ze kterého byla získána informace o cestě, atd.

Poznámka:

Obvykle platí: čím menší číslo metriky, tím lepší cesta. Přímo připojené adresy mají nejnižší metriku.

Současné routery mají své záznamy určeny *adresou sítě* a *délkou prefixu* (nebo maskou).

Kdy a jak se pracuje se směrovací tabulkou: pokud router přijme paket s cílovou IP adresou, potřebuje zjistit, na který port má tento paket přeposlat. Prochází směrovací tabulku a zjišťuje, kterému záznamu v tabulce tato adresa odpovídá, tedy zjišťuje, zda pro daný záznam adresa patří do podsítě v záznamu uvedené.

Pokud v tabulce existují dvě cesty k témuž cíli (tj. ve dvou záznamech je adresa podsítě, do které se dá zařadit adresa z paketu), obvykle se upřednostní ta, která má delší prefix (tj. je přesnější, „most specific“). Například pokud máme údaje

```
10.160.0.0/12
10.160.0.0/17
```

a adresa konkrétního zařízení, pro kterou hledáme údaj v tabulce, odpovídá oběma, druhý údaj bude upřednostněn. Jde o dva různé záznamy, protože se liší v délce prefixu, třebaže adresa v obou případech vypadá jako stejná. Druhá adresa zřejmě bude podsítí sítě určené první adresou (nebo hlouběji v hierarchii).

Pokud ve směrovací tabulce není nalezena žádná cesta, je buď paket předán *bráně* (geteway), protože paket zřejmě patří do jiné sítě, nebo zahozen (když žádná brána není dostupná).

Rozlišujeme protokoly *směrovatelné* (routed) a *směrovací* (routing). Směrovatelné protokoly (například IP) pracují na síťové vrstvě, ale vlastní směrování neprovádějí, to je záležitostí směrovacích protokolů (pracujících na L3 nebo jiné vrstvě).

9.3.2 Autonomní systém

Autonomní systém je skupina směrovačů patřících pod správu téže organizace. Každý autonomní systém je identifikován 16bitovým *číslem ASN*.

- *Vnitřní směrovací protokoly* (interior routing protocols) dokážou směrovat jen uvnitř vlastního autonomního systému (cestu ven neznají, všechno „cizí“ posílají prostě na jedinou bránu), nerozlišují jiné autonomní systémy (rozlišují jen „vlastní“ a „cizí“). Ve svých datových strukturách nemají pole pro evidenci autonomního systému.
- *Vnější směrovací protokoly* (exterior routing protocols) zvládají směrování i mezi různými autonomními systémy (dokážou pracovat s číslem autonomního systému, ve směrovací tabulce máme v každém záznamu i číslo směrovací oblasti), dokážou rozlišovat mezi různými „cizími“ cestami.

Poznámka:

Musí mít každá organizace svůj autonomní systém a své číslo ASN? Ne. Menším firmám stačí být součástí autonomního systému svého ISP, a jeho číslo ASN obvykle ani neznají. Pokud firma používá pouze vnitřní směrovací protokoly, není důvod mít vlastní ASN (hlavně za to platit).

Jednoprotokolový směrovač dokáže pracovat jen s jedním směrovacím protokolem, *multiprotokolový* směrovač dokáže kombinovat údaje od více směrovacích protokolů.

V multiprotokolovém směrovači je třeba navíc rozhodnout mezi cestami v různých tabulkách, které jsou vytvářeny podle různých kritérií a tedy tak jak jsou, nebývají navzájem porovnatelné. Z toho důvodu byla pro každý protokol stanovena hodnotící konstanta nazvaná *administrativní vzdálenost* (administrative distance, AD) umožňující porovnat více cest k témuž cíli ze směrovacích tabulek různých protokolů. Toto číslo reálně znamená důvěryhodnost zdroje (jak moc důvěřujeme, že tento zdroj dodává „dobré“ cesty).

Administrativní vzdálenosti jsou v tabulce 9.1. Všimněte si rozdílů u protokolů, které mohou pracovat jako vnitřní i vnější (například BGP). Na routerech se ve skutečnosti dá AD konfigurovat.

Pokud do daného cíle dokáže směrovat více protokolů, bude vybrána cesta toho směrovacího protokolu, který má menší číslo administrativní vzdálenosti. Jak vidíme, připojené rozhraní je nejdůvěryhodnější, protože router si je do směrovací tabulky řadí sám (sám sobě důvěřuje nejvíce) a cesta tímto směrem je nejkratší. Následují staticky přidané záznamy. Vnitřní EIGRP má AD=90, OSPF má AD=110, takže u EIGRP se předpokládá mírně lepší schopnost určování cest (ale na druhou stranu, EIGRP většinou umí jen zařízení od Cisca, takže se dá použít jen v homogenní síti).

Cesta zjištěná v rámci běžného směrování je *interní*, cesta zjištěná vně směrovacího procesu (například z jiné autonomní oblasti nebo od jiného protokolu) je *externí*. Většina protokolů upřednostňuje interní cesty, ale například u BGP je to naopak a jiné jejich spolehlivost nerozlišují (například OSPF).

9.3.3 Směrovací algoritmy

Každý směrovací protokol se řídí nějakým *směrovacím algoritmem*. Obvykle se jedná o jeden z následujících dvou algoritmů (případně modifikaci):

- algoritmus vektoru vzdáleností,
- algoritmus stavu spoje.

Typ cesty	Administrativní vzdálenost
Připojené rozhraní	0
Statické směrování	1
EIGRP summary route (souhrnné cesty pro skupinu sítí)	5
External BGP	20
Internal EIGRP	90
IGRP	100
OSPF	110
IS-IS	115
RIP	120
EGP	140
ODR (On Demand Routing)	160
External EIGRP	170
NHRP	250
Internal BGP	200
jiný (neznámý)	255

Tabulka 9.1: Administrativní vzdálenosti směrovacích protokolů

Algoritmus vektoru vzdáleností (Distance-Vector Algorithm) určuje metriku cesty především podle vzdálenosti (v jednodušším případě podle počtu routerů na cestě). Pro každou síť evidovanou ve směrovací tabulce máme tyto údaje: vzdálenost a vektor. Vektor určuje směr k dané síti, tj. buď přes které síťové rozhraní nebo přes kterého souseda vede cesta. Pozice cíle je určena pouze vektorem, zařízení nemají přehled o topologii sítě routerů.

Router pracující podle tohoto algoritmu má nejdřív v tabulce cesty do připojených sítí, tj. cesty k přímým sousedům, a od nich postupně dostává informace o jiných routerech a jejich podsítích. Pokud od svého souseda dostane paket (update) s informací o (pod)síti s konkrétní metrikou (podle toho souseda), přidá si tuto (pod)síť do směrovací tabulky:

- adresu (pod)sítě zjistil od souseda,
- rozhraním bude adresa či port dotyčného souseda (protože přes něj je ta síť dostupná),
- jako metriku použije o něco vyšší číslo než jaké mu poslal soused, v jednodušším případě číslo o 1 větší než jaká platí pro souseda (tj. k cestě od souseda k síti je třeba započítat i úsek od souseda ke mně).

Pro tento algoritmus je typické, že updaty (informace o aktuálním stavu směrování) se sousedům posílají v pravidelných intervalech.

Jak vidíme, může tady vzniknout podobný problém jako v síti switchů, kdy jsme museli pro odstranění smyček použít protokol STP. Smyčky mohou vzniknout i zde (máme síť routerů používajících daných algoritmus/protokol), přičemž důsledky by mohly být podobné. Představme si například situaci, kdy tatáž síť je dostupná přes dvě různé cesty. Router dostává informace o síti střídavě ze dvou směrů, což nesvědčí stabilitě směrovacích tabulek. Stabilita fungování algoritmu se vylepšuje různými způsoby, následuje jejich popis.

Při použití tohoto algoritmu může být definován *maximální počet přeskoků*, který určuje maximální velikost sítě (v počtu routerů na cestě) – obvykle 15 nebo 255. Pokud router dostane informaci o síti s metrikou vyšší než tato hranice (například u RIP s hodnotou 16), pak takovou síť považuje za nedostupnou.

Dále je určen *hold-down timer*, většinou nastavený na trojnásobek intervalu zasílání updatů sousedům. Pokud router obdrží informaci o nedostupnosti některé sítě, pak po dobu určenou hold-down timerem ignoruje zprávy o cestě do této sítě, považuje je za potenciálně chybné.

Poison reverse (otrávení cesty zpět) je oznámení směrovače o své nedostupnosti. Tento směrovač odešle svým sousedům aktualizaci cesty k sobě s hodnotou metriky max(TTL) a sousedi si k tomu ještě přičtou 1, tj. například u protokolu RIP to bude hodnota 16 nebo u jiných protokolů hodnota 256. Metrika větší než maximální stanovená označuje „nedosažitelnou“ cestu a tedy na port k odpojovanému směrovači nebudou zasílány žádné pakety.

Další možnost snížení zatížení sítě a rizika zacyklení je postup *split horizon* (rozložený horizont). Směrovač neodesílá jinému směrovači celou svou tabulku, ale jen ty záznamy, které nevedou k tomuto směrovači (není třeba informovat jiný směrovač o tom, o čem původně informoval on). Takto se také snižuje nebezpečí zasílání chybných informací.

Protokoly využívající algoritmus vektorů vzdáleností:

- RIP (Routing Information Protocol) varianty pro IP, IPX, atd.,
- IGRP (Interior Gateway Protocol) pro IP,
- EIGRP, nicméně tento algoritmus obohacuje o některé prvky typické spíše pro algoritmus stavu spoje, jeho metrika je považována za velmi dobrou.

Poznámka:

Typickou vlastností protokolů implementujících algoritmus vektoru vzdáleností je, že přímo „znají“ pouze své sousedy a o topologii sítě nemají žádnou informaci. Pouze vědí, že ke konkrétní (pod)síti se dá dostat přes konkrétního souseda a cesta trvá konkrétní počet přeskoků.

Algoritmus stavu spoje (Link-state Algorithm, algoritmus nejkratší cesty) vyžaduje budování *topologické databáze sítě* (jakési mapy sítě). Router s protokolem podle tohoto algoritmu si vede tři tabulky:

- tabulku sousedů, kterou si vytvoří jako první hned po zapnutí,
- tabulku reprezentující topologickou databázi sítě, tu si po zapnutí vytvoří postupně podle aktualizačních informací ostatních routerů,
- směrovací tabulku, kterou si vytvoří podle topologické databáze.

Každý router si neustále hlídá kvalitu (stav) spojů vedoucích k sousedům (to se týká první zmíněné tabulky). Pokud zjistí změnu (některý spoj/soused se stane nedostupným nebo změní některý parametr), informuje o tom své okolí. Routery, které dostanou tuto informaci, si změnu zanesou do topologické databáze (druhé tabulky) a podle ní propočítají novou cestu k cílům (aktualizují směrovací tabulku).

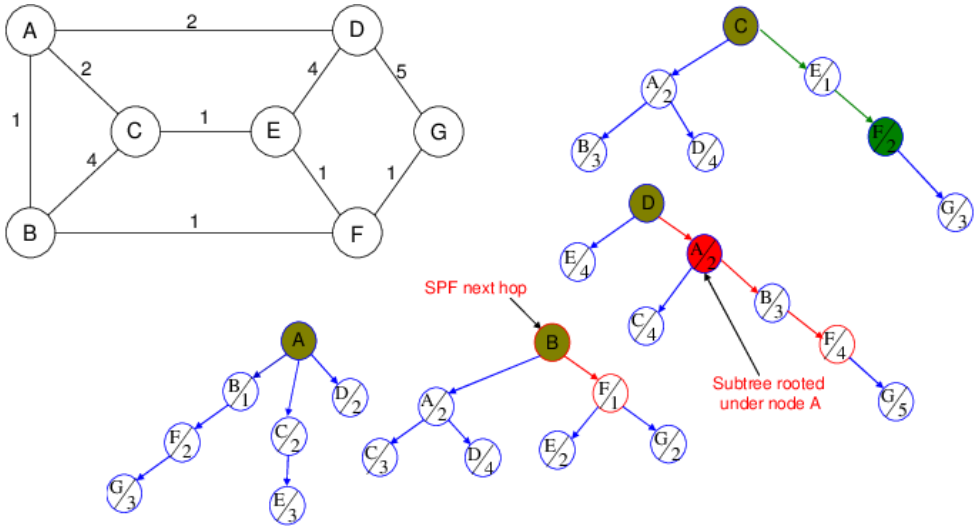
Pro protokoly podle algoritmu stavu spoje je typické, že mají přehled o topologii sítě, méně zahlcují síť (posílají se jen změny ke konkrétním spojům, a většinou jen při změnách topologie) a doba

konvengence je výrazně kratší (aktualizace se jen přeposílají, změnu si propočítává každý router sám, nemusejí na sebe navzájem čekat).

Samotné propočítávání tras a zjišťování nejkratší (nejoptimálnější) cesty k dané (pod)síti se provádí podle topologické databáze *Dijkstrovým algoritmem* (algoritmem nejkratší cesty, shortest path first, SPF) nebo jeho modifikací. Jedná se o jeden z nejznámějších grafových algoritmů, jehož autorem je holandský informatik Edsger Dijkstra.

Příklad

Podíváme se, jak funguje Dijkstrův algoritmus. Každý spoj mezi routery je ohodnocen číslem, přičemž menší číslo znamená kvalitnější spoj. Každý router si utváří topologickou databázi obsahující všechny routery v dané doméně a ke každé dvojici routerů informaci, zda mezi nimi vede spoj, a pokud ano, jaké je jeho ohodnocení. Topologickou databázi si můžeme představit jako tabulku, kde řádky i sloupce jsou označeny jednotlivými routery, v buňce pro určitou dvojici routerů je informace o spoji mezi nimi (nebo o tom, že tam spoj není).



Obrázek 9.1: Ukázka použití Dijkstrova algoritmu[1]

Na obrázku 9.1 je graf se sedmi routery (označenými A až G) s ohodnocením spojů mezi nimi. Topologická tabulka by mohla vypadat nějak takto:

	A	**B**	**C**	**D**	**E**	**F**	**G**
A	0	1	2	2			
B	1	0	4			1	
C	2	4	0		1		
D	2			0	4		5
E			1	4	0	1	
F		1			1	0	1
G				5		1	0

Tato tabulka by měla být stejná na všech routerech, každý router má tedy představu o topologii sítě. Podle topologické tabulky se vypočítává cena cesty a tedy nejvhodnější cesta ke všem prvkům sítě,

[1] Zdroj: https://www.researchgate.net/figure/Example-topology-and-routing-trees-of-node-A-and-its-neighbours_fig1_225143830

což je základ směrovací tabulky.

Postup stojí na přetvoření původního grafu (ve kterém máme redundantní cesty) do formy stromu – v kořeni stromu je ten router, na kterém probíhá tento výpočet (tj. každý router bude mít jiný strom, sebe dosadí na vrchol), z původních spojů některé odstraní a některé nechá. Zůstanou jen takové spoje, které jsou součástí optimálních cest.

Na obrázku 9.1 vidíme stromy pro routery A, B, C, D. Například router A nechal k routeru F pouze cestu přes souseda B, kdežto cesty přes další sousedy odstranil. K routeru G nechal cestu A→B→F→G, jejíž cena je $1 + 1 + 1 = 3$, nepoužil například cestu A→D→G, která sice vede přes méně mezilehlých uzlů, ale její cena je $2 + 5 = 7$, tedy horší.

Algoritmus SPF je jeden z nejpoužívanějších grafových algoritmů.

Nejznámějším protokolem stavu spoje je OSPF, tento algoritmus se používá také v IS-IS.

Poznámka:

Typickou vlastností protokolů implementujících algoritmus stavu spoje je, že mají přehled o své síti nebo její podstatné části (autonomní systém může být rozdělen na relativně samostatné *oblasti*, které jsou navzájem propojeny hraničními routery, a pak stačí, aby měl protokol přehled o oblasti, ve které se nachází). Další typickou vlastností je rychlá konvergence a menší náchylnost k zacyklení.

9.3.4 Směrovací protokol RIP

RIP (Routing Information Protocol) je jedním z nejstarších směrovacích protokolů, byl vyvinut společností Xerox už roku 1981. Jedná se o vnitřní směrovací protokol komunikující na portu 520 (využívá protokol UDP). Implementuje algoritmus vektoru vzdáleností, jako metrika se používá počet směrovačů na cestě k cíli.

RIPv1 (verze 1, RFC 1058) využívá třídy adres (nepodporuje beztřídní směrování), to znamená, že součástí směrovacích informací není maska podsítě ani délka prefixu. To je další nevýhoda, která RIPv1 prakticky vylučuje z větších sítí.

Nejvyšší akceptovaná metrika je 15, číslo 16 již označuje nedostupnou síť, a tedy v mechanismus Poison Reverse se právě číslo 16 používá pro oznámení nedostupnosti. V intervalech 30 s je všesměrově (broadcast) vysílána směrovací informace (celá směrovací tabulka), což ve velkých sítích může značně snižovat propustnost sítě.

Příklad

Adresy podsítí

`10.10.22.0/24`

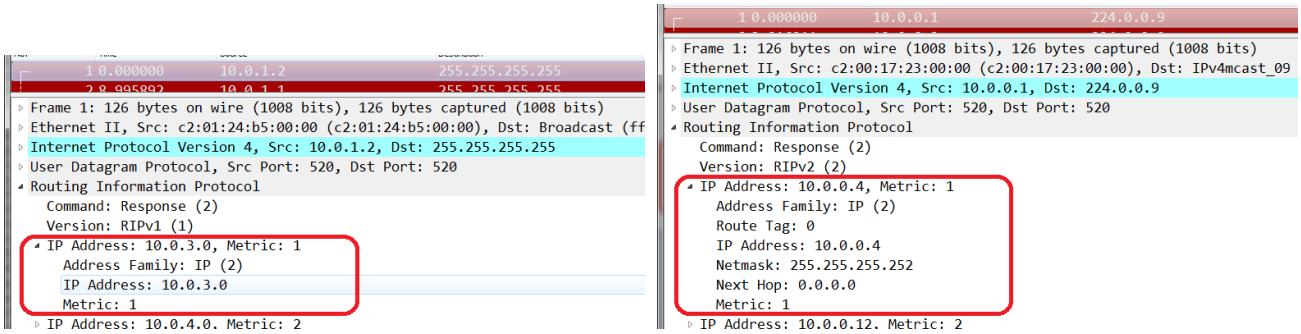
`10.20.10.0/24`

by byly považovány za stejnou síť, protože RIP směrovač by obě adresy považoval za adresu třídy A, kde je síť určena prvním oktetem. Pokud k těmto sítím vede cesta přes různé směrovače, nebude směrování probíhat správně (jeden údaj bude přepsán druhým, jedna ze sítí nebude dostupná).

RIPv2 (STD-56) je aktualizací protokolu RIP z poloviny 90. let 20. století. Hlavním úkolem bylo zohlednění využívání prefixů a beztřídního směrování (VLSM a CIDR). Rozdíly oproti RIPv1:

- lze používat beztřídní směrování (v aktualizačních paketech posílá záznamy o sítích včetně masky),
- aktualizace směrovacích informací se neposílají jako broadcast zprávy, ale jako multicast zprávy na adresu 224.0.0.9, v případě známých sousedů jako unicast pakety,
- podpora ověřování mezi dvěma směrovači, ale na bídné úrovni (buď nešifrovaně nebo šifrovaně s MD5).

Srovnání posílaných informací vidíme na obrázku 9.2 (vlevo RIPv1 posílající jen adresu a metriku, vpravo RIPv2 posílající i masku a další informace).



Obrázek 9.2: Srovnání informace o síti v updatu RIPv1 a RIPv2

U obou verzí RIP je metrika považována za nepříliš kvalitní, tedy použitelnost je pouze v malých sítích, kde to moc nevadí. Omezení na 15 skoků přetrvává.

RIPv2 může nebo nemusí používat *automatickou sumarizaci cest*. Sumarizace zde funguje tak, že pokud jsou ve směrovací tabulce dvě (pod)sítě se stejnou „třídní“ částí adresy, sloučí se do jednoho řádku směrovací tabulky. Slučování se provádí na hranici podle třídy adresy.

Příklad

Podsítě 10.1.0.0/16 a 10.2.0.0/16 mají adresy třídy A, tj. zajímá nás první byte adresy – v obou je to 10. Tyto dvě podsítě by byly při zapnuté autosumarizaci sloučeny do jednoho řádku směrovací tabulky. Pokud do obou vede cesta přes totéž síťové rozhraní, je to v pořádku, v opačném případě máme problém. Takže rozhodnutí zapnout/vypnout autosumarizaci je velmi důležité.

RIPng (vpodstatě třetí verze, next gen, RFC 2080) přidává podporu IPv6, tedy v updatech posílá IPv6 adresy. Další vlastnosti má podobné jako RIPv2.

9.3.5 Směrovací protokoly IGRP a EIGRP

IGRP (Interior Gateway Routing Protocol) je letitý proprietární protokol společnosti Cisco. Implementuje algoritmus vektoru vzdáleností, ale má obecně lepší vlastnosti než RIP, alespoň co se týče metriky. Součástí konfigurace protokolu je číslo autonomního systému (tudíž je slučitelný s jinými směrovacími protokoly na jednom zařízení). Bohužel nepodporuje beztřídní směrování (počítá pouze s třídami, žádná maska ani délka prefixu se nepoužívá).

Směrovací informace se posílají každých 90 s (ve výchozím nastavení), a to broadcastem. Metrika je vícekriteriální, využívá se kombinace kritérií

- zpoždění v síti (prodleva) za předpokladu nezatížené sítě v ms,
- propustnost v b/s, tj. šířka pásma,
- zatížení sítě, hodnota 255 znamená 100% zatížení,
- spolehlivost cesty, hodnota 255 znamená 100% spolehlivost.

První dvě kritéria jsou povinná (dají se odvodit z topologie sítě), další dvě kritéria jsou nepovinná (lze je odvodit z momentálního provozu). Další parametry trasy – počet směrovačů na cestě a MTU – se projevují při kompletaci výsledné metriky.

Zvolená kritéria se zjišťují pro každý spoj na cestě zvlášť a pak se do výsledku sečtou (například u zpoždění v síti) nebo se vybere nejhorší hodnota přes všechny spoje (například šířka pásma, to je vlastně úzké hrdlo celé cesty).

Při výpočtu metriky se postupuje podle zvoleného *typu služby* (obdoba QoS), hodnoty jednotlivých kritérií jsou násobeny konstantami specifickými pro daný typ služby (například může být upřednostněna propustnost sítě).

EIGRP (Enhanced IGRP) je vylepšení protokolu IGRP. Algoritmus směrování podle tohoto protokolu již není čistě algoritmem vektoru vzdáleností, objevují se některé prvky algoritmu stavu spoje, ale přesto je tento protokol řazen do skupiny využívající algoritmus vektoru vzdáleností. Metrika je (stejně jako u IGRP) vypočítávána z propustnosti a zpoždění v síti, je možné přidat kritéria zatížení sítě a spolehlivosti, ale obvykle se to nedělá.

Zatímco IGRP je classful, EIGRP je classless, tedy používá beztřídní směrování. Součástí informace o síti je i maska sítě. EIGRP v novější verzi podporuje také IPv6.

Podobně jako algoritmy stavu spoje, také EIGRP si udržuje tři tabulky: tabulku sousedů, topologickou tabulku a směrovací tabulku. Je zajímavé, že tyto tabulky jsou spravovány na aplikační vrstvě, tedy EIGRP je vlastně aplikační protokol, třebaže ovlivňuje fungování směrovací vrstvy. Tabulky jsou aktualizovány při změnách v topologii, a to multicast zprávami. Používá se multicast adresa 224.0.0.10, resp. pro IPv6 to je FF02::A.

RIP používá jednoduchý algoritmus počtu routerů na cestě, OSPF si vytváří směrovací tabulku pomocí Dijkstrova algoritmu, EIGRP používá algoritmus *DUAL* (Diffusing Update Algorithm). DUAL je poměrně složitý algoritmus, jehož cílem je nejen vypočítávat co nejoptimálnější cesty k cílům bez smyček, ale také udržuje údaje o některých dalších méně optimálních cestách. Na rozdíl od jiných algoritmů dokáže vyvažovat zátěž na více cest k témuž cíli (takže umí dodat do směrovací tabulky pro jeden cíl i více cest), a to i s různými metrikami (pak je posíláno více paketů optimálnější cestou a méně paketů cestou s horší metrikou). Navíc se uchovávají předvypočtené záložní cesty (backup routes), aby při změně topologie nebylo nutné znovu spouštět DUAL (je dost náročný na procesor).

Existuje více druhů zpráv, které se při používání EIGRP posílají (Hello, Update, Query, Reply, Ack), některé potvrzované, jiné nepotvrzované.

Na transportní vrstvě používá Cisco svůj vlastní speciální protokol Cisco RTP (Reliable Transport Protocol – neplést si se „standardním“ RTP Real-time Transport Protocol, navzdory stejné zkratce to jsou dva různé protokoly). Cisco RTP je „něco mezi“ TCP a UDP v tom smyslu, že dokáže fungovat potvrzovaně i nepotvrzovaně, podle toho, co aplikačního přenáší. Například Hello pakety přenáší nepotvrzovaně, kdežto Update pakety přenáší potvrzovaně.

EIGRP používá především dva timery: *hello timer* určuje, jak často si mají sousedi navzájem posílat hello pakety, hold timer určuje, jak dlouho má router čekat, na hello paket od souseda (po uplynutí hold bez odezvy od souseda je soused prohlášen za nedostupného).

Na jednom směrovači může běžet více instancí protokolu (E)IGRP i další směrovací protokoly, protože součástí konfigurace je číslo autonomní oblasti ASN.

Protokol EIGRP byl původně proprietární (společnost Cisco ho používala pouze na svých zařízeních), roku 2013 byl však uvolněn a dnes se jedná o volně použitelný standard, který může být implementován i na zařízeních jiných výrobců (nicméně nikdo se do toho moc nehrne, implementace EIGRP je přes nesporné výhody poměrně složitá).

Poznámka:

Zatímco IGRP se dnes prakticky nepoužívá, s protokolem EIGRP se i nadále setkáváme v praxi. Je považován za optimálnější algoritmus než OSPF, ale na druhou stranu není většinou výrobců podporován a je výpočetně náročný.

9.3.6 Směrovací protokol OSPF

OSPF (Open Shortest Path First) používá algoritmus stavu spoje, což znamená rychlou konvergenci sítě při každé změně topologie. Jde o otevřený protokol, a proto se s ním setkáme na zařízeních od mnoha různých výrobců. Sice se typicky používá jako vnitřní směrovací protokol, ale je možné ho použít i jako vnější (mezi autonomními systémy). Jde o beztřídní směrování, proto k informaci o adrese zadáváme i délku prefixu. OSPF verze 2 (RFC 1247 a RFC 2328) podporuje IPv4 adresy, kdežto OSPF verze 3 (RFC 2740) pracuje s IPv6 adresami.

Směrovací informace se zasílají okamžitě po změně topologie nebo minimálně každých 30 minut (to je pořád výrazně méně často než u předchozích protokolů). Ovšem neposílá se směrovací tabulka, posílá se pouze topologická informace.

Podobně jako EIGRP, i OSPF posílá několik druhů paketů (Hello, Database Description se stručným popisem topologie, Link-state Request s požadavkem na podrobnější info o spoji, Link-state Update s podrobnější informací o spoji, Link-state Ack s potvrzením), ale na rozdíl od EIGRP tyto pakety zapouzdřuje do IP paketů (tj. pracuje na vrstvě L3, podobně jako třeba ICMP).

Každý směrovač si udržuje topologickou databázi sítě nebo své oblasti v síti. Topologická databáze má formu orientovaného grafu s tím, že jedna cesta může mít v každém směru přiřazenu jinou metriku. Pro výpočet cest z topologické databáze se používá Dijkstrův algoritmus (SPF), který byl vysvětlen výše.

Metrika je označována pojmem *cena*. Cena je kombinací více kritérií – propustností sítě, náklady na spoje, atd., podle konfigurace, většinou se volí pouze výpočet z kritéria propustnosti sítě. Číslo 100 000 000 vydělíme šířkou pásma v b/s, tedy například 100 Mb/s má cenu 1, rozhraní s propustností 1.2 Mb/s má cenu 84, apod.

To je dost nepříjemné omezení, protože routery obvykle propojujeme spíše rychlejšími cestami, není výjimkou použití 10Gb spojů (jenže všechny spoje od rychlosti 100 Mb/s včetně výše by měly cenu 1). Naštěstí se koeficient pro přepočet dá změnit.

Příklad

Pokud je rozhraní rychlejší než 100 Mb/s, standardně má také cenu 1 (zaokrouhlujeme nahoru). Této *nesrovnalosti* se předchází změnou vzorce v konfiguraci (změna referenční šířky pásma). Například posloupnost příkazů

```
R1(config)# router ospf 1
R1(config-router)# auto-cost reference-bandwidth 1000
```

Číslo v prvním příkazu je číslo procesu (na routeru teoreticky může běžet několik instancí OSPF, v reálu se to nedoporučuje). Takže v OSPF s číslem procesu 1 měníme referenční šířku pásma na 1000 kb/s, tedy ve vzorci bychom použili číslo 1 000 000 000 (to je třeba provést ve všech routerech v doméně, abychom si nevyrobili nestabilní síť s nepředvídatelným chováním). Tímto odlišíme fast ethernetový spoj od gigabitového (ale 10Gb spoj bude mít stejnou cenu jako gigabitový). Pokud budeme chtít odlišit i 10Gb spoje, ještě jednu nulu přidáme:

```
R1(config-router)# auto-cost reference-bandwidth 10000
```

Cena celé cesty je součtem cen pro jednotlivé úseky cesty. Takže když SPF najde různé cesty k témuž cíli, sežte ceny spojů na každé cestě, porovná výsledky

OSPF podporuje *hierarchické směrování*. To znamená, že směrovače (a jejich sítě) lze v rámci směrovací domény rozdělit do *oblastí* (area). Každý směrovač si udržuje pouze topologickou bázi své oblasti, ostatní oblasti mu zůstávají skryty a aktualizace při změnách topologie se omezuje jen na danou oblast. Rozlišujeme *vnitřní směrovače* (uvnitř některé oblasti), *hraniční směrovače oblasti* (v několika oblastech – ABR, Area Border Router) a *hraniční směrovače autonomního systému* (zprostředkovávají komunikaci mezi různými autonomními systémy, včetně těch s jinými směrovacími protokoly).

Páteřní síť mezi směrovači je označována jako *oblast 0*, směrovače v páteřní síti jsou *páteřní směrovače*. Páteřní síť může být i WAN. V rozsáhlejších sítích by vždy měla být páteřní síť a všechny „nenulové“ oblasti by měly sousedit pouze s oblastí 0.

Součástí směrovací tabulky jsou samozřejmě také prefixy adres sítí. Podobně jako mnohé další protokoly, také OSPF používá *sumarizaci směrovacích informací* – CIDR. Sumarizace je zapnutá zejména na hraničních směrovačích, ať se zbytečně mezi oblastmi neposílá příliš mnoho updatů topologie.

V lokálních sítích (obecně v sítích s všesměrovým vysíláním) musí existovat jeden *pověřený směrovač* (Designated Router), který řídí veškerou aktualizaci směrovacích informací v síti. Pro případ, že by pověřený směrovač selhal, by měl existovat *záložní pověřený směrovač*. Pověřený směrovač provádí výpočet cest a tím snímá část zátěže z ostatních směrovačů v oblasti. Výběr pověřeného směrovače sice lze nechat na protokolu OSPF, ale lepší je provést to ručně, protože automatický výběr nedopadne vždy zrovna ideálně.

OSPF podporuje také autentizaci. Ve verzi 2 umí ověřovat pomocí MD5 nebo SHA kontrolních součtů, ve verzi 3 byl autentizační mechanismus přepracován – používá se autentizace a šifrování pomocí IPSec (který je vestavěn v IPv6).

Z dalších důležitých vlastností protokolu OSPF:

- podpora *rozložení provozu* k cíli mezi cesty se stejnou celkovou cenou (ale buď všechny pakety se stejnou cílovou – finální – IP adresou jdou stejnou cestou),
- podpora směrování podle typu služeb (ToS) IP.

V komerční sféře je OSPF momentálně nejrozšířenějším směrovacím protokolem.

9.3.7 Směrovací protokol IS-IS

IS-IS (Intermediate System to Intermediate System) byl původně součástí architektury ISO/OSI, ale po úpravě byl zařazen do TCP/IP. Je určen především pro směrování bez navazování spojení. Bývá oblíbený zejména u velkých ISP.

IS-IS používá *algoritmus stavu spoje* a princip jeho činnosti je hodně podobný protokolu OSPF. Taktéž se rozlišují směrovače uvnitř oblasti (první úroveň – L1) a hraniční směrovače (druhá úroveň – L2), pozor, neplést si s L1, L2, atd. u vrstev modelů. Neexistuje žádná oblast 0 s páteří, směrovače L2 poskytují propojení do všech oblastí. Také lze vytvořit virtuální spoj mezi dvěma L1 směrovači.

Pokud směrovač L1 obdrží paket nepatřící do jeho oblasti, odešle ho nejbližšímu směrovači L2. Potom paket putuje po směrovačích L2 tak dlouho, dokud se nedostane do oblasti, do které patří.

Zatímco OSPF posílá směrovací informace v IP paketech, IS-IS využívá rámce linkové vrstvy.

9.3.8 Směrovací protokol BGP

Pojem EGP má ve skutečnosti dva významy – označují se tak obecně vnější (externí) směrovací protokoly, ale také jeden konkrétní externí protokol (ten nejstarší).

EGP (Exterior Gateway Protocol) je jednoduchý vnější směrovací protokol pro výměnu informací o dostupnosti či nedostupnosti sítí (předchozí popsané protokoly byly vnitřní), číslo protokolu je 8. Směrovače si neustále ověřují funkčnost svých sousedů a také si s nimi vyměňují informace o dostupnosti připojených sítí. Nepoužívá se žádná metrika, a proto je vyloučena existence více paralelních cest k téže síti; není použitelný, pokud se na cestách vyskytuje smyčka (vyžaduje stromovou strukturu směrovačů). V současnosti se už s EGP nesetkáme.

BGP (Border Gateway Protocol) je také vnější směrovací protokol, ale již složitější, funkčnější a použitelnější. Propojuje různé autonomní systémy, slouží tedy ke směrování mezi autonomními systémy, nikoliv uvnitř (jakéhokoliv) autonomního systému. Ve skutečnosti se dá použít pro interní (iBGP, v rámci jednoho autonomního systému) i externí (eBGP, mezi různými autonomními systémy) směrování. Podporuje VLSM, CIDR (využívá beztřídní směrování, prefixy), agregaci cest a v novější verzi samozřejmě IPv6.

Zatímco jiné směrovací protokoly z důvodu větší pružnosti využívají nespolehlivé přenosy (UDP) nebo částečně spolehlivé (Cisco RTP), pokud tedy vůbec využívají nějaký transportní protokol, BGP využívá spolehlivý přenos (zapouzdřuje do TCP). Důvodem využití spojované komunikace je charakter WAN sítí, ve kterých se používá – tam se jinak než spojovaně nekomunikuje.

Směrovače využívající BGP udržují „sousedské vztahy“ (peering) – pravidelně vysílají zprávy *KeepAlive*, jejichž účel je ujišťování o vlastní funkčnosti. Své sousedy však nezjišťují automaticky, sousedství musí být ručně nakonfigurováno, u souseda v konfiguraci zadáváme i jeho ASN. Při první výměně směrovacích informací směrovač informuje o svém *prefixu* a čísle ASN a obdrží celou směrovací tabulku, při změnách pak jen části směrovací tabulky, kterých se změna týká.

Při oznámení svého prefixu směrovač doplní k prefixu své ASN. Během předávání mezi směrovači každý směrovač také přidá své ASN, tedy celková délka identifikačního řetězce cesty se neustále prodlužuje. Vyšší prioritu má kratší cesta, tedy cesta s menším počtem ASN, vedoucí přes méně BGP směrovačů.

Protože ke každé cestě eviduje sekvenci čísel ASN, vybere prostě tu cestu, která má tuto sekvenci kratší. Nicméně kvalitu cesty lze ve skutečnosti ovlivnit konfigurací několika dalších parametrů.

Jak vidíme, protokol BGP nelze přesně zařadit mezi algoritmy vektoru vzdáleností ani algoritmy stavu spoje. Využívá se *algoritmus vektoru cest*, který je hybridem algoritmu vektoru vzdáleností a algoritmu stavu spoje. Pokud k některému autonomnímu systému vede více cest, většinou vybere tu, která vede přes méně „průchozích“ autonomních systémů.

Ve směrovací tabulce je ke každé cestě především prefix příslušné sítě a sekvence čísel ASN (identifikátorů autonomních oblastí, přes které cesta vede).

Na protokol BGP je třeba si dát pozor. Protože se při jeho použití vlastně napojujeme do směrovací domény celého internetu, musíme dát velký pozor, jaké informace do internetu přes BGP pouštíme. Nejde jen o to, co nemá být vidět „z bezpečnostních důvodů“, ale při neopatrnosti můžeme ovlivnit fungování celého internetu. To už bylo průšvihů – legendární je například problém s chybnou konfigurací BGP směrovačů v Pákistánu, který způsobil kromě jiného několikahodinový výpadek serverů YouTube.

BGP se doporučuje používat jen tehdy, když náš autonomní systém je napojen na minimálně dva jiné autonomní systémy (tj. přes nás vede cesta mezi různými AS). Naopak BGP nemá smysl používat tehdy, když jsme napojeni jen na jeden jiný autonomní systém (a samozřejmě pokud jsme součástí většího AS), a taky když v BGP nejsme úplně kovaní...

Poznámka:

Protokol BGP je hlavním směrovacím protokolem na Internetu a WAN sítích (existují i jeho upravené varianty pro konkrétní typy sítí), jeho využití je poměrně časté.

Další informace:

- https://inl.info.ucl.ac.be/blogs/08-02-25-bgp-misconfigurations-strike-back-pakistan-and-affect-youtube.html
- https://nakedsecurity.sophos.com/2018/10/30/china-hijacking-internet-traffic-using-bgp-claim-researchers/
- https://www.networkworld.com/article/2272520/six-worst-internet-routing-attacks.html
- https://www.noction.com/blog/bgp-hijacking

9.3.9 Softwarový směrovač

Směrovače bývají obvykle hardwarové, ale pro tyto účely lze použít i starší počítač (směrování v menší síti není moc výpočetně náročné) se speciálním softwarem. Pokud si vystačíme se statickým směrováním, pak se dá použít to, co je v každé linuxové distribuci. Pro dynamické směrování to chce něco navíc.

Zebra[2] je velmi oblíbený svobodný software pro UNIXové systémy podporující protokoly RIPv1, RIPv2, OSPFv2, BGPv4.

Quagga[3] je open-source software pro UNIXové systémy (včetně Linuxu, FreeBSD, NetBSD, Solarisu) podporující kromě jiného protokoly OSPFv2, OSPFv3, RIPv1, RIPv2, BGPv4. Jedná se o klon Zebry, v Linuxu je momentálně nejdoporučovanější.

[2] http://www.zebra.org/
[3] http://www.quagga.net/

XORP[4] (Extensible Open Router Platform) je open-source směrovací platforma podporující prakticky všechny známé směrovací protokoly. Na stránkách projektu je dokonce možné si stáhnout demonstrační LiveCD.

NAT32 Windows Software Router[5] je software pro Windows (jen pro malé sítě, ze směrovacích protokolů podporuje RIP). Je volně ke stažení, platí se za podporu.

Další informace:

- https://www.abclinuxu.cz/serialy/ospf-dynamicke-routovani
- https://teklager.se/en/best-free-linux-router-firewall-software-2019/
- https://devconnected.com/how-to-configure-linux-as-a-static-router/
- https://www.root.cz/serialy/linux-jako-internetova-gateway/
- https://monoinfinito.wordpress.com/series/setting-up-a-linux-gatewayrouter-a-guide-for-non-network-admins/

9.4 Mechanismus DNS

Jmenná služba je obecným principem překladu jmenných názvů (textových řetězců) na číselné adresy. Typickým představitelem jmenných služeb je DNS, ale existují i další jmenné služby – například WINS (Windows Internet Name Service, slouží k překladu názvů NetBIOS na IP adresy).

Mechanismus DNS je typickým zástupcem distribuovaných systémů v oblasti počítačových sítí. Databáze jmenných názvů, se kterými DNS pracuje, je distribuovaná na mnoha síťových zařízeních v síti, přičemž každé zařízení si eviduje především adresy ze své vlastní sítě – *princip lokality*. Distribuovanost zde funguje především proto, že systém DNS adres je hierarchický.

Dále budeme předpokládat, že čtenář ví, co je to DNS a jak vypadá doménová adresa (jmenná adresa serveru). Soustředíme se na základní definice, funkčnost celého systému a formát PDU.

9.4.1 Domény a jmenné adresy

Definice (Doména, doménové jméno)

Doména je síť nebo skupina sítí pod společnou správou a sdílející společné (doménové) jméno. *Doménové jméno* musí splňovat tyto podmínky:

- může obsahovat písmena anglické abecedy, číslice a pomlčku, přičemž pomlčka nesmí být na začátku ani na konci,
- maximální délka jednoho jména je 63 znaků,
- maximální délka zřetězení doménových jmen je 255 znaků,
- v rámci domény jsou názvy subdomén jednoznačné.

Doména může mít přiřazeno více jmen. Jedno z nich je *kanonické jméno* (hlavní), ostatní jména nazýváme *aliasy*.

[4] http://www.xorp.org/
[5] http://www.nat32.com/

Doménový adresní prostor je taky *hierarchický*, ale ve směru zprava – zařízení v téže doméně mají pravou část jmenné adresy shodnou, domény jsou členěny hierarchicky. Jejich vztah můžeme taky zakreslit pomocí stromu.


Nejvyšší úrovně doménového stromu mají i své názvy:

- V kořeni stromu je *kořenová doména* (root domain).
- Domény typu *Top-Level Domain* (TLD) jsou v první úrovni stromu domén (například `.cz`).
- Dále hovoříme o doménách druhé (SLD – Second Level Domain), třetí úrovně atd.

Doménu nebo skupinu domén spravovuje konkrétní subjekt – *správce domény*. Podle našeho obrázku je například doména `.slu` včetně subdomén spravována Slezkou univerzitou, doména `.cz` je spravována hlavním registrátorem domén pro Českou republiku, organizací CZ.NIC.

TLD domény jsou několika typů:

- *národní domény* (ccTLD – Country Code TLD, dvoupísmenné) jsou pojmenovány podle mezinárodních kódů států (například `cz`, `uk` apod.), patří sem i doména `eu`,
- *generické domény* (gTLD, minimálně třípísmenné) pojmenované podle svého zaměření:
  - `com` – komerční organizace,
  - `edu` – vzdělávací instituce,
  - `gov` – státní správa v USA,
  - `mil` – armáda USA,
  - `net` – organizace související se správou a standardizací sítí, například `ieee.org`,
  - `org` – ostatní organizace,
  - nové generické domény, například `audio`, `cafe`, `download`, `help`, `chat`, `training`, atd. (stovky).


FQDN (Fully Qualified Domain Name) je plné jméno zařízení, úplná jmenná adresa. Tuto adresu sestavíme tak, že ve stromě jdeme od listu (dotyčného serveru) směrem nahoru ke kořeni a domény na cestě oddělujeme tečkou, například `www.slu.cz`. Maximální délka FQDN je 255 znaků.


Existuje možnost používání ne-ascii znaků v názvech domén (*IDN* – International Domain Names), ale překlad (nahrazení těchto znaků probíhá v klientské aplikaci, nevyskytují se v zónových souborech). Z toho vyplývá, že IDN musí být v klientské aplikaci implementován, včetně lokalizace pro danou znakovou sadu. Pokud tomu tak není, aplikace nedokáže s adresou *vůbec* pracovat (tj. ani zobrazit žádnou stránku z příslušného serveru).

IDN se týká pouze názvů domén. Zbytek adresy také může obsahovat ne-ascii znaky (pozor, často nefunguje, není zatím plně dopracováno), ale jde o IRI (International Resource Indicator).

9.4.2 Zóny a DNS servery


Jednu nebo více domén sdružujeme do společné *zóny* (typicky to bývá část podstromu domén). Pro každou zónu je stanovena konkrétní *autorita*, tedy zodpovědná organizace. Členění na zóny (coby skupiny domén) souvisí s odpovědností a technickou správou, zóny se obvykle nepřekrývají, až na jejich hranice (mezi zónami musí existovat určitá komunikace). Například celý podstrom `slu` tvoří zónu ve správě Slezské univerzity, kdežto nadřízená doména `cz` patří do zóny spravované organizací CZ.NIC.


Správu zóny zajišťujeme na *doménovém serveru* – *DNS serveru*, který plní tyto úlohy:

- vést tabulku adres (tabulku hostitelů – host table), je uložena v *zónovém souboru*,
- odpovídat podle této tabulky na DNS dotazy, tedy překládat jmennou adresu na IP adresu.

V zóně musíme mít jeden *primární DNS server* a dále tam můžeme mít pomocné servery – *sekundární DNS servery* a *caching-only servery*. Ve vztahu k zónovému souboru:

- Primární DNS server vede a garantuje zónový soubor s tabulkou hostitelů, změny v tabulce provádíme vždy právě na tomto severu a záznamy na tomto serveru jsou vždy důvěryhodné, *autoritativní*.
- Sekundární DNS servery si v pravidelných intervalech kopírují zónový soubor z primárního DNS serveru (synchronizují, tomu se říká *zone transfer*), jejich účelem je rozložení zátěže, aby primární server nebyl příliš zatěžován. Jejich odpověď je také *autoritativní*.
- Caching-only servery neobsahují celý zónový soubor. Když takový server obdrží dotaz na adresu, „zeptá se“ primárního nebo sekundárního serveru, ale odpověď si na určitou krátkou dobu uloží (do cache – dočasné paměti) a pokud je v krátké době tázán na tutéž adresu, použije údaj z cache (takže se nemusí dotazovat dále). Odpověď caching-only serveru není autoritativní, ale většinou dostačuje.

 Nejznámější DNS servery jsou BIND, MyDNS, TinyDNS, djbDNS (spíše jako cashing-only), v serverových Windows máme také roli DNS Server.

 *DNS resolver* (dotazovač) je komponenta, která zajišťuje dotazování a ve své cache paměti si po určitou dobu udržuje výsledky předchozích dotazů. Tuto komponentu najdeme v každém systému, který je připojen k síti používající jmenné názvy, tedy i v obyčejném počítači (včetně počítačů s Windows nebo mobilních zařízení). Z toho vyplývá, že i resolver ve Windows si po určitou dobu pamatuje výsledky proběhlých DNS dotazů, aby je nemusel zbytečně často opakovat.

Nejdůležitější informace, kterou DNS resolver potřebuje, je adresa příslušného DNS serveru. Právě tomuto serveru odesílá všechny své dotazy, které pak mohou být buď přímo zodpovězeny nebo v rekurzi předávány.

 **Postup (Možnosti vyhodnocení DNS dotazu na koncovém zařízení)**

Zařízení v síti, na kterém pracuje pouze DNS resolver, si ve skutečnosti některé údaje taky ukládá, jak bylo naznačeno výše. Koncové zařízení s DNS resolverem při vyhodnocení dotazu postupně zkouší tyto možnosti:

- v souboru `/etc/hosts`, resp. `...\system32\drivers\etc\hosts` je seznam dvojic IP adresa + jmenná adresa, které tam lze „staticky“ uložit,
- odpověď na dotaz je v cache paměti, pokud jsme v nejbližší době pokládali stejný dotaz,
- poslední možnost je dotázat se DNS serveru.

Standardně je odpověď hledána právě v uvedeném pořadí.

9.4.3 DNS dotazy

DNS server, jak bylo výše uvedeno, si v zónovém záznamu vede informace o své doméně, obsažených subdoménách, a také informaci o nadřízeném serveru (tj. DNS serveru nadřízené domény). Každý DNS server zná kořenový DNS server celého systému.

Dotaz na adresu obvykle znamená, že je zadána adresa FQDN a tazatel chce IP adresu, ale může tomu být i naopak.

Rozlišujeme *dopředné vyhledávání* (nalezení IP adresy podle doménového jména) a *zpětné vyhledávání* (reverzní, nalezení doménového jména podle IP adresy). Reverzní vyhledávání se provádí přes doménu `in-addr.arpa`. Zde je hlavním problémem to, že jak jmenné, tak IP adresy jsou hierarchické, ale každá „v jiném směru“ – doména TLD ve jmenné adrese je zcela vpravo, kdežto adresu sítě (hlavní část) v IP adrese najdeme zcela vlevo.

Dotazování v dopředném vyhledávání (máme jmennou adresu a chceme IP adresu) probíhá distribuovaně – DNS server často nedokáže na dotaz odpovědět okamžitě podle svých záznamů, proto se obrací nebo odkazuje na některý z kořenových serverů zadané domény první úrovně.

Rozlišujeme dva typy dotazů:

- rekurzivní dotazy – tazatel dostane již hotovou odpověď,
- iterativní dotazy – tazatel postupně získává odkazy na místa, kde může dostat odpověď.

Tazatel (ve skutečnosti resolver) svůj dotaz (jaká je IP adresa zařízení se jménem www.abcd.cz?) posílá tzv. *lokálnímu DNS serveru* (nejbližšímu, který zná). Pokud tento server již zná odpověď, poskytne ji. Pokud ne, další postup se liší podle používaného typu dotazování:

Rekurzivní dotaz. Pokud DNS server nezná odpověď (adresa není z jeho domény), obrátí se na kořenový server nebo DNS server té domény z adresy, kterou ještě zná (to může být TLD). Dotazovaný DNS server se chová stejně – pokud zná cíl, odpoví adresou, pokud cíl nezná, dotazuje se dále (obvykle ve svých subdoménách). Takto rekurzívně je dotaz posílán po stromě domén směrem dolů. Když se konečně dostane k DNS serveru, který zná odpověď, tento server odešle odpověď přímo lokálnímu DNS serveru, na kterém dotazování začalo, a ten ji odešle tazateli.

Takže postup je následující:

- Pokud je dotazovaný název z podřízené zóny (pravá část adresy je stejná), najde jmenný server ve směru zprava doleva první doménu v FQDN, která není jeho. Má odkazy na všechny domény na hranici s podřízenou zónou (například v zónovém souboru pro `cz` je odkaz na DNS server domény `slu`), tedy předá dotaz DNS serveru této podřízené domény. V podřízené doméně je dotaz vyřízen nebo opět předán níže. Rekurzívně směrem dolů je nalezena odpověď, která je pak stejnou cestou distribuována zpět tazateli.
- Pokud dotazovaný název není z vlastní ani podřízené zóny, předá DNS server dotaz DNS serveru z nadřízené zóny. Ten dotaz buď vyřídí nebo předá některému podřízenému nebo nadřízenému uzlu. Po vyřízení je odpověď opět distribuována stejnou cestou k tazateli.

Příklad

Pokud se některé zařízení z domény `fpf.slu.cz` dotáže na IP adresu serveru `www.fpf.slu.cz`, DNS server domény `fpf.slu.cz` hned odpoví, protože tento webový server patří do jeho zóny.

Pokud se některé zařízení z domény `fpf.slu.cz` dotáže na IP adresu serveru `www.cuni.cz`, nejdřív se dotazem zabývá DNS server domény `fpf.slu.cz`. Adresa je z jiného podstromu, tedy dotaz přepošle do domény `slu.cz`. To je podobný případ, proto je dotaz přeposlán DNS serveru domény `cz`. Tento server již pozná, že dotazovaná adresa patří do podřízené domény `cuni.cz` (protože pravá část adresy – `cz` – je shodná), v zónovém souboru najde kontakt na DNS server z podřízené domény, který již najde odpověď – IP adresu webového serveru `www.cuni.cz`, protože ten patří do jeho zóny. Odpověď půjde tazateli zpět stejnou cestou.

Iterativní dotaz. Zde je aktivita především na straně tazatele. Pokud dotazovaný DNS server (kterýkoliv, nejdřív lokální) nezná odpověď, nedotazuje se dále, ale tazateli odešle místo odpovědi seznam DNS serverů, které by mohly znát odpověď („nevím, zeptej se serveru/ů xxxx“, doporučí buď kořenový server, některý TLD, anebo servery ze svých subdomén). Tazatel si vybere některý z doporučených DNS serverů podle toho, jak odpovídají doménové adrese, a táže se dále. Každý dotazovaný server opět pošle buď přímo odpověď, a nebo doporučení s názvy dalších DNS serverů.

Příklad

Pokud potřebujeme zjistit IP adresu počítače *www.firma.cz*, obrátíme se opět na lokální DNS server. Ten nám doporučí, abychom dotaz odeslali na některý z kořenových serverů domény *cz*, což uděláme. Dotazovaný kořenový server buď adresu dohledá ve svých záznamech, anebo vrátí pouze adresy name serverů, které mohou znát odpověď.

Takto distribuovaně probíhá vyhodnocování dotazu (postupně podle vnořování domén různých úrovní), poslední tázaný name server vrátí požadovanou IP adresu.

V reálném světě se rekurzivní a iterativní způsob dotazování kombinuje. Rekurzivní způsob se používá v prvních fázích cesty, kdy se táže jednoduchý resolver, v dalších fázích probíhá iterativní dotazování (když se táže plnohodnotný DNS server).

9.4.4 Tabulka hostitelů

Klíčovým prvkem pro DNS překlad je *tabulka hostitelů* (host table) v zónovém souboru. Můžeme si ji představit jako tabulku, ve které má každá adresa svůj řádek, a na tom řádku máme tyto údaje (ve sloupcích):

- jmenná adresa,
- IP adresa,
- typ záznamu,
- TTL,
- třída (class) – většinou „IN“ jako Internet,
- případně další údaje k záznamu.

Údaj TTL je životnost záznamu v sekundách pro případ, že je tento záznam stažen do cache jiného zařízení (ať už koncového zařízení nebo caching-only serveru).

Z toho, že mezi údaji je i položka „typ záznamu“ (třetí odrážka předchozího výčtu), plyne, že existují různé typy DNS záznamů. Nejpoužívanější typy záznamů jsou:

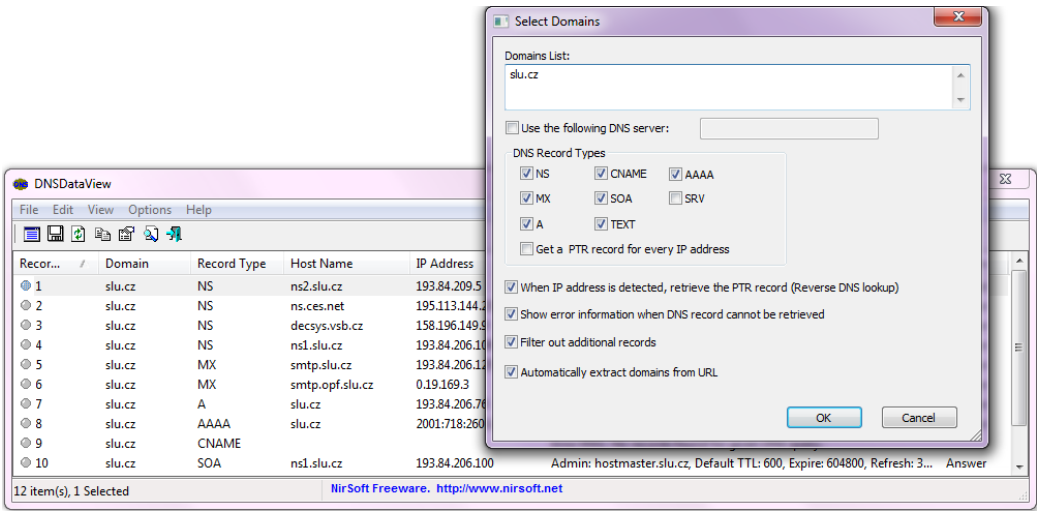
- typ A – pro překlad jmenné adresy na IPv4 adresu,
- typ AAAA – pro překlad jmenné adresy na IPv6 adresu (čtyři „A“) jsou tam proto, že IPv6 adresy jsou čtyřikrát delší než IPv4 adresy),
- typ CNAME (Cannonical Name) – pro překlad aliasu na kanonické jméno dotyčného serveru,
- typ NS (Name Server) – v tomto záznamu se dozvíme IP adresu autoritativního DNS serveru pro některou doménu (okolní domény nebo takové, se kterými se v naší doméně hodně komunikuje),
- typ MX (Mail eXchanger) – adresa e-mail serveru pro danou doménu,

- typ SOA (Start of Authority) – administrativní informace o doméně,
- typ TXT – záznam typu „děvče pro všechno“, používá se například pro ukládání veřejných klíčů, pomocí kterých si kdokoliv může ověřit, jestli e-mail poslaný z naší domény opravdu prošel naším mail serverem nebo byl podvržen,
- typ PTR – pro reverzní (obrácený) překlad, kdy známe IP adresu, ale potřebujeme jmennou adresu, atd.

Většinou potřebujeme právě záznamy typu A nebo AAAA, kdy známe jmennou adresu a potřebujeme buď IPv4 adresu nebo IPv6 adresu.

Příklad

Pro přístup k DNS záznamům můžeme použít příkaz `nslookup` nebo v UNIXových systémech lépe příkaz `dig`, ale zajímavou alternativou s grafickým rozhraním je aplikace *DNS Data View*[6] od společnosti Nirsoft. Okno s výstupem pro doménu `slu.cz` vidíme na obrázku 9.3.



Obrázek 9.3: Program Nirsoft DNS Data View

9.4.5 Protokol DNS a DNS paket

DNS není jen služba, stejně se nazývá i aplikační protokol, který tuto službu poskytuje. Tento protokol stanovuje, jak má vypadat DNS paket, jak se s ním má zacházet, jak funguje celý systém DNS včetně práce se zónovými soubory, jejich transferu mezi DNS servery (zone transfer) a vzájemné komunikace mezi DNS servery.

Protokol DNS definuje komunikaci typu klient-server, tedy rozlišujeme *dotaz* (query question) a *odpověď* (query response). Pro dotaz i odpověď se používá tentýž *formát DNS paketu*. V záhlaví je

- *identifikátor* (podobný jako se používá u IP) sloužící ke spárování dotazu a odpovědi,
- *pole příznaků*, z nichž je nejdůležitější ten, který určuje, zda jde o paket s dotazem nebo s odpovědí, nebo například v případě odpovědi zda je dotazovaný server autoritativní,

[6] Ke stažení na http://www.nirsoft.net/utils/dns_records_viewer.html

- několik dvouoktetových číselných hodnot (počet dotazů, odpovědí, autoritativních odpovědí, dodatečných informací v tomto paketu),
- dotaz(-y) – pokud se jedná o dotaz na překlad, pak tu najdeme jmennou adresu, která má být přeložena, typ záznamu, podle kterého chceme přeložit (A nebo AAAA) a třídu (obvykle IN jako Internet),
- odpověď(-i) – plné znění záznamu nalezeného v zónovém souboru (jmenná adresa, IP adresa, typ, třída, TTL).

Obrázek 9.4: DNS paket

Do paketu s odpovědí patří i dotaz, na který se odpovídá, vytvoření paketu pro odpověď probíhá takto:

- použijeme stejný identifikátor jako v dotazu,
- v poli příznaků nastavíme příznak pro odpověď a pak ještě pár dalších podle okolností,
- pole pro počet dotazů přejmeme, do pole pro počet odpovědí vložíme počet nalezených záznamů ze zónového souboru,
- pole pro dotaz zkopírujeme z paketu sdotazem,
- pole pro odpověď naplníme obsahem nalezených záznamů ze zónového souboru.

Autoritativní odpověď pochází z primárního nebo sekundárního serveru. Z DNS paketu také poznáme, jestli daná odpověď je či není autoritativní, které servery jsou autoritativní v dané zóně, případně si lze nastavením příznaku v dotazu vynutit autoritativní odpověď.

DNS paket se zapouzdřuje do UDP nebo TCP segmentu, na straně DNS serveru komunikuje *na portu 53* (pro UDP i TCP). Jsou upřednostňovány UDP segmenty, protože pro tento typ komunikace je důležitá rychlost. TCP se používá typicky tehdy, když je DNS paket nutné rozdělit do více segmentů.

Protože na straně klienta může být více různých tazatelů komunikujících s DNS serverem, musí každý takový tazatel (například okno webového prohlížeče, poštovní klient, Skype apod.) mít vlastní dynamické číslo portu, aby bylo možné tyto komunikace rozlišit.

Úkol

Pokud jste tak ještě neučinili, rozhodně si prohlédněte několik různých DNS paketů (v síti jich obvykle bloudí dost hodně, případně použijte webový prohlížeč). Srovnejte dotaz a k němu příslušnou odpověď. Všimněte si, jakým způsobem jsou dotaz a odpověď párovány – jak klient pozná, že konkrétní odpověď patří konkrétnímu dotazu.

9.4.6 Bezpečnost v DNS

V poslední době se hodně diskutuje o zabezpečení DNS. Překladu jmenné adresy na IP adresu obvykle věříme a mnoho uživatelů vůbec nenapadne, že ve skutečnosti mohou komunikovat s protistranou s jinou IP adresou, než kterou předpokládají. Důsledkem napadení DNS může být například provedení útoku

Man-in-the-Middle, přesměrování na napadený server za účelem šíření škodlivého kódu, odposlechnutí hesla, čísla karty apod.

V rámci zabezpečení DNS je tedy třeba zajistit, aby DNS pakety s informací o překladu nemohly být podvrženy. Je to vpodstatě obdoba mechanismu SEND, který zabezpečuje ARP.

DNSSEC (DNS Security Extensions) je bezpečnostní rozšíření systému DNS umožňující DNS klientovi ověřit původ dat (zda pocházejí od správného DNS serveru).

Při použití DNSSEC se používá asymetrické šifrování následovně:

- provozovatel DNS domény vygeneruje dvojici klíčů (soukromý a veřejný klíč),
- veřejný klíč uloží u nadřazené autority své domény (vytváří se jakási hierarchie důvěry),
- soukromým klíčem šifruje DNS server pakety, které odesílá,
- příjemce (DNS klient, resolver) je dešifruje veřejným klíčem, který získal od nadřazené autority DNS serveru.

Jedná se vlastně o elektronicky podepsané DNS pakety.

Také správce české TLD domény na svých DNS serverech používá DNSSEC, přičemž jeho nadřazenou autoritou (a místem, kde je příslušný veřejný klíč) je správce celosvětové sítě DNS serverů. Nadřízená autorita ručí za správnost překladu svých bezprostředních podřízených.

Další informace:

Další informace o DNS:

- http://www.fit.vutbr.cz/study/courses/ISA/public/xkupci00.pdf
- http://www.lupa.cz/clanky/neplechy-v-dns-a-jak-se-jim-vyhnout/
- http://www.dnssec.cz/
- http://www.samuraj-cz.com/clanek/dns-domain-name-system-zamereno-na-microsoft/
- http://www.root.cz/clanky/dnssec-cast-prvni-aneb-je-potreba-zacit-od-piky/
- http://www.abclinuxu.cz/clanky/site/nastaveni-dns

9.4.7 Databáze WHOIS

Jak bylo výše uvedeno, každá doména má svého odpovědného správce. Jak zjistit informace o doméně včetně odpovědnosti? K tomu slouží služba *WHOIS* (z angličtiny Who Is?).

Databáze WHOIS je distribuovaná, tedy rozložená mezi odpovědné organizace. Celá databáze je rozložena mezi jednotlivé RIR poskytovatele (každý si vede svou vlastní databázi pro svou doménu a subdomény), podobně se informace distribuují níže (LIR apod.).

Příklad

Pokud budeme chtít vyhledávat SLD doménu v rámci české TLD domény, hledáme ve WHOIS databázi vedené hlavním českým registrátorem CZ.NIC. Jestliže chceme vyhledávat TLD doménu z Evropy či většiny Asie, hledáme ve WHOIS databázi registrátora RIPE.

Jednodušší je však často využít „obecné služby“ serverů, které nejsou regionálně omezené, jako je například http://www.whois.com/whois/ nebo http://www.whois-search.com/.

Pokud chceme zjistit informace o určité konkrétní doméně, zadáme do příslušného vyhledávače FQDN název této domény (tj. včetně nadřízených domén v pravé části adresy). Volíme spíše obecnější název, protože bychom měli pro danou odpovědnou organizaci vyhledávat její nejvyšší doménu. Například místo `fpf.slu.cz` zadáme `slu.cz`.

V odpovědi bychom měli zjistit časové údaje (platnost registrace apod.), kdo je registrátorem, DNS servery, kontaktní informace a další.

Poznámka:

Shrňme si dosud probírané mechanismy pro překlad adres:

- protokoly ARP a NDP provádějí překlad logické IP adresy na fyzickou MAC adresu,
- protokol DNS provádí překlad jmenné (doménové) adresy na IP adresu,
- mechanismus NAT překládá mezi vnitřními a vnějšími IP adresami.

9.5 Adresářové služby a Active Directory

Pod pojmem *adresář* rozumíme databázi, která je organizována hierarchicky. Zatímco u jmenných služeb jde především o překlad názvů na čísla a hierarchie jmen je udržována za účelem fungování tohoto překladu, u adresářových služeb je právě adresář to hlavní a celá funkcionalita spočívá v práci s adresářem a řízení přístupu k němu. Adresářová služba je autentizační autorita (tak jako například RADIUS server nebo ve Windows LSA).

Standardem pro adresářové služby je X.500, protokol DAP (Directory Access Protocol). Tento standard je však příliš obsáhlý, a proto jej prakticky žádná funkční adresářová služba neimplementuje celý.

Většina adresářových služeb je postavena na *protokolu LDAP* (Lightweight Directory Access Protocol). Jde o aplikační protokol pracující nad TCP/IP. V názvu má lightweight (odlehčený), protože jde o zjednodušení protokolů, které se pro tyto účely používaly původně (X.500 zahrnující DAP, DSP a další). Důležitou vlastností, která zajišťuje snadnou přenositelnost, je komunikace mezi uzly přes protokoly rodiny TCP/IP.

Active Directory je implementace protokolu LDAP pro Windows od verze 2000. Je závislá na DNS (můžeme ji chápat jako nástavbu nad doménami DNS), například přejímá názvy domén DNS a využívá DNS při vyhledávání v doménách. V lokální síti musí existovat alespoň jeden server DNS, na klientských počítačích musí být nakonfigurován klient DNS třeba přes DHCP. Domény Active Directory však nejsou totožné s doménami DNS, i kdyby měly stejný název, jsou jinak reprezentovány, ukládají se jiné typy informací.

Používáme tyto pojmy:

- *adresářová databáze* (adresář) je databáze objektů, které jsou v systému spravovány, je hierarchicky uspořádaná (takže adresář je vlastně o objektech a vztazích mezi nimi, to vše je uloženo v souborech),
- *objekty* mohou být například uživatelé, skupiny, počítače, domény, apod., každý objekt má své vlastnosti (například přístupová práva), tyto vlastnosti se v hierarchické struktuře mohou dědit,

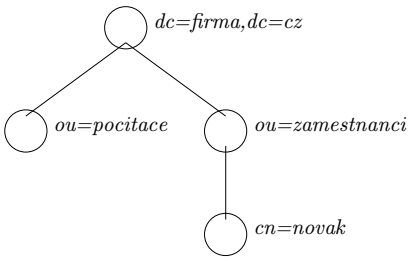
- *kontejner* je objekt, který může obsahovat další objekty (obdoba složek na disku),
- *AD schéma* popisuje objekty, které mohou být uloženy v adresáři Active Directory (jaké mohou mít atributy, co v nich může být uloženo, obdoba třídy),
- *doména* je skupina počítačů sdílejících společnou adresářovou databázi,
- *organizační jednotka* (OU) je podskupina domény (ale ne jakékoliv) oddělená za určitým účelem (například firma může mít jedinou doménu a tu rozčlení na organizační jednotky podle svých oddělení). OU mohou být i vnořené.

✎ V síti je Active Directory provozován na *doménových řadičích* (domain controller, vpodstatě jde o doménové servery), musí existovat alespoň jeden (primární řadič domény) a případně další (ale záleží na konkrétní verzi Windows, některé verze mají s dalšími řadiči trochu problém).

✎ V každé síti je nejméně jeden *Globální katalog* (první globální katalog se vytvoří na primárním řadiči domény). V Globálním katalogu jsou především souhrny informací obsažených v dalších doménových serverech sítě (ne všechny parametry, jen nejdůležitější), hovoříme také o *replikaci* (dynamickém vytváření kopií).

Globální katalogy slouží při vyhledávání informací v síti a také k *autentizaci* (uživatel se vlastně z technického hlediska přihlašuje ke globálnímu katalogu) a *autorizaci*. Takže přes doménové řadiče s globálními katalogy přistupujeme k objektům a také se na nich provádí autentizace (kontrola při přihlašování) a autorizace (při přístupu k objektům).

V Active Directory (také ve jmenných službách včetně DNS) se používá několik druhů názvů podle typu zanoření v doménách. Jsou to především



Obrázek 9.5: Názvy v doménách

✎ *Domain Component* (DC, uzel domény), *Organization Unit* (OU, organizační jednotka, to je Active Directory Container, obdoba složky) a *Common Name* (CN, objekt). Adresace (popis cesty k objektu) podle struktury na obrázku 9.5 je

```
cn=novak,ou=zamestnanci,dc=firma,dc=cz
```

Tento způsob adresace objektu se označuje *DN* (Distinguished Name). Další způsob adresace, UNC, známe z DNS názvů:

```
firma.cz/zamestnanci/novak
```

Existují i další způsoby adresace, především *RFC 822* (forma silně připomínající e-mail) a *názvy LDAP* (ve formě ldap://uncdomeny/CN=....,OU=....).

Na desktopu obvykle služba Active Directory není nainstalována, pracujeme zde pouze se *zásadami* (politikami) v nástrojích *Místní zásady zabezpečení* (`secpol.msc`) a *Zásady skupiny* (Group Policies Editor, `gpedit.msc`). V síti se zprovozněným Active Directory jsou zásady skupiny napojeny na Active Directory.

Dnes jsou běžné heterogenní sítě (tj. na počítačích v síti jsou různé typy operačních systémů. Může se zdát, že používání mechanismu Active Directory v heterogenní síti je problém, ale řešení existuje, spočívá v použití jakýchsi „překladatelů“ – protokolů, které zprostředkují komunikaci mezi počítači s různými operačními systémy. V případě použití Active Directory jde především o protokol LDAP implementovaný také na jiných operačních systémech včetně Linuxu, dále pro přístup k datům se používá protokol SMB.

Další informace:

- http://www.samuraj-cz.com/clanek/active-directory-komponenty-domain-tree-forest-site/
- http://www.mcmcse.com/microsoft/guides/ad.shtml
- http://www.petri.co.il/ad.htm
- http://www.learnthat.com/Software/learn/1295/Introduction-to-Active-Directory/
- http://www.samuraj-cz.com/serie/ldap-a-active-directory/
- http://www.samuraj-cz.com/clanek/adresarove-sluzby-a-ldap/
- http://ldap.zdenda.com/

9.6 QoS

QoS (Quality of Service) je mechanismus prioritizace daných typů provozu v síti. Existují metody, které nabízejí garanci určitých parametrů poskytované služby. Garantovat lze například:

- šířku pásma pro přenos, volnou kapacitu sítě,
- minimální ztrátovost datových jednotek (případně téměř nulovou ztrátovost),
- dynamiku ztrátovosti (tedy aby nedocházelo ke ztrátám ve shlucích, to by mohlo mít negativní vliv na přenos multimédií, která občasnou ztrátu datové jednotky tolerují),
- zpoždění, latenci,
- změny – dynamiku – zpoždění (jitter),
- atd.

Běžné síťové protokoly (e-mail, přenos souborů apod.) obvykle nemají velké nároky na QoS, ale naopak VoIP a videopřenosům mohou vadit vyšší hodnoty zpoždění, a také vyšší hodnoty jitter.

Technologie QoS je v některých specifikacích dostupná přímo, například v ATM, Frame Relay a MPLS. Jinde je nutné mechanismus QoS přidat. Jednou z možností, jak zajistit QoS ve velmi rozšířených IP sítích (i vzdálených), je DiffServ.

Díky IETF existují od 90. let 20. století tyto dva mechanismy:

- IntServ (Integrated Services)
- DiffServ (Differentiated Services)

Kromě toho existují i další mechanismy garance kvality služeb. Vlastní mechanismy mají také WAN sítě, včetně ATM a MPLS.

IntServ je starší jednoduchý mechanismus, který funguje takto:

- aplikace, která chce využít QoS, si rezervuje pásmo pro své přenosy,
- toto pásmo je aplikaci plně vyhrazeno po celou dobu trvání rezervace, ať už ho používá nebo ne,
- je simplexní, tedy rezervace platí v jednom směru komunikace.

Jednosměrnost obvykle není na závadu, protože se IntServ typicky používá pro multimediální přenosy, které jsou v principu asymetrické, případně není problém navázat dva simplexní spoje pro oba směry.

Rezervace by měla být platná na celé cestě, proto by příslušný protokol měl být podporován na všech uzlech, přes které cesta vede. Pokud je systém rezervací někde na cestě přerušen (třeba proto, že daný protokol tam není podporován), v daném úseku půjdou pakety běžným způsobem.

Nejznámější implementací mechanismu IntServ je *protokol RSVP*, který se často používá pro rezervaci komunikačního pásma například jako doplňující protokol protokolu SIP.

DiffServ pracuje jiným způsobem. Nerezervuje staticky šířku pásma ani nenutí aplikace k dopřednému oznamování svých potřeb, ale místo toho do záhlaví paketu dodává informaci o prioritě, přičemž tato informace bývá brána v úvahu na síťových prvcích na cestě tohoto paketu. PDU jsou tříděny do kategorií a každá kategorie má přiřazenu určitou prioritu nebo obecněji způsob zacházení při odbavení na síťových prvcích.

Pracuje s 6 bity označenými zkratkou *DSCP* (Differentiated Service Code Point), což znamená 64 možných tříd přístupu (v praxi se však využívá jen několik z nich). Třídy lze rozdělit do tří základních kategorií:

- *Urychlené předávání* (EF, Expedited Forwarding) – garance malého a téměř konstantního zpoždění, latence, šířky pásma; je složité, proto může být poskytnuto jen nízkému počtu přenosů, je vhodné například pro implementaci virtuálního okruhu.
- *Zajištěné předávání* (AF, Assured Forwarding) – funguje podobně jako využívání CIR u Frame Relay. Používá se u služeb, které upřednostňují volitelnou úroveň QoS.
- *Základní služba* (BE, Best Effort) – pro běžné datové přenosy.

Při vstupu do sítě jsou pakety klasifikovány, označeny třídou (pokud je to potřeba). Pokud ne (všechny bity jsou 0), jedná se o kategorii *Best Effort*.

Klasifikace probíhá pouze při vstupu do sítě (tj. sítě uplatňující DiffServ, hovoříme o *DiffServ doméně*), na následujících směrovačích je tato informace pouze využívána.

Tento způsob řízení QoS je poměrně jednoduše slučitelný s jinými QoS protokoly. Na hranici DiffServ domény (na hranovém, ingress edge, směrovači) probíhá výše popsaná klasifikace paketu, případně překlad z jiné QoS technologie (ATM, MPLS, atd.).

Konkrétní umístění pole bitů pro DiffServ závisí na tom, na které vrstvě a v kterém protokolu je tato technologie implementována (nemusí to být na síťové vrstvě s IP protokolem). Může to být uvnitř hlavičky datové jednotky (pokud je tam místo) nebo vně (před záhlavím). Také s konkrétními hodnotami jednotlivých bitů se někdy musí zacházet poněkud volně, protože DiffServ hodnoty se často mapují do polí v záhlavích PDU, která mají jiný počet bitů než jak DiffServ určuje.

V IPv4 paketu máme 8bitové pole třídy služby TOS (z toho se první 3 bity využívají podle IEEE 802.1p), v IPv6 paketu jde o pole pro prioritu. V MPLS záhlaví jsou pro QoS pouze tři bity.

Další informace:

- http://www.kiv.zcu.cz/~ledvina/Prednasky-PSI-2007/qos-text.pdf
- http://www.ipinfusion.com/pdf/IP_InfusionQoS_MPLS2.pdf
- http://www.svetsiti.cz/view.asp?rubrika=Tutorialy&temaID=70&clanekID=76
- http://www.cesnet.cz/doc/techzpravy/2000-6/diffserv.ps.gz

9.7 VoIP a videotelefonie

9.7.1 Princip

VoIP (Voice over IP, „internetová telefonie“, IP telefonie) je technologie pro přenos digitalizovaného hlasu v IP datagramech. Hovor může být veden i mezi více než dvěma účastníky. Koncová zařízení mohou být buď hardwarové VoIP telefony anebo jde o software, VoIP aplikaci.

Telefonovat můžeme

- v rámci sítě poskytovatele,
- mimo síť poskytovatele, ale pořád v rámci datové sítě,
- na běžné telefonní číslo do veřejné telekomunikační sítě.

Třetí možnost vyžaduje vytvoření spojení přes *VoIP bránu*, která tvoří rozhraní mezi datovou a veřejnou telekomunikační sítí. Volí se VoIP brána co nejblíž cíli, aby co nejdelší část spojení vedla po datových linkách (resp. aby část spojení po telekomunikačních linkách byla pokud možno „místní hovor“).

V současné době se setkáváme s komplexně řešenými pobočkovými ústřednami pro firmy, které si takto mohou zařídit vlastní VoIP síť. Jejich konkurencí jsou však nabídky mobilních operátorů, které jsou, zvláště pro větší firmy, velmi výhodné.

Obvykle se vyžaduje splnění těchto požadavků:

- rychlost připojení k Internetu přes 128 kb/s
- maximální zpoždění na spoji k serveru poskytovatele 150 ms
- maximální kolísání zpoždění 30 ms
- ztrátovost datových jednotek pod 2 %, lépe kolem 1 %

Dále si může poskytovatel stanovit požadavky na způsob implementace NAT apod. Tyto požadavky jsou však relativní, záleží na celkovém vytížení spoje (proto je důležité používat QoS) a obvyklém počtu hovorů na spoji vedených. VoIP vyžaduje sice menší šířku pásma než datové služby, ale zato pokud možno konstantní.

Některé typy připojení k Internetu nejsou pro VoIP moc vhodné (například některá mobilní připojení, jako je GSM, GPRS či EDGE), u Wi-Fi je využití diskutabilní (zvláště tehdy, když je oblast příliš zarušená, může ztrátovost datových jednotek dosáhnout příliš velké hodnoty).

Nejdřív je nutné navázat spojení. Navazování spojení provádí příslušný aplikační protokol, a to buď SIP nebo H.323.

Po navázání spojení se přenášený zvuk nejdřív digitalizuje – provádí se vzorkování na 8 kHz. Přenos se provádí pomocí kodeku (coder/decoder), který provádí kódování signálu a zároveň jeho kompresi.

VoIP je implementován i v různých aplikacích, například v notoricky známé aplikaci Skype.

9.7.2 Protokoly

SIP (Session Initiation Protocol) je textově orientovaný protokol vyvinutý přímo pro použití na Internetu (IETF), v roce 1999. Je to protokol aplikační vrstvy pro vytváření, udržování a ukončování interaktivních relací, kromě jiného VoIP. Sám o sobě nezajišťuje QoS, ale dokáže spolupracovat s protokoly, které jsou k tomu určeny. Jeho úkoly:

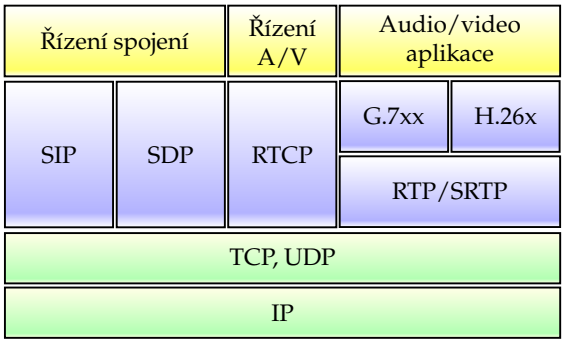
- lokalizuje příjemce volání,

- ověří vlastnosti použitého zařízení,
- zajistí přenos dat a zabezpečení u jiných protokolů.

SIP je *distribuovaný*, co nejvíc posouvá inteligenci přenosu ke koncovým zařízením.

Další významnou vlastností SIP je *pružnost*. Dokáže spolupracovat s mnoha protokoly a kromě VoIP je použitelný například i pro Instant Messaging a dokáže dobře spolupracovat s mobilními technologiemi. Lze používat jak číselné adresy (telefonní čísla), tak i URI (Universal Resource Identifier) – webovou adresaci.

Na transportní vrstvě je využíván především protokol UDP, což znamená vyšší rychlost přenosu (menší zpoždění).

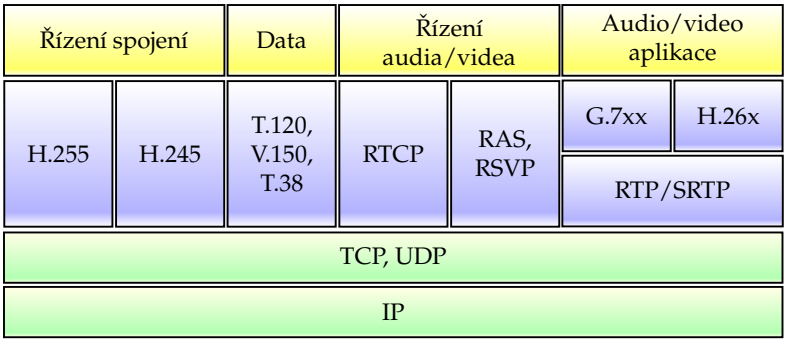


Obrázek 9.6: Protokolový zásobník pro SIP

H.323 je sada binárních protokolů pro konverzi signalizace paketového protokolu na signalizaci telefonní sítě. Je starší než SIP (rok 1996) a vychází ze signalizace obvyklé v telekomunikačních sítích. Jde o komplexnější přístup, H.323 řeší kromě navazování a udržování spojení také QoS a další charakteristiky přenosu. Při adresování používá telefonní čísla.

H.323 má větší zpoždění při navazování spojení a navíc jeho spolupráce s jinými protokoly je problematická. Jedná se o protokol do značné míry *centralizovaný*.

Na transportní vrstvě je využíván především protokol TCP, který má větší režii přenosu, to znamená větší zpožďování přenosu.



Obrázek 9.7: Protokolový zásobník pro H.323

Podpůrné protokoly transportní vrstvy:

- RTP (Real-time Transport Protocol) doplňuje činnost protokolu UDP v oblasti rychlé paketizace datového toku v reálném čase, synchronizuje přenos a umožňuje zjistit ztracený paket nebo nesprávné pořadí paketů.

- SRTP (Secure RTP) je bezpečnější varianta protokolu RTP.
- RTCP (Real-time Transport Control Protocol) zprostředkovává zpětnou vazbu pro protokol RTP, umožňuje využívat jeho výše popsané vlastnosti. Využitím RTCP lze regulovat datový tok, například dynamicky měnit rychlost přenosu podle požadavků přijímajícího uzlu.
- SDP (Session Description Protocol) se používá u protokolu SIP pro vyjednání parametrů multimediální komunikace.
- RSVP (Resource Reservation Protocol) slouží pro rezervaci zdrojů spoje, je to prostředek řízení QoS.
- G.711, G.729, g.723.1, atd. – audio kodeky, nad nimi běží audio aplikace.
- H.261, H.263, atd. – video kodeky, nad nimi běží video aplikace.
- T.120 – konferenční datové přenosy.
- RAS (Registration, Admission, Status) – podpůrný protokol pro vyjednání a udržení spojení při použití H.323. Je součástí protokolu H.255.0 sloužícího k administraci multimediální komunikace.

Z dalších protokolů se využívá například STUN nebo TURN pro podporu NAT, SCCP u zařízení Cisco, IAX v systému Asterisk, a další.

9.7.3 Videotelefonie a videokonference

✎ *Videotelefonie* je vzájemný hovor dvou či více účastníků se současným sledováním synchronizovaného videostreamu (obvykle se snímá obraz videokamery). Dnes obvykle nejde ani tak o samostatná zařízení (videotelefony), setkáváme se hlavně se softwarovou implementací (například výše zmíněný Skype nebo FaceTime od Applu, obojí na principu VoIP). Funkce videotelefonie je také obvykle implementována v softwaru pro pobočkové ústředny (např. ve výše zmíněném Asterisku).

Videotelefonie stojí především na protokolu *H.264*. Tento protokol standardizovaný ITU-T je vlastně notoricky známým videokodekem pro kompresi videa, mnohým bude hodně říkat zkratka MPEG-4. Používá se všude tam, kde je potřeba přenášet komprimované digitální video.

✎ Zatímco videotelefonie vyžaduje vytvoření spoje typu point-to-point, *videokonference* je již náročnější způsob komunikace. Jde vlastně o spoje typu multipoint-to-multipoint, signál je šířen formou multicastu. Videotelefonii můžeme považovat za speciální jednodušší případ videokonferencí. Existují řešení využívající webovou aplikaci, mobilní aplikaci, desktopovou aplikaci, nebo to vše kombinují.

✂ Nejmenší šířku pásma (do několika desítek kb/s) vyžaduje přenos zvuku. Větší šířku pásma vyžaduje přenos obrazu i v nižší kvalitě. Kromě přenosu zvuku a obrazu existují i další možnosti, jak využít kapacitu multimediálních přenosů, například *sdílená tabule* (účastníci „kreslí“ na tabuli, ta je viditelná a přístupná všem účastníkům videokonference).

✎ **MS Teams** je jednoduchý videokonferenční systém založený na cloudu, který je součástí MS Office. Nedá se nijak zvlášť široce konfigurovat, ale základní funkce obsahuje – vytváření týmů, plánování a navazování videokonferenčních hovorů, jejich nahrávání, sdílení plochy, sdílení souborů, funkce tabule. Uživatel je buď v roli vlastníka kanálu (učitele), nebo v roli člena nebo hosta (studenti).

✎ **BigBlueButton** je otevřený projekt umožňující provozovat videokonferenční systém ve vlastním privátním cloudu, ve skutečnosti se jedná o upravenou a obohacenou linuxovou distribuci (dá se získat i ve formě virtuálního stroje do VirtualBoxu pro různé operační systémy či VMWare Fusion na MacOS).

Vznikl v univerzitním prostředí a je především určen pro on-line výuku. Umí videokonference, sdílení obsahu, sdílení plochy, chat, sdílenou tabuli. Rozlišují se dvě role: moderator (učitel) a viewer (student).

Apache OpenMeetings je řešení spíše pro komerční sféru než pro školství, nicméně se také jedná o videokonferenční systém. Vznikl rozšířením Red5 media serveru a nabízí videokonference, schůzky, sdílení plochy, sdílení dokumentů, atd. Je integrován v některých CMS systémech.

Cisco Webex Meetings je řešení pro komerční sféru, placená verze zvládne videokonferenci s až 200 účastníky. Neplacená verze má určité limity, nicméně také je použitelná, když nemáme vysoké nároky. Webex se dá používat ve formě webové aplikace, ale desktopová aplikace nabízí širší možnosti konfigurace. Celkově se dá říct, že Webex je snad nejšířeji konfigurovatelné řešení.

Jitsi Meet je jednoduché open-source řešení pro menší týmy, které je postaveno na webovém rozhraní, takže není závislé na operačním systému a není třeba nic instalovat (ale existuje i mobilní aplikace). Nabízí běžné vlastnosti, které od videokonferenčních nástrojů čekáme, včetně nahrávání, sdílení obrazovky či okna, atd.

MBone je distribuovaný systém složený z volně dostupných nástrojů. Vyžaduje vhodně multimediálně vybavený počítač připojený do sítě, funguje nad protokolem IP. Podporuje široké spektrum operačních systémů od Windows po různé UNIXové systémy. Umožňuje organizovat videokonference, přihlásit se do existující videokonference, přenášet zvuk a obraz, sdílet text i multimédia (whiteboard – sdílená bílá tabule s obrázkem).

VRVS (Virtual Room Videoconferencing System) je komplexní systém poskytovaný zdarma ve formě služby. Je třeba mít instalovánu podporu buď MBone nebo H.323. K provozování VRVS potřebujeme počítač se systémem Solaris nebo Linux (omezení neplatí pro klientské stanice, tam je důležitá verze Java Virtual Machine). VRVS je velmi dobře zdokumentován. V poslední době má VRVS problém s dostupností.

Ekiga (GnomeMeeting) je svobodný software pro VoIP a videokonference vyvíjený v rámci projektu Gnome. Komunikuje i s jinými klienty podporujícími protokol SIP nebo H.323 (včetně MS NetMeeting). Klient je dostupný pro Linux i Windows.

Další informace:

- https://bigbluebutton.org/
- https://openmeetings.apache.org/
- https://www.webex.com/
- https://jitsi.org/jitsi-meet/
- http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=27320
- http://www.cesnet.cz/videokonference/mbone/prirucka/mbone.pdf
- http://oirt.rutgers.edu/cmn/video_desktop/vrvs.html
- http://www.ekiga.org/

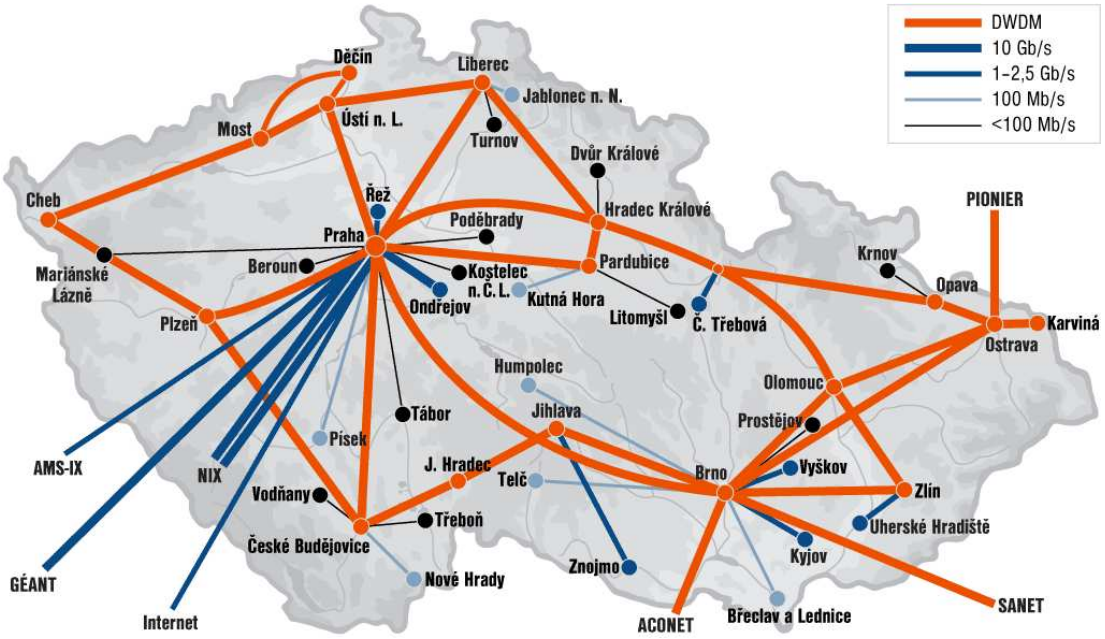
9.8 CESNET2

CESNET2 (Czech Education and Scientific NETwork) je národní vysokorychlostní počítačová síť určená pro vzdělávání, vědu, výzkum a vývoj, založená roku 1996. Propojuje především vysoké školy a výzkumné ústavy včetně všech ústavů Akademie věd České republiky, jsou připojeny také některé nemocnice a střední školy.

Jde už o druhou generaci této sítě. Původní síť CESNET první generace dosahovala rychlostí v desítkách až stovkách Mb/s, zatímco CESNET2 dosahuje rychlostí kolem 10 Gb/s, na některých místech až 100 Gb/s. V polovině 90. let se hodně používala síť ATM, později kolem roku 2000 byla postupně zaváděna technologie PoS (Packet over SDH) a MPLS a přenosová soustava byla postavena na DWDM.

http://www.cesnet.cz

Současná topologie sítě je naznačena na obrázku 9.8.



Obrázek 9.8: Topologie sítě CESNET2[7]

Účelem sítě CESNET2 je poskytování vysokorychlostních přenosů pro účely výzkumu a vzdělání, zabývá se také různými typy využití přenosové kapacity (VoIP, multimediální přenosy a videokonference, QoS, vysokorychlostní bezdrátové připojení, atd.). V současné době se vzhledem k momentální situaci v oblasti sítí hodně angažuje v bezpečnostních technologiích.

Zajímavý projekt je také *EduRoam.cz*, který umožňuje připojit se do bezdrátové sítě kteréhokoliv člena CESNETu (většinou jde o podporu mobilit zaměstnanců vysokých škol a výzkumných ústavů).

CESNET2 je také napojena na mezinárodní sítě s podobným zaměřením, například na evropskou síť GÉANT, slovenskou SANET, rakouskou ACONET a polskou PIONIER.

Metacentrum je aktivita CESNETu věnovaná provozu národního výpočetního gridu (distribuované výpočetní sítě). V gridu je v současné době (2015) několik výkonných výpočetních center zahrnujících

[7] Zdroj: http://www.cesnet.cz

celkem 10 712 procesorů. Implementuje různé metody pro BigData, včetně algoritmu Hadoop od společnosti Google. Grid je využíván především akademickými pracovníky pro náročné výpočty ve výzkumu a vývoji. Členství je bezplatné pro akademické pracovníky, kteří jsou zaměstnanci nebo studenty členů CESNETu.

9.9 Bezpečnostní týmy

Svůj bezpečnostní tým zasahující (nejen) proti kyberkriminalitě by dnes měla mít každá větší organizace. Setkáváme se zde se dvěma důležitými zkratkami:

- CERT (Computer Emergency Response Team),
- CSIRT (Computer Security Response Team).

V obou případech jde víceméně o totéž – tým odborníků určený k boji proti kyberkriminalitě, tedy lidé aktivně se zabývající bezpečností v oblasti ICT. Obě zkratky se používají pro označení bezpečnostních týmů jak na národní úrovni, tak i na úrovni soukromých společností. Ještě něco mají tyto týmy společného – spolupráci.

CERT je ve skutečnosti chráněnou značkou ve vlastnictví Carnegie-Mellon University, proto se v praxi setkáme častěji se zkratkou CSIRT nebo nějakou obdobou zkratky CERT.

CSIRT.CZ je CSIRT týmem provozovaným sdružením CZ.NIC (to je správce domény .CZ – české národní domény). Jedná se o hlavní CSIRT tým České republiky, a to od roku 2012, kdy bylo uzavřeno memorandum mezi CZ.NIC a Národním centrem kybernetické bezpečnosti (resp. Ministerstvem vnitra ČR), které má ze zákona povinnost zajistit provoz bezpečnostního týmu bojujícího proti kyberkriminalitě. Nicméně tento tým ve skutečnosti existuje už od roku 2007 (na státní úrovni se totiž o bezpečnostních týmech dlouho jen jednalo, a teprve potom, co vznikly iniciativou v soukromém sektoru, byla jejich činnost posvěcena zákonem).

Tým CSIRT.CZ plní tyto úkoly:

- řešení zjištěných bezpečnostních incidentů ve spolupráci s českými provozovateli sítí a zahraničními partnery,
- pomoc provozovatelům sítí při zřizování vlastních CSIRT týmů,
- poskytuje reaktivní i proaktivní služby, poradenství,
- provozuje *honeypoty* („vějičky“ pro útočníky – zařízení záměrně vypadající jako slabě zabezpečená), mapa je na https://honeymap.cz/,
- mohou zjišťovat možné autory DoS útoků a předávat informace orgánům, které pak na toto oznámení reagují, nebo informovat provozovatele sítě o napadení této sítě.

Na mezinárodní bezpečnostní scéně tým spolupracuje především s organizacemi TERENA (Trans-Europen Research and Education Networking Association) a FIRST (Forum for Incident Response and Security Team).

Proaktivními službami se rozumí služby v oblasti bezpečnostní prevence, například informační pomoc, nabídky školení, provoz IDS/IPS (systémů detekujících možnost nebo pravděpodobnost napadení) apod. *Reaktivní služby* naopak reagují na již proběhlý nebo probíhající bezpečnostní incident, v tomto případě například přijímání oznámení a podnětů o incidentech a reakce na ně.

K novějším službám týmu patří služba bezplatného penetračního testování webu (tzv. Skener webu), a dále taktéž bezplatná služba zátěžových testů odpovídajících svou intenzitou DDoS útokům, které na několik dnů roku 2013 silně zpomalily české sítě.

CESNET-CERTS je bezpečnostní tým společnosti CESNET, provozovatele sítě propojující akademické a výzkumné organizace (především vysoké školy a ústavy Akademie věd). Pracuje už od roku 2004 a nese odpovědnost za bezpečnost především v sítích nějak souvisejících s akademickou a vědeckou sférou (včetně domény eduroam.cz). Povinnosti a pravomoci jsou vpodstatě podobné jako u CSIRT.CZ, jen vztažené na sítě, kde má tým pravomoc.

Poznámka:

CSIRT týmy (ani celostátní) nemají žádné pravomoce k řešení bezpečnostních incidentů na jiné než technické úrovni, nejde o represivní orgány. Mohou pouze monitorovat a radit, pravomoci k razantnějším reakcím mají pouze ve své vlastní síti.

Další: Jak bylo výše napsáno, svůj bezpečnostní tým by měly mít všechny větší organizace. Vlastní CSIRT týmy působí například v sítích ISP (poskytovatelů Internetu), bank, poskytovatelů energií, provozovatelů energetických společností (jako je u nás například ČEZ), velkých automobilek, velkých poskytovatelů internetových služeb (Google, Seznam, Facebook), atd. Svůj bezpečnostní tým má také Slezská univerzita.

Poznámka:

Každý takový tým se určitým způsobem prezentuje – musí mít někde uvedeny kontaktní informace, vymezení pole působnosti a odpovědnosti, seznam služeb, které poskytuje (především v rámci sítě svého provozovatele). Musí být zřejmé, kdo se na tým může obrátit, s jakým problémem k řešení a jakým způsobem (může to být třeba formulář na webu). Pro tyto účely obvykle tým pořádá i různá školení (to je vpodstatě jedno z proaktivních opatření).

Další informace:

- Národní centrum kybernetické bezpečnosti: http://www.govcert.cz/cs/
- https://www.csirt.cz/
- http://www.cesnet.cz/sluzby/reseni-bezpecnostnich-incidentu/,
- http://www.cert.org/csirts/
- https://csirt.cesnet.cz/
- http://www.earchiv.cz/b08/b0408002.php3
- http://www.lupa.cz/clanky/csirt-cz-prichazi-kyberzlocincum-navzdory/

Kapitola 10

Bezpečnost a správa

10.1 Typy útoků

V počítačové síti je třeba čelit různým typům útoků, z nichž některé jsou vzájemně provázány. Především nás zajímají tyto útoky:

Mapping (mapování) – jde vlastně o jakousi přípravu k následujícím útokům. Hacker získává co nejvíc informací (otevřené porty, použité rozsahy IP adres, operační systémy a jejich verze, atd.), aby vlastní útok mohl být co nejúspěšnější.

Network sniffing (naslouchání na síti) – packet sniffer (zachytávač nebo odposlouchávač paketů) odkloní provoz na síti, zachytává pakety a získává z nich informace (pak je posílá dál k cíli, nedoručené pakety by znamenaly jeho odhalení).

Účelem sniffingu je především získávat hesla přenášená v textovém tvaru a také další citlivé informace. Také je možné zachytávat a pozměňovat obsah paketů anebo dokonce číst a pozměňovat informace na uzlech v síti (včetně konfiguračních souborů nebo hesel).

Účinnou obranou je především spolehlivé šifrování (včetně autentizace po síti), a také fyzické zabezpečení sítě, protože tento typ útoku vyžaduje fyzický přístup k síti. To může být problém u bezdrátových typů připojení.

Packet sniffer (třeba Wireshark) však může být zcela legálně využíván administrátorem ke sledování provozu na síti.

SQL Injection – hackerovi se podaří podstrčit (inject) SQL příkaz někam, kde nemá co dělat, případně vhodnou volbou řetězců „vyrobit“ a poslat serveru SQL příkaz. Většinou se jedná o zneužití formulářů na webu nebo řetězce předávaného přes HTTP GET, kdy útočník vyplní místo běžného řetězce „škodlivý“ řetězec obsahující symboly se speciálním významem. Typický zneužívaný symbol je třeba apostrof.

Proti tomuto typu útoku se celkem dá bránit, pokud programátor webové aplikace (tedy webového rozhraní k databázi) zajistí pečlivou analýzu vstupů od uživatele, a především je třeba mít správně nastavená oprávnění u databáze (například požadavky na destruktivní operace typu DROP, DELETE, ale také změny dat nebo vyžádání si chráněných informací nemají být prováděny běžným uživatelem z nechráněné části sítě, potažmo Internetu).

IP Spoofing – podvrhnutí IP adresy. Komunikující uzel předstírá, že je vlastníkem IP adresy, která ve skutečnosti patří jinému uzlu (nebo nepatří žádnému uzlu). Obvykle jde o případ, kdy se útočník snaží předstírat, že je řádným členem privátní sítě používající určitý rozsah IP adres, případně se pokouší vydávat za konkrétní uzel v síti.

Častým způsobem použití je zneužití některých protokolů, včetně protokolu SMTP pro odesílání e-mailů. Používá se také jako prostředek pro sofistikovanější útoky, například DoS (v paketech, které jsou zasílány na napadené zařízení, je zdrojová adresa podvržená).

Další typy spoofingu – podvržení identity se dá „spáchat“ i jinak, často se jedná o hacknutí některých tabulek s adresami, například:

- ARP Spoofing znamená podvržení záznamů v ARP tabulkách na zařízeních (je napaden mechanismus překladu IP adresy na MAC adresu),
- DNS Spoofing je podvržení záznamů v zónovém souboru na DNS serveru (je napaden mechanismus překladu jmenné adresy na IP adresu).

DoS (Denial of Service) – odmítnutí služby. Jde o vynucení odmítnutí služby legitimním uživatelům. Tento útok probíhá buď využitím chyby v kódu (některého protokolu nebo operačního systému), vyvolaným přetížením sítě nebo podvrženými zprávami – pakety o stavu sítě. Server je zahlcen žádostmi o spojení (TCP handshake) nebo žádostmi o data, které není schopen vyřídit, a proto i následný provoz je zadržen.

DDoS (Distributed DoS) – velmi nebezpečná varianta předchozího typu útoku, proti které se téměř nelze bránit. Častým nedobrovolným „účasníkem“ DDoS útoků jsou sítě botů. *Bot* je napadený počítač ovládaný na dálku hackerem (bez dovolení a většinou také bez vědomí právoplatného majitele). Sítě botů, a to i velmi rozsáhlé (resp. jejich služby), jsou „prodávány“ na černém trhu kromě jiného právě k DDoS útokům.

DDoS útok se dá provést i pomocí jinak celkem užitečných mechanismů. Například typ útoku *Ping Flood* (čte se [flad], znamená „záplava“) probíhá tak, že napadený stroj je zahlcen zprávami ICMP Echo Request (to jsou ty, které posílá příkaz `ping`). Tím se zahlcuje jak jeho vstupní kapacita (příjem), tak i jeho výstupní kapacita (odesílání) – když se pokouší odpovídat. Aby pachatel skryl svou identitu, v IP paketech zapouzdřujících tyto zprávy falšuje zdrojovou IP adresu. DDoS varianta útoku znamená, že ICMP zprávy posílají zařízení zapojená do botnetu.

Smurf Attack („šmoulí“ útok) má ještě vyšší míru distribuovanosti. Spočívá v tom, že velké množství zařízení (typicky serverů a routerů) dostává zprávy ICMP Echo Request, jejichž zdrojová adresa je podvržená – nastavená na adresu oběti. Tato zařízení odpovídají zprávami ICMP Echo Reply na adresu oběti, čímž zahlcují její vstupní kapacitu.

Man-in-the-Middle – hacker se dostane mezi dva komunikující počítače a odposlouchává nebo dokonce pozměňuje komunikaci mezi nimi. Tento typ útoku souvisí s některými předchozími – únos spojení, zjišťování hesla apod.

Hijacking (únos spojení) – jde především o útoky související s vytáčeným spojením, ale také obecně s jakýmkoliv placeným spojením vyžadujícím autentizaci (například soukromé Wi-Fi sítě). Účinnou obranou je použití vhodného šifrovacího algoritmu při autentizaci.

Útoky na zjištění hesla – heslo lze zjistit například brute-force útokem (hrubá síla), použitím trojského koně a některými výše popsanými metodami. Další možností je sociální inženýrství, které

je v současné době na vzestupu (uživatel defacto tento údaj prozradí sám a dobrovolně, je z něho podvodně vylákán).

Brute-force útok může znamenat zkoušení všech možných kombinací znaků v řetězci o „vhodné“ délce, ale může být veden jako *slovníkový*, tj. automaticky jsou zkoušeny všechny řetězce z vytvořeného slovníku (nemusí jít jen o anglická slova!). Proti tomu se lze bránit nastavením maximálního počtu neúspěšných přihlášení, po překročení tohoto limitu je přístup zcela zablokován. Dále je možné po uživatelích vyžadovat používání tzv. „silného“ hesla (dostatečně dlouhého, obsahujícího jak písmena, tak i číslice a další znaky – dvojtečka, procento, zavináč, apod.), s čímž ale bývají problémy (uživatelé raději volí takové heslo, které si dokážou zapamatovat).

Lidský faktor – „nepřítel“ může být i uvnitř sítě, a to dokonce i takový, který to o sobě netuší. Zvláště v poslední době jsou pro útoky často voleny *metody sociálního inženýrství*. Sociální inženýrství je metoda, jak z uživatele vytáhnout potřebné informace třeba i bez použití techniky, a to jeho uvedením v omyl (obelstěním). Uživatel je přesvědčen, že informace předává důvěryhodné osobě z naprosto nutných důvodů.

Útočník se vydává například za zaměstnance bezpečnostního oddělení, opraváře, zaměstnance telefonní společnosti, nového kolegu, apod. a nenápadně ze svých obětí vytáhne vše potřebné (hlavně hesla), případně si „vyzkouší“ nový skvělý počítač nebo se jinak dostane k technice na pracovišti. Stává se to především ve větších firmách, kde se nepočítá s tím, že by se všichni zaměstnanci znali, ale kontakt může probíhat i „neosobně“ přes telefon nebo mail. Právě do telefonu jsou zaměstnanci schopni říci o své firmě neuvěřitelné věci včetně těch, které jsou obchodním tajemstvím.

Popřípadě mnoho uživatelů nepochopitelně důvěřuje e-mailu: stačí například poslat e-mail s informací, že zřejmě došlo ke kompromitaci účtu, přičemž pro kontrolu je třeba zpět odeslat přihlašovací údaje, aby administrátor ověřil, zda je vše v pořádku (navíc e-mail musí být „správně“ graficky vyveden), a spousta uživatelů slepě poslechne.

Sociální inženýrství může mít i „materiální“ povahu – například volně „pohozená“ přenosná zařízení, která jsou pro mnoho lidí neodolatelná. Podle DHS[1] je kolem 60 % běžných uživatelů schopno náhodně nalezený USB flash disk připojit ke svému počítači, třebaže netuší, zda se na něm nenachází škodlivý software. Podobně lze zneužít i jiná přenosná zařízení, zde je nejlepší ochranou poctivost, tedy odevzdat nalezený předmět do ztrát a nálezů.

Sociální inženýrství je v současné době jedna z nejpoužívanějších (a taky nejefektivnějších) metod získávání utajených informací. Na to by měli myslet zejména administrátoři firemních sítí a zajistit patřičné školení všech, kdo se do firemní sítě připojují.

Jak se bránit?

- Nevěřit všemu, co kdo povídá. Zvláště když jsem například administrátor firemní sítě, musím si vše ověřovat (totožnost žadatele o nové heslo po jeho zapomenutí, nutnost provedení požadovaného zásahu do systému, apod.).
- Heslo či jiné podobné údaje si nenecháváme na papírku přilepeném k monitoru (nebo na jiných „obvyklých“ místech). Volíme silná hesla. Je vhodné nepoužívat totéž heslo u více systémů: pokud útočník prolomí heslo na jednom systému, obvykle se totéž heslo pokouší použít i jinde, kde zjistí účet oběti.

[1] DHS (Department of Homeland Security, http://www.dhs.gov)

- Každý systém zabezpečený heslem (včetně operačních systémů) lze nastavit tak, aby po stanoveném počtu chybných pokusů o přihlášení byl účet zablokován.
- Banky ani jiné instituce nechtějí po svých zaměstnancích zasílání hesel, certifikátů, TAN, čísla kreditní karty a podobných údajů mailem ani žádným jiným nezapezpečeným způsobem (jen přes zabezpečené stránky přímo při přihlašování). A rozhodně o ně nežádají mailem ani telefonem.
- Správce sítě by měl mít každou žádost o zásah do účtů zaměstnanců nebo systému vždy „papírově“ podloženou.
- Útočníci používající sociální inženýrství dělají „domácí úkoly“ – shánějí co nejvíc informací (včetně osobních), které by mohli využít. Firma by měla zvážit, co zveřejní (a totéž platí i o domácích uživatelích, také například na diskusních fórech a sociálních sítích). Skartovačka není zbytečné zařízení.

10.2 Síťový analyzátor

Síťový analyzátor je zařízení, které se připojuje mezi některý prvek sítě (PC, server, most, router, atd.) a LAN, také může být součástí jiného zařízení. Samostatný síťový analyzátor (přenosný) může být i velmi malý, může připomínat mobilní telefon.

Obrázek 10.1: Síťové analyzátory[2]

Pracuje na fyzické nebo vyšších vrstvách, na tom záleží, které charakteristiky dokáže sledovat. Na fyzické vrstvě především stav média, provoz (zatížení), dokáže generovat vlastní provoz, na vyšších vrstvách může podporovat také virtuální sítě, sledování stavu dostupných uzlů sítě, sledování rámců, příp. paketů, atd.

Síťový analyzátor „dosáhne“ pouze tam, odkud se k němu dostanou data. U uzlu sítě je tedy omezen na jedinou kolizní doménu (tj. k nejbližšímu přepínači nebo směrovači). Proto bývá vhodné umístit síťový analyzátor na takový uzel sítě, který se vyskytuje ve více kolizních doménách, například právě router.

Další informace:

Přehled některých síťových analyzátorů najdeme například na
http://www.fluketestery.cz/produkty-sitove-analyzatory.html.

[2] Zdroj: http://www.fluketestery.cz/

Hardware of firmy Fluke je v této oblasti všeobecně znám. Nejpoužívanější přístroje:

- *LinkRunner* – pracuje na fyzické vrstvě, identifikace vlastností Ethernetového spojení – negociace (10/100/1000 Mb, duplex apod.), detekce poruch kabelů včetně vzdálenosti k poruše, zásuvek, provoz na segmentu, atd.
- *NetTool* – síťová vrstva, testy na fyzické vrstvě (kabeláž, negociace, využívání segmentu), spojové (detekce kolizí a chybových rámců, VLAN, vyhledávání MAC adres v síti, apod.), síťové (vyhledávání IP adres v síti, podsítí, routerů a dalších zdrojů, ping, atd.), měření PoE, monitoring parametrů VoIP, atd.
- *AirCheck* – analýza bezdrátových sítí, včetně vytížení kanálů, zjišťování připojených zařízení a vyhledávání „černých“ zařízení.

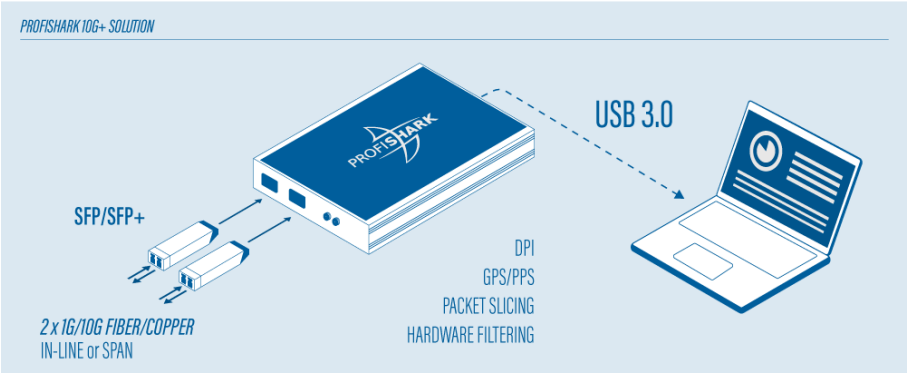
Hardware of firmy Embedded Technologies obsahuje obvykle vlastní variantu embedded Linuxu (tj. Linuxu upraveného pro speciální účely, zde pro síťová zařízení) zároveň s dalším potřebným softwarem. Například:

- *Síťový analyzátor* – vestavěný sniffer Wireshark, NAT, firewall, atd.
- *Diagnostický switch* pro Ethernet

Něco podobného si lze vytvořit vlastnoručně ze staršího (nadbytečného) počítače s potřebným množstvím hardwarových rozhraní, distribuce embedded Linuxu jsou dostupné na Internetu.

Softwarové síťové analyzátory: Síťové analyzátory mohou být také softwarové – sledují a případně ovlivňují síťový provoz související s počítačem, na kterém jsou nainstalovány (ideálně server, ale může to být kterýkoliv počítač v síti). Asi nejznámější softwarový síťový analyzátor je *Wireshark* (dříve se nazýval Ethereal) dostupný na http://www.wireshark.org/.

Network Tap (síťový odposlech) je speciální analyzátor, jehož úkolem je „sbírat“ provoz na daném síťovém segmentu a (obvykle) přeposílat na jeden či dva vybrané porty (u novějších i například na USB výstup). Může fungovat jako IDS, procházet všechny pakety, nebo může sledovat jenom konkrétní typ paketů (třeba multimediální, případně pro konkrétní protokol), bývá začleněn do systému ochrany sítě.



Obrázek 10.2: Schéma zapojení síťového odposlechu[3]

První podmínka se zdá jednoduchá, ale je třeba si uvědomit, že požadavek „v obou směrech“ znamená vlastně dvojnásobek běžného provozu, protože dnes se komunikuje v plném duplexu. Plně duplexní provoz musíme do odposlechu dostat „v jednom směru“ (směrem k odposlouchajícímu počítači).

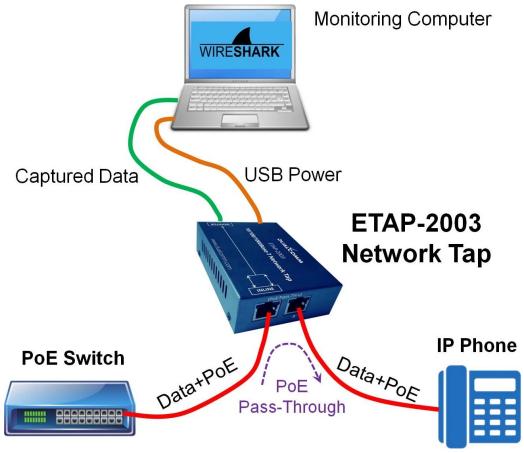
[3] Zdroj: https://www.profitap.com/profishark-10g-plus/

Zařízení určená pro použití v sítích s velkým provozem mívají běžně pro tento směr minimálně dva porty, pro každý původní směr komunikace jeden.

Na obrázku 10.2 vidíme příklad zapojení síťového odposlechu v síti. Toto zařízení má:

- vstupy vedoucí ze sousedících zařízenení, na obrázku jde o SFP moduly (například pro ethernetové optické kabely),
- výstup(-y), to může být jedno nebo několik síťových rozhraní, u novějších obvykle USB rozhraní vyšší verze (novější verze USB už mají dostatečnou propustnost, ale samozřejmě záleží na datovém toku mezi sledovanými zařízeními),
- napájení – obvykle bývá realizováno přes výstup (pokud jde o kroucenou dvojlinku, pak PoE, pokud jde o USB, pak se používají napájecí vodiče v USB kabelu, případně lze použít přídavný USB kabel pro napájení).

Na spoji vedoucím mezi sledovanými zařízeními taky může být používáno PoE. V tom případě by se tap neměl „přiživovat“, napájení by měl transparentně přeposílat, jak vidíme na obrázku 10.3.



Obrázek 10.3: Schéma zapojení síťového odposlechu s PoE[4]

Další informace:

Další informace o síťových analyzátorech najdeme na

- http://www.root.cz/serialy/pruvodce-programem-ethereal/
- http://knihy.cpress.cz/DataFiles/Book/00003877/Download/kapitola_3_K1599.pdf
- http://www.proficomms.cz/index.php?module=shop_catalog&action=view&category_id=60&id=51
- http://www.etech.cz/cs/products/lan.htm

  https://www.profitap.com/profishark-10g-plus/ https://www.garlandtechnology.com/network-tap

  https://www.dualcomm.com/collections/network-tap
- http://hrazdil.info/school/pds-sitovy-analyzator/
- http://computerworld.cz/testy/nastroje-pro-analyzu-provozu-wlan-proveri-vase-pakety-1-4555
- http://www.trinstruments.cz/clearsight-analyzer

[4] Zdroj: https://www.dualcomm.com/collections/network-tap

10.3 Firewall

10.3.1 Princip firewallu

Firewall je síťový prvek (hardwarový nebo softwarový), který slouží k řízení provozu mezi sítěmi s různou úrovní důvěryhodnosti či zabezpečení. Ve firewallu jsou definována *pravidla* pro komunikaci mezi sítěmi, podle kterých reaguje buď povolením komunikace, jejím zakázáním a nebo žádostí o vyjádření uživatele. Pravidla se obvykle týkají těchto údajů:

- zdroj a cíl komunikace, může být zadán IP adresou,
- číslo portu, přes který se komunikuje (tj. můžeme zablokovat používání některého portu), může jít o zdrojový i cílový port,
- používaný protokol,
- stav spojení, atd.

Firewall může být buď jen *jednosměrný* (filtruje pouze příchozí pakety) nebo *obousměrný* (filtruje příchozí i odchozí pakety). Lepší je samozřejmě obousměrný, dokáže efektivněji zachytit případné vynášení citlivých informací.

Kromě softwarových firewallů existují také *hardwarové firewally* fungující jako mezilehlé prvky sítě. Dnes jsou většinou součástí směrovačů (routerů) nebo jiných běžných síťových prvků (na typu zařízení závisí, k jakým informacím se firewall dostane). Hardwarový firewall má velkou výhodu v nižším riziku napadení (není závislý na operačním systému klientského počítače). Další výhodou je nezávislost na výkonu počítače – pokud je procesor počítače hodně zatížen, má to vliv i na činnost (především rychlost) softwarového firewallu na tomto počítači nainstalovaného. Hardwarový firewall (třeba vestavěný v jiném síťovém zařízení) takto ovlivněn není.

Z hlediska bezpečnosti je v domácnostech a menších firmách za ideální považována kombinace jednoduchého hardwarového firewallu ve směrovači (například ADSL router) a softwarového routeru na počítači, každý z nich jiného typu.

Softwarové firewally slouží buď k ochraně koncových uzlů, a nebo mohou běžet v operačním systému nainstalovaném na (téměř) jakémkoliv síťovém prvku.

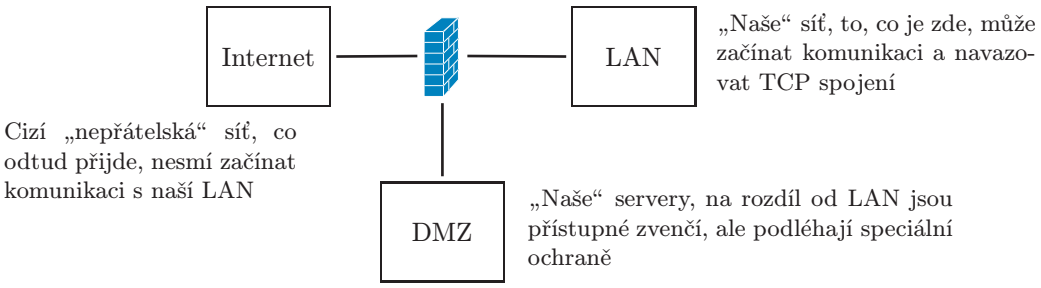
Oblasti podle firewallu. Z pohledu firewallu členíme síť do několika oblastí:

- *vnitřní síť* – do této oblasti patří vše, co je „naše“, čemu můžeme důvěřovat, uzel z vnitřní sítě může (s ohledem na přístupová oprávnění) iniciovat spojení k vnitřní i vnější síti,
- *vnější síť* – typicky Internet,
- *demilitarizovaná zóna* (DMZ) – zóna „nikoho“, z bezpečnostního hlediska někde mezi vnitřní a vnější sítí.

Do DMZ lze umístit například to, co je sice „naše“, ale má být přístupné z vnější sítě (například web server, mail server, DNS server). Servery v DMZ by měly být zabezpečeny tak, jako by byly opravdu přímo na Internetu, třebaže jsou od Internetu odděleny firewallem.

Další možnost využití DMZ je propojení k třetí straně, které víceméně důvěřujeme, typicky dodavateli služby, kterou si nemůžeme zajistit vlastními silami. Obě možnosti můžeme zkombinovat a používat dvě demilitarizované zóny (nebo více).

Na síťovém zařízení podporujícím DMZ máme obvykle jeden nebo více (hardwarových) portů takto označených, případně můžeme některé porty sami nakonfigurovat tak, aby se s nimi zacházelo jako



Obrázek 10.4: Oblasti z pohledu firewallu

s DMZ (například tehdy, když chceme starší počítač využít jako hardwarový firewall, ukázky najdeme v příloze).

TCP/UDP porty. Běžný uživatel si většinou s nastavením (softwarových) portů neví rady (viz část první kapitoly věnovanou TCP/UDP paketům). Na internetu najdeme stránky se seznamy známých a registrovaných portů používaných různými protokoly.[5] Tyto seznamy jsou však použitelné pro sledování odchozího provozu nebo při nastavování portů na serveru. Ve skutečnosti aplikace mohou používat i jiné porty, než které jsou běžné pro protokoly, se kterými pracují.

10.3.2 Typy filtrování

Rozlišujeme různé typy filtrování podle toho, na kterou vrstvu ISO/OSI lze danou metodu zařadit (a tedy na které informace v záhlavích „dosáhneme“). Obvykle se jedná především o filtrování na síťové vrstvě (L3), protože tam lze pracovat s IP adresami.

Paketový filtr. Jedná se o nejjednodušší filtrování na L3 a částečně L4, v pravidlech se uvádějí jen IP adresy a čísla portů. Je to jednoduché a rychlé řešení (provoz je zdržován jen minimálně), které se dříve běžně uplatňovalo především na mezilehlých síťových prvcích (například starší verze operačního systému IOS pro směrovače), dnes je najdeme v některých nejlevnějších směrovačích. Nevýhodou je neschopnost nahlížet do komunikace probíhající v složitějších protokolech.

ACL na síťové vrstvě. ACL (Access Control List – seznam řízení přístupu, přístupový seznam) je metoda široce používaná jak v desktopových a serverových operačních systémech, tak i v síťových zařízeních. Účelem je určovat pravidla přístupu pro různé uživatele/komunikace. Vpodstatě se jedná o funkční nástavbu metody paketového filtru.

Přístupový seznam je vlastně seznam položek označených kategorií:

- *deny* (nepustit) – to, co nemá projít,
- *permit* (pustit) – to, co může projít,
- *deny all else* (nepustit nic kromě...) – co nebylo zmíněno, nesmí projít.

ACL je většinou implementován ve formě „co není dovoleno, je zakázáno“, takže najdeme spíše jen položky *permit* (a co není v těchto položkách, to neprojde). Případně se můžeme setkat s celou strukturou navzájem provázaných ACL. Položky se procházejí sekvenčně, jedna po druhé, a hledá se shoda. Také pořadí položek je důležité.

[5] Seznam portů a případně služeb najdeme například na http://www.iana.org/assignments/port-numbers, http://en.wikipedia.org/wiki/List_of_TCP_and_UDP_port_numbers, http://www.ports-services.com/. Na české Wikipedii není seznam úplný.

Filtruje se obvykle podle cílové IP adresy, podle zdrojové IP adresy, podle jejich kombinace, a nebo případně podle dalších kritérií (například podle položek v PDU síťové vrstvy – protokolu IP, ICMP, podle TCP/UDP portu, se kterým se navazuje spojení ze síťové vrstvy, apod.). Adresa obvykle bývá adresou podsítě, tedy je uložen prefix a jeho délka (resp. maska, aby bylo zřejmé, u kolika bitů adresy se má hledat shoda).

Další informace:

Ukázky ACL najdeme například na

- http://skola.sssbb.sk/~badani/cisco/semester2/Sem2-11%20ACL.pdf
- http://www.cs.vsb.cz/grygarek/PS/projekt0304/acl_priklad.pdf
- http://www.samuraj-cz.com/clanek/cisco-ios-18-inter-vlan-routing-a-acl-smerovani-mezi-vlany/

Stavová inspekce paketů. Přesuneme se o vrstvu výše – na transportní vrstvě (L4, konkrétněji teď jsme na L3 a L4) se provádí filtrování *SPI* (Stateful Packet Inspection, stavová inspekce paketů, také stavový paketový filtr). Zatímco na L3 se pakety berou jako „jedináčci“ bez vzájemné vazby, na L4 je možné brát při filtrování v úvahu jejich vzájemné vztahy. například můžeme odlišit pakety, které navazují spojení, od paketů, které patří do již existujícího spojení.

Paket navazující spojení je důkladně prověřen (údaje z vrstev L3 a L4, například zdrojová a cílová adresa, protokoly, zdrojové a cílové porty, příznaky nastavené podle jednotlivých protokolů, cokoliv, co je v záhlavích PDU) a pokud projde, vytvoří se v *stavové tabulce* záznam povolující dané spojení. Pokud paket kontrolou neprojde úspěšně, záznam se nevytvoří.

Pakety, které patří do již navázaného spojení (nebo se za takové vydávají), se filtrují podle toho, zda jejich spojení je zaznamenáno ve stavové tabulce.

V současné době je SPI vpodstatě standardem kvalitních firewallů. Oproti běžnému paketovému filtru nabízí větší bezpečnost při relativně malém zpomalení provozu při filtrování.

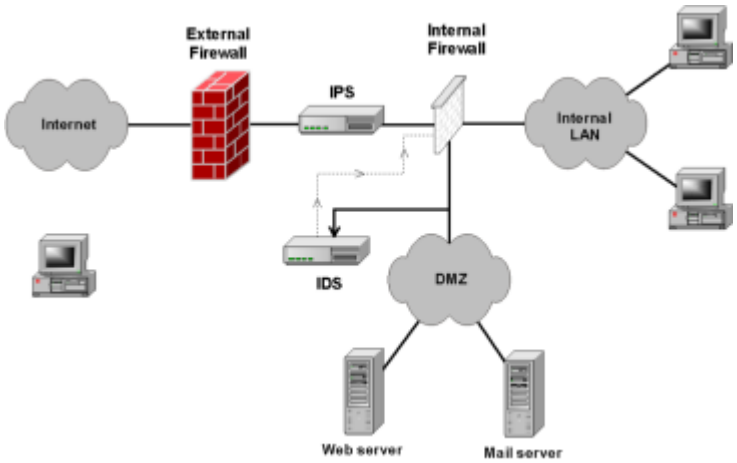
IDS/IPS, DPI. SPI můžeme brát jako základ pro IDS/IPS systémy:

1. *IDS* (Intrusion *Detection* System, systém pro detekci útoků) – pracuje podobně jako antivirus, tedy používá
   - signatury útoků pro rozpoznání známých typů útoků (vede si jejich databázi, kterou je nutné aktualizovat nebo „mít k dispozici v cloudu“),
   - heuristiky (využívá statistické metody vyhodnocující provoz na síti) – hledá v provozu na síti podezřelé pakety,
   - detekci neobvyklého chování sítě (zjišťují se odchylky od běžného provozu na síti).

   Detekuje pokusy o průnik do systému a podá informaci zařízení, které dokáže na útok reagovat.
2. *IPS* (Intrusion *Prevention* System) – podobně jako IDS provádí detekci útoků, ale navíc aktivně reaguje. Reakce má útoku zabránit, proto také konkrétní místo IPS sondy je třeba naplánovat tak, aby mohla případně také zasahovat do konfigurace jiných síťových zařízení (například přenastavit pravidla ve firewallu).

Firewall může mít integrovanou funkci (modul) IDS/IPS, ale obecně je bezpečnější (zvláště u větších sítí) nasadit IDS/IPS odděleně od firewallu. Setkáváme se také s názvem *Stavový paketový filtr s hloubkovou*

kontrolou (Deep Packet Inspection), to je právě firewall s integrovaným modulem IDS/IPS. Tento typ firewallu přidává další možnosti definování pravidel související s obvyklými vlastnostmi komunikace s danou (známou) aplikací či protokolem. Pokud například zjistí, že protokol HTTP je používán pro jiný typ komunikace než s WWW serverem, tento požadavek zablokuje jako podezřelý.



Obrázek 10.5: Schéma možného zapojení IDS/IPS[6]

Na obrázku 10.5 vidíme možné schéma zapojení firewallů a IDS/IPS ve větší síti. Celou síť chrání externí firewall, za kterým je IPS – přes IPS jde veškerá komunikace, tedy může reagovat na cokoliv podezřelého v kterémkoliv paketu. Následuje vnitřní firewall, který rozděluje síť na oblasti tak, jak bylo výše naznačeno – odděluje vnější „nedůvěryhodnou“ síť (Internet), vnitřní (relativně důvěryhodnou) lokální síť a demilitarizovanou zónu pro servery. DMZ má dodatečnou ochranu ve formě IDS zařízení.

Poznámka:

Všimněte si, že IDS není „přímo na cestě“ k DMZ, ale vede k němu „odbočka“. Provoz směřující do DMZ (nebo opačně) je zrcadlen (kopírován) na port vedoucí k IDS zařízení. Proč tomu tak je? Protože nechceme, aby zařízení IDS „bylo vidět“. Pokud se útočníkovi podaří vloudit se do sítě a začne skenovat naši síť (zjišťovat, kde co máme, jakou co má adresu apod.), IDS pro něj zůstane neviditelným ze dvou důvodů – pouze detekuje (nedává o sobě vědět žádnými aktivními reakcemi) a zároveň není na cestě k žádnému síťovému prvku. Takže máme v síti skrytý prostředek, který můžeme následně použít proti útočníkovi, resp. sledovat jeho činnost bez toho, aby to útočník zpozoroval.

Proč máme IDS právě u DMZ? Protože DMZ je zranitelnější než vlastní lokální síť. Zatímco směrem do LAN nesmí být z Internetu navazována žádná spojení (cokoliv takového na firewallu hned „zařízneme“, u DMZ to udělat nesmíme, protože potřebujeme, aby na naše servery přistupovali klienti z vnějšku. Takže ochrana DMZ je složitější a každý ochranný prvek navíc se hodí.

Další informace:

- http://www.actinet.cz/bezpecnost_informacnich_technologii/l19/cl25/st1/j1/Uvod_do_IDS/IPS.html
- http://www.symantec.com/connect/articles/intrusion-detection-terminology-part-one

[6] Zdroj: http://www.actinet.cz

- http://www.symantec.com/connect/articles/intrusion-detection-terminology-part-two
- http://www.systemonline.cz/clanky/systemy-pro-detekci-neopravneneho-pruniku.htm

Proxy na aplikační vrstvě. Posuneme se ještě o vrstvu výše – na aplikační vrstvě TCP/IP pracují proxy firewally (také aplikační brány). Pracují s aplikačními protokoly, například rozumí protokolu HTTP, FTP, IMAP, dokážou pracovat s pakety obsahujícími prvky pro ActiveX, apod.

Tento typ firewallu odděluje sítě až do té míry, že počítač (server) ve vnější síti nezná IP adresu počítače ve vnitřní síti, se kterým komunikuje. Veškerá komunikace je zpracovávána pomocí tzv. *proxy* – softwarových bran buď naprogramovaných pro konkrétní typ komunikace (protokol) s poměrně vysokým stupněm zabezpečení (například pro protokol FTP nebo HTTP), nebo pomocí generické proxy použitelné obecně pro různé protokoly.

Proxy defacto zcela odděluje vnitřní a vnější síť (v podobném smyslu jako NAT) a při filtrování využívá informace prakticky ze všech vrstev TCP/IP, protože při „prokopávání“ k záhlaví protokolů vrstvy L7 prochází přes záhlaví předchozích vrstev. Rozlišujeme dva typy proxy:

1. *Běžný (standardní) proxy* – pro něj platí vše, co bylo dříve o proxy napsáno. Filtruje všechny pakety podle údajů z PDU protokolů aplikační vrstvy a nižších vrstev.
2. *Dynamický proxy* – chová se odlišně k různým paketům; k zahajujícím spojení se chová stejně jako první typ proxy, ale k paketům patřícím do již vytvořeného spojení se chová jako SPI (tj. na nižší vrstvě TCP/IP). Důsledkem je zrychlení odbavování příchozího i odchozího provozu.

Další informace:

V přílohách najdeme sekce o firewallu ve Windows a v Linuxu včetně potřebných příkazů pro konfiguraci. Na straně 298 je sekce o firewallu ve Windows, na straně 327 je sekce o firewallu v Linuxu (velmi podrobná včetně příkladů pravidel).

Další informace o firewallech:

- http://www.actinet.cz/bezpecnost_informacnich_technologii/l19/cl44/st4/j1/Soucasnost_a_trendy_reseni_firewallu.html
- http://www.actinet.cz/bezpecnost_informacnich_technologii/l19/cl25/st1/j1/Uvod_do_IDS/IPS.html
- http://www.root.cz/serialy/openwrt/
- http://bravenec.org/cs/clanky/openwrt/openwrt1
- http://bravenec.org/cs/clanky/openwrt/openwrt2
- http://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=15131

Úkol

V příloze od strany 327 je sekce o firewallu *NetFilter*, ve které jsou na příkladech vysvětleny nejdůležitější funkce firewallů. Projděte si dotyčnou sekci a ujasněte si princip filtrování, SPI, překladu adres a dalších.

10.3.3 OpenWRT

Zajímavým projektem je OpenWRT. Jedná se o embedded Linux (tj. Linux upravený pro určitý konkrétní typ zařízení, typicky firmware takového zařízení) pro routery některých výrobců, například firmy Mikrotik (RouterBoard). Jeho výhodou oproti „originálnímu“ operačnímu systému (například u desek RouterBoard je to RouterOS taktéž založený na Linuxu) je to, že má k běžným linuxovým distribucím mnohem blíže než jiné embedded Linuxy. Zatímco obvykle se firmware aktualizuje jako celek (to může být celkem nebezpečné, dotyčné zařízení se při poruše při aktualizaci může zcela odrovnat), OpenWRT je pružný systém, který lze aktualizovat, konfigurovat a jakkoliv měnit prakticky za provozu a můžeme při tom používat jak běžný balíčkovací systém, tak při konfiguraci třeba textový editor.

Jak bylo výše zmíněno, OpenWRT používá balíčkovací systém a tedy lze doinstalovat různé balíčky podle potřeby, výběr je veliký. Do distribuce patří samozřejmě firewall NetFilter s přístupovým programem iptables, podporuje ftp, samba, telnet, ssh, lze doinstalovat X Window a nějakého vhodného správce oken, konfigurujeme obvykle přes ssh (případně lze přes ssh bezpečně provozovat grafické prostředí). Ve výchozím stavu je tato distribuce vybavena typicky pro Wi-fi router, což ale můžeme změnit podle svých vlastních potřeb.

10.4 VPN

VPN (Virtual Private Network) je zabezpečené spojení procházející nedůvěryhodným prostředím (obvykle Internetem). Jedná se o vytvoření tunelu, komunikačního kanálu mezi dvěma body (to mohou být konkrétní koncová zařízení nebo třeba routery, pak propojujeme sítě).

Používá se v těchto případech (tj. dělení podle typu ukončujících bodů):

- *Remote Access*: mobilní zaměstnanec potřebuje na svých cestách zabezpečený přístup do firemní (lokální) sítě, aby mohl přistupovat do firemního informačního systému a dalších zabezpečených zdrojů, nebo se jedná o zaměstnance pracujícího doma (Home Office).
- *Site-to-Site* (nebo také *Network-based*): je třeba propojit vzdálené lokální sítě (například pobočky téže firmy).
- Je třeba komunikovat s obchodním partnerem zabezpečeným komunikačním kanálem, ale zároveň ho nechceme pustit přímo do naší lokální sítě. Může se jednat o *site-to-site* nebo *remote access*, podle skutečné konfigurace VPN, tato možnost je vlastně speciálním případem obou předchozích.

Ve všech těchto případech je třeba vybudovat zabezpečený šifrovaný *tunel*, přes který povede komunikace nedůvěryhodným prostředím – zajišťujeme předně vhodné zapouzdření s případným překladem adres (protože pakety půjdou přes cizí síť s jinými adresními rozsahy a případně jinými protokoly), a dále zajišťujeme šifrování (na to je někdy potřeba použít přídavné řešení). Ve třetím případě navíc musíme použít firewall, který bude propouštět pouze komunikaci povolenou pro tento účel.

VPN tunel do firemní sítě ústí většinou buď přímo na routeru s firewallem na hranici lokální sítě, který je viditelný na Internetu, anebo v demilitarizované zóně (tam se často umísťuje VPN koncentrátor, což je vlastně jakási VPN brána), záleží, kam v lokální síti je třeba přes tunel přistupovat.

VPN řešení by mělo zajistit následující:

- autentizace – je třeba ověřit totožnost obou komunikujících bodů (např. uživatel s notebookem na pracovní cestě a firewall s podporou VPN v LAN firmy), případně se zajišťuje autentizace přenášených PDU,

- autorizace – stanovení konkrétních přístupových oprávnění,
- zajištění důvěrnosti dat – přenos je vždy šifrován,
- zajištění integrity dat – detekce pozměnění či poškození paketu po cestě.

Tunelovací problémy. Někdy bývá problém se zprovozněním tunelu – všechno se zdá v pořádku, ale v případě, že má být přenesen větší paket, nic nedorazí a do zdroje nepřijde ani ICMP zpráva se zdůvodněním. Důvodem je právě velikost paketu, která je příliš blízko hodnotám MTU na cestě. Nezapomeňte, že příliš velký paket je cestou buď fragmentován nebo zahozen (když se nedá fragmentovat), a přitom VPN obvykle funguje tak, že se původní paket/rámec zapouzdřuje do obalového a nosného paketu – přidávají se další záhlaví, velikost paketu naroste.

Většinou se na síťových zařízeních počítá s tím, že by měly projít rámce zapouzdřující maximálně 1500 B, a s tím taky počítá odesílající koncové zařízení – obvykle už na transportní vrstvě se nastaví maximální velikost segmentu tak, aby po přidání IP záhlaví tato hodnota nebyla překročena (až delší PDU jsou rozděleny do více segmentů). Jenže odesílatel „neví“ o tom, že jeho dílo nabobtná tunelováním.

Dobře, a proč se tedy nefragmentuje? IPv6 se po cestě fragmentovat nesmí, a u IPv4 se často používá mechanismus *Path MTU Discovery*, který je sice užitečný (zjistí MTU po celé cestě a už před odesláním upraví paket tak, aby prošel na všech prvcích), ale automaticky zakazuje fragmentaci po cestě (nastaví příznak DF v záhlaví IP paketu). O problému se často řádným způsobem nedozvíme, protože o zahození nefragmentovatelného paketu je odesílatel informován zprávou ICMP Destination Unrecheable, kód 4, a zároveň na některých routerech jsou paušálně zahazovány všechny ICMP zprávy (ano, takové jsou pak následky).

Takže jaké je řešení? Například můžeme na transportní vrstvě určit menší hodnotu pro maximální délku segmentu. Vezmeme obvyklé minimum MTU (1500) a odečteme délku všech záhlaví, která se během zapouzdřování přidávají. Například u GRE se dá použít hodnota 1220.

Dále bude představeno několik běžných VPN řešení. Jedná se o protokoly pracující na různých vrstvách ISO/OSI, další odlišnosti jsou v náročnosti nasazení, (ne)nutnosti mít externího poskytovatele služby, a také v tom, zda je v řešení zahrnuto i šifrování a další bezpečnostní mechanismy.

Některé VPN protokoly jsou typu *multipoint* (je možné sestavit síť tunelů typu mesh, kde každý bod může přímo nebo zprostředkovaně komunikovat s více dalšími body) nebo typu *point-to-point* (každý tunel musí být nakonfigurován zvlášť, všechny tunely mají právě dva konce).

10.4.1 IPSec VPN – na síťové vrstvě

Protokol IPSec (Internet Protocol Security) umožňuje vytvářet zabezpečené tunely typu point-to-point. Zajišťuje jak vytvoření tunelu, tak i šifrování. Pracuje na síťové vrstvě, díky tomu je transparentní pro aplikační protokoly. Používá se jak pro řešení site-to-site, tak i pro remote access.

Tunel se vytváří zapouzdřením takto:

- uvnitř je PDU přenášeného protokolu, většinou IP (ale může být také IPX, NetBEUI nebo jiný), nebo jenom jeho datová část, přičemž záhlaví se použije v třetím kroku,
- následuje obalový protokol, což bývá obvykle IPSec (může být např. GRE, PPTP, L2TP nebo jiný), zapouzdřená PDU je zašifrována a je přidáno bezpečnostní záhlaví,
- nosný protokol síťové vrstvy (obvykle IP) v sobě zapouzdří PDU vytvořenou v předchozím kroku, aby bylo možné přenést paket přes „nechráněný“ Internet či jinou veřejnou síť.

Bonusem je možnost takto zapouzdřit i takové protokoly, které nejsou běžně ve vnějším prostředí podporovány, případně zajistit využívání soukromých IP adres (vnitřní IP PDU je adresován soukromou cílovou IP adresou, ale aby bylo možné celou PDU doručit, vnější IP datagram má veřejnou cílovou IP adresu hraničního zařízení firemní sítě podporujícího VPN).

IPSec se také často kombinuje s protokoly GRE nebo L2TP pracujícími na vrstvě L2. Jedním z důvodů těchto kombinací je problém IPSec při průchodu přes NAT, který modifikuje IP záhlaví. I pro tento případ však existuje řešení, lze využít mechanismus *NAT-T* (NAT Traversal), který zapouzdří paket do UDP a tím znemožní pozměnění IP záhlaví. Označuje se IPSec over UDP nebo IPSec over NAT-T.

Šifrování a zapouzdření v IPSec se provádí jedním z těchto dvou způsobů:

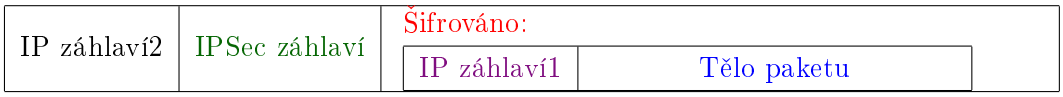
- *tunelovací režim* – celý původní IP paket se zašifruje a připojí se záhlaví IPSec (a pak samozřejmě záhlaví nosného protokolu s novou IP adresou),
- *transportní (přenosový) režim* – šifruje se datová část zapouzdřovaného paketu bez záhlaví, pak se přidá IPSec záhlaví, před ně se předstune původní záhlaví zapouzdřených dat.

Tunelovací režim je bezpečnější (celé zapouzdřené IP záhlaví je skryto, šifrováno), transportní režim je pružnější (můžeme používat prvky z původního IP záhlaví, například pro QoS) a pakety jsou kratší (tedy menší dopad na propustnost sítě). Srovnání vidíme na obrázku 10.6.

Tunelovací režim je používán především při spojeních *site-to-site*, zatímco transportní režim je typičtější pro připojení vzdáleného počítače s firemní sítí (*remote acces*).

Původní IP paket:

- *tunelovací režim*:



- *transportní (přenosový) režim*:

Obrázek 10.6: PDU před a po zašifrování do IPSec tunelu

Technicky vzato, existují dva typy *IPSec záhlaví*: AH (Authentication Header) a ESP (Encapsulated Security Payload). Zatímco AH je jednodušší, zajišťuje pouze autentizaci a integritu dat, ale ne šifrování, záhlaví ESP je bezpečnější (zajišťuje autentizaci a šifrování). V současné době se používá obvykle jen ESP (lze použít dokonce obojí najednou, tedy AH/ESP).

Při vytváření jakéhokoliv (tedy i IPSec) tunelu je třeba nejdřív dojednat *parametry bezpečného přidružení* (SA – Security Association). IPSec pro tento účel používá *protokol IKE* (Internet Key Exchange), nyní ve verzi 2.

Poznámka:

V protokolu IPv6 je již IPSec nativně implementován (tak to bylo už v původním návrhu IPv6, k verzi

IPv4 byl IPSec dodatečně přidán jako „pomocný“ protokol zajišťující šifrování). Proto přechod na IPv6 má jeden bonus – snadnější vytváření IPSec tunelů.

Jak se IPSec konfiguruje. IPSec se konfiguruje podobným způsobem jako například firewall – definujeme pravidla a další parametry (v tzv. tabulce politik). Určujeme například, že pakety příchozí z konkrétní adresy (vzdálené pobočky) se mají zpracovat mechanismem IPSec (dešifrovat apod.), pakety odchozí na tutéž adresu naopak zašifrovat, přidat IPSec záhlaví apod., pakety směřující do vnitřní sítě se mají nechat tak jak jsou, atd. Také je obvykle potřeba upravit směrovací tabulku.

V Linuxu záleží na konkrétní distribuci. V distribucích odvozených z Debianu bývá k dispozici (nebo se doinstaluje) balíček *Freeswan*, kde se konfigurace provádí v souboru `/etc/ipsec.conf` a příkazem `ipsec`, v jiných distribucích máme balíček *IPSec Tools* s příkazy `setkey` a `racoon`, konfigurační soubory jsou `/etc/racoon/racoon.conf` a `/etc/racoon/setkey.conf`.

Ve Windows a na síťových zařízeních konfigurovaných přes webové rozhraní se konfigurace provádí většinou „myší“, například ve Windows Server máme k dispozici konzolu v *Start – Programs – Administrative Tools – Local Security Settings*. Kromě toho je ve Windows Server k dispozici program `ipsecpol` pro definování IPSec politik (zásad).

Na klientovi je připojení vcelku jednoduché. VPN je předem nakonfigurována na serveru, na klientovi vpodstatě konfigurujeme nové připojení k síti: *Vytvořit nové připojení*, zvolíme připojení k firemní síti, VPN, atd. V průvodci potřebujeme IP adresu VPN serveru, se kterým budeme komunikovat. Ve vlastnostech připojení dále nakonfigurujeme protokol IPSec, včetně bezpečnostních nastavení.

Další informace:

- Podrobnosti především o Linuxu najdeme v odkazech na konci sekce o VPN, především v odkazu http://www.ipsec-howto.org/ipsec-howto.pdf
- Konkrétní postup konfigurace na Linuxu je v https://www.root.cz/clanky/tuneluji-tunelujes-tunelujeme-ipsec/, je to jeden z dílů seriálu o tunelování.

10.4.2 GRE tunely

Protokol *GRE* (Generic Routing Encapsulation) pracuje na síťové vrstvě a vytváří point-to-point tunely, typicky site-to-site. Je standardizován IETF jako RFC 2784.

Obvyklý postup je takový, že původní paket je opatřen GRE záhlavím (velice jednoduchým, pár polí) a následně je přidáno záhlaví nosného protokolu, obvykle IP (v něm je v poli Protocol/NextHeader číslo 47 určující protokol GRE). Všimněte si, že v postupu není ani slovo o šifrování, to totiž neumí.

Jeho výhodou je univerzálnost, dokáže zapouzdřit i jiné typy PDU síťové vrstvy ISO/OSI než jen IP. Dokonce může zapouzdřit i protokoly jiných vrstev, včetně rámců z vrstvy L2. Dokáže přes tunel transportovat i multicast a broadcast vysílání. Další výhodou je, že tento protokol je podporován prakticky všude (v Linuxu, ve Windows, na síťových zařízeních různých výrobců).

Na druhou stranu, velkou nevýhodou GRE je, že neprovádí šifrování, proto se ve skutečnosti nejedná o „pravý“ VPN tunel. V praxi je GRE zapouzdřován do tunelů VPN (například v kombinaci s IPSec), aby byla konverzace zároveň šifrována.

Konfigurace v Linuxu je jednodušší než IPSec, vystačíme si s vestavěnými nástroji (příkaz `ip tunnel` a další varianty příkazu `ip`).

V příloze na straně 317 je ukázka nastavení jednoduchého GRE tunelu mezi dvěma vzdálenými sítěmi (mezi routery s nainstalovaným Linuxem).

Další informace:

- https://www.ietf.org/rfc/rfc2784.txt
- https://www.root.cz/clanky/tuneluji-tunelujes-tunelujeme-jak-a-k-cemu/

Úkol

V příloze si projděte postup konfigurace GRE tunelu.

10.4.3 L2TP tunely

Protokol L2TP je potomkem starších protokolů PPTP (Microsoft) a L2F (Cisco). Je standardizován organizací IETF – verze L2TPv3 z roku 2005 je standard RFC 3931.

Jak název napovídá, tento protokol pracuje na vrstvě L2 (Layer 2 Tunneling Protocol). VPN protokoly linkové vrstvy jsou zajímavé tím, že obvykle dělají „smyčku“ na vyšší vrstvu – zapouzdřují provoz z nadřízené vrstvy, ale samy se zapouzdřují do TCP nebo UDP segmentu.

Hlavním účelem protokolu *L2TP* je tunelování (zapouzdřování) paketů protokolu PPP (ten se používá pro přenos dat přes telefonní síť, včetně ADSL/VDSL), je velmi oblíbený u různých ISP. Zapouzdřuje se do UDP segmentů, nepoužívá TCP (tunel tedy nemá oporu na vrstvě L4).

Stejně jako GRE, ani L2TP neprovádí šifrování, tedy je obvykle kombinován s IPSec v transportním módu, což je standardizováno v RFC 3193. Takže k paketu, který má být takto přenesen:

- se nejdřív na L2 přidá PPP záhlaví a zápatí (PPP zajistí navázání a kontrolu průběhu spojení, základní autentizaci, v záhlaví je identifikace zapouzdřeného protokolu, v zápatí kontrolní součet), ale tento krok není povinný (L2TP dokáže zapouzdřit i IP pakety),
- pak se přidá záhlaví L2TP (v záhlaví jsou parametry pro tunel a relaci a další údaje),
- tento L2TP rámec se zapouzdří do UDP (takže smyčka nahoru),
- dále se může přidat IPSec záhlaví (plus zápatí) a následně se přidá záhlaví nosného protokolu, obvykle IP.

L2TP je podporován v Linuxu i ve Windows. V Linuxu se dnes doporučuje použití OpenL2TP (s využitím IPSec), ve Windows se nachází klient L2TP/IPSec pro transportní mód.

Poznámka:

Ve Windows se také můžeme setkat s protokolem SSTP (Secure Socket Tunneling Protocol), který přenáší PPP nebo L2TP šifrované s využitím protokolu SSL.

10.4.4 OpenVPN

Zajímavou implementací VPN je projekt *OpenVPN*. Běží na většině známých UNIXových systémů včetně Linuxu, existuje i varianta pro Windows. Je to otevřené řešení pro remote access, které zvládá jak vytvoření tunelu, tak i šifrování, šifrují se pouze přenášená data.

Toto řešení pracuje na vyšších vrstvách (nad L3), což znamená, že například není třeba řešit problémy s NAT či jinými překlady IP adres. Pro šifrování, autentizaci a správu klíčů se používá protokol SSL nebo TLS, na transportní vrstvě se používá většinou UDP, ale je možné použít i TCP. Protokol TLS je sice standardizován, ale celé řešení OpenVPN nikoliv.

Zatímco IPSec je síťové řešení (transparentní pro různé aplikace), SSL/TLS je aplikační řešení (tj. musí být podporováno konkrétní aplikací, jejíž komunikace má být šifrována). Ve webových prohlížečích je podpora SSL/TLS již dávno zabudována, takže s funkčností tohoto řešení nebývají problémy alespoň v případech, kdy se komunikuje přes protokol HTTP nebo podobné. Pokud aplikace nepodporuje SSL/TLS, je možné to řešit například pomocí *STunnel*.[7]

SSL je považováno za sice méně bezpečné, ale zato pružnější řešení než IPSec (například s mechanismem NAT má IPSec problémy, kdežto SSL si s ním poradí celkem bez problémů).

10.4.5 SSH

O SSH už něco víme z kapitoly o některých aplikačních protokolech, zde se pouze soustředíme na SSH spojení, které taktéž může být chápáno jako tunel.

SSH je protokolem vyšších vrstev a komunikuje přes TCP. Na rozdíl od TLS a některých dalších podobných protokolů SSH-2 nabízí správu více navázaných relací najednou (TLS pouze jedno spojení).

Jedná se o tunely typu remote access, point-to-point, na vyšších vrstvách. Tunelování funguje na principu předávání (forwardování) provozu na porty, na které je napojen SSH tunel. SSH server naslouchá na TCP portu 22.

Existuje více různých implementací SSH. K nejoblíbenějším patří nástroj *OpenSSH*, pro který existuje klientská i serverová varianta (spíše pro UNIXové systémy). Instalace a používání jsou popsány v příloze na straně 340. Pro Windows existují mezi volně šiřitelným softwarem spíše klienty, například *PuTTY*.

Když používáme SSH, měli bychom využívat bezpečné verze softwaru pro přenos dat. V UNIXu (včetně Linuxu) používáme příkazy `scp` (kopírování, místo `cp`) a `sftp` (místo `ftp`). Existuje také souborový manažer *WinSCP* implementující obdobu těchto příkazů.

http://blog.trackets.com/2014/05/17/ssh-tunnel-local-and-remote-port-forwarding-explained-with-examples.html

Úkol

Ve zmíněné příloze si projděte instalaci a používání SSH. Jsou tam také odkazy na další informace.

[7] http://www.stunnel.org/

10.4.6 MPLS VPN

MPLS sítě se používají spíše pro implementaci propojení firemních poboček tunelem (příp. firmy a obchodního partnera), tedy VPN typu site-to-site, spoje jsou typu point-to-point. S MPLS sítěmi jsme se již seznámili a víme, že se používá systém záhlaví, z nichž jedno je určeno právě pro virtuální sítě. Řešení pracuje na rozhraní vrstev L2 a L3.

V síti MPLS VPN rozeznáváme tato zařízení:

- CE (Customer Edge) – hraniční uzel sítě zákazníka, přes který se data přenášejí do MPLS tunelu,
- PE (Provider Edge) – hraniční uzel sítě poskytovatele řešení, je přímo spojen s uzlem CE,
- P (Provider) – vnitřní uzly MPLS sítě poskytovatele, vpodstatě Core MPLS směrovače.

Uzly CE nejsou součástí MPLS sítě, jsou na nich vyžadovány pouze protokoly TCP/IP (konkrétně hlavně IP). Uzly PE jsou hraničními (LER) uzly MPLS sítě a implementují protokol BGP, příp. iBGP (distribuce tabulek mezi uzly). K jednomu PE uzlu může být připojeno více CE uzlů.

IP paket přicházející z CE na uzlu PE je opatřen dvěma MPLS hlavičkami:

- vnitřní (B=1) bude využita až na druhém konci tunelu, určuje CE příjemce,
- vnější (B=0) určuje cestu v MPLS síti přes P uzly.

VRF tabulka (VPN Routing and Forwarding Table) je tabulka směrovacích informací pro konkrétní VPN a sítě. Nachází se na uzlech PE, jeden PE může obsahovat i více VRF.

V tabulce je konkrétní port (rozhraní) asociován s konkrétní VRF (patří do příslušné VPN), pokud paket přijde z rozhraní napojeného na určitou VRF, podle dané VRF se rozhoduje, co s ním. To znamená, že VRF určuje, ke kterému CE se má dostat paket příchozí z daného rozhraní.

Na obrázku 10.7 vidíme ukázku MPLS VPN sítě, kde například LAN 2 (Site 2) patří do VPN A a VPN B, proto ve VRF tabulce na příslušném PE určené pro tuto LAN najdeme směrování do všech sítí patřících do těchto dvou VPN (tj. sítí 1, 2 a 3).

Pro MPLS VPN existují dvě základní řešení – MPLS Layer-3 VPN, MPLS Layer-2 VPN. Zapouzdřovat se mohou jak IP pakety, tak i ethernetové rámce (řešení Ethernet over MPLS – EoMPLS), případně PDU jiných protokolů na těchto vrstvách.

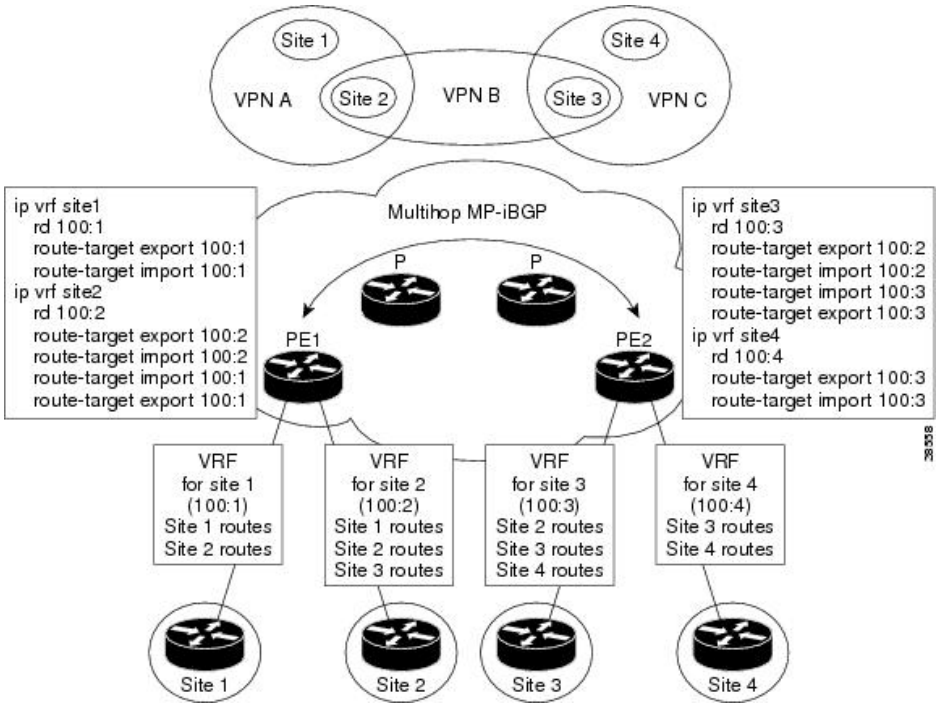
Nevýhodou MPLS VPN je, že potřebujeme externího poskytovatele této služby (obvykle ISP, přes jehož WAN síť náš tunel povede), tyto tunely si nemůžeme vytvořit sami. Naopak výhodou je, že pakety přenášené tunelem neprocházejí přes nezabezpečený Internet, konfigurace je na straně poskytovatele služby a součástí služby je i QoS, takže i přes poměrně vysoké finanční náklady si tato služba své zákazníky nachází.

10.4.7 Další možnosti řešení VPN

IP-in-IP je jednoduše zapouzdření IP paketu do IP paketu. Může jít o stejné verze nebo různé verze, často se například tímto způsobem řeší transport IPv6 paketů přes úsek sítě, který „nerozumí“ IPv6 (takže IPv6 paket zapouzdříme do IPv4 paketu).

Důležitým prvkem Ip-in-IP je změna IP adres v záhlaví, což je vlastně jedna z vlastností tunelování: ve vnějším záhlaví se jako zdrojová použije IP adresa zařízení (routeru) na začátku tunelu, jako cílová

[8] Zdroj: http://www.cisco.com/en/US/docs/net_mgmt/vpn_solutions_center/1.1/user/guide/VPN_UG1.html



Obrázek 10.7: Schéma MPLS VPN a VRF tabulek[8]

se dosadí adresa konce tunelu. Na rozdíl od NAT (což je vlastně také překlad adres) se původní záhlaví zachovává (NAT zasahuje do původního záhlaví, nové nevytváří).

V případě Ipv6 je přímo v tomto mechanismu vestavěna možnost šifrování (ve vnějším IP záhlaví by pak bylo volitelné bezpečnostní záhlaví), u IPv4 bychom šifrování museli zajistit jinak (třeba pomocí protokolu IPSec).

VPLS (Virtual Private LAN Services) je multipoint řešení pracující na linkové vrstvě, vpodstatě multipointová obdoba EoMPLS určená pro aplikace, které vyžadují funkčnost vzdálené multicast a broadcast komunikace. Poskytovatel této služby vlastně z pohledu klienta simuluje switch (jsme na L2) propojující vzdálené LAN sítě.

DMVPN (Dynamic Multipoint VPN) je řešení postavené na protokolu NHRP (Next Hop Resolution Protocol), což je rozšíření protokolu GRE pro využití na multipoint spojích. Pracuje na síťové vrstvě a nevyžaduje externího poskytovatele (konfiguraci si můžeme provést u sebe, přenos probíhá běžně přes Internet, ovšem v tunelu). Dá se říct, že jde o multipoint alternativu ke GRE a IPSec.

Další informace:

Další informace o VPN:

- http://wh.cs.vsb.cz/sps/images/0/04/Kratos-MPLS-VPN.pdf
- http://www.cisco.com/en/US/docs/net_mgmt/vpn_solutions_center/1.1/user/guide/VPN_UG1.html
- http://www.linuxsoft.cz/article.php?id_article=1800
- http://www.ipsec-howto.org/ipsec-howto.pdf
- http://openvpn.net/
- http://www.root.cz/clanky/openvpn-vpn-jednoduse/
- http://www.root.cz/clanky/openvpn-vpn-jednoduse-2/

- http://idoc.vsb.cz/cit/servery/ldap/navod_prechod_na_ldaps/ar01s05.html
- http://www.svetsiti.cz/view.asp?rubrika=Tutorialy&temaID=219&clanekID=230
- http://www.samuraj-cz.com/clanek/vpn-1-ipsec-vpn-a-cisco/
- http://www.openl2tp.org/documentation
- http://www.soom.cz/index.php?name=articles/show&aid=319
- http://wh.cs.vsb.cz/mil051/index.php/%C5%A0ifrov%C3%A1n%C3%AD_IP_provozu_protokolem_IPSec
- http://dolezel.net/post/2008/03/09/Konfigurace-L2TPIPSec-klienta.aspx
- http://www.cs.vsb.cz/grygarek/TPS/projekty/0607Z/L2TP.pdf
- http://www.cs.vsb.cz/grygarek/TPS/projekty/0506Z/krs008_TPS_projekt.pdf
- http://www.datacom.cz/Professional-Computing-clanek-SSLVPN
- http://www.cisco.com/en/US/tech/tk436/tk428/technologies_tech_note09186a0080125b01.shtml
- http://net.infocom.uniroma1.it/corsi/Network%20Infrastructures/lucidi/MPLS-QoS&TE.pdf
- http://www.itu.int/ITU-D/arb/COE/2009/MPLS/Documents/Doc6-MPLS%20VPN-ppt.pdf

10.5 Správa sítě

10.5.1 NMS

Správa sítě v sobě zahrnuje tyto činnosti:

- dohled, kontrola (monitorování) – na fyzické vrstvě (stav média apod.), linkové vrstvě (chybovost rámců, atd.) i vyšších, interpretace shromážděných údajů,
- diagnostika a řešení poruch, předcházení poruchám,
- správa jednotlivých prvků sítě, řešení změn topologie (například přidání nového uzlu),
- účtování využívání zdrojů.

V rámci modelu ISO/OSI většina těchto činnosti probíhá na aplikační vrstvě, a to vždy v některé formě na všech uzlech sítě.

Správa sítě je služba využívající řadu nástrojů, aplikací a zařízení k monitorování, údržbě a zabezpečování sítě, a to v co nejvyšší míře automatizace.

 Rozlišujeme *řídicí* a *řízené entity*. Řízené entity odesílají zprávy obsahující hlášení o událostech nebo problémech, řídicí entity na ně adekvátně reagují, například

- zpraví operátora mailem nebo jiným způsobem,
- zapíšou událost do LOG souboru (logování událostí),
- vypnou nebo restartují systém, odhlásí „problémového“ uživatele, odstřelí „problémový“ proces,
- provedou automatickou úpravu konfigurace systému, přenastaví určené limity.

Softwarové agenty (řízené entity) sbírají informace z koncových zařízení a odesílají řídicím entitám.

 **Network Management System (NMS)** je systém řízení sítě zahrnující řídicí entity, databázi a protokoly řízení sítě. Nejznámější protokoly pro NMS:

- SNMP (Simple Network Management Protocol)
- CMIP (Common Management Information Protocol)

10.5.2 ISO model řízení sítě

Řízení sítě lze rozdělit do několika oblastí.

Řízení výkonu (performance management) – účelem je zajistit, aby výkon sítě byl na přijatelné úrovni. Patří zde proměnné veličiny jako je průchodnost sítě, lhůty pro odezvu uživatelů, optimální využívání kabelů a jiných spojových cest a prvků, atd. Zahrnuje:

1. shromáždění souvisejících dat o veličinách
2. analýza dat a stanovení úrovně pro normální provoz
3. stanovení horní (resp. u některých veličin dolní) hranice s tím, že překročení této hranice je indikováno jako problém, který je třeba řešit

Při překročení stanovených hranic je informován NMS.

Řízení konfigurace (configuration management) znamená monitorování konfigurace sítě a připojených systémů (hardware i software) a jejich ovlivňování. Pro každou sledovanou veličinu (verze operačního systému s aktualizacemi, verze protokolu TCP/IP, apod.) je vytvořeno pravidlo, a pokud není splněno, musí být vyvozeny důsledky.

Řízení uživatelských účtů (accounting management) znamená definování regulačních parametrů pro uživatele a skupiny uživatelů. Vhodná regulace minimalizuje problémy v síti, především při využívání zdrojů (hardware, vyhrazené místo v paměti, atd.), a maximalizuje férovost využívání zdrojů v síti vzhledem k ostatním uživatelům.

Řízení chyb (fault management) představuje detekci, evidenci (log soubory), oznamování a, pokud je to možné, také automatické řešení problémů, které v síti mohou vzniknout. o jeden z nejdůležitějších modulů systémů řízení sítě, protože neodchycené chyby mohou způsobit zpomalení komunikace, ztrátu dat nebo obecně neakceptovatelné zhoršení celkového provozu na síti (případně zhroucení sítě).

Modul využívá údaje získávané a uložené v rámci ostatních funkcí řízení sítě, detekuje možné problémy (aby byly co nejdříve zachyceny), navrhuje a testuje řešení.

Řízení bezpečnosti (security management) znamená řízení přístupu ke zdrojům na síti podle stanovených pravidel, zabránění poškození systému (ať už úmyslnému nebo neúmyslnému) a vynesení citlivých informací.

V hierarchické síti jsou stanoveny oblasti autorizované a neautorizované, autorizace může být také víceúrovňová.

10.5.3 Management v ISO/OSI

Jedná se o centralizovaný systém založený na protokolu *CMIP* (Common Management Information Protocol). Management můžeme rozdělit do tří částí:

- *systémový management* (také síťový, network management) – působí v aplikační vrstvě, slouží jako rozhraní k ostatním částem managementu, tj. provádí úkoly související s více vrstvami,
- *vrstvový management* – pracuje v rámci jedné vrstvy,
- *protokolová operace* – také v rámci jedné vrstvy, ale pouze pro jediný případ komunikace.

Jedná se o objektový model (pracuje s objekty a jejich vlastnostmi – atributy, operacemi, vztahem k jiným objektům, používá ISA hierarchii):

- *manažer* (správce) – programové vybavení na stanici síťového managementu (centrální),

- *agent* – programové vybavení na jednotlivých uzlech sítě, posílá zprávy o (mimořádných) událostech manažerovi,
- *MIB* (Management Information Base) – objektová databáze řízených objektů, v tomto modelu binární.

MIB je tedy objektová databáze řízených objektů. Pro každý objekt je zde

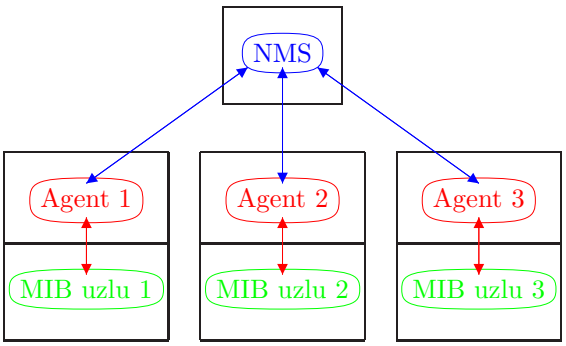
- jméno, třída objektu,
- atributy jako uspořádané dvojice typ–hodnota,
- chování – reakce předpokládaná a skutečná na řídicí operace
- hlášení – o událostech souvisejících s objektem včetně informace, co má být agentem sděleno manažerovi
- výčet operací, které je možno s objektem provádět (u třídy)

Protokol CMIP pracuje na aplikační vrstvě uzlů sítě. Při komunikaci používá službu se spojením (tj. pokud nelze navázat zcela spolehlivě spojení, CMIP nemůže komunikovat s agenty), což může být svým způsobem nevýhoda. Pro reprezentaci řídicích informací se používá jazyk ASN.1 (Abstract Syntax Notation One), který je pro tyto účely běžný (setkáme se s ním i u SNMP).

10.6 Management v TCP/IP

10.6.1 SNMP

V TCP/IP je správa prováděna pomocí protokolu *SNMP* (Simple Network Management Protocol). Ve skutečnosti může běžet i nad jiným protokolem než IP, ale téměř výhradně se setkáme s použitím nad IP.



Obrázek 10.8: Komunikace v SNMP

SNMP (resp. jeho nástavby) je v současné době nejpoužívanějším protokolem pro správu sítě a je široce podporován prakticky v každém zařízení, které je možné připojit k síti (také tiskárny, nejrůznější čidla a analyzátory, přístupové body, atd.).

Protokol SNMP existuje ve třech verzích. Současná verze je SNMPv3, ale přesto je momentálně nejpoužívanější verze SNMPv2, přesněji její varianta SNMPv2c. Hlavní, a velmi důležitý rozdíl mezi verzí 3 a předchozími je úroveň zabezpečení komunikace. Starší verze používají při autentizaci pouze (textové) heslo – *community string* (přesněji dvě – read comminity string pro čtení, write community string pro

povolení zápisu). SNMPv3 již využívá bezpečnější metody – přístupové jméno a heslo, kontrolní součty MD5, šifrování DES, řízení přístupu k objektům.

SNMPv2 přinesl oproti předchozí verzi například podporu komunikace mezi různými správci (ve verzi SNMPv1 se s více správci ani moc nepočítalo) a vylepšení zabezpečení komunikace. I nadále se sice používají community strings, ale byly přidány nové mechanismy jednoznačné identifikace entit.

✎ Hlavní vlastností SNMP je jednoduchost. Podobně jako CMIP, i SNMP používá koncept *agent–správce* (manager), ale funkce těchto modulů jsou jinak rozděleny. Každý uzel sítě (spravované zařízení – *managed device*) může obsahovat jakékoliv množství agentů specializovaných na určitou činnost. V síti musí existovat alespoň jeden správce, může jich být více.

✎ Komunikace mezi agentem a správcem probíhá především pomocí protokolu UDP. Tento protokol však nepodporuje potvrzení doručení (ale na druhou stranu je rychlý a zasílání obvykle funguje), proto SNMPv2 a vyšší obsahuje vlastní mechanismus kontroly doručení. SNMP podporuje dva typy komunikace:

- *Dotaz–odpověď* – aktivita je na straně správce, správce odesílá dotazy agentům a přijímá jejich reakce. Podle SNMPv2 správce posílá dotazy agentovi na port 161, agent odpovídá z portu 161 na dynamický port správce (tzn. číslo portu se mění). Podle SNMPv3 agent naslouchá na jiném portu – číslo 10 161.
- *Trap* – aktivita je na straně agentů, agenty odesílají trapy (oznámení) správci například při výskytu definované události, překročení zadané hodnoty, změně topologie sítě (například připojení nového uzlu) nebo v pravidelných intervalech. Agent podle SNMPv2 posílá správci trapy ze svého dynamického portu (s různými čísly) na port 162. Podle SNMPv3 správce naslouchá na portu 10 162.

  Tento typ komunikace je běžnější.

Jak vidíme, mezi verzemi 2 a 3 jsou odlišnosti dokonce i v číslech portů.

✎ O skupině NMS v rámci jedné administrativní domény hovoříme jako o *komunitě*. Každá komunita je jednoznačně identifikována řetězcem *community name*. Ve starších verzích SNMP se používá jako jednoznačný identifikátor při autentizaci.

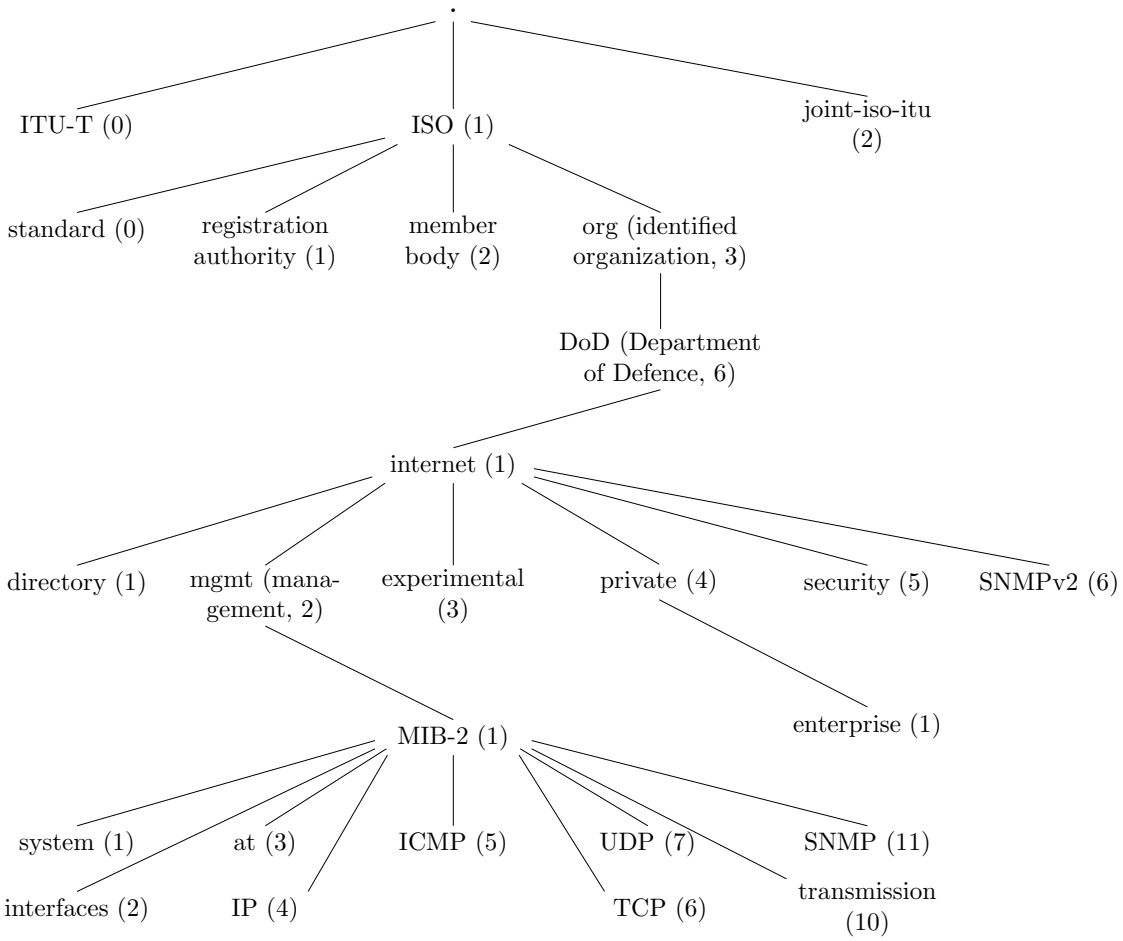
Protokol SNMP je asynchronní, transakčně orientovaný, typu klient/server (komunikace je většinou typu dotaz–odpověď).

10.6.2 MIB-II

✎ Také se používá *databáze MIB*, ale má obecně jinou strukturu, data jsou uložena v textové formě (obslužný modul MIB je naprogramován v jazyce ASN.1, který je pro tyto účely v TCP/IP typický). Je sice objektová (v tom smyslu, že pracujeme s objekty), ale na rozdíl od OSI MIB je řízená atributy (nepoužívá plně objektový přístup, přistupujeme přes vlastnosti objektů).

Ve skutečnosti se v současných zařízeních setkáme vždy s podporou MIB-II, tedy druhé verze MIB. V následujícím textu budeme pro stručnost používat název MIB, ale vždy půjde o MIB-II.

✎ MIB je tvořena stromem s jediným kořenem. Kořen obsahuje „prázdnou hodnotu“, jeho účelem je pouze spojovat z něho vycházející větve. Všechny uzly kromě kořenu mají přidělen textový název a číselnou hodnotu – *OID* (Object ID), OID jednoznačně odlišuje uzly se stejným „rodičem“ – viz obrázek 10.9. Textový název je určen spíše pro „lidskou orientaci“, nemá v databázi žádný jiný význam.



Obrázek 10.9: SNMP MIB (zkrácená)

Uzly (objekty) v MIB, které jsou přímými potomky kořene, jsou nazvány podle organizací, jejichž protokoly využívají podstromy těchto uzlů (například ISO nebo ITU). Některé části (větve) MIB jsou standardizovány orgranizací IETF, další vydávají výrobci síťových zařízení. Fyzicky je tedy MIB uložena obvykle ve více než jednom (textovém) souboru, aby zpracování zde uložených dat bylo dostatečně pružné a aby byla snadněji rozšiřitelná.

Adresa objektu v MIB je sekvence

- *názvů uzlů* na cestě od kořene stromu – pro lidi, objekty oddělujeme lomítkem nebo tečkou, NEBO
- *čísel OID uzlů* na cestě od kořene stromu – pro kohokoliv/cokoliv jiného, objekty oddělujeme tečkou (včetně prázdného názvu kořene, tedy celý řetězec vlastně začíná tečkou).

Příklad

Například podle obrázku 10.9 má uzel *enterprise* určený pro firemní uzly (zařízení různých výrobců) tuto textovou a OID adresu:

- .iso.org.dod.internet.private.enterprise
- .1.3.6.1.4.1

Každé síťové zařízení, protokol či hodnota (resp. přiřazený objekt), které lze přes MIB spravovat, má svou unikátní OID adresu, která je stejná v jakékoliv MIB, tedy všechny OID adresy musí být globálně jedinečné.

Listy stromu MIB jsou *skalární objekty*, jedná se o *proměnné* obsahující konkrétní hodnotu využívanou pro účely řízení sítě (například počet paketů procházejících routerem). Proměnné mohou být uspořádány i do tabulky (*tabulkové objekty*). SNMP poskytuje omezenou možnost pohybovat se v tabulce – po sloupcích. Typy proměnných v MIB:

1. číselné
   - běžný *Integer*
   - *Counter* – pro čítače, při dosáhnutí maximální hodnoty se vynulují, slouží především pro zjištění rychlosti sledování změn
   - *Gauge* (míra) – pokud sledovaná veličina přesáhne danou hranici, hodnota Gauge zůstává na maximální hodnotě a „nepřeteče“
   - *Time Ticks* – sleduje čas (v setinách sekundy) od zadané události, například od spuštění zařízení
2. *IP Address* – zde bývá uložena IP adresa
3. *Octet String* – posloupnost oktetů, používá se pro ukládání znakových řetězců
4. *Object Identifier* – OID řetězec některého objektu
5. *tabulka* hodnot výše uvedených typů.

K proměnným se přistupuje poněkud zvláštním způsobem. Pokud se jedná o jednoduchou proměnnou (takovou, která není tabulkou), syntaxe je

`.cesta.název_proměnné.0`

Pokud se jedná o tabulku, pak místo čísla 0 na konci použijeme index v tabulce.

Příklad

K proměnné *sysUpTime* ve větvi *system* vede cesta

`.iso.org.dod.internet.mgmt.mib-2.system.sysUpTime.0` nebo číselně

`.1.3.6.1.2.1.1.3.0`

K hodnotám v tabulkách přistupujeme tak, že místo čísla 0 na konci dosadíme index v tabulce (od 1), například k hodnotám v tabulce ukazujícím stav portů zařízení (živý/mrtvý, v Ethernetu to obvykle znamená je/není něco připojeno) přistupujeme takto:

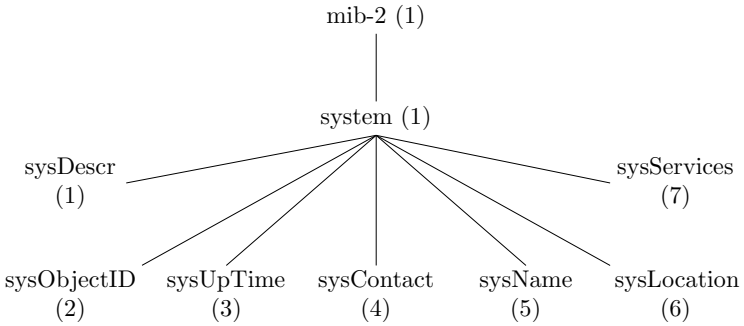
```
.iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifOperStatus.1
.iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifOperStatus.2
.iso.org.dod.internet.mgmt.mib-2.interfaces.ifTable.ifEntry.ifOperStatus.3
```

atd., hodnota může být buď `up(1)` nebo `down(2)`. V *interfaces* je jednoduchá číselná proměnná *ifNumber*, ve které máme uložen počet rozhraní v systému (number of interfaces). Je zřejmé, že na pracovní stanici bude `ifNumber.0 = 2` (jedno fyzické rozhraní a jedno loopback), případně nějaký ten výtvor virtualizačního softwaru, na směrovači bude rozhraní více.

Uvádíme zde sice „absolutní“ adresy, ale je možné adresovat také relativně uvnitř daného MIB souboru (skupiny), například ve skupině *interfaces*, *ip* nebo *system*.

Struktura MIB je celkem logická. Jak bylo výše uvedeno, ve větvi *mib-2* najdeme proměnné, které se netýkají přímo konkrétního výrobce (tj. obecné informace), kdežto ve větvi *private.enterprise* jsou proměnné týkající se výrobků od konkrétních výrobců (záleží, co konkrétně v příslušném síťovém zařízení najdeme). Zařízení, která nejsou pro MIB standardizovaná, jsou obvykle umístěna ve větvi *experimental*.

Obecné informace o zařízení, s jehož MIB pracujeme, najdeme především v podstromu uzlu *system*. Proměnné v něm obsažené vidíme na obrázku 10.10. Je zde popis zařízení (*sysDescr*), adresa OID vedoucí do větve *enterprises* s podrobnějšími informacemi (*sysObjectID*), doba od zapnutí zařízení v setinách sekundy (*sysUpTime*), kontaktní údaje ke správci zařízení (*sysContact*), název zařízení přiřazený adminem (*sysName*), info o fyzickém umístění zařízení (*sysLocation*) a číslo určující vrstvy v ISO/OSI, na kterých zařízení pracuje (*sysServices*).



Obrázek 10.10: Podstrom uzlu *system*

Příklad

Vraťme se k hodnotě *sysServices* určující vrstvy v ISO/OSI, na kterých dané zařízení (to, na kterém je uložena databáze daného agenta) pracuje. Jednotlivé bity této hodnoty jsou nastaveny podle podpory vrstev (bity zprava doleva od nejméně významného bitu).

Například `sysServices.0 = 10` znamená, že zařízení pracuje na L2 a L4 ($2^{2-1}+2^{4-1}$, tedy je nastaven druhý a čtvrtý bit zprava) a znamená to, že jde zřejmě o switch (L2) a obsahuje funkcionalitu L4 pro účely správy (transportní vrstva).

Dalším užitečným uzlem v obecné části MIB je uzel *interfaces*. Zde zjistíme informace o rozhraních zařízení, informace o jednotlivých zařízeních jsou v tabulce *ifTable* (do níž jsme už trochu nahlédli). Tato tabulka má mnoho sloupců, například *ifSpeed* (nominální rychlost přenosu na rozhraní), *ifOperStatus* (operační stav rozhraní), počet oktetů, přijatých/odeslaných/zahozených unicast paketů, non-unicast paketů (*ifOctets*, *ifInUcastPkts*, ...), atd.

Skutečnou rychlost rozhraní zjistíme pouze tak, že dvakrát za sebou zjistíme počet přijatých oktetů, odečteme a vydělíme délkou intervalu, který uplynul mezi těmito dvěma načteními hodnot.

10.6.3 Používání protokolu SNMP

Protokol SNMP definuje příkazy (pro agenty a správce):

- `get-request` – správce žádá informace z MIB; uvede název proměnné, jejíž hodnotu požaduje, získá hodnotu a název této proměnné,
- `get-next-request` – účel je stejný (žádost o informace z MIB), správce také uvede název proměnné, ale získá hodnotu proměnné a dále název následující proměnné z databáze, která je potomkem

téhož uzlu (tj. pokud žádal o proměnnou `...2.1.1.3`, dostane informaci o existenci proměnné `...2.1.1.4`, na kterou se může dotazovat následně),

- `set-request` – správce ukládá informace do MIB, je nutný autorizovaný přístup správce,
- `trap` – agent předává správci nevyžádanou informaci o mimořádné události,
- `get-response` – odpověď agenta na předchozí `get-request` od správce, kromě hodnot z MIB obsahuje také původní dotaz, protože SNMP neumožňuje správci párovat dotaz/odpověď (nestavové chování),
- `get-bulk` – podobně jako `get-request`, ale je možné žádat i několik řádků tabulky (od SNMPv2),
- `inform` – komunikace mezi dvěma správci (od SNMPv2).

Příkaz `get-next-request` použitý ve vhodné smyčce umožňuje získat postupně hodnoty stejných atributů přes všechny objekty stejného typu (sloupcová operace).

Konkrétně můžeme tyto příkazy (ať už v textové nebo grafické podobě) zadávat v některém z nástrojů vytvářejících rozhraní k SNMP. Lze použít například tyto nástroje:

- *net-snmp*[9] je open-source nástroj pracující v textovém režimu, a to pod Linuxem, dalšími UNIXovými systémy a také pod Windows. Je to ve skutečnosti balík programů, který umožňuje využívat plně SNMP v kterékoliv jeho verzi, včetně příjmu trapů.
- *MIB Browser*[10] je volně šiřitelná aplikace s grafickým rozhraním umožňující pracovat s MIB.
- *GNetWatch*[11] je open-source aplikace s grafickým rozhraním pro real-time monitoring sítě přes SNMP a ICMP.

Další informace:

- http://www.samuraj-cz.com/clanek/snmp-simple-network-management-protocol/
- http://www.samuraj-cz.com/clanek/zarizeni-v-siti-pod-kontrolou/
- http://www.cisco.com/en/US/docs/internetworking/technology/handbook/SNMP.html
- http://www.svetsiti.cz/view.asp?rubrika=Tutorialy&temaID=129&clanekID=133
- http://www.abclinuxu.cz/clanky/system/monitoring-pomoci-nastroju-z-baliku-net-snmp
- http://www.snmplink.org/
- http://www.fi.muni.cz/~kas/p090/referaty/2007-podzim/ct/snmp.html

10.7 Syslog

Syslog je mechanismus logování a souvisejících úloh dostupný na každém zařízení, na kterém běží (téměř) jakýkoliv UNIXový systém včetně Linuxu. Jádrem mechanismu je démon[12] *syslogd*.

[9] *net-snmp* je ke stažení na http://www.net-snmp.org/.
[10] *MIB Browser* je ke stažení na http://www.ks-soft.net/hostmon.eng/mibbrowser/index.htm.
[11] *GNetWatch* je dostupný na http://gnetwatch.sourceforge.net/.
[12] Pro ty, kteří zapomněli: démon je v UNIXových systémech něco podobného jako služba ve Windows. Jde o systémový proces, který běží na pozadí. Názvy démonů obvykle končí písmenem „d“.

Démon *syslogd* čte svůj vstup ze zařízení (socketu) `/dev/log`. Tento vstup filtruje (vybírá to, co ho zajímá) podle konfigurace, která se nachází v souboru `/etc/syslog.conf` (to je tedy jeho konfigurační soubor) a pak podle nastavení ukládá hlášení o takto zjištěných událostech do jednoho nebo více LOG souborů.

Data, která syslogd přijímá, se skládají ze tří částí:

1. *kategorie* určuje typ nebo odesílatele události, existují tyto kategorie:
   - auth – autentizace uživatelů,
   - authpriv – informace o autentizaci určená pro administrátora,
   - cron – zprávy od démona cron (plánování procesů),
   - daemon – zprávy od různých démonů,
   - kern – zprávy od jádra (kernel),
   - lpr – zprávy od tiskového subsystému,
   - mail – zprávy týkající se elektronické pošty,
   - mark – časová razítka (timestamps), které se pravidelně zapisují do logu,
   - news – diskusní skupiny,
   - security – totéž co auth,
   - syslog – vlastní zprávy syslogu (i z jiného uzlu v síti),
   - user – obvykle zprávy od aplikací v uživatelském režimu,
   - local0–local7 – pro tyto kategorie lze definovat vlastní význam,
2. *priorita* určuje důležitost události:
   - emerg – systém je nepoužitelný nebo vážně ohrožen (emergency),
   - alert – je nutný okamžitý zásah,
   - crit – kritická situace,
   - err – chyba,
   - warning – varování,
   - notice – normální, avšak významná, zpráva,
   - info – informativní zpráva,
   - debug – ladicí zpráva (debugger),
3. vlastní *text* zprávy.

Tento typ informací tedy *syslog* získá. Jak bylo výše uvedeno, ve svém konfiguračním souboru má určeno, co s takovým záznamem provést. V tomto souboru jsou záznamy ve formě

`kategorie.priorita` TAB `cíl`	*tato a vyšší priority*
`kategorie.=priorita` TAB `cíl`	*pouze tato priorita*
`kategorie.!priorita` TAB `cíl`	*všechny priority kromě této a vyšších*
`kategorie.!=priorita` TAB `cíl`	*všechny priority kromě této*

(priorit může být i více, oddělují se středníkem, totéž platí i o dvojicích kategorie.priorita). Pokud je před prioritou jen tečka, nastavení platí pro uvedenou a všechny vyšší priority. Pokud chceme nastavení omezit jen na zadanou prioritu, přidáme před ni „=“. Symbol „!“ znamená negaci, tedy pokud je uveden, daná priorita (a všechny vyšší) nebude zahrnuta. Lze kombinovat: „!=“, nebude zahrnuta pouze uvedená priorita. Kategorii můžeme zadat symbolem `*`. To znamená, že se konfigurace týká jakékoliv kategorie.

Mezi prioritami a cílem musí být vždy *alespoň jednou stisknutý tabulátor*, mezera nestačí.

Cíl určuje, kam konkrétně má být událost oznámena, logována. Může to být buď konkrétní log soubor (výchozí nebo jakýkoliv jiný), nebo výpis na konzolu či přeposlání na jiný počítač v síti. Pokud chceme, aby byla událost oznámena více cílům (například zobrazena určitému uživateli a zároveň uložena do souboru), oddělíme cíle čárkou. Možnosti:

- soubor – výchozí je `/var/log/messages`, ale můžeme si určit názvy souborů například pro různé kategorie nebo priority,
- `@server.firma.cz` nebo `@IPadresa` – událost bude přeposlána,
- `user1`, `user2`, ... – událost bude oznámena zadanému uživateli (to může být root, admin, operator, apod., podle toho, jaké máme vytvořené uživatele), ale jen tehdy, když je uživatel právě přihlášen,
- `*` – událost bude oznámena všem přihlášeným uživatelům,
- `@loghost` – můžeme použít, pokud v souboru `/etc/hosts` je definován cíl `loghost`.

Lze použít také přesměrování do pojmenované roury, ze které pak může číst jakýkoliv proces, který určíme (a průběžně zpracovávat oznámení o událostech), stačí uvést název souboru a před něj napsat symbol roury:

```
kern.=err          |/var/log/jadro
```

Postup

Takto nějak mohou vypadat záznamy v souboru `/etc/syslog.conf`:

`mail.*` `/var/log/maillog`

Všechny události z kategorie *mail* jsou uloženy do souboru `/var/log/maillog`

`security.*;security.!=debug` `/var/log/secure`

Všechny události z kategorie *security* kromě událostí s prioritou *debug* jsou uloženy do souboru `/var/log/secure`

`cron.*` `/var/log/cron`

Všechny události z kategorie *cron* jsou uloženy do souboru `/var/log/cron` (to znamená události související s plánovaným spouštěním procesů)

`kern.debug;auth.notice` `/dev/console`

Události týkající se ladění jádra a běžné a přesto významné události autentizace jsou vypsány na systémové konzoli

`authpriv.*` `/var/log/secure`

Všechny „citlivější“ události autentizace jsou uloženy do souboru `/var/log/secure`

`lpr.info` `/var/log/lpd-info`

Informační zprávy a všechny s vyšší prioritou o událostech z tiskového subsystému se uloží do `/var/log/lpd-info`

`*.err` `admin,/var/log/errors`

Všechny chybové a vážnější zprávy jsou okamžitě oznámeny uživateli *admin* (pokud je přihlášen) a zároveň uloženy do souboru `/var/log/errors`

`*.=crit` `/var/log/messages,@loghost,root`

Kritické události jsou uloženy do souboru `/var/log/messages`, poslány na adresu definovanou pod aliasem *loghost* a zároveň je okamžitě informován *root*, pokud je přihlášen

`*.emerg` `*,/var/log/emergency`

O mimořádně nebezpečných událostech jsou informováni všichni přihlášení uživatelé a zároveň je přidán záznam do souboru `/var/log/emergency`

 Pokud chceme, aby *syslog* přijímal zprávy z jiných systémů, musíme spustit démona *syslogd* s parametrem `-d` (platí v Linuxu):

```
/usr/sbin/syslogd -m 0 -r
```

Přepínač `-m` určuje délku intervalu, v jakém se *syslogd* ozývá, tj. do logu zapisuje, že „žije“. Nastavením na 0 toto chování zrušíme. Parametr `-r` znamená „remote“, *syslogd* pak naslouchá na portu 514 a přijímá všechny UDP pakety.

Na FreeBSD je nutné použít jinou syntaxi:

```
/usr/sbin/syslogd
```

(bez parametrů, protože naslouchání na portu 514 je zde výchozí chování). Na FreeBSD je také možné určit uzly v síti, jejichž UDP pakety budou přijímány. Odlišnou syntaxi (i od této) mají systémy OpenBSD, Solaris a jiné, je tedy třeba vždy prostudovat manuálovou stránku:

```
man syslogd
```

Také je možné, že budeme muset povolit naslouchání služby *syslogd* na UDP portu. To se provede v `/etc/services` řádkem `syslog 512/UDP`, ale je pravděpodobné, že tam takový záznam už je. Na zařízeních, ze kterých jsou UDP pakety přijímány, je pak nastaveno výše popsaným způsobem zasílání na tento „sběrný“ počítač.

Pokud *syslogd* právě běží a my jsme provedli změnu jeho konfigurace (v souboru `/etc/syslog.conf`), musíme *syslogd* restartovat, aby zaregistroval změny ve své konfiguraci.

 Zatím jsme si ukázali, jaké údaje dostává *syslogd* na svůj vstup, podle čeho je filtruje a kam je ukládá. Zbývá nám podívat se, v jakém formátu. Na výstupu je záznam obsahující

- čas, kdy k události došlo,
- kategorii (kernel, mail apod.),
- text zprávy.

Součástí není priorita, protože ta už byla uplatněna při filtrování. Může zde být například záznam:

```
Feb 21 10:00:28 IOL kernel: device eth0 left promiscuous mode
```

 Když syslog nastavujeme, hodí se možnost vyzkoušení jeho funkčnosti. K tomu může sloužit nástroj *logger*. Například událost kategorie *daemon* a priority *info* se zadanou zprávou vygenerujeme takto:

```
logger -p daemon.info "testujeme syslog"
```

Ruční správa logů může být celkem náročná. Je třeba hlídat obsah souborů a navíc sledovat jejich délku (včas umazat starší záznamy), stroj naslouchající na UDP portu by měl být dostatečně chráněn (je zde vysoké riziko DoS útoků, tedy firewall je rozhodně na místě). S některými úlohami mohou pomoci přídavné nástroje. Například rotaci logů (včetně označování či odstraňování starších záznamů) zvládne *logrotate*.

Existují také propracovanější verze *syslog*u, například *syslog-ng* (umí komunikovat i přes TCP a zvládne také regulární výrazy, nejen hvězdičku), *nsyslogd* (komunikuje přes TCP/SSL), *Secure Syslog*, *Modular Syslog* a další. Volně šiřitelný program *NTSyslog*[13] (vlastně služba, kterou můžeme instalo-

[13] *NTSyslog* najdeme na http://ntsyslog.sourceforge.net.

vat), umožňuje používat *syslog* také ve Windows (logy z Windows lze posílat na vzdálený systém a tam například vyhodnocovat). Nástroj *logwatch*[14] můžeme použít k sumarizaci logů, provádí analýzu logů za stanovené období a vytváří souhrnnou zprávu. Program *swatch*[15] je užitečný, když naopak potřebujeme být o určité události informováni pokud možno okamžitě – detekuje námi definované situace a stanoveným způsobem informuje, a protože je psán v perlu, je to velmi pružný nástroj využívající regulární výrazy.

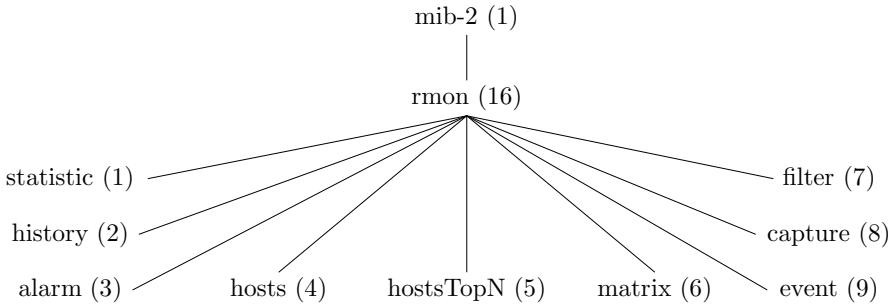
Další informace:

- http://www.linux.cz/noviny/2001-04/clanek03.html
- http://linux.about.com/od/commands/l/blcmdl8_syslogd.htm
- http://www.precision-guesswork.com/sage-guide/syslog-overview.html
- http://books.google.com/books?id=8KUzFBDAT6EC&pg=PA189
- http://www.root.cz/clanky/linux-jako-internetova-gateway-9/
- http://www.softpanorama.org/Logs/Syslog/syslog_configuration_examples.shtml
- http://UNIXhelp.ed.ac.uk/CGI/man-cgi?logger+1
- http://www.root.cz/clanky/synchronizace-casu/

10.8 Monitorování sítě

10.8.1 Protokol RMON

RMON (Remote Monitoring) je protokol používaný při monitorování sítě. Je úzce spjat s TCP/IP a SNMP, své zprávy posílá přes SNMP. Informace jsou uloženy v MIB ve větvi rmon (OID 1.3.6.1.2.1.16). Narozdíl od SNMP agentů dokáže kromě současného stavu sledovat i historii (vývoj) dané veličiny a na základě vyvozených důsledků reagovat.



Obrázek 10.11: Větev *rmon* v MIB-II

Z hlediska protokolu RMON rozlišujeme dva druhy zařízení v síti:

- RMON-compliant console manager (*správce*) – správce celé LAN

[14] *logwatch* získáme na http://www.logwatch.org.
[15] Program *swatch* (Simple Watch) je dostupný na http://swatch.sourceforge.net.

- RMON probe (*sonda*) – zasílá informace správci, jedna sonda pracuje v rámci jednoho segmentu LAN (například uvnitř jedné kolizní domény)

✎ RMON pracuje s tzv. *skupinami*. Každá skupina v sobě zahrnuje určitý typ informace, která je sledována. Například skupiny pro Ethernet jsou

- *Statistics* – statistika (okamžitý stav) sledovaných zařízení (množství odeslaných paketů, broadcast a multicast paketů, oktetů, chyb v CRC součtech, kolizí apod., také čítače, například pro počty paketů o velikosti z daného intervalu),
- *History* – totéž jako statistika, ale evidováno v historii,
- *Alarms* – pro některé sledované veličiny jsou stanoveny prahové hodnoty, a když je tato prahová hodnota překročena, generuje se příslušná událost,
- *Hosts* – vedou se statistiky typické pro jednotlivé uzly v síti (segmentu sítě) – adresa, odeslané/přijaté pakety, chyby v CRC součtech a zahozené pakety či rámce,
- *HostsTopN* – podobně jako předchozí, uzly jsou seřazeny v tabulkách podle konkrétní sledované veličiny,
- *Matrix* – sledují se veličiny související s konverzací mezi dvěma uzly v síti, pro každou komunikující dvojici uzlů,
- *Filters* – umožňují definovat filtry pro zachycení některých paketů nebo generování určité události při průchodu určených paketů,
- *Packet Capture* – definují se podmínky pro zachytávání paketů (například velikost vyrovnávací paměti pro zachycené pakety, počet zachycených paketů, kdy vyvolat alarm apod.),
- *Events* – generování a ověřování událostí (eviduje se typ události, její popis a časový údaj posledního vygenerování této události daným zařízením).

Zařízení připojená v síti obvykle podporují většinu těchto skupin, v ideálním případě všechny. Tedy sledujeme ty proměnné v MIB, které nás zajímají.

Hlavním přínosem RMON je přenos velké části zpracování dat od správců na agenty (sondy), čímž se také snižuje množství přenášených dat v síti.

10.8.2 Snort

✎ Snort[16] je jednoduchý volně šiřitelný systém (open-source licence, ale pro aktualizace je nutné se registrovat, „okamžité“ aktualizace jsou navíc placené), který můžeme zařadit mezi *NIDS* (Network Intrusion Detection System).

✎ Snort pracuje v jednom ze tří možných režimů:

- *sniffer* („prohlížeč“ provozu) – slidič, data z hlaviček paketů vypisuje na obrazovku nebo někam posílá,
- *logování provozu* – data ukládá na disk a/nebo do databáze,
- *plný NIDS* včetně používání pravidel – analýza hlaviček paketů.

Výstupy dokáže Snort buď ukládat do LOG souboru, nebo zasílat *syslog*u, posílat jako SNMP trap, ukládat do databáze a nebo dokonce podle nich nastavovat konfiguraci některých síťových prvků.

[16] Snort získáme na http://www.snort.org.

Snort funguje na principu *detekce signatur* v kombinaci s *pravidly*. Mnoho pravidel je vytvořeno už při instalaci (je jich opravdu hodně, někdy to chce trochu je promazat), a také je možné přidat pravidla vlastní podle konkrétní situace v síti. Využívání pravidel dává systému obecně větší sílu (například kombinace „mírně podezřelých“ akcí je *velmi podezřelá* a narozdíl od systémů založených pouze na signaturách systém generuje méně falešných poplachů.

Obecný tvar pravidel je

```
action protocol sourceIP sourcePort direction destIP destPort (options)
```

Action akce) může být například pass (předání, tj. ignorování paketu), log (zaznamenání), alert (výstraha), drop (zahodit paket), atd. Následují *vlastnosti paketu*, podle nich pozná Snort, že má pravidlo použít (protokol, třeba TCP, dále zdrojová a cílová adresa a port, a také směr přenosu paketu zachycený „šipkou“). Poslední částí pravidla jsou *options* – volby, které ve skutečnosti popisují, co se má stát, když Snort zachytí paket odpovídající údajům v pravidlu a také co dalšího má být testováno (například pole TTL paketu nebo ICMP informace). Například:

```
alert tcp any 3389 -> 81.28.192.0/24 3389 (msg: "TCP paket na Net5";)
```

To znamená, že má být generována výstraha, pokud je nalezen TCP paket z jakékoliv adresy na portu 3389 mířící do sítě s danou IP adresou (používají se CIDR adresy s prefixem), a to na tomtéž portu. S výstrahou se má vygenerovat zpráva v zadaném znění (zpráva je nahlášena a uložena do logu). Ve skutečnosti bývají pravidla poněkud sofistikovanější a mohou obsahovat také například popis porovnání se signaturou a další volby.

V uvedeném příkladu jsou konkrétní adresy, ale lze využít také proměnné, ve kterých má Snort předdefinovanou adresu vnitřní sítě a některé další adresy, například

```
alert icmp $EXTERNAL_NET any -> $HOME_NET any
    (msg:"ICMP Large ICMP Packet"; dsize: >800;)
```

Snort funguje podobně jako předchozí popsané mechanismy včetně SNMP (správce+agenty). Agenty se nazývají *sondy* a nacházejí se v různých oblastech infrastruktury, podle potřeby (umístění stanovíme podle typu informací, které chceme získávat, záleží na tom, jestli je sonda před či za firewallem, v DMZ, v dané kolizní doméně apod.).

Obecně jde o dobře rozšiřitelný systém. Pokud máme více sond a na síti je větší provoz (což znamená velké množství záznamů), doporučuje se použít některý databázový systém, například MySQL. Existují také frameworky (rozhraní), které zjednodušují přístup a analýzu dat Snortu, například *Prelude*,[17] což je framework použitelný nejen pro Snort, ale také například pro Nessus.

Další systém, který zjednodušuje používání Snortu, je *ACID*[18] (Analysis Console for Intrusion Databases). ACID je zajímavý už proto, že se s jeho použitím počítá v doporučeních pracovních skupin CERT (setkali jsme se s nimi u CESNETu). ACID potřebuje webový server (třeba Apache), funkční PHP a databázi, do které Snort ukládá záznamy (třeba MySQL).

Další informace:

- http://i.iinfo.cz/r/kd/Intrusion_Detection_with_SNORT.pdf
- http://www.cs.vsb.cz/grygarek/SPS/projekty0405/IDS/ids.html

[17] Informace o *Prelude* najdeme na http://www.prelude-technologies.com.
[18] *ACID* najdeme na http://www.andrew.cmu.edu/user/rdanyliw/snort/snortacid.html.

- http://www.cs.vsb.cz/grygarek/SPS/projekty0506/Snort.pdf
- http://www.scribd.com/doc/6777057/Snort-Manual
- http://www.snort.org/docs/writing_rules/
- http://www.dusatko.org/cs/content/snort-network-intrusion-detection-system
- http://connect.zive.cz/node/315
- http://www.inguardians.com/research/docs/snortguis.pdf
- http://www.cert.org/kb/aircert/
- http://www.actinet.cz/bezpecnost_informacnich_technologii/l19/cl18/st1/j1/Jak_nasadit_system_detekce_pruniku.html
- http://www.dusatko.org/cs/node/121
- http://www.prelude-technologies.com/en/development/documentation/index.html

10.8.3 NetFlow

 NetFlow je protokol vyvinutý společností Cisco. S jeho podporou se tedy setkáme především na zařízeních od této společnosti, ale také v zařízeních Juniper a některých dalších výrobců (přesněji: tato zařízení mohou exportovat informace ve stejném formátu jako NetFlow). Nemusí jít jen o routery a switche, ale existují také jednoduchá přenosná zařízení, která plní pouze roli sledování sítě tímto způsobem.

Existuje více verzí NetFlow s odlišnými vlastnostmi. Pokud potřebujeme podporu IPv6, je nutné používat minimálně NetFlow verze 9. NetFlow verze 10 byl standardizován organizací IETF jako RFC 7011 pod názvem IPFIX (IP Flow Information Export).

 NetFlow pracuje na principu *toků*. Tok (flow) je to, co obvykle považujeme za úplnou síťovou konverzaci. V případě přenosů realizovaných spojovými stavovými protokoly (jako je například TCP) do jednoho toku patří nejen odeslání jednoho paketu, ale komunikace v rámci celého spojení (ale pouze jedním směrem, tedy obousměrná komunikace je technicky ve dvou tocích). Do jediného společného toku také patří například sekvence ICMP paketů vyslaných v rámci jediného příkazu `ping`. Každý tok je jednoznačně popsán těmito údaji:

- zdrojová IP adresa
- cílová IP adresa
- zdrojový port
- cílový port
- protokol

Protokoly, které se ve všech těchto parametrech shodují, jsou řazeny do téhož toku. Tady je důvod, proč jsou různé směry komunikace v různých tocích – opačné směry mají v záhlavích přehozené adresy a porty.

Komunikace také může být rozdělena do více toků, pokud trvá příliš dlouho (na směrovačích Cisco je to například 30 minut), nebo je delší dobu neaktivní anebo je zaplněna vyrovnávací paměť zařízení (toky není kam ukládat, proto ty starší budou vymazány).

Místo zjišťování informací (s protokolem NetFlow) se nazývá *Observation Point* (také *NetFlow Exporter*). Toto zařízení zjišťuje z průchozích paketů potřebné informace a odesílá je flow kolektoru. *Flow kolektor* je hardware nebo software, který přijímá data o tocích a zpracovává. Jeden flow kolektor může přijímat data od více observation pointů (takže opět jde o architekturu agenti–správce). Na flow kolektor se informace o toku dostává až po ukončení tohoto toku.

Ke každému toku se kromě určujících parametrů evidují také další informace – počet paketů v toku (na Observation Pointu se během evidence toku tento údaj při každém identifikovaném paketu zvyšuje o 1), doba trvání (před odesláním Flow kolektoru se srovnají časová razítka začátku a konce toku), celkové množství přenesených dat (s každým paketem se tato hodnota postupně zvyšuje).

Kromě ukládání dat na Flow kolektoru ještě potřebujeme aplikaci, která by zjednodušila analýzu těchto dat. Existuje více takových aplikací, mnohé volně šiřitelné.

- *OSU Flow-Tools*[19] jsou volně šiřitelný balík programů pro textový režim vyvíjený na univerzitě v Ohiu. V balíku najdeme nástroje na shromažďování dat na flow kolektoru a zpracování údajů o tocích.
- *NFDump*[20] je volně šiřitelný kolektor (distribuovaný pod BSD licencí). Existuje pro něj také grafická nástavba *NFSen*,[21] která je vlastně webovým rozhraním.

Díky „koncentrovanějším“ informacím, které NetFlow umožňuje získávat, lze odhalit například DoS útok (všechny pakety v rámci tohoto útoku mají společné posuzované parametry, patří do jednoho toku), ale ne například DDoS útok (je z mnoha zdrojů, obvykle z počítačů některého rozsáhlého botnetu) ani skenování uzlů v síti (různé cílové adresy).

Oproti předchozím technologiím má NetFlow výhodu v tom, že administrátora nezahlcuje informacemi: do Flow kolektoru se dostane jen jediný záznam o kompletním (jednosměrném) toku, i kdyby to byly stovky paketů.

NetFlow je výborným nástrojem také pro drobné poskytovatele internetového připojení, protože takto mohou jednoduše provádět monitoring a správu sítě, mít přehled o vytíženosti sítě, provádět optimalizaci (včetně nastavení QoS) nebo účtování na základě vytížení sítě, a hlavně mohou splnit prováděcí vyhlášku 357/2012 Sb. o uchovávání, předávání a likvidaci provozních a lokalizačních údajů doplňující Zákon o elektronických komunikacích.

https://www.flowmon.com/cs/solutions/use-case/netflow-ipfix (včetně odkazu na video)

10.9 WBEM

10.9.1 Princip WBEM

WBEM (Web-Based Enterprise Management) je skupina technologií a postupů vytvořená za účelem sjednotit správu distribuovaných počítačových prostředí (tj. včetně vzdálené správy). Správa se provádí buď přes webové rozhraní nebo pomocí specializovaných shellů.

Data jsou organizována v modelu *CIM* (Common Information Model), který je otevřeným standardem. Účelem je jednotným způsobem popisovat, uchovávat a poskytovat potřebné informace bez ohledu

[19] *OSU Flow-Tools* získáme na http://www.splintered.net/sw/flow-tools/.
[20] *NFDump* je dostupný na http://nfdump.sourceforge.net/.
[21] *NFSen* je dostupný na http://nfsen.sourceforge.net/.

na jejich zdroj a použitý protokol. Jedná se vlastně o objektově orientovanou databázi jako MIB, kde jednotlivé objekty jsou zapouzdřeny a přistupuje se k nim přes unifikované rozhraní.

Rozlišujeme *WBEM server* (ukládá a poskytuje informace, provádí příkazy) a *WBEM klienta* (reprezentován především rozhraním, se kterým pracuje administrátor, a příslušným API). Klient odesílá žádosti, server konfrontuje se skutečným stavem a provádí je (případně zajišťuje proces autentifikace a autorizace). Spolu komunikují přes protokol HTTP nebo HTTPS. Právě podle těchto protokolů je v názvu „web-based“. (Pozor, rozhodně to neznamená, že se komunikuje přes webový prohlížeč!)

Architektura je postavena na vzoru model–obraz. Klient čte data z „obrazu“, server přistupuje k „modelu“ reprezentovanému skutečným stavem hardwaru a softwaru v síti a při jakékoliv změně modelu aktualizuje obraz.

WBEM nemá nahradit protokoly CMIP a SNMP, ale je jakousi nástavbou těchto modelů zjednodušující přístup administrátora k informacím.

WBEM má více různých implementací, například:

WMI (Windows Management Instrumentation) je implementace Microsoftu pro Windows. Umožňuje (i vzdáleně) spravovat počítače s Windows v síti, je předinstalována na všech verzích od Win 2000. Je postavena na VBScriptu a PowerShellu, WMI těmto skriptovacím jazykům poskytuje přístup prakticky k čemukoliv ve Windows. Některé další nástroje, které nejsou instalovány se systémem, je možné stáhnout ze stránek Microsoftu http://www.microsoft.com/downloads/details.aspx?familyid=6430F853-1120-48DB-8CC5-F2ABDC3ED314&displaylang=en.

OpenWBEM od Novellu je implementace určená pro Novell Netware, Linux a některé jiné UNIXové systémy (včetně Solarisu a MacOSX). Je používána v komerční i nekomerční sféře a stejně jako WMI nabízí rozsáhlé možnosti zásahů do konfigurace sítě a monitorování. Je ke stažení na http://www.openwbem.org/.

OpenPegasus je další otevřená implementace pro různé UNIXové systémy včetně Linuxu, a Windows. Je ke stažení na http://www.openpegasus.org/.

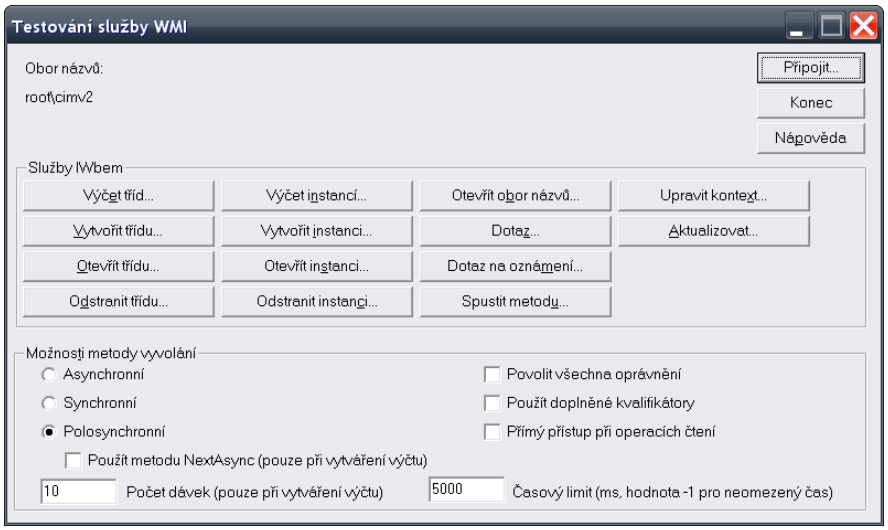
V Linuxu (v menších sítích) se hodně používá nástroj *Webmin*, který zřejmě není přímo implementací WBEM, ale má hodně podobné možnosti. Poskytuje sjednocené rozhraní v internetovém prohlížeči a umožňuje monitorovat, vyhodnocovat a konfigurovat uzly v síti. Varianta s omezenými právy, kterou mohou používat běžní uživatelé, se nazývá *usermin*. Dále existuje varianta Virtualmin pro hromadnou správu virtuálních hostitelů (například v serveru Apache). Webmin je ke stažení na http://webmin.com/.

10.9.2 WMI

Jak bylo výše uvedeno, WMI je implementace modelu WBEM pro správu Windows. Databáze CIM je taktéž objektová, jde o objekty plnohodnotné s implementací tříd, vlastností, metod, událostí, využívající dědičnost a kompozici.

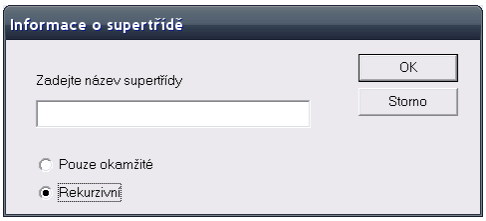
Třídy jsou psány v objektovém jazyce *MOF* (Managed Object Format), který je interpretovaný (dá se říct skriptovací). Většina komponent WMI (včetně tříd) je v adresáři `...\System32\Wbem`. Najdeme tam hodně souborů s příponou MOF.

Pokud nás zajímá, jak vlastně vypadají třídy WMI, můžeme se na ně podívat v nástroji, který je součástí Windows od verze XP. *WbemTest* se spouští příkazem `wbemtest` a má grafické rozhraní. Po spuštění se objeví základní okno, které vidíme na obrázku 10.12.



Obrázek 10.12: Úvodní okno aplikace WbemTest

Nejdřív je nutné se připojit k určitému oboru názvů (zobrazuje se v levém horním rohu okna), na obrázku 10.12 jsme připojeni k oboru názvů *CIMV2* (k oboru názvů se jednoduše připojíme tak, že klepneme na tlačítko *Připojit* a zadáme název). Pak můžeme volně pracovat se třídami, které do zvoleného oboru názvu patří (také vytvářet nové).



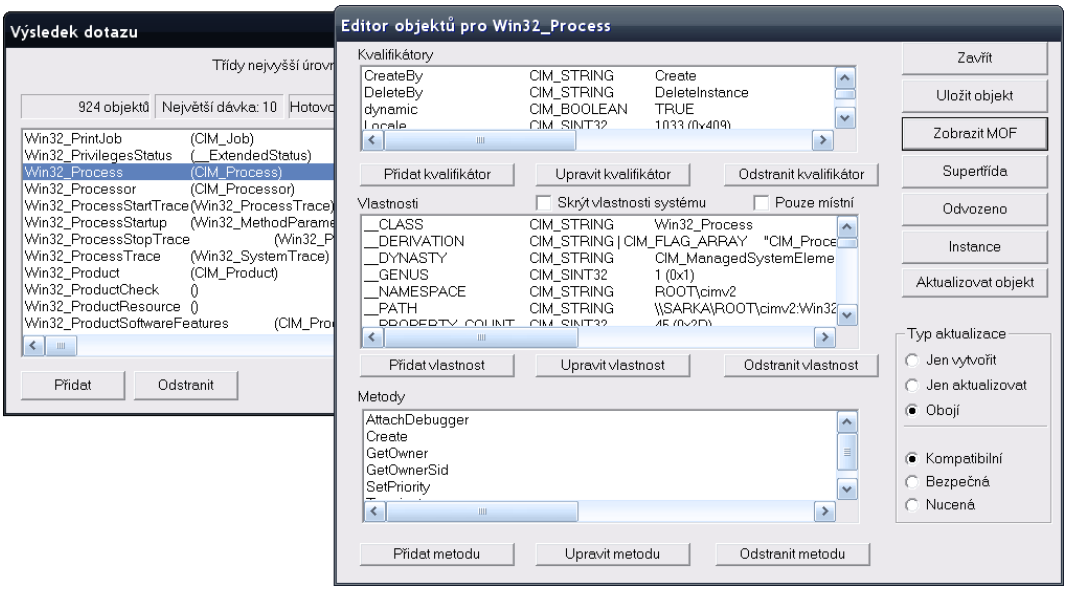
Obrázek 10.13: Stanovení třídy

Pokud si chceme prohlédnout (nebo upravit) vlastnosti některé třídy daného oboru názvů, klepneme v hlavním okně na tlačítko *Výčet tříd*. Zobrazí se dialogové okno (obrázek 10.13), do kterého buď zadáme přímo název třídy, a nebo zvolíme „rekurzivní“ výčet (to znamená, že budou vypsány všechny třídy z oboru názvů). V seznamu pak poklepeme na vybranou třídu a zobrazí se okno se všemi jejími vlastnostmi a metodami (obrázek 10.14).

Kromě výše uvedeného nástroje WbemTest lze ke službě WMI přistupovat přes konzolu *Řízení služby WMI*. Spouštíme ji souborem `wmimgmt.msc`, ale je dostupná také v konzole *Správa počítače*, jak vidíme na obrázku 10.15. V kontextovém menu položky *Řízení služby WMI* zvolíme *Vlastnosti* a získáme okno, které vidíme na obrázku 10.15 vpravo. Nastavujeme obecné vlastnosti řízení WMI jako je způsob protokolování a umístění protokolu, zálohování (máme možnost obnovit databázi WMI ze zálohy) a určujeme zabezpečení prvků databáze.

Dalším velmi užitečným nástrojem pro řízení WMI je program `wmic.exe` (WMI Console). Pomocí tohoto nástroje můžeme přistupovat k databázi WMI a získávat z ní nejrůznější informace. Tomuto příkazu se věnujeme v příloze (kapitola A.5, strana 295).

Rozhraní WMI se velmi často používá ve skriptech (pomocí VB skriptu nebo v PowerShellu se takto dostaneme prakticky k čemukoliv, co na počítači potřebujeme, i přes síť). Skripty můžeme vytvářet buď ručně, a nebo v některém nástroji pro generování WMI skriptů. V minulosti se pro tyto účely hodně používaly nástroje *WMI Code Creator* a *Scriptomatic*, ale u obou Microsoft ukončil podporu a nevyvíjí nové verze (poslední jsou pro Windows XP).



Obrázek 10.14: Vlastnosti zvolené třídy `Win32_Process`

Další informace:

V novějších verzích PowerShellu se dá k WMI přistupovat celkem snadno a na webu najdeme také hodně ukázek, například:

- https://www.darkoperator.com/blog/2013/1/31/introduction-to-wmi-basics-with-powershell-part-1-what-it-is.html
- https://www.simple-talk.com/sysadmin/powershell/powershell-day-to-day-admin-tasks-wmi-cim-and-pswa/
- http://www.tomsitpro.com/articles/working-with-wmi-powershell,1-3581.html

Informace o skriptování (včetně PowerShellu) najdeme na stránce Microsoft Script Center: https://technet.microsoft.com/en-us/scriptcenter/bb410849.aspx

Úkol

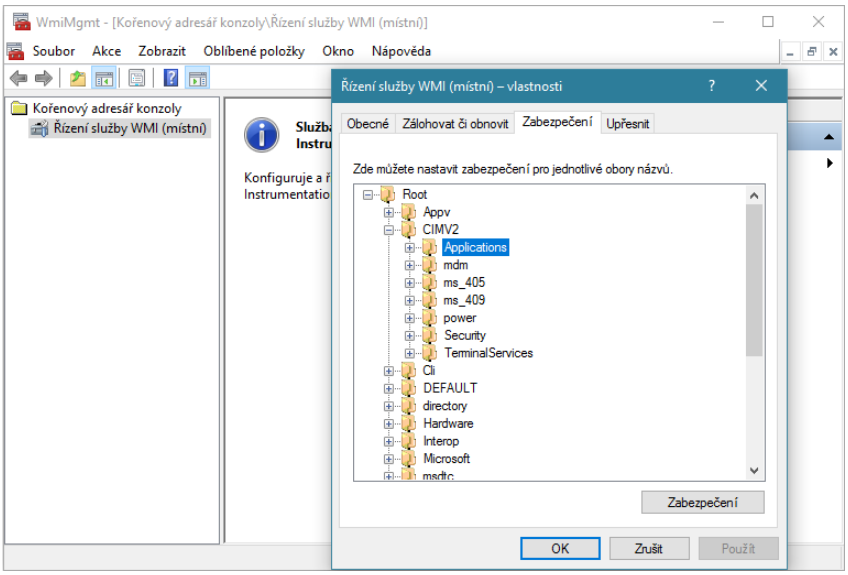
V příloze najděte sekci o programu `wmic` a vyzkoušejte některé z uvedených ukázek jeho používání.

10.10 Dohledové systémy

Dohledový systém (Network Management Software) je systém, který dokáže monitorovat stav sítě (různé veličiny), tedy sbírat informace z různých zdrojů na síti, generovat reporty a v případě problémů (abnormální hodnoty sledovaných veličin, velké výkyvy apod.) vhodně reagovat (například odeslat informaci adminovi).

Když si vybíráme dohledový systém, bereme v úvahu:

- co vše může být monitorováno (množství a typy monitorovaných služeb),
- konkrétnost hlášení (některé poruchy jsou „hierarchické“, tedy pokud něco selže v důsledku selhání něčeho jiného, neměl by report zahrnovat vše, ale pouze „kořen problému“),



Obrázek 10.15: Vlastnosti v konzole *Řízení služby WMI*

- uživatelské rozhraní (dnes je velmi běžné webové rozhraní), mělo by být dostatečně přehledné a vhodně hierarchicky členěné, hodně záleží na tom, co si lze zobrazit a v jaké formě, možnost vizualizovat nejrůznější datové toky či cokoliv dalšího, co se může s časem měnit,
- způsob zasílání hlášení – přes SMTP server (příp. může být i integrovaný), SMS,
- možnosti nastavení uživatelských oprávnění pro přístup k evidovaným informacím,
- funkční omezení – způsob komunikace s hlídanými zařízeními (přes který protokol, např. SNMP), způsob logování, atd.

Existují jak open-source, tak i komerční dohledové systémy. Ke komerčním patří například WhatsUp Gold,[22] SonicWALL,[23] a další. Na některé open-source produkty se podíváme v následujícím textu.

10.10.1 Nagios

Nagios je velmi oblíbený open-source monitorovací (dohledový) systém. Zvládá nejen samotné monitorování, ale i vizualizaci. Dokáže monitorovat různé typy síťových služeb (HTTP, POP3, ICMP, SMTP), nemá problémy s šifrováním, monitoruje také využívání prostředků na uzlech sítě (rozumí si s různými operačními systémy včetně Windows), zvládá vizualizaci stavu sítě (také dokáže vykreslit topologii sítě včetně 3D grafu). Pro konfiguraci lze použít *webové rozhraní* (ale je možné využívat jiná rozhraní, například Centreon v kombinaci s MySQL).

Vznikl nový fork (odnož) Nagiosu, který se nazývá *Icinga*.[24] Podle vývojářů tohoto forku je účelem především zrychlit a zpružnit vývoj.

Další informace:

- http://www.abclinuxu.cz/clanky/site/nagios-plus-centreon-plus-mysql-instalace-a-zakladni-konfigurace

[22] http://www.annexnet.cz/ipswitch-whatsupgold-popis-produktu/
[23] http://www.sonicwall.com/
[24] Systém Icinga včetně zdůvodnění jejího vzniku najdeme na http://www.icinga.org/.

- http://www.nagiosbook.org/
- http://nagios.sourceforge.net/docs/3_0/quickstart.html

10.10.2 Zabbix

Zabbix[25] je dalším oblíbeným open-source monitorovacím systémem. Potřebujeme Zabbix Server (ten nainstalujeme na dohledový server, měl by na něm běžet některý UNIXový systém včetně Linuxu) a Zabbix agenty (na klientských zařízeních, různé operační systémy včetně Windows). Konfigurace se provádí opět přes webové rozhraní, tedy po základní orientaci nic moc složitého.

Funkčnost je podobná, snad mírně nižší, než v případě Nagiosu, obvykle je Zabbix považován za jednodušší ve vybavenosti a konfiguraci (Nagios je zase považován za stabilnější). Taktéž podporuje SNMP a agenty je možné instalovat na různé operační systémy.

Další informace:

- http://www.root.cz/serialy/dohledovy-system-zabbix/
- http://www.zabbix.com/documentation.php

10.10.3 OpenNMS

OpenNMS[26] je open-source dohledový systém, který je narozdíl od předchozích napsán v Javě. Je určen pro monitorování rozsáhlých sítí s velkým množstvím sledovaných zařízení. Monitoring může být distribuovaný (na více serverech), díky tomu lze monitorovat tak velké množství zařízení.

Jde o dobře rozšiřitelný systém. Monitoring různých služeb je řešen pomocí pluginů (podporu monitorování další služby lze přidat jednoduše přidáním pluginu).

Konfigurace se opět provádí přes webové rozhraní. Práci se systémem je možné si vyzkoušet na demo aplikaci na stránkách projektu.

Další informace:

- http://www.howtoforge.com/opennms_network_management
- http://demo.opennms.org/opennms/acegilogin.jsp

10.10.4 Zenoss

Zenoss[27] je další open-source dohledový systém (jedna varianta je open-source a volně ke stažení, druhá – Enterprise – komerční). Je postaven na myšlence využití existujících open-source projektů.

Jeho jádrem je objektový webový server *Zope* napsaný v jazyce Python, a také v Pythonu je možné Zenoss dále rozšiřovat (dokonce lze údajně používat i pluginy z Nagiosu, který je také napsán

[25] *Zabbix* je dostupný na http://www.zabbix.com/.
[26] Informace o OpenNMS najdeme na http://www.opennms.org/wiki/Main_Page.
[27] *Zenoss* je dostupný na http://www.zenoss.com/.

v Pythonu). Dále se přidává databázový systém *MySQL* a celý systém běží nad *Twisted*, což je událostmi řízené rozhraní pro programování síťových aplikací, které si rozumí s mnoha různými protokoly na více vrstvách ISO/OSI.

Důležitou součástí celého systému je *RRDTool* (Round Robin Database Tool). Tento nástroj slouží k ukládání, zpracovávání a vytváření grafické reprezentace (grafů) jakýchkoliv čísel, která mu předáme. Základem je databáze organizovaná jako kruhová fronta (odtud název Round Robin Database) se statickou velikostí, ve které jsou starší data přepisována novějšími.

Zenoss si rozumí s protokoly SNMP, ICMP, protokoly rodiny TCP/IP, komunikuje s UNIXovými systémy (včetně syslogu) a také s Windows (podporuje také WMI). Na webu firmy najdeme také informaci o tom, že jsou podporována také cloud zařízení (pojem Cloud Computing je velmi populární). Konfigurace probíhá přes přehledné webové rozhraní.

Další informace:

- https://www.zenoss.com/product
- http://www.root.cz/serialy/prechadzame-na-rrdtool/
- http://oss.oetiker.ch/rrdtool/tut/
- https://help.ubuntu.com/community/Zenoss

10.10.5 Cacti

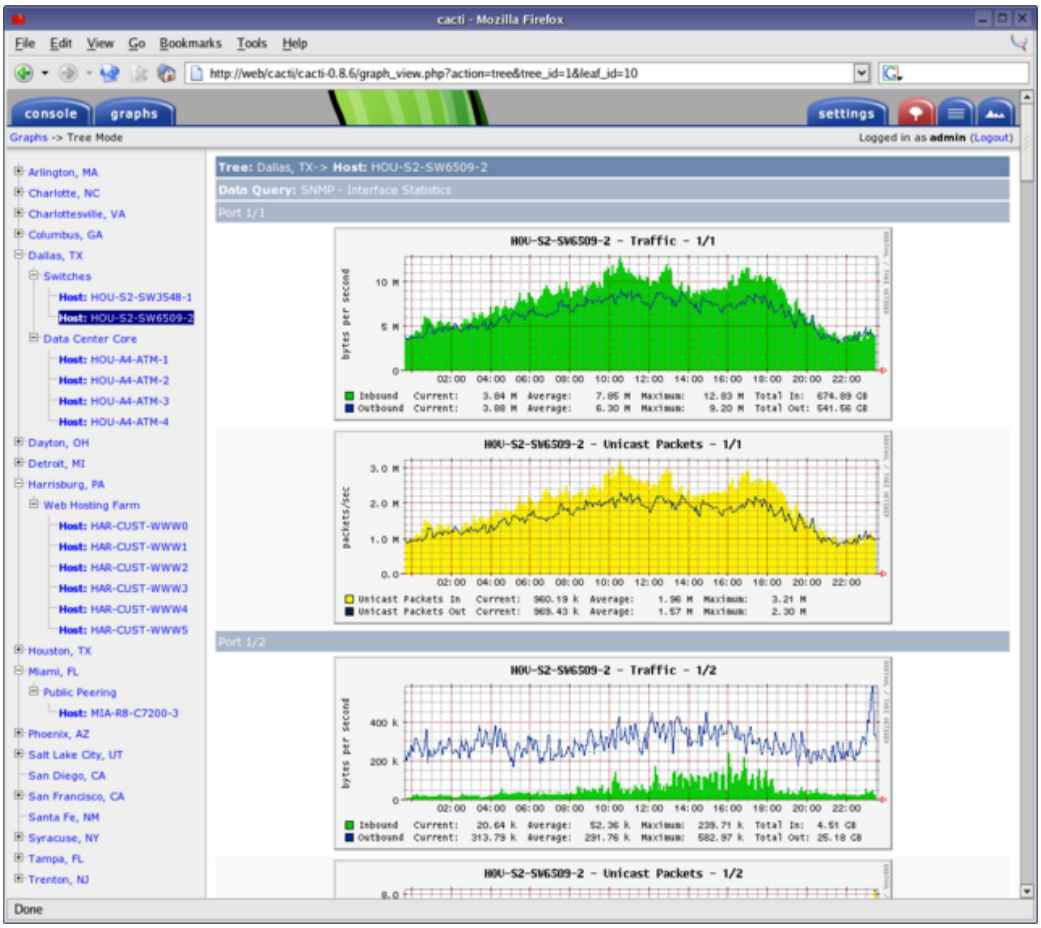
Cacti[28] je další open-source nástavba nástroje RRDTool, podobně jako Zenoss. Také v Cacti jde o to, jak získat data, vhodně je uložit a následně analyzovat a zobrazit. Data mohou být získávána z různých zdrojů, například od SNMP (Net-SNMP), skriptů v nejrůznějších skriptovacích jazycích nebo WMI, a uložena buď do SQL databáze nebo pomocí RRDTool. Systém je určen pro menší a střední sítě (stovky, možná tisíce, uzlů sítě).

Komunita kolem systému Cacti se utěšeně rozrůstá, a tak existuje hodně pluginů a také šablon pro různé typy hodnot. Stejně jako u předchozích nástrojů, také zde máme k dispozici přehledné webové rozhraní pro konfiguraci a manipulaci s daty.

Další informace:

- http://www.root.cz/clanky/cacti-vse-dulezite-v-jednom-monitoru/
- http://www.codewalkers.com/c/a/Server-Administration/Monitoring-Temperatures-with-Cacti/
- http://www.ubuntugeek.com/install-and-configure-cacti-monitoring-tool-in-ubuntu-9-10-karmic-server.html
- http://www.root.cz/clanky/cacti-ziskavani-vlastnich-dat-pomoci-ssh/
- http://www.samuraj-cz.com/clanek/cacti-snmp-monitoring-a-grafy/
- http://forums.cacti.net/about30438.html

[28] *Cacti* najdeme na http://www.cacti.net/.
[29] Zdroj: http://www.cacti.net/screenshots.php



Obrázek 10.16: Dohledový systém Cacti[29]

Úkol

Vyberte si jeden ze zmiňovaných dohledových systémů a najděte o něm co nejvíce informací. Pokud máte možnost, vyzkoušejte ho.

Další informace:

A ještě pár odkazů týkajících se zabezpečení:

- http://www.scycore.com/papers/low_linux_intro.pdf
- http://www.networksecuritytoolkit.org/nst/index.html
- http://www.abclinuxu.cz/clanky/bezpecnost/linuxove-dmz-i
- http://www.networksecuritytoolkit.org/nst/index.html

10.11 SIEM

SIEM (Security Information and Event Management) může svou definicí připomínat dohledový systém, je však více orientován na bezpečnost. Tyto systémy slouží jako nástroje pro shromažďování,

analýzu a prezentaci informací získaných z různých napojených zařízení, ať už běžných součástí sítě nebo zařízení bezpečnostního charakteru (kamery, nejrůznější senzory, IDS/IPS). Obvyklé úlohy:

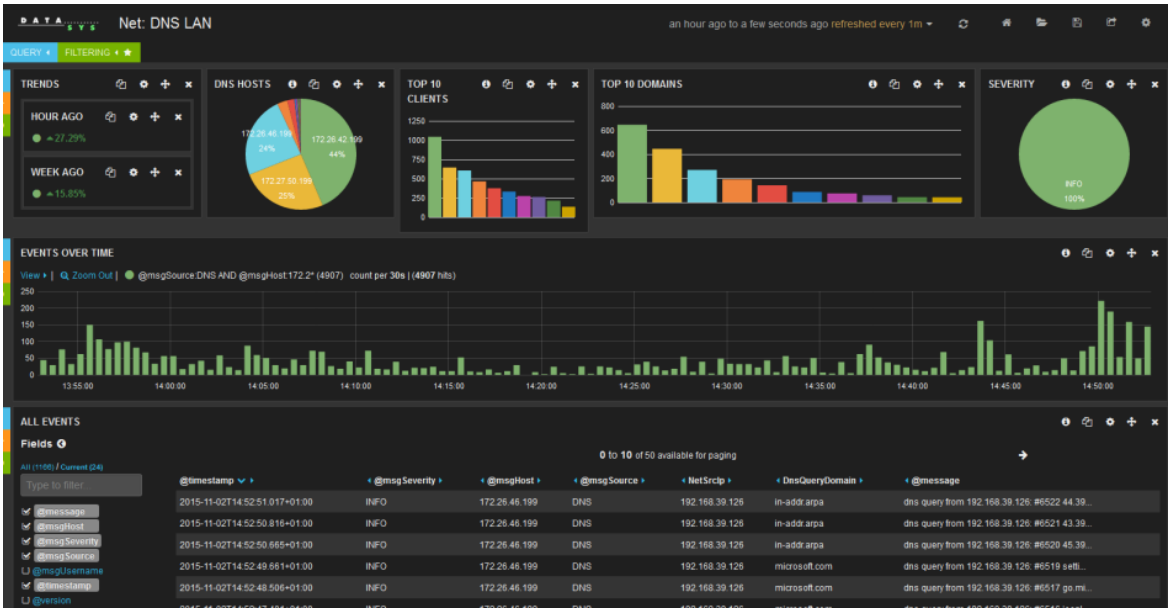
- shromažďování dat z různých zdrojů, práce s logy, normalizace údajů (převod do jednotného formátu),
- korelace – hledání a vyhodnocování vztahů mezi daty, např. mezi různými událostmi v síti,
- okamžité varování při výskytu problému,
- prezentace informací – vytváření přehledů, sestav, grafů, statistik.

Účelem je mít na jednom místě všechny informace potřebné při řešení bezpečnostních incidentů a mít možnost co nejrychleji na ně reagovat.

Můžeme najít buď kompletní SIEM řešení, nebo zvlášť SIM (Security Information Management, tj. práce s logy, třídění informací, reportování, vizualizace apod.) a SEM (Security Event Management, tj. především práce s logy, analýza v reálném čase, hledání závislosti mezi událostmi a rychlá reakce na události). SIEM používají při analýze jak běžné metody z dohledových systémů (jako je práce s logy), tak i metody podobné těm, které používají antivirové programy (práce se signaturami, heuristická analýza apod.).

Nabídka SIEM je poměrně rozsáhlá. Můžeme si vybrat open-source řešení nebo komerční produkt, ovšem vybírat bychom měli především podle vlastních potřeb. Důležitým kritériem je počet zařízení, která v síti máme a na jejichž logy bude SIEM napojen, mělo by nás zajímat, s kterými typy zařízení dokáže SIEM nativně pracovat a naopak která zařízení bude problém napojit, s velikostí sítě souvisí údaj EPS (Events per Second), tedy kolik událostí za sekundu bude muset SIEM zvládat, atd.

Vlastní SIEM nabízí Cisco, z dalších je oblíbený Splunk (má více variant), Helix Security Platform (od FireEye), Qradar SIEM (od IBM), AlienVault OSSIM (od AT&T Cybersecurity), AlienVault USM (od téhož výrobce), FortiSIEM (od Fortinetu), McAfee Enterprise Security Manager, LogRhythm Next-Gen SIEM Platform, Trustwave SIEM, Rapid7 InsightIDR, Juniper JSA SIEM, nebo český produkt ELISA (od DataSys, staví na Zabbixu, Xlogu a NetFlow).



Obrázek 10.17: Kibana – rozhraní používané v SIEM Elisa[30]

Další informace:

- https://www.nagios.com/products/nagios-xi/ (informace o dohledovém systému Nagios a screenshoty)
- http://www.hw-group.com/software/pd_snmp_cz.html (přehled dohledových systémů, hodně komerčních, pár volně šiřitelných, příp. některé komerční, ale s malým počtem sledovaných uzlů zdarma)
- http://www.monitortools.com/
- Kříž, Lukáš. *SIEM: Náročná cesta k pokročilému zajištění kybernetické bezpečnosti* [online], 2019. Dostupné na: https://ictrevue.ihned.cz/c3-66593380-0ICT00_d-66593380-siem-narocna-cesta-k-pokrocilemu-zajisteni-kyberneticke-bezpecnosti

[30] Zdroj: https://docplayer.cz/36323951-Datasys-elisa-log-management-rizeny-zabbixem-lukas-maly-dis-it-konzultant-bezpecnost-a-monitoring.html

Seznam doporučené literatury

[1] *CESNET2*. URL: http://www.cesnet.cz/
[cit. 20. 5. 2010]

[2] Donahue, G. A. *Kompletní průvodce síťového experta*. Brno, Computer Press, 2009.

[3] Empson, S. *CCNA: Kompletní přehled příkazů. Autorizovaný výukový průvodce*. Brno, Computer Press, 2009.

[4] Harper, A. et al. *Hacking – manuál hackera*. Praha, Grada, 2008.

[5] Horák, J. — Keršláger, M. *Počítačové sítě pro začínající správce*. Brno, Computer Press, 2008.

[6] Hubert, B. et al. *Linux Advanced routing & Traffic Control* [online].
URL: http://lartc.org/howto/, další na http://lartc.org/
[cit. 1. 7. 2011]

[7] Janeček, J. *Distribuované systémy* [online].
URL: https://dsn.felk.cvut.cz/wiki/_media/vyuka/cviceni/x36pko/skripta_dsy.pdf
[cit. 20. 5. 2010]

[8] Jones, D. *Automatizace správy a skriptování Microsoft Windows*. Brno, Computer Press, 2006.

[9] *Linux Home Networking* [online]. Knihy o správě počítačových sítí v Linuxu.
URL: http://www.linuxhomenetworking.com/wiki/index.php
[cit. 1. 7. 2011]

[10] Minasi, M. — York, D. *Linux pro administrátory Windows*. Brno, Computer Press, 2004.

[11] Oulehla, Milan a Roman Jašek. *Moderní kryptografie*. Praha: IFP Publishing, 2017. ISBN 978-80-87383-67-4.

[12] Peterka, J. *eArchiv: Co je čím v počítačových sítích* [online].
URL: http://www.earchiv.cz/i_coje.php3
[cit. 20. 5. 2010]

[13] Peterka, J. *eArchiv: Principy počítačových sítí* [online].
URL: http://www.earchiv.cz/i_pri.php3
[cit. 20. 5. 2010]

[14] Price, B. *Active Directory*. Brno, Computer Press, 2005.

[15] Pužmanová, R. *Moderní komunikační sítě od A do Z*. 2. aktualizované vydání. Brno, Computer Press, 2006.

[16] Pužmanová, R. *TCP/IP v kostce*. 2., upr. a rozš. vyd. České Budějovice: Kopp, 2009. ISBN 978-80-7232-388-3.

[17] Ruest, D. — Ruest, N. *Virtualizace: Podrobný průvodce*. Brno, Computer Press, 2010.

[18] Satrapa, P. *Internetový protokol IPv6*. Praha, CZ.NIC, z.s.p.o., 2008. 358 stran. Kniha je dostupná v elektronické formě na
http://knihy.nic.cz/files/nic/edice/pavel_satrapa_ipv6_2008.pdf
[cit. 1. 7. 2011]

[19] Scott, C. — Wolfe, P. — Erwin, M. *Virtual Private Networks*. 2nd edition. O'Reilly, USA, 1999. Náhled většiny stránek je dostupný na Google Books.

[20] Schroder, C. *Linux: Kuchařka administrátora sítě*. Brno, Computer Press, 2009. Z originálu Linux Networking Cookbook vydaného nakladatelstvím O'Reilly Media.

[21] *Svět sítí, tutoriály* [online]. URL: http://www.svetsiti.cz/tutorialy.asp
[cit. 20. 5. 2010]

[22] *System × Virtualization Strategies Development Space* [online]. IBM Redbooks.
URL: http://www-01.ibm.com/redbooks/community/display/REDP4480/Home
[cit. 20. 5. 2010]

[23] Thomas, T.M. *Zabezpečení počítačových sítí*. Autorizovaný průvodce. Brno, Computer Press, 2005.

[24] Trulove, J.. *Sítě LAN: hardware, instalace a zapojení*. Praha: Grada, 2009, 384 s. ISBN 978-80-247-2098-2.

[25] *Výukové materiály Cisco* [online]. URL:
http://www.cisco.com/en/US/docs/internetworking/technology/handbook/ito_doc.html
[cit. 20. 5. 2010]

Přílohy

Příloha A

Práce s adresami a sítěmi ve Windows

V této příloze jsou popsány a demonstrovány příkazy, které se používají ve Windows při práci se sítěmi, Linuxu se věnujeme v další příloze. Můžeme je zatím brát jako inspiraci pro procvičování a taky jako předehru k zatím neexistujícím cvičením z tohoto předmětu.

A.1 Problémy při používání příkazů ve Windows

V serverových Windows podobně jako v desktopových variantách je možné pro téměř vše použít nástroje grafického rozhraní. Velmi často je však praktičtější použít Příkazový řádek – příkazy někdy umějí více než nástroje grafického režimu, lze je snadněji používat pro vzdálenou správu a hlavně příkazy je možné skriptovat a tedy automatizovat zpracování některých úloh.

Ve Windows je s příkazy pro Příkazový řádek podobný problém jako v grafickém rozhraní: různé verze Windows se liší vybaveností a některé příkazy pracují v různých verzích odlišně. Komplikace se projevují především v heterogenních sítích, ve kterých jsou uzly s různými verzemi a případně edicemi Windows (když píšeme skript, musíme dbát na to, aby byl použitelný na všech uzlech, kde ho potřebujeme spouštět).

Jistou komplikací je také změna funkčnosti *RunAs mechanismu* – od verze Vista sice pořád existuje příkaz `runas` pro spuštění procesu s odlišnými přístupovými oprávněními, ale je problematický. Pokud chceme ve Windows od Visty výše spustit příkaz s jinými (typicky administrátorskými) oprávněními než pod kterými právě pracujeme, zvolíme v kontextovém menu ikony programu Příkazového řádku volbu „Spustit jako správce“ a tedy spustíme s vyššími oprávněními už program `cmd.exe`. Svým způsobem to může být bezpečnostní problém, protože někteří uživatelé zřejmě budou na Příkazovém řádku s vyššími oprávněními pracovat i tehdy, když to není nutné.

Pokud chceme některé úlohy provádět vzdáleně, také můžeme narazit na přístupová oprávnění. Obvykle pomůže používat doménový správcovský účet, který lze využít na všech zařízeních v doméně.

A.2 Soubory související se sítěmi

Se sítí souvisí využívání některých (textových) souborů – naštěstí není vše v registru, některé z níže uvedených souborů budou zmíněny u popisovaných příkazů.

Především z důvodu kompatibility se síťovými protokoly a ne-windows síťovými klienty existují ve složce `...\system32\drivers\etc` tyto soubory:

- `networks` – obsahuje doménové a IP adresy lokálních sítí,
- `hosts` – urychluje mapování IP adres na známé doménové adresy (hostitele, anglicky *hosts*),
- `services` – informace o známých síťových službách,
- `protocol` – informace o známých síťových protokolech.

V 64bitových Windows je třeba hledat pomocí 64bitového správce souborů, případně Průzkumníka; v 32bitové aplikaci se totiž některé složky (včetně této) nezobrazují.

Vzpomeňte si, že v UNIXových systémech jsou v adresáři `/etc` nejrůznější konfigurační soubory včetně síťových, zde si Microsoft bral inspiraci.

Úkol

Prohlédněte si obsah souborů `networks`, `hosts`, `services` a `protocol` zmíněných v této sekci.

A.3 Práce s adresami a síťovými kartami

A.3.1 Základní práce s IP adresami – `ipconfig`

Pro zobrazení informací o adresách můžeme použít příkaz `ipconfig`.

`ipconfig /?` (nebo `ipconfig -?`) zobrazí seznam parametrů

`ipconfig /all` vypíše podrobné informace (IP a MAC adresu, adresu brány, masku podsítě)

`ipconfig /release název_karty` uvolnění IP adresy přidělené zadané síťové kartě (když neuvedeme název karty, jsou uvolněny IP adresy všech karet, je možné také použít hvězdičkovou konvenci)

`ipconfig /renew název_karty` obnovení přidělení IP adresy pro zadanou síťovou kartu (když není název karty uveden, provede se pro všechny dostupné karty)

`ipconfig /displaydns` zobrazí záznamy adres přidělených v systému DNS (informace k příslušným DNS jménům včetně hodnoty TTL), a to jak ve směru doménové jméno → IP adresa, tak i ve směru opačném (reverzní adresy, ty jsou v záhlaví označeny řetězcem „in-addr.arpa“); vždy jsou zobrazeny alespoň dva záznamy (localhost a jeho reverzní adresa)

`ipconfig /flushdns` vymaže DNS cache (tj. dotazy na doménové adresy, které takto z cache smažeme, se musejí provádět znovu), použijeme, pokud DNS překlad funguje nekorektně (máme v cache chybné či neaktuální záznamy)

Příklad

Výstup příkazu může vypadat například takto (záleží na verzi Windows, nejdřív ve Windows XP):

```
C:\Windows> ipconfig /all

Konfigurace protokolu IP systému Windows

        Název hostitele . . . . . . . . . . : PCPRACE
        Primární přípona DNS. . . . . . . . :
        Typ uzlu  . . . . . . . . . . . . . : neznámý
        Povoleno směrování IP . . . . . . . : Ne
```

```
        WINS Proxy povoleno . . . . . : Ne

Adaptér sítě Ethernet Připojení k místní síti:

        Přípona DNS podle připojení . . . :
        Popis . . . . . . . . . . . . . . : Broadcom NetXtreme Gigabit Ethernet
        Fyzická Adresa. . . . . . . . . . : 00-0F-FE-12-34-56
        Protokol DHCP povolen . . . . . : Ne
        Adresa IP . . . . . . . . . . . . : 193.84.195.15
        Maska podsítě . . . . . . . . . . : 255.255.255.128
        Výchozí brána . . . . . . . . . . : 193.84.195.1
        Servery DNS . . . . . . . . . . . : 193.84.192.10
```

Výstup téhož příkazu ve Windows 7 je hodně dlouhý, vyjmeme pouze část výstupu odpovídající jedné ethernetové kartě:

```
...
Adaptér sítě Ethernet Připojení k místní síti:

   Přípona DNS podle připojení . . . : fpf.slu.cz
   Popis . . . . . . . . . . . . . . : Atheros AR8131 PCI-E Gigabit Ethernet
Controller (NDIS 6.20)
   Fyzická Adresa. . . . . . . . . . : E0-CB-4E-45-67-89
   Protokol DHCP povolen . . . . . : Ano
   Automatická konfigurace povolena  : Ano
   Místní IPv6 adresa v rámci propojení . . . : fe80::ec8b:a3a6:68bc:9b2f%11
(Preferované)
   Adresa IPv4 . . . . . . . . . . . : 193.84.194.100(Preferované)
   Maska podsítě . . . . . . . . . . : 255.255.255.128
   Zapůjčeno . . . . . . . . . . . . : 9. června 2011 15:25:14
   Zápůjčka vyprší . . . . . . . . . : 9. června 2011 21:25:14
   Výchozí brána . . . . . . . . . . : 193.84.194.129
   Server DHCP . . . . . . . . . . . : 193.84.192.10
   IAID DHCPv6  . . . . . . . . . . .  : 248613215
   DUID klienta DHCPv6. . . . . . . .  : 00-01-00-01-12-F4-7B-66-E0-CB-4E-45-67-89

   Servery DNS . . . . . . . . . . . : 193.84.192.10
   Rozhraní NetBios nad protokolem TCP/IP. . . . . . . . : Povoleno
...
```

Řetězec DUID (DHCP Unique Identifier) jednoznačně identifikuje klienta serveru DHCPv6 a naopak, je jedinečný pro danou konverzaci s DHCP serverem při stanovení konfiguračních parametrů. Všimněte si vazby na MAC adresu.

Číslo IAID (Identity Association Identifier) jednoznačně identifikuje sadu adres přidělených DHCP klientovi. Klient může mít k danému síťovému rozhraní více sad IPv6 adres (plus další potřebné adresy, například musí znát adresu DNS serveru), každá taková sada musí mít jiné IAID.

Příklad

Pokud se nedaří navázat síťové spojení nebo spojení nefunguje korektně (například tehdy, když naše síťová karta omylem získala IP adresu, která je již přidělena jinému zařízení, a nebo nebyly korektně získány adresy DNS serverů), může pomoci znovuzískání IP adresy a souvisejících informací (adresa brány, adresy DNS serverů). Použijeme postupně tyto příkazy:

`ipconfig /flushdns` pročistíme DNS cache, tím se zbavíme starších potenciálně chybných záznamů (tento krok nemusí být nutný)

`ipconfig /release` uvolnění IP adresy přidělené všem síťovým kartám, ale často nebývá nutné

`ipconfig /renew` obnovení přidělení IP adresy všech síťových karet

`ipconfig /all` ověříme si, zda je vše vpořádku

Tento postup samozřejmě nebudeme používat, pokud máme pevnou (statickou) IP adresu. Kdybychom se o to pokusili, obdržíme chybové hlášení sdělující, že daný adaptér (tj. síťová karta) není ve službě DHCP povolen (proto nemůže žádat na DHCP serveru jinou adresu).

Když budeme chtít takto obnovit adresu pro jedinou konkrétní kartu, zadáme v příkazech název (můžeme používat i hvězdičkovou konvenci pro zkrácení názvu), například

```
ipconfig /release Připojení*1
```

(pokud máme síťovou kartu s názvem „Připojení k místní síti 1“).

A.3.2 Překlad adres – `arp` a `nslookup`

ARP je protokol převádějící IP adresy na fyzické (MAC) adresy. Fyzická adresa je v síti zjišťována odesíláním tzv. ARP paketů s dotazy. Aby nebylo nutné opakovaně posílat tyto pakety, udržuje každé síťové zařízení (tj. také zvlášť každá síťová karta) mezipaměť, do které ukládá dosud zjištěné dvojice IP a fyzických adres (ARP tabulky). ARP tabulku je možné prohlížet a přidávat i mazat záznamy. Ukážeme si využití příkazu `arp`, který slouží právě k práci s ARP tabulkou.

`arp -a` zobrazí ARP tabulku, také vidíme, které záznamy jsou dynamické (automaticky vytvořené z komunikace) a které statické (ručně vložené)

`arp -a -n 10.0.128.10` zobrazí ARP tabulku pouze pro síťovou kartu, která má přidělenu zadanou IP adresu

`arp -s 123.123.123.123 00-12-34-56-78-9A` přidá se do ARP tabulky statický záznam se vztahem zadané IP adresy a MAC (fyzické) adresy, MAC adresa se zadává v hexadecimálních číslech s pomlčkami

`arp -d 123.123.123.123` odstraní z ARP tabulky záznam pro zadanou IP adresu

`arp -d *` odstraní z ARP tabulky všechny záznamy

Lze přidat parametr s určením IP adresy síťové karty, pro kterou daná ARP tabulka platí (každá karta má svou tabulku), to má smysl jen v případě, kdy je v provozu víc než jedna síťová karta.

Příklad

ARP tabulka může vypadat takto (na Windows 7):

```
C:\Windows> arp -a

Rozhraní: 193.84.194.100 -- 0xb
  internetová adresa    fyzická adresa        typ
  193.84.194.129        00-0d-66-22-11-00     dynamická
  193.84.194.162        00-1e-4f-33-22-11     dynamická
  193.84.194.185        00-1e-4f-dd-bb-00     dynamická
  193.84.194.186        00-26-2d-ff-ee-11     dynamická
  193.84.194.188        00-1f-d0-66-77-88     dynamická
  193.84.194.212        00-1e-4f-aa-11-bb     dynamická
  193.84.194.220        00-1e-4f-dd-cc-bb     dynamická
  193.84.194.227        00-24-be-77-66-55     dynamická
  193.84.194.229        00-1e-4f-ee-dd-44     dynamická
```

```
  193.84.194.255        ff-ff-ff-ff-ff-ff     statická
  224.0.0.2             01-00-5e-00-00-02     statická
  224.0.0.22            01-00-5e-00-00-16     statická
  224.0.0.252           01-00-5e-00-00-fc     statická
  239.255.255.250       01-00-5e-7f-ff-fa     statická
  255.255.255.255       ff-ff-ff-ff-ff-ff     statická
...
```

Ve Windows nižších verzí je ARP tabulka výrazně kratší. V tabulce vidíme dynamické i statické záznamy, u všech je IP a MAC adresa. Všimněte si skupinových a všesměrových IP a MAC adres (ke skupinové IP adrese patří skupinová MAC adresa a naopak).

Příklad

Určitě jste si všimli, že ve výstupu příkazu `arp` jsou pouze IPv4 adresy. Pokud je třeba vypsat IPv6 sousedy, musíme použít příkaz `netsh`, vypisujeme tabulku protokolu NDP:

```
netsh interface ipv6 show neighbors
```

Zatímco protokol ARP slouží k překladu IP adresy na MAC adresu, protokol DNS slouží k překladu doménové adresy na IP adresu. Pro základní přístup k tomuto překladu lze i na klientských stanicích využít příkaz `nslookup`. Příkaz lze využívat běžným způsobem i interaktivně.

`nslookup www.seznam.cz` vypíše se nejdřív adresa DNS serveru, který poskytl odpověď, a pak samotná odpověď (IP adresy a aliasy pro zadaný doménový název)

`nslookup 77.76.75.3` reverzní překlad, takto zjistíme doménovou adresu k této IP adrese

`nslookup` přechod do interaktivního režimu, tento režim ukončíme příkazem `exit`

Plnou nápovědu k tomuto příkazu získáme pouze v interaktivním režimu: tedy výše uvedeným příkazem přejdeme do interaktivního režimu a pak zadáme `?` nebo `help`.

Příklad

Ukážeme si práci v interaktivním režimu:

`nslookup` spustíme program `nslookup`, prompt se změní na symbol `>`

`?` zobrazíme nápovědu, je značně podrobnější než `nslookup /?` vně interaktivního režimu; také lze použít vnitřní příkaz `help`

`www.google.com` napíšeme doménovou adresu, získáme IP adresu, výstup (v různých verzích Windows se ve výstupech setkáme s více či méně kostrbatou češtinou):

```
Server:  decsu.fpf.slu.cz
Address:  193.84.192.10

Neautorizovana odpoved:
Název:   www.l.google.com
Addresses:  209.85.149.147
          209.85.149.99
          209.85.149.103
          209.85.149.104
          209.85.149.105
          209.85.149.106
Aliases:  www.google.com
```

`set all` tato forma příkazu `set` nic nenastavuje, ale informuje nás o momentálním nastavení pro poslední zadanou adresu (v našem případě pro `www.google.com`; pokud jsme předtím nic nepřekládali, pak obecně platná nastavení), výstup:

```
Vychozí server:   decsu.fpf.slu.cz
Address:  193.84.192.10

Hostitel =   www.l.google.com
Addresses:  209.85.149.147
          209.85.149.99
          209.85.149.103
          209.85.149.104
          209.85.149.105
          209.85.149.106
Aliases:  www.google.com

Nastaveni moznosti:
  nodebug
  defname
  search
  recurse
  nod2
  novc
  noignoretc
  port=53
  typ=A+AAAA
  trida=IN
  doba vyprseni casoveho limitu=2
  obnoveni=1
  koren=A.ROOT-SERVERS.NET.
  domena=fpf.slu.cz
  MSxfr
  verze IXFRversion=1
  srchlist=fpf.slu.cz/slu.cz
```

Takže se například dozvíme, že se použil rekurzivní dotaz, jedná se o záznam adresy pro IPv4 i IPv6 (typ A+AAAA), atd.

`set norecurse` od této chvíle bude požadován iterativní (nerekurzivní) typ dotazů, zpět nastavíme příkazem `set recurse`

`ls fpf.slu.cz` chceme vypsat adresy (počítače) v zadané doméně, výpis je celkem dlouhý

`ls -t ns fpf.slu.cz` vypíšou se pouze doménové servery (typ je NS, tedy name server)

`ls -t aaaa slu.cz` vypíšou se zařízení s IPv6 adresou (tj. záznamy typu AAAA) v zadané doméně

A.3.3 Směrování – `route`

Pokud posíláme data na určitou IP adresu, pakety obvykle (ne vždy) procházejí přes bránu do Internetu či jiné sítě. Proto každý uzel sítě včetně klientských počítačů zná adresu brány a případně další směrovací informace (tj. kterým směrem poslat data do určité (pod)sítě), vede si svou směrovací tabulku.

Ve Windows se s touto tabulkou pracuje příkazem `route`.

`route` (nebo s jakýmkoliv nesprávným parametrem) zobrazí nápovědu k příkazu

`route print` vypíše směrovací tabulku, v nejnovějších verzích Windows vlastně dvě tabulky – zvlášť pro IPv4 a IPv6; na začátku výpisu také máme seznam všech síťových rozhraní včetně virtuálních, tento seznam může být také užitečný

`route print 193.*` vypíše pouze ty cíle ze směrovací tabulky, které odpovídají zadanému výrazu (můžeme používat hvězdičkovou konvenci, také symbol `?` pro jeden znak)

`route add 193.84.197.0 mask 255.255.255.128 193.84.195.8 metric 4` do tabulky přidá statický záznam určující, že k síti s IP adresou `193.84.195.8` a maskou `255.255.255.128` se dá dostat přes bránu s IP adresou `193.84.195.63` a metrika (cena trasy) je 8

`route -p add 193.84.197.0 mask 255.255.255.128 193.84.195.8 metric 4` totéž jako předchozí, ale záznam zůstane v tabulce i po restartu systému (bez přepínače `-p` by byl záznam platný jen do konce činnosti systému)

`route change 193.84.197.0 mask 255.255.255.128 193.84.195.9 metric 3 if 2` změna existující položky tabulky (změna platí i po restartu), změnili jsme adresu brány, metriku a navíc jsme nastavili směrování na síťové rozhraní (kartu) číslo 2, k danému cíli zadáváme vždy jen ty položky, které chceme změnit

`route del 193.84.197.0` odstraníme záznam z tabulky (lze použít i hvězdičkovou konvenci)

Záznamy ve směrovací tabulce lze tímto příkazem také mazat a měnit. Když máme více síťových karet, lze v příkazu pro přidání do tabulky použít i parametr určující síťové rozhraní.

Příklad

Příkaz `route print` ve Windows XP vypíše tento výstup:

```
===========================================================================
Seznam rozhraní
0x1 ........................... MS TCP Loopback interface
0x2 ...00 0f fe 12 34 56 ...... Broadcom NetXtreme Gigabit Ethernet - Packet
 Scheduler Miniport
===========================================================================
===========================================================================
Aktivní směrování:
       Cíl v~síti     Síťová maska           Brána        Rozhraní Metrika
          0.0.0.0          0.0.0.0     193.84.195.1   193.84.195.15      10
        127.0.0.0        255.0.0.0        127.0.0.1       127.0.0.1      1
     193.84.195.0  255.255.255.128    193.84.195.15   193.84.195.15      10
    193.84.195.15  255.255.255.255        127.0.0.1       127.0.0.1      10
   193.84.195.255  255.255.255.255    193.84.195.15   193.84.195.15      10
        224.0.0.0        240.0.0.0    193.84.195.15   193.84.195.15      10
  255.255.255.255  255.255.255.255    193.84.195.15   193.84.195.15      1
Výchozí brána:      193.84.195.1
===========================================================================
Trvalé trasy:
  Žádné
```

V seznamu síťových rozhraní jsou pouze dvě, jedno je loopback (místní smyčka), druhé je ethernetová síťová karta. Ve vyšších verzích bychom zde viděli více záznamů (reálné síťové karty ethernetové, Wi-fi, Bluetooth, ale i virtuální zařízení), a také virtuální počítače si instalují vlastní virtuální síťové karty. Například pokud používáme virtuální počítač od VMWare, najdeme v seznamu rozhraní podobné záznamy:

```
20...00 50 56 c0 00 01 ......VMware Virtual Ethernet Adapter for VMnet1
21...00 50 56 c0 00 08 ......VMware Virtual Ethernet Adapter for VMnet8
```

Výpis byl pořízen na stroji, jehož síťová karta (jediná) má přiřazenu IP adresu 193.84.195.15, všimněte si, kde všude se tato adresa v tabulce vyskytuje (například u místních a multicast směrování).

Z výpisu vidíme, že žádné statické (trvalé) trasy nejsou definovány. Všimněte si, že poslední řádek tabulky obsahuje informaci o výchozí bráně – co není směrováno podle výše uvedených pravidel, to odchází na zadanou výchozí bránu.

Pokud zadáme síťové zařízení či název sítě symbolickým názvem, příkaz tento název musí přeložit na IP adresu. K tomu využije obsah souboru

- `...\windows\system32\drivers\etc\networks`, pokud jde o název cíle směrování (obvykle to bývá síť či podsíť),
- `...\windows\system32\drivers\etc\hosts`, pokud jde o bránu (brána je konkrétní uzel v síti, tedy hostitel protokolu, anglicky *host*).

Do těchto souborů můžeme přidávat vlastní symbolické názvy (je to celkem jednoduché, soubory jsou okomentovány). V souboru `hosts` je vždy nejméně jeden název uzlu, a to *localhost*.

Úkoly

1. Zjistěte svou IP adresu, masku, adresu brány, adresu DNS serveru.
2. Máte statickou IP adresu nebo ji získáváte dynamicky od DHCP serveru? Kde to zjistíte? Pokud máte dynamickou adresu, uvolněte ji (deaktivujte síťovou kartu) a získejte novou.
3. Zobrazte ARP tabulku.
4. Zjistěte IP adresu serveru `extrahardware.cz`.
5. Zjistěte adresu svého DNS serveru, zda při komunikaci s DNS serverem používáte rekurzivní nebo iterativní dotazy, na kterém portu komunikujete, v jaké jste doméně. Vypište seznam uzlů ve vaší doméně.
6. Vypište směrovací tabulku na vašem počítači.

A.3.4 Malá síť a skupina (workgroup)

Na systémech s Windows se setkáváme také se zkratkou *NBT* (nebo NetBT, NetBIOS over TCP/IP). Samotný NetBIOS funguje defacto jako jmenná, spojovací a komunikační služba v malých sítích s Windows (skupiny počítačů v síti typu peer-to-peer). Umožňuje ve skupinách sdílet různé prostředky (složky, tiskárny). Názvy počítačů, které zadáváme v grafickém rozhraní Windows, jsou vlastně názvy podle protokolu NetBIOS.

Aby bylo možné názvy NetBIOSu používat i mimo malé sítě, pracuje dnes NetBIOS nad protokoly TCP/IP ve formě NBT. Pro práci s názvy v NBT můžeme využít příkazy `hostname` a `nbtstat`:

`hostname` zobrazí jméno našeho počítače v NBT

`nbtstat` zobrazí nápovědu příkazu `nbtstat`

`nbtstat -n` zjistíme všechny názvy podle protokolu NBT, které souvisejí s tímto zařízením (tj. obvykle jen název našeho počítače, pokud počítač patří do skupiny, tak i název této skupiny), jestliže máme více síťových karet (tj. více rozhraní), pak se zobrazí zvlášť tabulky pro všechna aktivní rozhraní (která mají přidělenu IP adresu), i když u všech najdeme stejný NBT název

`nbtstat -c` vypíše se seznam ostatních počítačů ve skupině (NetBIOS názvy a IP adresy)

`nbtstat -a` *`počítač`* totéž, ale pro jiný (zadaný) počítač patřící do naší skupiny, zadáváme NetBIOS jméno

`nbtstat -A` *`IP_adresa`* totéž jako předchozí, ale počítač zadáváme jeho IP adresou

Také můžeme zobrazit tabulku relací s daným uzlem v síti (skupině).

Příklad

Předpokládejme, že máme dva počítače připojené ve skupině *mojeskupina*:

- *pcnotebook* s IP adresou 10.0.0.1,
- *pcdesktop* s IP adresou 10.0.0.3,

Na prvním z nich nejdřív vypíšeme tabulku místních NBT názvů:

```
C:\Windows> nbtstat -n

Připojení k místní síti:
Adresa IP uzlu: [10.0.0.1] ID oboru: []

               Tabulka místních názvů systému NetBIOS

    Název                Typ          Stav
-----------------------
PCNOTEBOOK      <00>  UNIQUE      Registrovaný
MOJESKUPINA     <00>  GROUP       Registrovaný
PCNOTEBOOK      <20>  UNIQUE      Registrovaný
```

Teď na tomtéž počítači vypíšeme seznam dostupných počítačů:

```
C:\Windows> nbtstat -c

Připojení k místní síti:
Adresa IP uzlu: [10.0.0.1] ID oboru: []

               Název vzdálené paměti NetBIOS Tabulka

    Název                Typ      Adr. hostitele   Životnost [sek]
---------------------------------
PCDESKTOP       <20>  UNIQUE       10.0.0.3           92
```

Pokud dostaneme jen chybovou hlášku „V mezipaměti nejsou žádné názvy“, důvodem může být to, že počítače nejsou ve stejné skupině (tj. nevidí se), ale také se tento problém může/nemusí objevit, pokud na těchto počítačích máme různé verze Windows.

Můžeme také vypsat NBT tabulku druhého počítače (jsme pořád na tomtéž počítači):

```
C:\Windows> nbtstat -a pcdesktop

Připojení k místní síti:
Adresa IP uzlu: [10.0.0.1] ID oboru: []
```

```
            Tabulka názvů vzdálených počítačů NetBIOS

   Název               Typ          Stav
----------------------
PCDESKTOP       <00>  UNIQUE      Registrovaný
MOJESKUPINA     <00>  GROUP       Registrovaný
PCDESKTOP       <20>  UNIQUE      Registrovaný

Adresa MAC = 00-1D-0F-12-34-56
```

Jiný přepínač použijeme, pokud příkaz zadáme s využitím IP adresy:

`nbtstat -A 10.0.0.3`

Výstup by byl tentýž jako v předchozím případě. Příkaz bohužel nepřijímá regulární výrazy ani hvězdičkovou konvenci, tedy případnou automatizaci (například vypsání údajů pro více různých adres) bychom mohli řešit například příkazem `for` (viz skripta pro Windows z předmětu Operační systémy).

Úkol

Pokud jste v LAN s více počítači, které jsou ve stejné skupině, vyzkoušejte použití příkazu `nbtstat` podle ukázek v příkladech. Pokud ne, vyzkoušejte alespoň výpis NBT názvů na vlastním počítači.

A.4 Testování a statistiky

A.4.1 Testování cesty a průchodnosti

To, zda je dostupná doménová nebo IP adresa (resp. příslušné zařízení na síti), zjistíme příkazem `ping`. Tento příkaz odesílá k danému zařízení ICMP pakety (zprávy *ICMP Echo Request*) a podle zaslaných odpovědí určuje, zda je počítač dostupný a jaká je odezva připojení (pokud má vzdálený počítač firewall nakonfigurovaný tak, aby požadavky `ping` byly ignorovány, může se počítač jevit jako nedostupný)

`ping www.google.com` zjistí, zda je zadaný server dostupný (je dostupný prakticky vždy, tedy pokud se zobrazí hláška, že tomu tak není, zřejmě nejsme připojeni k síti nebo je někde na cestě porucha)

`ping -n 2 www.google.com` tento parametr omezí (nebo u většího čísla navýší) počet odesílaných testovacích paketů, zde na 2; výchozí hodnota je 4 pakety

`ping -t www.google.com` počet odesílaných paketů není stanoven, posílají se opakovaně až do přerušení klávesovou zkratkou Ctrl+C

Příklad

Příkaz `ping` můžeme použít i pro otestování funkčnosti vlastní síťové karty:

`ping loopback` nebo

`ping 127.0.0.1`

Příklad

Jestliže se síťové spojení podle parametrů zdá v pořádku, ale ve skutečnosti nefunguje, ověříme, zda je funkční DNS překlad adres (doménových adres na IP adresy):

`ping www.google.com` skončí chybovým hlášením, když spojení nefunguje (můžeme použít adresu jakéhokoliv serveru na Internetu, u kterého máme jistotu, že odpovídá na žádosti příkazu `ping`)

`ping 74.125.79.104` provedeme totéž, ale použijeme IP adresu (zde se jedná o IP adresu serveru z předchozího příkazu)

Pokud první příkaz selhal a druhý byl úspěšný, pravděpodobně nefunguje DNS server. Měli bychom tedy reagovat buď zjištěním správné adresy (například u svého poskytovatele připojení) nebo použitím adresy veřejného DNS serveru.

Takových serverů již existuje dostatek, například Google má veřejné DNS servery na adresách `8.8.8.8` a `8.8.4.4`, případně můžeme použít DNS servery sdružení CZ.NIC `217.31.204.130` nebo `217.31.204.131`. Ve Windows se adresy DNS serverů nastavují v grafickém režimu, v Síťových připojeních (vlastnostech protokolu TCP/IP).

Návod najdeme například na adrese
http://www.nic.cz/page/847/jak-nastavit-verejne-dns-servery-cz.nic/.

Program `tracert` používáme, když chceme znát cestu, po které procházejí naše pakety (tj. adresy směrovačů na cestě). Ve Windows se používá metoda ICMP paketů posílaných v IP paketech s postupně se zvyšující hodnotou TTL.

`tracert -?` vypíše nápovědu (pozor, UNIXová syntaxe, přepínače píšeme s pomlčkou)

`tracert www.google.com` chceme trasu k zadanému serveru, postupně (celkem pomalu) se vypíšou všechny směrovače na cestě včetně odezvy v ms počítané od předchozího uzlu

Můžeme zadat také IP adresu. Metoda ICMP paketů je poměrně pomalá. Navíc některé směrovače jsou z bezpečnostních důvodů nastaveny tak, aby neodpovídaly na ICMP pakety (pak od dotyčného uzlu obdržíme informaci o vypršení časového limitu žádosti), také je obvyklé, že se takto nedostaneme přímo ke konkrétnímu serveru, ale jen ke směrovači na okraji sítě, ve které je daný server.

Program `pathping` slouží k podobným účelům, také dokáže vypsat trasu k zadanému serveru (jednotlivé směrovače) a navíc pro každý tento úsek vypočívává statistiku stejnou jako příkaz `ping`. Pracuje na principu zpráv *ICMP Echo Request* a *ICMP Echo Reply* stejně jako příkaz `ping`.

`pathping -?` vypíšeme nápovědu

`pathping www.google.com` spustíme výpis směrovačů a výpočet statistik

Zjištění uzlů na cestě může být mírně rychlejší než u `tracert`, ale počítání statistiky naopak zabere více než 100 sekund a opět se nedostaneme moc daleko (dokonce můžeme „uváznout“ v síti našeho ISP). Pokud se nám zdá výpočet příliš dlouhý, program lze předčasně ukončit stisknutím Ctrl+C.

A.4.2 Statistiky v `netstatu`

Velice silný nástroj použitelný pro analýzu statistik souvisejících s protokoly rodiny TCP/IP je `netstat`. Stejně jako jiné příkazy pro práci se sítěmi vznikl podle podobně pojmenovaných nástrojů z UNIXu, u příkazu `netstat` se však setkáme se zřejmě největší podobností. Příkaz nejen zachovává

UNIXovou syntaxi (přepínače se píší za pomlčku), ale také dokonce umožňuje sdružovat jednopísmenné přepínače k jedné pomlčce. Množina přepínačů je odlišná v různých verzích Windows!

`netstat` vypíše základní statistiku (nevšímá si aplikací, které jen naslouchají, vypisuje TCP spojení)

`netstat -a` vypíše celou statistiku

`netstat -a -o` vypíše celou statistiku, přidá sloupec s PID příslušného procesu

`netstat -ao` totéž (přepínače můžeme sdružovat do jediného řetězce)

`netstat -ano` navíc místo doménových vypisuje IP adresy, tato forma je velmi používaná a je dobré si ji zapamatovat

`netstat -ab > seznam.log` vypíše všechny procesy (názvy i PID), které právě komunikují se sítí (jakkoliv), výsledek je směrován do zadaného souboru

`netstat -e` základní statistika pro Ethernet (počet odeslaných a přijatých paketů, počet chyb apod. – týká se protokolů rodiny TCP/IP)

`netstat -es` podrobná statistika *všech* protokolů z rodiny TCP/IP

`netstat -sp tcp` podrobná statistika, přepínač `-p` s přidaným parametrem `tcp` určuje, že chceme pouze statistiku protokolu TCP (můžeme použít parametr `tcp`, `udp` nebo `ip`)

`netstat 2` vypisuje výstup opakovaně každé dvě sekundy (můžeme zadat jakýkoliv interval a také kombinovat s jinými přepínači), činnost příkazu lze ukončit jen klávesovou zkratkou Ctrl+C

`netstat -r` zobrazí směrovací tabulku, stejný výstup jako příkaz `route print`

Příklad

Po několika prohlédnutých stránkách ve webovém prohlížeči může statistika IP vypadat takto:

```
C:\Windows> netstat -sp ip

Statistika protokolu IPv4

  Počet přijatých paketů             = 30736
  Přijato s chybami hlaviček         = 0
  Přijato s chybami adresy           = 3509
  Předané datagramy                  = 0
  Přijato s neznámým protokolem      = 0
  Zahozené přijaté pakety            = 1078
  Doručené přijaté pakety            = 26369
  Požadavky na výstup                = 23914
  Zahozená směrování                 = 0
  Zahozené výstupní pakety           = 2
  Nesměrovatelné výstupní pakety     = 0
  Vyžadováno nové sestavení          = 0
  Úspěšná nová sestavení             = 0
  Chybná nová sestavení              = 0
  Úspěšně fragmentované datagramy    = 0
  Neúspěšně fragmentované datagramy  = 0
  Vytvořené fragmenty                = 0
```

Úkoly

1. V předchozím textu najděte IP adresy veřejných DNS serverů (jeden od Googlu, jeden od CZ.NIC) a vyzkoušejte na ně příkaz `ping`. Všimněte si časových údajů ve výpisech.

2. Všimněte si rozdílu (včetně časového) ve výstupech příkazů
   `tracert www.google.com` `pathping www.google.com`
3. Zjistěte PID všech procesů, které právě komunikují po síti (případně naslouchají).
4. Vypište statistiku protokolů rodiny TCP/IP.
5. Vypište podrobnou statistiku protokolu IP.
6. Vypiste podrobnou statistiku protokolu TCP, přitom zobrazte i sloupec s PID komunikujícího procesu (přepínač `-o`, například `netstat -osp tcp`). Zjistěte, kterým procesům patří řádky s navázaným spojením nebo čekající (time-wait) – může to být například Skype, mail klient, webový prohlížeč, antivirus, apod. O který proces jde u konkrétního PID, zjistíte například ve Správci úloh (zobrazte si sloupec s PID), Process Exploreru nebo ve výstupu příkazu `tasklist`.

A.5 Služba WMI a program `wmic`

 Program `wmic` (WMI Console) je dostupný od verze Windows XP/Server 2003. Umožňuje využívat rozhraní WMI na Příkazovém řádku. pracuje ve dvou režimech – klasickém (externím) a interaktivním. Při použití v klasickém režimu zadáváme příkaz začínající názvem programu (`wmic`) následovaný parametry, do interaktivního režimu se dostaneme zadáním příkazu `wmic` bez dalších parametrů (prompt se změní na `wmic:root\cli>` a zadáváme interní příkazy).

Nápovědu získáme buď v grafickém režimu, nebo příkazem `wmic /?`, a nebo v interaktivním režimu této konzoly příkazem `/?`. Ještě podrobnější nápovědu zobrazíme příkazem `/?:FULL`.

Příkaz `wmic` je velmi komplexní. Má tuto syntaxi:
`WMIC přepínače předmět sloveso parametry`, kde

- *přepínače* nastavují obecné chování příkazu, mohou být například
  - `/node:`*`počítač`* – příkaz bude proveden na zadaném počítači (před názvem počítače nebudeme dávat opačná lomítka)
  - `/user:`*`uživatel`* – při vyhodnocování bude použit zadaný uživatel s jeho přístupovými oprávněními (budeme dotázáni na heslo)
  - `/output:`*`soubor`* – výpis bude směrován do zadaného souboru
- *předmět* (také alias) určuje to, s čím chceme manipulovat nebo na co se dotazujeme, následuje *zkrácený* seznam (všechny možnosti zjistíme v nápovědě):

bios	diskdrive	memPhysical	onBoardDevice	registry
bootconfig	fsdir	netuse	OS	service
cpu	irq	NIC	pageFile	share
dcomApp	job	NICConfig	printer	temperature
desktop	memLogical	NTEvent	process	useraccount

- *sloveso* určuje požadovanou akci, která má být provedena s předmětem:
  - LIST – toto sloveso lze použít na všechny předměty, zobrazí obecnou informaci o předmětu, můžeme upřesnit, jak podrobné informace chceme (full – všechny, brief – základní v tabulce, writeable – které lze měnit, atd.),

- GET – získání podrobnějších informací o všech nebo vybraných vlastnostech předmětu,
- SET – změna vlastností předmětu,
- ASSOC – vrátí instance zadaného objektu (tedy s ním asociované),
- CREATE, DELETE – vytvoření nové instance, odstranění instance nebo třídy,
- CALL – spuštění metody zadané třídy WMI.

 Příkaz *v neinteraktivním režimu* používáme takto:

`wmic bios list` vypíše se informace o BIOSu (úplná)

`wmic os list brief` stručná informace o operačním systému

`wmic os list full` úplná informace o operačním systému

`wmic memorychip get` bude zobrazena tabulka s informacemi o paměťových čipech

`wmic memorychip get capacity,devicelocator,name,partnumber` vypíše vybrané informace o paměťových čipech (oproti předchozímu příkazu jen některé sloupce)

`wmic diskdrive get model,size,interfacetype,mediatype` příkaz zobrazí seznam všech paměťových zařízení (i výměnných), a to vlastnosti z výčtu oddělené čárkou

`wmic /output:D:\procinfo.txt cpu get` informace o procesoru (v tabulce) budou uloženy do zadaného souboru

`wmic cpu get > D:\procinfo.txt` totéž

`wmic service where state="running" get caption, name` vypíše seznam běžících služeb (pouze ty vlastnosti, které byly specifikovány), všimněte si syntaxe podobné SQL (pracujeme s databází)

`wmic process call create notepad.exe` spustí zadaný proces (zavolá proceduru create třídy `process` pro vytvoření procesu)

`wmic /node:počítač process call create notepad.exe` totéž, ale na jiném počítači (to příkaz `start` neumí), název počítače se zadává bez úvodních opačných lomítek

`wmic os call reboot` restart systému

`wmic /node:počítač os call shutdown` vzdálené vypnutí systému

`wmic service "spooler" call startservice` spustí zadanou službu (Zařazování tisku)

`wmic /node:ucetni process where name="explorer.exe" call terminate`
zadaný proces bude ukončen, a to na uvedeném počítači (vypadá to, že účetnímu bude ukončeno grafické prostředí, ale tento proces se obvykle po ukončení znovu spustí, tedy jde vlastně o restart grafického prostředí)

`wmic /node:ucetni /user:dadmin process where name="explorer.exe" call terminate` jako předchozí, ale v příkazu na zadaném počítači vystupujeme s oprávněními zadaného uživatele

Vidíme, že můžeme používat i dotazy filtrované podle vlastností (where) podobně jako v SQL. Pokud se jedná o řetězcovou hodnotu, musíme ji uzavřít do uvozovek (například `name="explorer.exe"`), ale čísla nebo hodnoty `true`/`false` do uvozovek neuzavíráme.

 **Příklad**

Ukážeme si práci v interaktivním módu.

`wmic` spustíme interaktivní režim, prompt je `wmic:root\cli>`

`os /?` dotážeme se, co lze použít na předmět `os` (operační systém)

`os list /?` chceme upřesnit parametry pro sloveso `list`

`os list brief` získáme tabulku s několika základními informacemi

`os list full` získáme seznam (už ne tabulku) s dvojicemi vlastnost=hodnota

`os list free` tabulka s hodnotami volného místa (ve fyzické paměti, v stránkovacích souborech, ve virtuální paměti)

`/output:D:\procinfo.txt cpu get` do souboru se uloží hodně široká tabulka vlastností procesoru

`/output:D:\procinfo.txt cpu list full` totéž, ale místo široké tabulky máme v souboru seznam položek vlastnost=hodnota

`cpu get /?` zeptáme se, co se dá zjistit o procesoru

`cpu get NumberOfLogicalProcessors` získáme údaj o počtu logických procesorů (počet jader nebo jeho dvojnásobek, pokud procesor podporuje hyperthreading)

`cpu get AddressWidth,Caption,CurrentClockSpeed,DataWidth,Description,ExtClock` vypíše se tabulka s vybranými sloupci

`process get` vypíše se seznam běžících procesů, u každého je příkaz, kterým byl spuštěn (všimněte si, že v seznamu je i `wmiprvse.exe`, který zajišťuje vyhodnocování dotazů na WMI)

`process list brief` získáme tabulku procesů s některými informacemi

`process where ThreadCount>8 list brief` podobný výstup, ale vypíšou se pouze procesy, které mají více než 8 vláken

`service get name,state,serviceType` tabulka s názvy, stavy a typy služeb

`service where desktopInteract=true get name,state` takto získáme seznam interaktivních služeb (všimněte si, že u neběžících služeb je PID=0)

`service where (desktopInteract=true and startMode="auto") get name,processid` výběrová kritéria můžeme kombinovat (jako v SQL), ale v tom případě je uzavřeme do závorky

`service where desktopinteract=true get name,processid,status /every:5` dotaz bude automaticky opakován v intervalu 5 sekund (stisknutím některé klávesy opakování přerušíme)

`quit` ukončíme interaktivní režim (funguje také příkaz `exit`)

Mechanismus WMI se hodně používá právě ve skriptu, a to typicky v jazyce VBScript, JScript, Perl nebo PowerShell (je třeba mít nainstalován interpret daného jazyka, což u druhého a zejména třetího uvedeného musíme zajistit sami).

Při psaní WMI skriptu bychom měli především zajistit příslušná oprávnění, zvláště když je skript spouštěn na vzdáleném počítači v LAN. Jde o oprávnění v modelu DCOM na daném počítači (WMI je totiž postaveno na COM objektech). Problém lze zjednodušit spouštěním pod některým doménovým správcovským účtem, pak není třeba řešit oprávnění zvlášť na každém počítači.

Další informace:

S mnoha WMI skripty se můžeme seznámit například ve zdroji [8] ze seznamu literatury o automatizaci správy a skriptování. Další informace o WMI (včetně WMI skriptů) najdeme například na

- http://64.4.11.252/en-us/library/aa561208%28BTS.10%29.aspx
- http://www.computerperformance.co.uk/vbscript/wmi.htm

- http://etutorials.org/Server+Administration/Active+directory/Part+III+Scripting+Active+Directory+with+ADSI+ADO+and+WMI/Chapter+26.+Scripting+with+WMI/
- http://flylib.com/books/en/2.679.1.8/1/

Úkol

Pokud máte dostatečná přístupová oprávnění, vyzkoušejte si alespoň „vypisovací“ příkazy z výše uvedeného příkladu.

A.6 Firewall ve Windows

Firewall ve Windows XP je pouze jednosměrný. Ve verzi Vista a Server 2008 se objevil obousměrný firewall, ale filtrování odchozích paketů je v desktopové verzi standardně vypnuto (dá se ale zapnout), tedy funguje jako jednosměrný. Ve vyšších verzích už firewall pracuje tak, jak má, je obousměrný.

Příklad

Prohlédneme si konfiguraci firewallu. K firewallu budeme přistupovat ve Windows (zde verze XP SP2, ve vyšších je to obdobné), a to přes *NetShell*.

`netsh` spustíme NetShell

`firewall` přesun do kontextu *firewall*, následující příkazy provádíme v tomto kontextu

`show` nejdřív si ověříme, co vše lze zobrazit

`show config` zobrazíme konfiguraci firewallu (je celkem obsáhlá)

`show state` zobrazí se momentální stav firewallu

`show opmode` zobrazí se provozní (operační) režim (porovnejte, co se rozumí stavem z předchozího příkazu a provozním režimem z tohoto příkazu)

`set ?` tímto příkazem zjistíme, co vše lze nastavovat příkazem `set` (tj. daná vlastnost už je definována, ale my ji můžeme změnit)

`set opmode enable` nastavíme provozní mód na dostupný

`show allowedprogram` zobrazí aplikace, které mají povoleno přistupovat ven do sítě

`add allowedprogram ?` zajímá nás, jak lze do seznamu přidat další aplikaci

`show portopening` zjistíme porty s nastavenou výjimkou (otevřené)

`delete portopening ?` ze seznamu zjistíme, že je v něm port, který by tam neměl být, tedy nás zajímá, jak ho uzavřít

`show service` vypíšou se služby přistupující na síť

`exit` skončíme

Příklad

Od verze Vista výše se s firewallem zachází trochu jinak. Předně se už nerozlišují povolené/zakázané aplikace, protokoly a porty, ale vše jsou pravidla (rules). Tedy zobrazujeme pravidla, přidáváme pravidla, odstraňujeme je.

`advfirewall firewall` přesuneme se do kontextu *advfirewall* a hned v tomtéž příkazu do jeho podkontextu `firewall`

`show` ověříme, co vše lze zobrazit, zjistíme, že použitelné je `show rule`

`show rule ?` požádáme o podrobnosti

`show rule name=all` pokusíme si nechat vypsat všechna nastavená pravidla, ale seznam je příliš dlouhý, pravidla vypadají nějak takto:

```
Název pravidla:                         Windows Live Messenger (UPnP-In)
----------------------------------
Povoleno:                               Ano
Směr:                                   In
Profily:                                Doména,Privátní,Veřejná
Seskupení:                              Windows Live Messenger
LocalIP:                                Any
Vzdálená IP adresa:                     LocalSubnet
Protokol:                               TCP
Místní port:                            2869
Vzdálený port:                          Any
Funkce Edge traversal:                  No
Akce:                                   Allow
```

Podobně vypadají i další pravidla. Pravidlo je povoleno, směr je „In“, tedy se jedná pravidlo pro příchozí provoz (pro odchozí by bylo „Out“), jsou stanoveny síťové profily, pro které pravidlo platí, pravidla jsou seskupována (group) pro snazší správu, dále jsou určeny adresy, protokol, port. Pokud nám nestačí konec výpisu (záleží na velikosti vyrovnávací paměti Příkazového řádku, kolik údajů bude viditelných), pak musíme příkaz spustit mimo prostředí NetShellu a výstup směrovat do souboru:

```
netsh advfirewall firewall show rule name=all > d:\vystupfirewall.txt
```

(ale pak je třeba použít editor, který zvládá znakovou sadu Latin2, například PSPad (je třeba nastavit v menu *Zobrazit* ; *Zobrazení znaků MSDOS*)

Bohužel musíme zadat buď „all“ nebo konkrétní název pravidla, dále lze filtrovat – omezovat výpis (ale moc možností nemáme)

`show rule name=all type=dynamic dir=in profile=public` zkonkrétnili jsme dotaz, chceme všechna pravidla, která jsou dynamicky vytvořená, pro příchozí provoz a veřejnou síť

`add rule name="povolit udp 5000" dir=out protocol=udp localport=5000 action=allow` povolili jsme odchozí komunikaci na UDP portu 5000

`add rule name="šifrovat 80" protocol=TCP dir=in localport=80 security=authdynenc action=allow` takto jsme si vynutili šifrování příchozí komunikace na TCP portu 80 (o jaký port se asi tak jedná?)

A.7 Další příkazy

Z dalších užitečných příkazů (s mnohými jsme se seznámili v Operačních systémech):

`net` příkaz pro základní správu lokální sítě (sdílení zdrojů v LAN, případně nastavení NTP serveru) a další (včetně správy uživatelů, skupin, zásad účtů apod.)

`ftp` FTP klient pro přenášení souborů po síti, ve Windows také najdeme jednoduchou variantu TFTP

`ipseccmd` konfigurace protokolu IPSec, na klientech nemusí být dostupný

`openfiles` slouží ke správě otevřených souborů, můžeme si nechat vypsat soubory otevřené na tomto nebo jiném počítači v síti, případně otevřené uzavřít, na klientské stanici lze pro podobné účely používat také varianty příkazu `net`

Na serverových verzích Windows najdeme mnoho dalších příkazů, které na klientech nejsou instalovány, především jde o příkazy pro správu Active Directory (např. `ntdsutil` nebo `netdom`), či pro správu DNS serveru (např. `dnsutil`) a další. Seznam s komentáři najdeme v jednom z níže uvedených odkazů. U těchto příkazů bychom měli dát pozor především na to, že jejich syntaxe se může v různých verzích systému mírně lišit.

Další informace:

Seznam příkazů pro Příkazový řádek ve Windows (ne všech) najdeme například na

- http://technet.microsoft.com/en-us/library/cc772390%28WS.10%29.aspx
- http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/ntcmds.mspx?mfr=true.
- http://commandwindows.com/

Příloha B

Práce s adresami a sítěmi v Linuxu

V této příloze jsou popsány a demonstrovány příkazy, které se používají v Linuxu při práci se sítěmi. Podobně jako ve Windows, i tuto přílohu můžeme brát jako inspiraci pro procvičování a jako předehru k zatím neexistujícím cvičením z tohoto předmětu.

B.1 Specializované distribuce

Na routerech nižší a střední třídy se běžně setkáváme s některou distribucí *embedded (vestavěného) Linuxu* (často to ani není znát, pokud při správě využíváme webové rozhraní), protože tento systém je velmi dobře vybaven nástroji pro práci v síti. Obecně – embedded systém je systém je určen k oživení zařízení, která nejsou běžnými „univerzálními“ počítači, například zmíněných síťových zařízení, ale také průmyslových strojů, spotřební elektroniky, „chytrých“ zařízení, atd.

Podobný router či bránu nebo switch s Linuxem si můžeme vyrobit sami. Lze využít starší počítač (ovšem musíme počítat s poměrně vysokou spotřebou a hlučností a potřebujeme vhodnou síťovou kartu s více rozhraními), nebo můžeme pořídit malý počítač průmyslového typu nebo přímo zařízení určené k práci v síti.

Zbývá zvolit vhodnou distribuci Linuxu. Použitelná je vpodstatě jakákoliv (až na ty typicky multimediální), typicky Slackware nebo Debian (Debian je výhodný pro svou univerzálnost co se týče hardwarových platforem), ale můžeme si stáhnout distribuci specializovanou na využití na síťovém zařízení. Sice budeme zřejmě „ochuzeni“ o rozbujelé grafické rozhraní, ale zato budeme mít méně práce s optimalizací distribuce pro naše využití, distribuce bude mít menší nároky na zdroje (především paměť, většinou je spodní hranice 12 nebo 16 MB), instalace bývá někdy možná i přes USB (například IPCop) a pravděpodobněji bude podporovat hardwarové platformy, které se na specializovaných zařízeních používají (záleží na tom, jaký máme hardware, i podle toho vybíráme operační systém).

Příklady vhodných distribucí:

- *IPCop*[1] je předpřipraven pro instalaci na firewallu a internetové bráně (podporuje i VLAN), lze ho spravovat přes webové rozhraní nebo konzolu. Nabízí poměrně dost služeb – DHCP klient a server, NTP klient a server, Dynamic DNS, IDS (program Snort), SSH server, HTTP/FTP proxy (program Squid), stavový firewall, IPSec VPN, atd.

[1] http://www.ipcop.org/

- *SEntry Firewall CD*[2] je zajímavý tím, že se spouští z bootovacího CD (tudíž je použitelný především na zařízeních vybavených optickou mechanikou, třeba starších počítačích), „proměnlivou“ konfiguraci zapisujeme na výměnné médium (třeba na disketu, když použijeme starší počítač, disketa má výhodu velmi snadné a rychlé ochrany proti zápisu). Může sloužit jako firewall, brána (software ebtables), IDS (Snort), router (Zebra a IPRoute2), server, podporuje OpenVPN, atd.
- *Freesco*[3] (ze slov „Free Cisco“) je volně šiřitelný systém obdobný systémům pro zařízení Cisco, Nortel, 3-Com apod. Může plnit roli routeru (až 10 ethernetových portů), firewallu, serveru pro protokoly DHCP, NTP (čas), DNS, FTP, HTTP, SSH, RAS, PPPoE, PPtP, apod., protože jde o Linux, tak samozřejmě i firewall a NAT.
- *Pyramid Linux*[4] je určen především pro bezdrátové přístupové body nebo může plnit roli firewallu, je odvozen z Ubuntu. Typickou hardwarovou platformou je buď x86 nebo jednodesková zařízení.
- *Bering uClibc*[5] je mimořádně malá distribuce určená pro routery s bránou. Její zdrojový kód je natolik upraven směrem k minimalizaci, že není kompatibilní s jinými distribucemi, tedy se při doinstalovávání dalších balíčků musíme spolehnout na repozitář této distribuce (nicméně ten je zcela dostatečně vybaven). Obsahuje celkem běžnou sadu softwaru včetně firewallu, IDS, sledování sítě, podpora bezdrátu, VPN, atd.
- *Voyage Linux*[6] je distribuce odvozená z Debianu, určená pro hardwarové platformy založené na x86 (včetně jednodeskových zařízení). Používá se na firewallech, bezdrátových access pointech, NAS, případně routerech, má také vestavěnou podporu pro VoIP bránu (software Asterisk, ve variantě Voyage ONE).
- *Embedded Debian* (EmDebian)[7] je Debian upravený pro použití na embedded zařízeních, odlehčený (spíše osekaný), schopný běžet na zařízeních s velmi nízkou spotřebou. Předpokládají se úpravy systému křížovou kompilací (cross-compilation).
- *Embedded Gentoo*[8] je derivát Gentoo, opět se počítá s cross-kompilací.
- *QPlus*[9] je embedded distribuce Linuxu pro různé typy zařízení včetně routerů.

Pokud se rozhodneme použít některou běžnou distribuci, měli bychom zajistit co nejlepší zabezpečení našeho zařízení. Velmi dobrou volbou je instalace nástroje *Bastille Linux*,[10] což není distribuce, ale sada skriptů, které formou průvodce zjistí informace od systému a uživatele (s uživatelem konverzuje formou otázek a odpovědí) a pak na pokyn (se souhlasem uživatele) provede potřebná bezpečnostní nastavení. Skripty je možné procházet postupně a případně volby rušit či se k nim vracet.

Další informace:

Velmi důkladný popis přizpůsobení či vylepšení (jakékoliv) distribuce Linuxu pro použití na síťovém

[2] http://www.sentryfirewall.com/
[3] http://www.freesco.org/
[4] http://code.google.com/p/pyramidlinux/
[5] http://leaf.sourceforge.net/bering-uclibc/, http://www.csg.ethz.ch/education/lectures/pps_firewall/SS2006/doc/bering_installation_guide.pdf
[6] http://linux.voyage.hk/
[7] http://www.emdebian.org/
[8] http://www.gentoo.org/proj/en/base/embedded/, http://www.gentoo.org/proj/en/base/embedded/handbook/
[9] http://sourceforge.net/projects/qplus/
[10] http://bastille-linux.sourceforge.net/

zařízení najdeme především ve zdroji [20] ze seznamu literatury. Další informace o embedded Linuxu:

- *Embedded Linux Platform Specification*. LinuxFoundation.org. URL: http://www.linuxfoundation.org/en/ELC
- *Embedded Linux Interfacing*. URL: http://www.embeddedlinuxinterfacing.com/
- *eLinux (Embedded Linux) wiki*. URL: http://elinux.org/Main_Page
- *Embedded Linux*. Felk.cvut.cz. URL: http://support.dce.felk.cvut.cz/osp/cviceni/2/
- Thelin, J. *Introduction: a Typical Embedded System* [online]. Linux Journal, 2009. WWW: http://www.linuxjournal.com/magazine/introduction-typical-embedded-system

Poznámka:

Mnohé z dále uvedených příkazů vyžadují vyšší přístupová oprávnění. Může se také stát, že příkaz bude fungovat i bez vyšších přístupových oprávnění, ale jeho výstup bude jiný (obvykle kratší nebo prázdný), to se týká například příkazů skupiny `ip neighbor`. Proto je lepší alespoň „vypisovací“ příkazy zkoušet s vyššími oprávněními.

B.2 Soubory související se sítěmi

Vybavenost UNIXových systémů nástroji pro práci se sítěmi je obecně vysoká, ovšem v každém UNIXovém systému svým způsobem specifická. Některé firmy distribuující UNIXové systémy přidávají své vlastní typické nástroje.

Odlišnosti jsou také v souborech a adresářích, které se sítěmi souvisejí. Většina konfiguračních skriptů obvykle bývá v adresáři `/etc/sysconfig/network` a jeho podadresářích, v některých linuxových distribucích to je adresář `/etc/rc.d/rc.inet1` (Slackware), `/etc/conf.d/net` (Gentoo) nebo `/etc/network` (Debian).

Příklad

Předpokládejme, že konfigurace sítě je v adresáři `/etc/sysconfig/network-scripts`. Přesuneme se do tohoto adresáře. Měl by tam být soubor `ifcfg-eth0`, ve kterém je uložena konfigurace prvního síťového rozhraní (karty) – IP adresa, maska, způsob získání IP adresy (pokud dynamicky, tak zde IP adresu nenajdeme) apod. Pokud máme více síťových rozhraní, pro každé z nich zde bude jeden takový soubor.

V souboru `/etc/resolv.conf` nastavujeme kromě jiného adresu DNS serveru (tj. když nevíme, kde tuto adresu zadat, podíváme se do tohoto souboru). Jde o řádek (nebo o více řádků, když chceme zadat i záložní DNS servery) ve formátu

`nameserver` *`adresa`*, například

```
nameserver 193.84.192.10
```

Zadaný DNS server bude využíván okamžitě po uložení souboru, není třeba restartovat systém ani žádný proces.

Příklad

Předpokládejme, že z počítače chceme vytvořit router. To není až tak těžké, stačí mít zařízení s více síťovými rozhraními a provést několik nastavení. Jedno z nich je povolení forwardingu (přeposílání). To se provede menší změnou v souboru `/etc/sysctl.conf`:

- najdeme řádek `net.ipv4.ip_forward=0` (je možné, že místo teček budou lomítka, pravidla syntaxe umožňují obojí)
- změníme na `net.ipv4.ip_forward=1`
- aby změna začala platit, musíme buď restartovat systém a nebo jednoduše přimět démona sysctl, aby znovu načetl své konfigurační soubory (včetně toho, který jsme pozměnili a samozřejmě uložili):

  ```
  sysctl -p
  ```

Ovšem zapnutí forwardingu nestačí, je třeba na počítačích v dotyčných sítích (které router propojuje) nastavit toto zařízení jako bránu (například pokud má karta eth0 IP adresu 193.84.200.1, u počítačů v síti, do které je připojena, nastavíme tuto adresu jako adresu brány, IP adresu karty eth1 zase použijeme v druhé síti, samozřejmě vždy tu adresu, která je v dané síti viditelná).

A pak samozřejmě musíme nakonfigurovat firewall a další bezpečnostní mechanismy.

Všimněte si, že není třeba restartovat počítač, třebaže jsme provedli podstatnou změnu v nastavení systému.

V adresáři `/etc` najdeme hodně užitečných souborů a adresářů, jak víme, například `/etc/networks`, `/etc/hosts`, `/etc/ethers`.

Dynamické (momentální) nastavení sítě obvykle najdeme v podadresářích a souborech v systému `/proc`, například `/proc/net/arp`.

Příklad

Do některých souborů v `/proc` lze za určitých okolností zapisovat, například příkazem

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

zapneme forwardování (přeposílání mezi sítěmi) na takovém zařízení, které má dvě síťová rozhraní (třeba dvě síťové karty) a je tedy připojeno do dvou různých sítí. Pokud je v tomto souboru hodnota 0, nepřeposílá se, hodnota 1 znamená, že se pakety mají přeposílat mezi rozhraními.

Toto nastavení však platí jen do restartu počítače (to platí obecně o čemkoliv, co změníme v `/proc`), proto je vhodné tento příkaz uložit také do některého skriptu, který se spouští při startu systému, například do `/etc/rc.d/rc.local`, podle konkrétní distribuce.

Zatím jsme si ukázali dvě možnosti, jak zapnout forwardování (jiný způsob je v předchozím příkladu). Rozdíl mezi nimi je v tom, že zápis do souboru v souborovém systému `proc` platí jen do restartu počítače, zatímco změna v souboru `/etc/sysctl.conf` je trvalá.

Úkol

Projděte si zde probírané soubory na umístěních platných ve vaší distribuci, včetně „síťových“ částí souborového systému `/proc`.

B.3 Starší příkazy pro práci s adresami

Nejdřív se podíváme na příkazy, které se v UNIXových systémech používají pro práci s adresami již desítky let. V novějších distribucích Linuxu se však místo `ifconfig`, `route` a `arp` doporučuje používat nový příkaz `ip`, z důvodu kompatibility a podpory nových technologií.

Příkaz `ifconfig` slouží ke konfiguraci síťových rozhraní (interfaces) jádra (na rozdíl od Windows jsou v Linuxu síťová rozhraní součástí jádra).

`ifconfig` zobrazí informaci o všech síťových rozhraních

`ifconfig eth0` zobrazí informaci o síťovém rozhraní eth0 (to obvykle bývá ethernetová karta)

`ifconfig eth0 down` „shodí“ zařízení eth0 (zneaktivní tuto kartu), provádíme například tehdy, když chceme tomuto rozhraní přiřadit jinou adresu

`ifconfig eth0 up` aktivuje zařízení eth0

`ifconfig eth0 down hw ether 00:00:00:00:00:02` kromě toho, že zneaktivní zařízení eth0, také mu přiřadí MAC adresu; vnitřní příkaz `hw ether` *`adresa`* znamená, že jde o ethernetovou kartu, které přiřazujeme danou adresu (před podobnými změnami je vždy nutné kartu zneaktivnit)

`ifconfig eth0 172.19.124.104 netmask 255.255.255.0 up` nastaví IP adresu a masku podsítě pro eth0, pak je zaktivní (předpokládáme, že předtím byla tato karta neaktivní)

`ifconfig eth0 promisc` zapne promiskuitní mód (tj. karta přijímá/odposlouchává všechny pakety, nejen ty, které jsou jí adresované)

`ifconfig eth0 -promisc` vypne promiskuitní mód

Tento příkaz má další volby, kterými lze například určovat multicast a broadcast adresy, přidělovat zdroje (I/O paměť, IRQ apod.), nastavovat metriky, atd.

Příklad

Příklad `ifconfig` na počítači s jednou síťovou kartou bude vypadat takto:

```
sarka@notebook:~$ ifconfig

eth0      Zapouzdření:Ethernet  HWadr 00:1D:72:31:0A:A0
          inet adr:10.0.0.2  Všesměr:10.0.0.255 Maska:255.255.255.0
          inet6-adr: fe80::21d:72ff:fe31:aa0/64 Rozsah:Linka
          AKTIVOVÁNO VŠESMĚROVÉ_VYSÍLÁNÍ BĚŽÍ MULTICAST  MTU:1500  Metrika:1
          RX packets:561 errors:0 dropped:0 overruns:0 frame:0
          TX packets:556 errors:0 dropped:0 overruns:0 carrier:0
          kolizí:0 délka odchozí fronty:1000
          RX bytes:379302 (370.4 KiB)  TX bytes:100735 (98.3 KiB)
          Přerušení:169

lo        Zapouzdření:Místní smyčka
          inet adr:127.0.0.1 Maska:255.0.0.0
          inet6-adr: ::1/128 Rozsah:Počítač
          AKTIVOVÁNO SMYČKA BĚŽÍ  MTU:16436  Metrika:1
          RX packets:270 errors:0 dropped:0 overruns:0 frame:0
          TX packets:270 errors:0 dropped:0 overruns:0 carrier:0
          kolizí:0 délka odchozí fronty:0
          RX bytes:24914 (24.3 KiB)  TX bytes:24914 (24.3 KiB)
```

Výpis samozřejmě může být celý v angličtině, například místo `délka odchozí fronty` bychom pak našli řetězec `txqueuelen`, nebo místo `AKTIVOVÁNO VŠESMĚROVÉ_VYSÍLÁNÍ BĚŽÍ MULTICAST` by bylo `UP BROADCAST NOTRAILERS RUNNING MULTICAST`.

Srovnejte s výstupem příkazů `ip link show` a `ip addr show dev eth0` v příkladu na straně 308, všechny tyto příkazy byly spuštěny na tomtéž počítači, ale v jiné situaci (tj. například nebudou sedět počty RX a TX bytes/packets).

Pro rychlé odpojení a aktivování síťového rozhraní existují také zkrácené verze příkazu:

`ifdown eth0` deaktivace síťové karty

`ifup eth0` aktivace karty

Pro práci se směrovací tabulkou lze využít příkaz `route`.

`route` zobrazí hlavní směrovací tabulku (k jiným směrovacím tabulkám se tímto příkazem nedostaneme)

`route -n -A inet` první parametr způsobí vypisování číselných adres místo doménových (u některých příkazů také ve formě `-no-dns`, ale většina rozumí pouze `-n`), druhý stanoví, že se mají vypsat pouze IPv4 adresy (pro IPv6 by bylo `-A inet6`)

`route add -net 193.90.100.0 netmask 255.255.255.128 gw 193.90.220.3 dev eth0` přidáme do směrovací tabulky nový záznam (jde o adresu sítě se zadanou maskou, posledním parametrem je brána, na kterou se mají posílat pakety určené pro danou síť)

`route add default gw 193.88.50.10 dev eth0` určíme výchozí bránu pro zadané síťové rozhraní

`route del -net 193.90.100.0 netmask 255.255.255.128 gw 193.90.220.3` odstraníme řádek směrovací tabulky (obvykle stačí zadat jen údaj o IP adrese, kterou chceme odstranit z tabulky)

`route del default` odstraníme směrovací informaci o výchozí bráně, typicky když chceme nastavit jinou

`route -Cn` zobrazí se cache pro směrování, a to s číselnými IP adresami

Místo příkazu `route` se doporučuje používat příkaz `ip route`.

V Linuxu se také používá příkaz `arp`, a to s podobnou syntaxí jako ve Windows:

`arp -a -n` zobrazí ARP tabulku, druhý přepínač znamená, že všechny adresy se zobrazí v číselném tvaru (místo doménových názvů), ale můžeme použít samozřejmě `arp -a` (ovšem když dochází k překladu IP adres na doménové, příkaz pracuje pomaleji)

`arp -n -i eth1` zobrazí ARP tabulku zadaného síťového rozhraní (to následuje za přepínačem `-i`)

`arp -s 123.123.123.123 00:12:34:56:01:08 -i eth1` přidání statického záznamu do ARP tabulky pro kartu eth1

`arp -d 123.123.123.123 -i eth1` odebrání záznamu z ARP tabulky pro kartu eth1

Je možné, že v některých distribucích nebude v nejnovějších verzích některý ze starších příkazů fungovat. Z toho a z dalších důvodů je lepší zvyknout si na příkaz `ip`.

Úkoly

1. Zjistěte, jaká síťová rozhraní na počítači máte. Zjistěte příslušné MAC adresy a přidělené IP adresy. Pokuste se některé síťové zařízení deaktivovat a pak znovu aktivovat (ne loopback!), pokud získáváte IP adresu od DHCP serveru, srovnejte adresy před deaktivací a po aktivaci.

2. Zjistěte, zda máte přidělenu také IPv6 adresu. Pokud ano: doplňte vynechané nulové sekce adresy (stačí jedna nula na sekci) a oddělte prefix od klientské části adresy. Byla klientská část získána autokonfigurací? (tj. srovnejte s MAC adresou, zda odpovídá EUI-64)
3. Zobrazte hlavní směrovací tabulku tak, aby se IP adresy nepřekládaly na doménové. Srovnejte s výpisem obsahujícím doménové adresy. Zjistěte, jakou IP adresu má výchozí brána v hlavní tabulce. Zobrazte směrovací cache (opět s IP adresami místo doménových).
4. Zobrazte ARP tabulku (opět tak, aby se IP adresy nepřekládaly na doménové, srovnejte s výpisem s doménovými adresami).

B.4 Mechanismus `iproute2` (příkaz `ip`) – adresy, síť, směrování

Ve všech novějších distribucích se setkáme s ještě komplexním příkazem `ip`, který je součástí balíčku `iproute2`. Tento příkaz byl zařazen jako náhrada příkazů `ifconfig`, `route` a `arp` (pro úplnost jsme se seznámili i s nimi), v současné době se místo těchto tří doporučuje používat spíše příkaz `ip`. Jeho konfigurační soubory najdeme v `/etc/iproute2`. Má specifické vnitřní příkazy, postupně probereme nejdůležitější varianty.

B.4.1 Konfigurace síťového rozhraní a adres

Nejdřív se podíváme na formu příkazu `ip` pro práci se síťovými rozhraními a IP adresami.

`ip link ...` pracujeme na vrstvě L2 (linkové) ISO/OSI modelu, vpodstatě i L1, tedy pracujeme s MAC adresami a síťovým rozhraním (například můžeme kartu odpojit či připojit)

- `ip link show` zobrazí stav spojení (tj. stav síťového rozhraní), místo `show` je možné všude používat `list` nebo `ls`, na multihomed zařízeních může být výpis velmi dlouhý; na rozdíl od `ifconfig` se výpišou i *ta síťová rozhraní, která* z nějakého důvodu *nelze řádně aktivovat*
- `ip -s link show eth0` zadali jsme název rozhraní (tj. nebudou se vypisovat informace pro všechna, pokud jich je více), a navíc parametr `-s` znamená podrobnější výpis
- `ip link set dev eth0 up` aktivuje rozhraní eth0
- `ip link set dev eth0 down` deaktivuje rozhraní eth0
- `ip link set dev eth0 mtu 1500` změní hodnotu MTU na 1500 oktetů (Maximum transmission unit, maximální velikosti IP paketu, kterou je možné přijmout a odeslat)
- `ip link set dev eth0 address 02:00:00:00:11:22` změníme MAC adresu síťové karty eth0, všimněte si prvního oktetu; pozor, změna MAC nemusí někdy dopadnout dobře, jde o potenciálně nebezpečnou operaci

`ip addr ...` pracujeme na vrstvě L3 (síťové), tedy s IP adresami (místo `addr` lze použít `address` nebo `a`)

- `ip addr show` takto zjistíme IP adresu a související informace, případně můžeme přidat i název karty, která nás zajímá, jedna karta může mít i více než jednu IPv6 adresu (jednu primární a ostatní sekundární)
- `ip address show` totéž, také funguje `ip a show` nebo `ip a ls`
- `ip addr show dev eth0 primary` zjistíme primární adresu zařízení eth0

`ip addr add 193.90.220.42/25 brd + dev eth0` nastavíme IP adresu pro rozhraní eth0,[11] broadcast adresu necháme nastavit na výchozí (pokud chceme určit jinou než výchozí, místo symbolu `+` napíšeme novou broadcast adresu), všimněte si, že místo masky zadáváme délku prefixu hned u adresy

`ip addr del 193.90.220.42/25 brd + dev eth0` odebereme kartě eth0 zadanou adresu; pozor – pokud se jedná o primární adresu, odeberou se i všechny sekundární

`ip addr flush dev eth0` pročištění (flush, odstranění) všech adres na daném rozhraní, zůstane pouze IPv6 adresa získaná autokonfigurací, vnitřnímu příkazu `flush` se spíše vyhýbáme (můžeme omylem odstranit adresy, které by měly zůstat)

`ip -f inet6 addr flush dynamic` před vnitřním příkazem je použit parametr určující, že se bude pracovat pouze s IPv6 adresami (místo `-f inet6` je možné napsat zkráceně `-6`), odstraní se adresy IPv6 získané dynamicky (tj. kromě adresy získané autokonfigurací), pokud chceme provést totéž pro IPv4 adresy, napíšeme

`ip -f inet addr flush dynamic` nebo

`ip -4 addr flush dynamic` nebo

`ip -4 a flush dynamic`

Příklad

Podíváme se na výstupy několika příkazů:

```
sarka@notebook:~$ ip link show

1: lo: <LOOPBACK,UP,10000> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:1d:72:31:0a:a0 brd ff:ff:ff:ff:ff:ff
3: sit0: <NOARP> mtu 1480 qdisc noop
    link/sit 0.0.0.0 brd 0.0.0.0

sarka@notebook:~$ ip -s link show

1: lo: <LOOPBACK,UP,10000> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    RX: bytes  packets  errors  dropped overrun mcast
    24914      270      0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    24914      270      0       0       0       0
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:1d:72:31:0a:a0 brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    379366     562      0       0       0       9
    TX: bytes  packets  errors  dropped carrier collsns
    100799     557      0       0       0       0
3: sit0: <NOARP> mtu 1480 qdisc noop
    link/sit 0.0.0.0 brd 0.0.0.0
    RX: bytes  packets  errors  dropped overrun mcast
    0          0        0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
    0          0        0       0       0       0
```

[11] Jedno síťové rozhraní může mít více IPv6 adres. Jestliže již má IPv6 adresu přiřazenu, další bude sekundární. Pokud chceme jednomu síťovému rozhraní přiřadit více IPv6 adres, měli bychom každé z těchto adres přiřadit label – viz `man ip`, především z důvodu jejich rozlišení v příkazech, například `ifconfig` s tím mívá problémy. Taktéž lze IP adresy jednoho rozhraní rozlišit pomocí aliasů, což je rozlišení na nižší úrovni než v případě labelů, například příkazem `ifconfig eth1:0 193.84.190.99 netmask 255.255.128.0 up` vytvoříme alias eth1:0 k rozhraní eth1 s vlastní IP adresou, který lze také deaktivovat bez ovlivnění rozhraní eth1.

Všimněte si, že v ostrých závorkách za názvem rozhraní jsou příznaky rozhraní (například zda jde o loopback, jestli je aktivní, povolení broadcastu, apod.).

```
sarka@notebook:~$ ip addr show

1: lo: <LOOPBACK,UP,10000> mtu 16436 qdisc noqueue
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 00:1d:72:31:0a:a0 brd ff:ff:ff:ff:ff:ff
    inet 10.0.0.2/24 brd 10.0.0.255 scope global eth0
    inet6 fe80::21d:72ff:fe31:aa0/64 scope link
       valid_lft forever preferred_lft forever
3: sit0: <NOARP> mtu 1480 qdisc noop
    link/sit 0.0.0.0 brd 0.0.0.0
```

Kdyby nás zajímalo pouze zařízení eth0, napsali bychom

```
ip addr show dev eth0
```

Úkoly

1. Vypište údaje o svém síťovém rozhraní, a pak je vypište včetně statistiky (podrobnější informace). Srovnejte oba výpisy. Určete, kolik máte síťových rozhraní (na koncové stanici to odpovídá počtu síťových karet plus případných virtuálních rozhraní), jakou máte velikost MTU, jaké máte MAC adresy, příp. jaká je broadcast MAC adresa, jaký je provoz na vstupu/výstupu rozhraní.
2. Zjistěte IP adresy (IPv4 i IPv6) na svých síťových rozhraních. Máte jen primární adresu, nebo i nějaké sekundární?
3. Pokud je na rozhraní definováno více adres, jak zobrazíte pouze tu primární? Co se stane, když síťovému rozhraní s více IP adresami odebereme primární/sekundární adresu?

B.4.2 Směrování a filtrování

Příkaz `ip` se používá při práci se směrovacími tabulkami a při definování pravidel pro filtrování obdobně jako u firewallu.

V Linuxu neexistuje pouze jedna směrovací tabulka. Ve výchozím nastavení máme vždy alespoň hlavní (*main*) směrovací tabulku, která je zároveň výchozí (*default*), a *local* – lokální tabulku (v lokální směrovací tabulce najdeme především loopback (místní cesty) a směrování broadcastů a multicastů), další tabulky si můžeme dle potřeb vytvořit. Každá tabulka je identifikována svým jménem a číslem, v příkazech můžeme používat cokoliv z toho. Co se týče čísel směrovacích tabulek, tak u nově vytvořených můžeme použít čísla z intervalu 1 až 252, čísla předdefinovaných tabulek vidíme ve výpisu níže.

Zatímco příkaz `route`, se kterým jsme se již dříve seznámili, dovoluje přistupovat pouze k hlavní tabulce, příkaz `ip` umožňuje pracovat s jakoukoliv tabulkou.

Seznam směrovacích tabulek je uložen v souboru `/etc/iproute2/rt_tables`. Výchozím obsahem je

```
255     local
254     main
253     default
0       unspec
```

(plus řádky s komentáři, komentářový symbol je `#`).

 Novou směrovací tabulku vytvoříme jednoduše přidáním nového řádku do tohoto souboru (další informace najdeme u příkazu `ip rule` od strany 311). Obvykle soubor needitujeme přímo, ale použijeme přesměrování s přidáním na konec:

`echo` *`číslo název`* `» /etc/iproute2/rt_tables`

například

```
echo 15 novatab » /etc/iproute2/rt_tables
```

Všimněte si, že při přesměrování jsme použili dvojitou „šipku“, protože nechceme přepsat původní obsah (na to pozor!), ale přidat nový záznam na konec souboru.

Příkazy:

 `ip route ...` pracujeme se směrovací tabulkou, taktéž na vrstvě L3 (ale nad protokolem IP)

- `ip route show` zobrazí hlavní směrovací tabulku (pozor, ne všechny)
- `ip route show table local` zobrazí lokální směrovací tabulku
- `ip route show table 12` zobrazí směrovací tabulku číslo 12 (na tabulku se odkazujeme jménem nebo číslem)
- `ip route show cache` zobrazí vyrovnávací paměť směrování; velmi dlouhý výstup, je dobré přidat ještě IP adresu, která nás zajímá, a nebo výstup nějak filtrovat (třeba `grep`em)
- `ip -s route show cache 193.90.102.36` zadali jsme IP adresu, která nás jako cíl zajímá, navíc přepínačem `-s` vyžadujeme podrobnější výstup (tj. vlastně statistiku směrování na danou adresu)
- `ip route add default via 193.90.220.1` nastavení výchozí brány pro všechny cíle, které ve směrovací tabulce nejsou přímo uvedeny
- `ip route add 193.90.100.0/25 via 193.90.220.3` přidání nového (statického) řádku do směrovací tabulky (zadáváme adresu sítě s prefixem a dále adresu zařízení, přes které se k první zadané adrese dá dostat)
- `ip route add 193.90.100.0/25 via 193.90.220.3 table administrativa` záznam jsme přidali do zadané směrovací tabulky `administrativa`, nikoliv do hlavní směrovací tabulky
- `ip route delete 193.90.100.0/25` odstranění řádku tabulky (příp. lze opět zadat tabulku)
- `ip route add prohibit 193.221.88.0/28` zakážeme směrování na zadanou adresu, tj. defacto tuto adresu znepřístupníme ze zařízení s touto tabulkou (používají například zaměstnavatelé k zamezení přístupu z daného počítače/podsítě k určitým oblastem), zadaná podsíť či uzel je ohlášen(a) jako „no route to host“ (ICMP zpráva)
- `ip route add prohibit 193.221.88.0/28 from 193.90.102.29` podobně jako předchozí, ale zablokovali jsme přístup pouze z jednoho (zadaného) uzlu, z jiných uzlů bude směrování fungovat (klíčové slovo `from` použijeme typicky na směrovači s Linuxem, na koncové stanici nemá moc smysl)
- `ip route add blackhole 69.63.189.11` pokud někdo z lokální sítě odešle paket na tuto adresu (mimochodem, je to jedna z IP adres serverů Facebooku), paket bude zahozen a dotyčný nebude informován

`ip route add unreachable 69.63.189.11` podobně jako předchozí, ale odesílatel je informován o tom, že paket nedorazil, ICMP zprávou „ICMP unreachable“;

router tedy může paket „záměrně“ zahodit třemi různými způsoby (prohibit, blackhole, unreachable).

`ip route add nat 192.205.120.56 via 193.90.102.29` stanoví NAT překlad pro příchozí pakety (přicházející do LAN): když tento router dostane paket s první („virtuální“) adresou jako cílovou, přepíše ji na druhou uvedenou (skutečnou v místní síti) a pošle dál

Příklad

Ukážeme si rozdíl mezi hlavní a lokální směrovací tabulkou na klientské stanici:

```
sarka@notebook:~$ ip route show

193.84.195.0/25 dev eth0  proto kernel  scope link  src 193.84.195.30
default via 193.84.195.1 dev eth0

sarka@notebook:~$ ip route show table local

broadcast 127.255.255.255 dev lo  proto kernel  scope link  src 127.0.0.1
broadcast 193.84.195.0 dev eth0  proto kernel  scope link  src 193.84.195.30
local 193.84.195.30 dev eth0  proto kernel  scope host  src 193.84.195.30
broadcast 193.84.195.127 dev eth0  proto kernel  scope link  src 193.84.195.30
broadcast 127.0.0.0 dev lo  proto kernel  scope link  src 127.0.0.1
local 127.0.0.1 dev lo  proto kernel  scope host  src 127.0.0.1
local 127.0.0.0/8 dev lo  proto kernel  scope host  src 127.0.0.1
```

Řádek začínající `default` v hlavní tabulce řádku určuje bránu lokální sítě.

Směrování lze spojit s filtrováním či podrobnějším řízením směrování podle zadaných *pravidel* (politik), čemuž říkáme *Advanced Routing* (pokročilé směrování). Zatímco příkaz `ip route` pracuje pouze s adresami, pomocí `ip rule` lze směrování ovlivnit také obsahem jiných polí IP paketu (směrování podle zásad), takto lze implementovat i mechanismus NAT.

`ip rule ...` pracujeme s pravidly (politikami, zásadami) pro směrování

`ip rule show` výpis pravidel (vlastně zásad) pro směrování, jsou tam minimálně tyto řádky:

```
0:      from all lookup local
32766:  from all lookup main
32767:  from all lookup default
```

(případně místo `default` může být přímo číslo výchozí směrovací tabulky 253), výchozí hodnoty pouze odkazují na lokální, hlavní a výchozí směrovací tabulku, číslo na začátku je tzv. priorita pravidla, toto číslo by mělo být jedinečné pro každé pravidlo (čím menší číslo, tím vyšší priorita)

`echo 12 pomocnatab » /etc/iproute2/rt_tables` trochu jsme odbočili, tímto příkazem jsme vytvořili směrovací tabulku s číslem 12 a názvem `pomocnatab`

`ip rule add from 195.200.94.0/24 table 12 priority 354` všechny pakety s uvedenou adresou jako zdrojovou (resp. adresou ze zadané sítě) budou směrovány podle tabulky 12 vytvořené v předchozím příkazu, hodnota pro prioritu pravidla by měla být u pravidel unikátní

(pokud zvolíme již existující prioritu, jádro přidělí nejbližší menší volné číslo; když prioritu neuvedeme vůbec, jádro nějakou zvolí samo), čím menší číslo, tím vyšší priorita; pokud teď zadáme `ip rule show`, bude výstup následující (srovnejte s prvním výpisem pravidel):

```
0:      from all lookup local
354:    from 195.200.94.0/24 lookup pomocnatab
32766:  from all lookup main
32767:  from all lookup 253
```

`ip rule add from 195.80.94.0/24 to 194.200.24.0/24` určíme, že pakety ze sítě s první zadanou adresou do sítě s druhou zadanou adresou budou směrovány podle hlavní směrovací tabulky (případně můžeme zadat, která tabulka se má použít), prioritu pravidla zvolí jádro (všimněte si, které číslo jádro zvolilo: o něco nižší číslo než poslední uživatelské), ve výstupu příkazu `ip rule show` přibude tento řádek:

```
353:    from 195.200.94.0/24 to 194.200.24.0/24 lookup main
```

`ip rule add from 193.90.102.29 nat 192.205.120.56` NAT pravidlo pro odchozí pakety (tj. odcházející ze sítě): pokud má odejít ven ze sítě paket se zdrojovou adresou první uvedenou (skutečnou), bude zdrojová adresa nastavena na druhou uvedenou („virtuální“; všimněte si, jaký je vztah mezi adresami v podobném výše uvedeném příkazu `ip route`, musíme mít provedeny oba příkazy, aby to fungovalo)

`ip rule add iif lo table 10 priority 560` takto můžeme oddělit směrování provozu generovaného přímo na tomto zařízení (lo, loopback, poněkud netypické použití loopbacku) od směrování přeposílaného (forwarding) provozu, tedy provoz generovaný procesy na tomto zařízení bude směrován podle tabulky 10 (případně můžeme takto oddělit provoz směrovaný z konkrétního rozhraní, třeba eth2 coby třetí ethernetové síťové karty), do seznamu pravidel přibývá nový řádek (předpokládejme, že tabulka 10 se nazývá `dalsitabulka`):

```
560:    from all iif lo lookup dalsitabulka
```

`ip rule del priority 560` odstraníme pravidlo se zadanou prioritou (připomeňme si, že priorita je vlastně pořadové číslo v tabulce pravidel, pravidlo je takto jednoznačně identifikováno)

Příklad

Podíváme se na možnost vytvoření jednoduchého one-to-one stateless (bezstavového) NAT, tedy překládáme mezi jednou vnitřní a jednou vnější IP adresou. Toho lze docílit buď příkazem `ip`, nebo mechanismem *NetFilter* (to, jak víme, je modul jádra obvykle používaný jako firewall, jehož obrazem v uživatelském režimu je `iptables`). Pokud použijeme příkaz `ip`, bude náš router odpovídat na ARP dotazy na NAT IP adresu, což může být užitečné (mechanismus *NetFilter* tuto vlastnost nemá).

Předpokládejme, že chceme zpřístupnit stroj s IP adresou 195.159.200.28 ven ze sítě pod adresou 207.189.240.28, může se například jednat o některý server v DMZ:

`ip route add nat 207.189.240.28 via 195.159.200.28` nejdřív zajistíme překlad cílové adresy v příchozím (inbound) provozu, tedy DNAT

`ip rule add nat 207.189.240.28 from 195.159.200.28` zajistíme překlad zdrojové adresy v odchozím provozu, tedy SNAT (Source NAT, viz str. 336)

`ip route flush cache` aby překlad fungoval, musíme vyprázdnit směrovací cache se starými záznamy

`ip route show table all` ve výstupu by měl být kromě jiného řádek

`nat 207.189.240.28 via 195.159.200.28 table local scope host`

(řádků je tam obvykle hodně, výstup můžeme profiltrovat například příkazem `grep`)

`ip rule show` podobně zkontrolujeme seznam pravidel pro směrování, mezi nimi by měl být řádek

`32386: from 195.159.200.28 lookup main map-to 207.189.240.28`

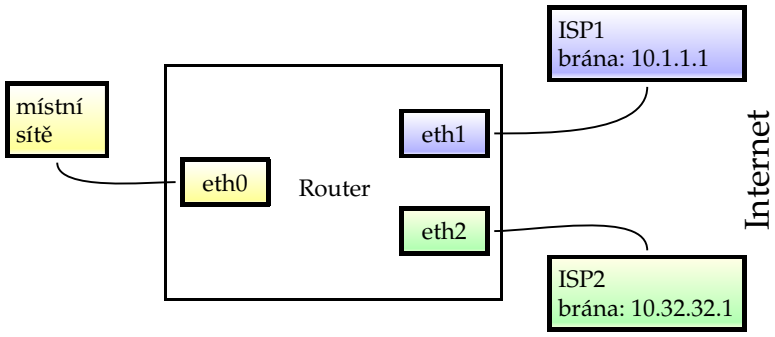
(pravděpodobně s jinou hodnotou priority)

Totéž lze provést s použitím mechanismu *NetFilter* zároveň s určením dalších údajů pro překlad (například čísla TCP portu), ukázku máme na straně 337.

Příklad

Předpokládejme, že máme dva ISP (poskytovatele připojení k Internetu) připojené k témuž routeru s Linuxem. Tento případ není až tak exotický, můžeme se s ním setkat u některých středních/větších společností (respektive, nemusí jít ani o více různých ISP, stačí, když máme více bran).

Budeme chtít vytvořit dvě směrovací tabulky, zvlášť pro každého ISP. Situace je znázorněna na obrázku B.1.



Obrázek B.1: Situace pro více ISP přes jeden router

Nejdřív „vytvoříme“ dvě směrovací tabulky, pro každého ISP zvlášť:

```
echo 10 ISPprvni » /etc/iproute2/rt_tables
echo 20 ISPdruhy » /etc/iproute2/rt_tables
```

Teď se musíme rozhodnout, jak se bude dělit provoz mezi jednotlivé brány (a tedy i mezi jednotlivá rozhraní routeru). Předpokládejme, že provoz související s lokální sítí 10.192.0.0/24 půjde přes první bránu (rozhraní eth1) a bude směrován tabulkou 10, provoz sítě 10.192.128.0/24 půjde přes druhou bránu a bude směrován tabulkou 20, zbytek půjde také přes druhou bránu a bude směrován hlavní směrovací tabulkou.

Nakonfigurujeme směrovací tabulku a pravidla:

`ip route add default via 10.32.32.1 dev eth2` nastavíme výchozí bránu pro pakety ze sítí neuvedených jinde

`ip route add default via 10.1.1.1 dev eth1 table 10` do tabulky 10 (ISPprvni) jsme přidali nový záznam – pro vše, co bude směrováno podle této tabulky, platí výchozí brána 10.1.1.1, ke které se dostaneme přes rozhraní eth1

`ip route add default via 10.32.32.1 dev eth2 table 20` podobně pro další tabulku, určíme bránu a port

`ip rule add from 10.192.0.0/24 table 10` vše, co přijde ze sítě se zadanou adresou, bude směrováno podle tabulky 10

`ip rule add from 10.192.128.0/24 table 20` podobně pro další síť, určujeme směrovací tabulku

Provoz lze směrovat i podle jiných kritérií.

Úkoly

1. Najděte seznam směrovacích tabulek (vypište soubor `/etc/iproute2/rt_tables`). Dále zjistěte, jak jsou nastavena přístupová oprávnění k tomuto souboru:
   - `ls -la /etc/iproute2/rt_tables` vypíše základní oprávnění,
   - `getfacl /etc/iproute2/rt_tables` vypíše seznamy POSIX ACL,
   - `lsattr /etc/iproute2/rt_tables` zobrazí atributy souboru v daném souborovém systému,
   - `getfattr /etc/iproute2/rt_tables` zobrazí rozšířené atributy souboru.
2. Vypište hlavní směrovací tabulku a porovnejte s výpisem pomocí příkazu `route` (je v předchozí sekci). Dále vypište lokální směrovací tabulku, a pokud existují další, také je vypište. Najděte adresu výchozí brány. Jak byste ji změnili?
3. Vypište seznam definovaných politik (zásad, pravidel) pro směrování. Všimněte si, na které směrovací tabulky tyto zásady odkazují.

B.4.3 Objevování sítě

Protokol IPv4 používá pro objevování sítě protokol ARP, u IPv6 to je NDP (Neighbour Discovery Protocol), respektive mechanismus NDis (Network Discovery). Pro práci s „okolím“ máme k dispozici další vnitřní příkaz příkazu `ip`.

Jednotlivé uzly sítě mohou být v některém z následujících stavů (označujeme NUD, Neighbour Unreachability Detection):

- *none* – stav „nevyplněn“
- *noarp* – soused je platný a dosažitelný, nemá být ověřován ARP dotazy, záznam má stanovenu dobu platnosti (po této době je vyřazen z ARP cache)
- *permanent* – podobně jako noarp je platný, dosažitelný a nemá být ověřován ARP dotazy, nemá stanovenu dobu platnosti (tj. platnost neomezená), z tabulky sousedů smí být odstraněn jen administrátorem
- *reachable* – soused je platný a dosažitelný, záznam má stanovenu dobu platnosti (životnost záznamu), po uplynutí této doby bude opětovně dynamicky zjišťován ARP dotazem (tento stav je běžný pro dynamicky zjišťované sousedy)
- *incomplete* – soused je „zjišťován“, proces objevování souseda není ukončen (obvykle znamená, že k dané IP adrese, na kterou se dotazuje některý protokol vyšší vrstvy, neexistuje žádný záznam)

- *stale* – soused je považován za platného, ale jeho dosažitelnost musí být neustále ověřována (po každém použití záznamu tento záznam přechází do stavu *delay*)
- *delay* – je třeba záznam ověřit, byl naplánován ARP dotaz na tuto adresu, pokud tento soused bude generovat provoz, stav se mění zpět na *stale*
- *probe* – sousedovi s příznakem `delay` vypršel limit na odpověď (nebo generování provozu), jádro odešle ARP dotaz
- *failed* – ARP dotaz selhal, záznam není platný

Většina sousedů je ve stavu *reachable*, *stale* nebo *permanent*.

Pokud je soused ve stavu *stale* a je připojen, pak v tomto stavu tráví většinu času (občas přechází do *delay*). Pokud je soused ve stavu *reachable*, obvykle v něm zůstává, ale když ho odpojíme, postupně přechází mezi stavy *reachable–stale–delay–probe–failed*, dotaz na danou IP adresu pak končí výpisem stavu *incomplete*. Příkazy:

`ip neighbour ...` pracujeme s vazbami mezi adresami L2 a L3 (tj. MAC a IP) ve stejné síti (tj. se „sousedy“, odtud klíčové slovo), v případě IPv4 jde o práci s ARP tabulkami, u IPv6 jsou to neighbour tabulky (slovo `neighbour` lze zkracovat na `neighbor`, `neigh` nebo `n`)

`ip neigh show` zobrazí momentální stav ARP nebo NDP cache (tabulky)

`ip neigh show dev eth0` zobrazí seznam sousedů připojených k rozhraní eth0

`ip neigh show to 193.168.200/24` zobrazí se info o sousedech ze zadané podsítě

`ip -s neigh show to 10.0.0.1` podrobnější statistika souseda se zadanou adresou (navíc počet uživatelů záznamu a dále před kolika sekundami byl záznam použit/potvrzen/aktualizován)

`ip neigh show nud permanent` zobrazí všechny sousedy, jejichž stav je „permanent“, obvykle jde o ručně přidané sousedy

`ip neigh add 193.168.200.1 lladdr 00:0a:1b:2c:3d:4e dev eth2 nud permanent` přidali jsme do ARP tabulky statický záznam o sousedovi (zadáváme IP adresu, MAC adresu – Link Layer Addr, rozhraní, přes které je soused dosažitelný, a pak stav)

pokud přidáváme nový záznam, můžeme hodnotu `nud` nastavit na jednu z hodnot `noarp`, `reachable`, `permanent` nebo `stale`, čímž také určíme, zda záznam bude mít omezenou platnost a jestli má být ověřován

`ip neigh del 193.168.200.1 dev eth2` odstranili jsme záznam z tabulky

Příklad

Zatímco mezilehlé síťové zařízení (router, switch) má sousedů poměrně hodně, koncové zařízení připojené přes ethernet (konektor RJ-45) má obvykle jediného souseda – router nebo switch.

Na koncovém zařízení vypadají výpisy následovně:

```
sarka@notebook:/home/sarka# ip neigh show

10.0.0.138 dev eth0 lladdr 00:21:63:e2:0f:98 STALE

sarka@notebook:/home/sarka# ip -s neigh show

10.0.0.138 dev eth0 lladdr 00:21:63:e2:0f:98 ref 12 used 172/172/152 STALE
```

Druhý výpis obsahuje podrobnější údaje o sousedovi (což je router, jehož uvnitř viditelná IP adresa je 10.0.0.138). Na řádku postupně vidíme IP adresu, rozhraní, přes které jsme k sousedovi připojeni, dále počet odkazů na toto připojení a tři časové údaje (počet sekund od chvíle, kdy bylo připojení naposledy použito/potvrzeno/aktualizováno).

Úkoly

1. Zobrazte seznam svých sousedů (ARP/NDP cache). Pokud máte více aktivních síťových rozhraní, pak zvlášť pro několik rozhraní. Vyzkoušejte zobrazení se statistikou (podrobnějšími informacemi) o daném propojení.
2. Přidejte do ARP cache „virtuálního“ souseda, jehož MAC adresa je 03:00:01:a5:b6:c7, IP adresa 10.0.0.12, je připojen na rozhraní eth0 (i když v reálu není), stav spojení *permanent*. Pak vypište seznam sousedů a zkontrolujte, zda je v seznamu přidaný záznam. Dále vypište seznam sousedů, jejichž připojení je ve stavu *permanent*. Potom nový záznam odstraňte a opět zkontrolujte, jestli záznam opravdu zmizel.
3. Zopakujte si celý postup aktivace síťového rozhraní (zkompletujte příkazy):
   - pokud je rozhraní aktivní, je třeba ho deaktivovat
   - nastavíme IP adresu (včetně délky prefixu)
   - nastavíme bránu
   - nastavíme adresu DNS serveru
   - aktivujeme rozhraní

B.4.4 Tunely

Mechanismus `iproute2` slouží také k vytváření IP/IP tunelů. Lze použít pro vytvoření VPN tunelu, zapouzdření IPv6 do IPv4 na cestě bez podpory IPv6, vytvoření mostu mezi více síťovými rozhraními na jednom zařízení patřícími do různých sítí, vytvoření vzdáleného mostu, apod.

Je možné pracovat s několika různými typy tunelů, ale předně musíme mít v jádře načten příslušný modul pro daný typ tunelu. Nejběžnější jsou

- *IP-IP tunely* (modul `ipip`) – dokážou zapouzdřit jen unicast IP pakety (tunely point-to-point),
- *GRE tunely* (modul `ip_gre`) – zapouzdřují také jiné typy paketů, a to spojení unicast i multicast, jsou poměrně hodně používány,
- *SIT tunely* (Simple Internet Transition, modul `ipv6`) – k propojení IPv6 sítí přes IPv4 sítě.

V příkazech pro práci s tunely se typ tunelu projevuje v povinném parametru `mode` (tedy mód tunelu), určuje způsob, jakým se má s transportovaným paketem zacházet (především jak a do čeho se má zapouzdřit). Používáme následující příkaz:

`ip tunnel ...` zabezpečený přenos, tunelování

`ip tunnel show mujtunel` zobrazí informaci o vytvořeném tunelu s názvem `mujtunel`, zobrazí se obvykle typ (mód) tunelu, vzdálená a místní IP adresa (konce tunelu), pak další zadané (nepovinné) vlastnosti jako název síťového rozhraní, ze kterého tunel vede, hodnota TTL pro pakety jdoucí do tunelu, hodnota TOS, apod. (vpodstatě se zobrazuje to, co se zadalo při vytvoření tunelu)

`ip -s tunnel show mujtunel` kromě výše uvedených informací se zobrazí také statistika tunelu podobná výstupu příkazu `ip -s link show` (jde vlastně o stejný typ informace, jen se místo síťového rozhraní týká tunelu), viz výstup na straně 308

`ip tunnel add mujtunel mode ipip local 194.50.20.42 remote 195.84.152.140 ttl 32` vytvořili jsme nový tunel typu IP-IP se zadanou místní a vzdálenou adresou a hodnotou TTL

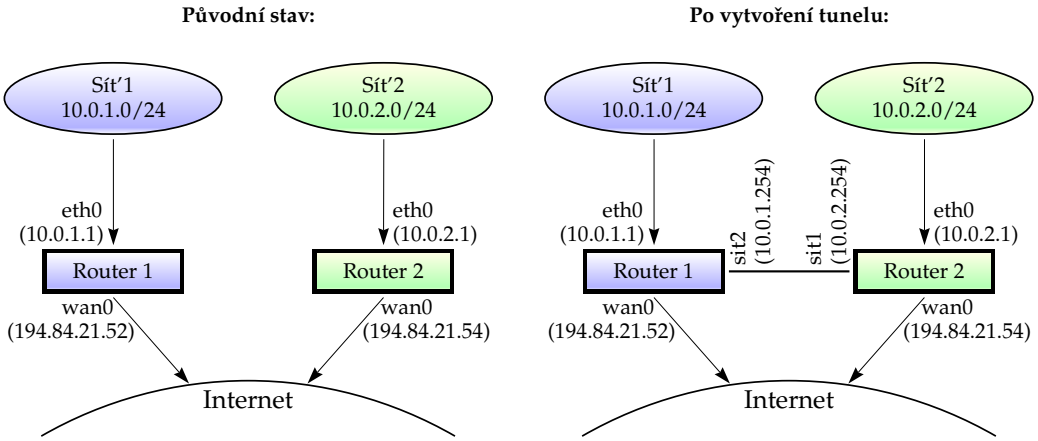
`ip tunnel del mujtunel` zrušili jsme tunel

`ip tunnel change mujtunel remote 195.35.84.15` změna nastavení tunelu (změnili jsme vzdálenou adresu)

Abychom mohli tyto příkazy používat, musíme mít předně v jádře načten příslušný modul.

Příklad

Ukážeme si vytvoření GRE tunelu, který je zřejmě nejčastěji používán (protože dokáže zabalit cokoliv, zvládá i multicast a je všeobecně podporován prakticky v čemkoliv). Předpokládejme, že chceme propojit tunelem dvě vzdálené sítě (patřící stejné firmě), ve kterých jsou soukromé IP adresy. Rozsahy adres v těchto sítích by se neměly překrývat, aby nedocházelo k problémům při směrování.



Obrázek B.2: Vytvoření GRE tunelu

Situaci včetně adres máme znázorněnu na obrázku B.2 vlevo. Podle prefixů poznáme, že rozsahy adres v sítích se nepřekrývají. U routerů, přes které se sítě připojují do Internetu, vidíme také adresu viditelnou uvnitř (je z rozsahu adres vnitřní sítě) a adresu viditelnou vně (tu máme přidělenou ISP providerem). Aby bylo možné vybudovat funkční tunel, musíme všechny tyto adresy znát (zadávají se do příkazů).

Na routeru *Router1* zadáme tyto příkazy:

`modeprobe ip_gre` načteme do jádra modul pro podporu GRE tunelů

`ip tunnel add sit2 mode gre local 194.84.21.52 remote 194.84.21.54 ttl 255` vytvoříme tunel, všimněte si vyšší hodnoty TTL (nevíme, přes kolik routerů tunel vede, tedy je lepší použít vyšší hodnotu)

`ip link set sit2 up` aktivujeme nové virtuální rozhraní (tedy náš tunel)

`ip addr add 10.0.1.254 dev sit2` přidělíme našemu konci tunelu IP adresu, měla by být viditelná v naší síti a neměla by být použita pro jiné zařízení (to si musíme pohlídat, případně upravit rozsahy adres)

`ip route add 10.0.2.0/24 dev sit2` do směrovací tabulky přidáme pravidlo pro směrování do sítě za tunelem, bližší konec tunelu bude fungovat jako brána

Obdobně postupujeme na routeru *Router2*:

`modeprobe ip_gre` načteme do jádra modul pro podporu GRE tunelů

`ip tunnel add sit1 mode gre local 194.84.21.54 remote 194.84.21.52 ttl 255` vytvoříme tunel

`ip link set sit1 up` aktivujeme nové virtuální rozhraní (tedy náš tunel)

`ip addr add 10.0.2.254 dev sit1` přidělíme našemu konci tunelu IP adresu

`ip route add 10.0.1.0/24 dev sit1` do směrovací tabulky přidáme pravidlo pro směrování do sítě za tunelem

Na obrázku B.2 vpravo je znázorněna situace po vytvoření tunelu. Pak bychom měli vyzkoušet, zda jsou pakety doručovány správně, například na prvním routeru vyzkoušíme `ping` na druhý konec tunelu:
`ping 10.0.2.154`

Pokud nefunguje, je možné, že pakety byly zahozeny některým firewallem. Pokud jsou povoleny ICMP pakety a nemáme žádné restrikce na námi vytvořené virtuální rozhraní, měli bychom ještě zkontrolovat, zda mohou procházet GRE pakety (protokol číslo 47), například v chainu FORWARD bychom protokol 47 povolili takto (podrobnosti najdeme v samostatné sekci o firewallu v Linuxu):
`iptables -A FORWARD -p 47 -j ACCEPT`
Dále může být problém v zakázání čísla portu používaného tunelem, tedy pokud nepomůže povolení protokolu 47, můžeme vyzkoušet
`iptables -A FORWARD -p tcp -dport 1723 -j ACCEPT`
Případně v jiných tabulkách nebo s dalšími parametry, podle momentálního nastavení firewallu.

Měli bychom si uvědomit, že GRE tunely nejsou šifrovány.

Další informace:

- http://www.root.cz/clanky/tuneluji-tunelujes-tunelujeme-jak-a-k-cemu/
- http://tier.cs.berkeley.edu/drupal/howto/ip-tunnel-using-gre-on-linux
- http://kovyrin.net/2006/03/17/how-to-create-ip-ip-tunnel-between-freebsd-and-linux/
- http://www.abclinuxu.cz/clanky/site/ipv6-konfigurace-site-tunely
- IP-IP tunel: http://fengnet.com/book/ICUNA/ch11lev1sec5.html
  GRE tunel: http://fengnet.com/book/ICUNA/ch11lev1sec6.html
- http://www.informit.com/articles/article.aspx?p=29039&seqNum=3
- http://www.linuxfoundation.org/collaborate/workgroups/networking/tunneling
- http://www.policyrouting.org/iproute2.doc.html

Poznámka:

V této sekci nejsou popsány všechny možnosti mechanismu `iproute2`. Existují například vnitřní příkazy pro monitoring, práci s multicast adresami, multicast směrování, práce s různými směrovacími protokoly včetně OSPF, atd.

Další informace:

Další informace o používání mechanismu `iproute2` (velmi užitečné tutoriály a jiné návody):

- http://linux-ip.net/gl/ip-cref/
- http://linux-ip.net/html/index.html
- http://www.policyrouting.org/iproute2-toc.html
- http://lartc.org/howto/index.html
- http://www.faqs.org/docs/Linux-mini/Remote-Bridging.html

A dále samozřejmě `man ip` použité v Linuxu nebo na Googlu.

B.5 Překlad názvů

B.5.1 Název počítače

Také v Linuxu se pracuje s názvy počítačů. Předně je důležité vědět, jak zjistit název svého počítače. Pro tyto účely používáme příkaz `hostname` a jeho varianty:

`hostname` (bez parametrů) vypíše název našeho počítače; stejný údaj, jak víme, zjistíme také v souboru `/etc/sysconfig/network` nebo případně `/etc/hostname`, pokud existuje

`hostname` *`novynazev`* nastavíme název našeho počítače, potřebujeme vyšší přístupová oprávnění (tudíž použijeme příkaz `su` nebo `sudo`, podle toho, který naše distribuce podporuje)

`hostname -F` *`soubor`* nastavíme název našeho počítače, tento název je uložen v zadaném souboru, taktéž potřebujeme vyšší přístupová oprávnění

`domainname` (bez parametrů) vypíše název naší NIS domény (pozor, ne DNS; NIS je obdoba Active Directory pro UNIXové systémy, postupně nahrazován různými implementacemi LDAP), pokud ovšem máme NIS vytvořen

`dnsdomainname` (bez parametrů) vypíše název naší DNS domény (to je to, co v plném názvu počítače následuje za první tečkou), podobně `hostname -d`

B.5.2 Resolver a soubor `resolv.conf`

Konfigurace *resolveru* (tj. části mechanismu překladu adres, která zadává dotazy, je na každém klientovi) je tedy v souboru `/etc/resolv.conf`. V tomto souboru mohou být záznamy několika typů:

- `nameserver` *`adresa`* doménový server, obvykle zadaný IP adresou, takových záznamů může být více (pak je ale důležité jejich pořadí, jako první dáme ten server, na který se chceme obracet nejčastěji, tj. obvykle nejbližší)

- `domain` *doména* název domény, zároveň s názvem našeho počítače tvoří plné jméno počítače, pokud nejsme v žádné síťové doméně, pak je zde záznam
  `domain localdomain`
- `search` *doména doména ...* seznam domén, které jsou při vyhledávání určitého názvu přidávány pro zkompletování plného jména (před odesláním dotazu DNS serveru), například pokud je v seznamu doména `fpf.slu.cz`, pak název uzlu `axpsu` lze doplnit na `axpsu.fpf.slu.cz`
- `options volby` volby pro konfiguraci resolveru (uvádějí se všechny na jednom řádku), např.
  - počet pokusů pro vyřízení DNS dotazu na každý známý DNS server se nastavuje takto:
    `options attempts:3`
    (výchozí hodnota je 2, zvýšili jsme počet pokusů, zřejmě máme méně spolehlivé spojení k DNS serverům, obvykle není nutné měnit)
  - podporu IPv6 zapneme volbou `inet6`,
  - volba `no-check-names` se používá především tehdy, když máme v síti také počítače s nainstalovanými Windows; Windows totiž často porušují RFC 952, které v názvech dovoluje pouze písmena, číslice a pomlčky, a bez použití této volby by uzly s Windows s názvem obsahujícím nedovolené názvy nebyly dostupné, atd.

  Pokud chceme použít více voleb, musíme je umístit na jeden řádek, například
  `options attempts:1 ipv6 no-check-names`

Záznamy typu `domain` a `search` by teoreticky neměly být oba najednou použity (dá se říct, že `domain` je speciálním případem `search`, ve kterém zadáváme jen jednu doménu). Prakticky se vyskytovat mohou, ale při načítání konfigurace se bere v úvahu jen poslední uvedený z těchto záznamů.

V Linuxu se běžně používá DNS server *BIND* běžící jako démon `named`, jeho popis je však nad rámec těchto skript. Postup konfigurace včetně ukázkových konfiguračních souborů najdeme například v knize [10] ze seznamu literatury.

B.5.3 Testování DNS

Podobně jako ve Windows, i v Linuxu máme k dispozici příkaz `nslookup`. Způsob využívání je podobný jako ve Windows:

`nslookup www.google.com` takto zjistíme IP adresy zadaného serveru
`nslookup 209.85.148.99` zjistíme jméno přislušející zadané IP adrese

Do interaktivního režimu se dostaneme jedním ze dvou způsobů:

- jednoduše zadáním příkazu bez parametrů,
- příkaz s jediným parametrem – názvem DNS serveru, před který napíšeme pomlčku, aby byl odlišitelný od názvu, který chceme přeložit.

Název DNS serveru zadáváme tehdy, když výchozí server neposkytuje autorizované odpovědi a nevíme jistě, zda v jeho cache jsou správné záznamy. Pokud už jsme v interaktivním režimu, nastavíme používaný DNS server příkazem `server` *adresa* (zadáme adresu DNS serveru).

Vnitřní příkazy vpodstatě odpovídají tomu, co jsme si ukazovali u příkazu pro Windows, tedy například, překlady podobné jako v neinteraktivním režimu, zobrazení parametrů, určení typu dotazu na DNS server, zobrazení hostitelů (uzlů) v doméně, určení typu zobrazovaných hostitelů (výchozí je typ A) atd. Z interaktivního režimu se dostaneme příkazem `exit`.

Podobný účel má příkaz `dig` (zkratka z Domain Information Gropher). Tento nástroj je u administrátorů oblíbenější než `nslookup`, zvláště pro účely testování nastavení DNS serverů, už proto, že jeho výstup je obsáhlejší a odpovídá údajům v DNS paketu.

`dig www.root.cz` získáme vyčerpávající odpověď se všemi potřebnými informacemi, kromě žádané IP adresy (resp. více adres) například adresu DNS serveru, který odpověděl, jak dlouho trvala komunikace, ke každé adrese informaci o třídě záznamu (kde je platný), typ záznamu, atd.

`dig -x 91.213.160.118` reverzní dotaz, chceme doménový název k zadané IP adrese

`dig www.root.cz +short` krátká odpověď ve stylu `nslookup` (tj. vypíše se pouze IP adresa)

`dig google.com ANY` chceme všechny typy záznamů týkající se daného serveru (výchozí jsou záznamy typu A, tj. IPv4 adresy)

`dig seznam.cz NS +short` vypíšou se doménová jména DNS serverů v zadané doméně, jejich IP adresy můžeme zjistit následným dotazem podle vypsaných doménových jmen

Příklad

Vyzkoušíme práci s příkazem `dig`:

```
sarka@notebook:~$ dig www.root.cz

; <<>> DiG 9.7.1-P2 <<>> www.root.cz
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 55084
;; flags: qr rd ra; QUERY: 1, ANSWER: 2, AUTHORITY: 2, ADDITIONAL: 2

;; QUESTION SECTION:
;www.root.cz. IN A

;; ANSWER SECTION:
www.root.cz. 5 IN CNAME root.cz.
root.cz. 5 IN A~91.213.160.118

;; AUTHORITY SECTION:
root.cz. 5 IN NS ns.iinfo.cz.
root.cz. 5 IN NS ns6.adminit.cz.

;; ADDITIONAL SECTION:
ns6.adminit.cz. 5 IN A~81.91.85.230
ns6.adminit.cz. 5 IN AAAA 2001:1568:b:149::1

;; Query time: 18 msec
;; SERVER: 192.168.184.2#53(192.168.184.2)
;; WHEN: Mon Jul 11 07:09:03 2011
;; MSG SIZE  rcvd: 152
```

Ve výpisu jsou tyto položky:

- v záhlaví výpisu jsou informace o tom, jakým způsobem byl příkaz volán a kterou máme verzi, dále část s početními údaji (čísla odpovídají počtům záznamů uvedených v následujících sekcích), příznaky apod.,
- sekce „Question section“ obsahující údaje dotazu (zde doménová adresa, třída „INternet“, typ záznamu „A“)

- sekce „Answer section“ obsahuje údaje z odpovědi (v záhlaví máme uvedeny dva záznamy v odpovědi, proto zde budou dva řádky obsahující doménový název, TTL, třídu, typ záznamu a hodnotu záznamu),
- v sekci „Authority section“ máme seznam doménových adres DNS serverů, které o dotazovaném názvu poskytují autoritativní odpověď (pokud získaným informacím nedůvěřujeme, můžeme se zeptat některého z těchto serverů),
- v následující sekci najdeme IP adresy DNS serverů uvedených v předchozí sekci,
- výpis končí statistickými informacemi.

Teď podobně zjistíme informace o hostiteli `mail.google.com`, ale zeptáme se jiného DNS serveru. Název DNS serveru musíme uvést symbolem „zavináče“, aby bylo zřejmé, že se nejedná o název, který má být přeložen, za zavináčem můžeme zadat doménovou nebo IP adresu. Zde se (shodou okolností) dotážeme DNS serveru poskytovaného Googlem:

```
sarka@notebook:~$ dig @8.8.4.4 mail.google.com

; <<>> DiG 9.7.1-P2 <<>> @8.8.4.4 mail.google.com
; (1 server found)
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 20135
;; flags: qr rd ra; QUERY: 1, ANSWER: 5, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;mail.google.com. IN A

;; ANSWER SECTION:
mail.google.com. 86399 IN CNAME googlemail.l.google.com.
googlemail.l.google.com. 299 IN A~74.125.232.246
googlemail.l.google.com. 299 IN A~74.125.232.248
googlemail.l.google.com. 299 IN A~74.125.232.247
googlemail.l.google.com. 299 IN A~74.125.232.245

;; Query time: 56 msec
;; SERVER: 8.8.4.4#53(8.8.4.4)
;; WHEN: Mon Jul 11 07:21:14 2011
;; MSG SIZE  rcvd: 124
```

Je zřejmé, že odpověď jsme sice dostali, ale vyžádali jsme si ji od serveru, který není pro daný záznam autoritativní (údaje bere ze své cache nebo se dotazuje jinde).

Dále vyzkoušíme zkrácený výpis (použijeme parametr `+short`), chceme DNS servery v doméně `seznam.cz`.

```
sarka@notebook:~$ dig seznam.cz NS +short

ns.seznam.cz
ms.seznam.cz
```

Příklad

Příkaz `dig` můžeme také použít pro zjištění celé trasy našeho dotazu:

```
sarka@notebook:~$ dig seznam.cz +trace
```

```
; <<>> DiG 9.7.1-P2 <<>> seznam.cz +trace
;; global options: +cmd
. 5 IN NS i.root-servers.net.
. 5 IN NS j.root-servers.net.
. 5 IN NS k.root-servers.net.
. 5 IN NS l.root-servers.net.
. 5 IN NS m.root-servers.net.
. 5 IN NS a.root-servers.net.
. 5 IN NS b.root-servers.net.
. 5 IN NS c.root-servers.net.
. 5 IN NS d.root-servers.net.
. 5 IN NS e.root-servers.net.
. 5 IN NS f.root-servers.net.
. 5 IN NS g.root-servers.net.
. 5 IN NS h.root-servers.net.
;; Received 272 bytes from 192.168.184.2#53(192.168.184.2) in 12 ms

cz. 172800 IN NS f.ns.nic.cz.
cz. 172800 IN NS d.ns.nic.cz.
cz. 172800 IN NS a.ns.nic.cz.
cz. 172800 IN NS c.ns.nic.cz.
cz. 172800 IN NS b.ns.nic.cz.
;; Received 334 bytes from 192.5.5.241#53(f.root-servers.net) in 10 ms

seznam.cz. 18000 IN NS ms.seznam.cz.
seznam.cz. 18000 IN NS ns.seznam.cz.
;; Received 93 bytes from 194.0.14.1#53(c.ns.nic.cz) in 43 ms

seznam.cz. 300 IN A~77.75.72.3
;; Received 43 bytes from 77.75.77.77#53(ms.seznam.cz) in 9 ms
```

Pokud nezadáme DNS server, kterému má být zaslán dotaz, příkaz `dig` použije DNS servery uvedené v souboru `/etc/resolv.conf`.

B.5.4 Zjistění informací o doméně

Pokud chceme zjistit informace o určité doméně, použijeme příkaz `whois`. Tento příkaz nám vypíše veškeré dostupné informace o doméně – kontakt na vlastníka domény, doba, na kterou je doména registrována apod. Příkaz se hodí například tehdy, když chceme registrovat vlastní doménu a chceme si ověřit, zda už není někým registrována, a nebo pokud z některé domény přichází malware a chceme upozornit vlastníka, aby „si udělal pořádek“.

`whois koupaliste.cz` zjistíme údaje o zadané doméně

`whois -r koupaliste.cz` výpis bude odlišný, ale základní informace budou shodné (uvedeným přepínačem jsme si vyžádali informace z WHOIS databáze RIPE obsahující především evropské servery)

V manuálové stránce příkazu bychom zjistili přepínače pro další možné WHOIS databáze (pozor, manuálové stránky různých UNIXových systémů se liší v tom, jak moc jsou podrobné při komentování seznamu přepínačů tohoto příkazu, pak je dobré použít manuálové stránky na Internetu).

Další informace:

Další informace o doménách v Linuxu:

- http://www.madboa.com/geek/dig/
- http://www.brennan.id.au/08-Domain_Name_System_BIND.html

Úkoly

1. Zjistěte název svého počítače. Dále zjistěte, jaké DNS servery používáte.
2. Zobrazte příkazem `dig` podrobnosti o serveru `seznam.cz`, a to všechny typy DNS záznamů (tj. ANY). Srovnejte s výpisem pomocí příkazu `nslookup`.
3. To, že se ptáme jiného DNS serveru než je ten od našeho poskytovatele, nemusí znamenat, že dostaneme komplexnější informace. Vyzkoušejte dotaz na doménu `root.cz` postupně bez zadání DNS serveru a pak se zadáním některého DNS serveru (například `ns.seznam.cz` nebo `8.8.4.4`). Pokud příkaz zkoušíte v síti některé vysoké školy, kde se učí Linux, tak bude výstup od výchozího serveru komplexnější.
4. Zjistěte údaje o několika doménách druhé úrovně podle vlastního výběru (například `seznam.cz`, `slu.cz`, apod.).

B.6 Testování a statistiky

B.6.1 Základní nástroje

Příkaz `ping` slouží ke stejným účelům jako ve Windows:

`ping www.google.com` v pravidelných intervalech odesílá cílovému zařízení ICMP pakety, po přerušení klávesou Ctrl+C ukončí zasílání a vypíše souhrnnou statistiku (kolik paketů bylo odesláno, přijato, podíl ztracených, dále odezvu cílového počítače – nejrychlejší, nejpomelejší, střední hodnota), cílový počítač můžeme zadat také IP adresou

`ping -c 4 www.google.com` chová se stejně jako ve Windows, tedy zadáme počet zasílaných ICMP paketů (bez tohoto parametru by byly zasílány tak dlouho, dokud bychom nestiskli Ctrl+C)

`ping -c 1 -R www.google.com` parametr `-R` (pozor, velké písmeno) znamená, že se vypíšou všechny routery na cestě podobně jako například u `traceroute` (ale zde získáme méně podrobný výpis)

V manuálové stránce bychom zjistili další možné parametry.

Existuje také příkaz `arping`, který má podobný účel jako `ping`, ale používá jiný typ paketů (ARP Request) a je určen především pro testování v lokální síti.

`arping -f -c 4 -I eth0 209.85.143.99` zadanému cíli (poslední parametr) přes zadané rozhraní odešle maximálně čtyři ARP pakety, parametr `-f` určuje, že se nemusejí deslat všechny 4, ale pouze tolik, než vzdálený systém odpoví

Zatímco ve Windows máme příkaz `tracert`, v UNIXových systémech včetně Linuxu se setkáme s příkazem `traceroute`.

`traceroute www.google.com` vypíše trasu k zadanému zařízení (zadáváme doménovou nebo IP adresu), použije UDP pakety (pouze metoda s využitím UDP paketů je použitelná kterýmkoliv uživatelem)

`traceroute -I 209.85.143.99` podobně, ale použije zprávy ICMP Echo Request (přepínač je velké „i“), ekvivalentem je příkaz `tracert` (s ním se setkáme i u Windows)

`traceroute -6 209.85.143.99` vynucení použití IPv6 (také `traceroute6`), k vynucení IPv4 slouží přepínač `-4`

Tento příkaz má ještě několik dalších přepínačů (viz manuálové stránky) včetně u těchto příkazů běžného `-n` zakazujícího mapování IP adres na doménové názvy, určení maximální hodnoty TTL, atd.

Podobný je příkaz `tracepath` (v některých distribucích ho však nenajdeme). Slouží především ke zjištění hodnot MTU na cestě:

`tracepath 209.85.143.99` vypíše údaje o cestě k zadanému uzlu sítě (ke každému úseku zjistíme hodnotu odečítaného TTL na tomto úseku, časový údaj a hodnotu MTU (Path MTU)

`tracepath -n 209.85.143.99` místo doménových se vypisují IP adresy

Samozřejmě nesmí chybět příkaz `netstat`. Parametry jsou obdobné těm ve Windows (odkud se tento příkaz asi do Windows dostal), tedy si zde uvedeme jen několik dalších motivačních příkladů:

Příklad

Zobrazíme seznam otevřených portů, postupně: výpis numerických IP adres (n), výpis UDP portů (u), TCP portů (t), všechny údaje (a, naslouchající i nenaslouchající), vypsat PID procesů, které port používají (p). Výpis příkazu následuje.

```
sarka@notebook:/home/sarka# netstat -nutap

Aktivní Internetová spojení (servery a~navázaná spojení)
Proto Přích-F Odch-F Místní Adresa          Vzdálená Adresa
Stav      PID/Program name
tcp        0      0 127.0.0.1:2208          0.0.0.0:*
LISTEN     2725/hpiod
tcp        0      0 127.0.0.1:49702         0.0.0.0:*
LISTEN     2728/python
tcp        0      0 0.0.0.0:113             0.0.0.0:*
LISTEN     3038/inetd
tcp        0      0 127.0.0.1:631           0.0.0.0:*
LISTEN     2850/cupsd
tcp        0      0 127.0.0.1:25            0.0.0.0:*
LISTEN     3018/exim4
tcp        0      0 10.0.0.2:54255          74.125.232.216:443
SPOJENO    3517/firefox-bin
tcp        0      0 10.0.0.2:54260          74.125.232.216:443
SPOJENO    3517/firefox-bin
tcp        0      0 10.0.0.2:54265          74.125.232.216:443
SPOJENO    3517/firefox-bin
tcp        0      0 10.0.0.2:39412          74.125.232.213:443
SPOJENO    3517/firefox-bin
tcp        0      0 10.0.0.2:32819          74.125.232.194:80
SPOJENO    3517/firefox-bin
tcp        0      0 127.0.0.1:47622         127.0.0.1:111         TIME_WAIT  -
tcp        1      0 10.0.0.2:55360          91.189.90.132:80
CLOSE_WAIT 3597/python
udp        0      0 0.0.0.0:32768           0.0.0.0:*
        2929/avahi-daemon:
udp        0      0 0.0.0.0:68              0.0.0.0:*
        3152/dhclient
udp        0      0 0.0.0.0:714             0.0.0.0:*
```

```
        3082/rpc.statd
udp        0      0 0.0.0.0:5353            0.0.0.0:*
        2929/avahi-daemon:
udp        0      0 0.0.0.0:631             0.0.0.0:*
        2850/cupsd
```

Vidíme, že (aktivně) komunikuje především Firefox, další démony spíše naslouchají. Najdeme tam například nechvalně známý *Avahi* (provádí tzv. Zero-conf automatické nastavení parametrů sítě, což je odborníky hodně kritizováno), *cupsd* (tiskový subsystém), *hpiod* (démon pro podporu tisku kompatibilní s HP zařízeními), *exim4* (síťová podpora pro mail klienty, v současné době se spíše doporučuje použít *postfix*), *inetd*,[12] *dhclient* (DHCP klient), atd. (výpis byl záměrně mírně zkrácen).

Příklad

V Linuxu si pomocí příkazu `netstat` můžeme prohlédnout tabulku síťových rozhraní tak, jak ji vidí jádro:

```
sarka@notebook:/home/sarka# netstat -i

Tabulka rozhraní v jádru
Iface   MTU Met   RX-OK RX-ERR RX-DRP RX-OVR    TX-OK TX-ERR TX-DRP TX-OVR Flg
eth0   1500 0      2218      0      0      0     1919      0      0      0 BMRU
lo    16436 0        48      0      0      0       48      0      0      0 LRU
```

Co je to MTU, už víme. Následuje metrika, údaje pro příchozí (RX) a odchozí (TX) pakety, kdy se vypisuje počet úspěšně přijatých/odeslaných, chybných, zahozených. Poslední sloupec obsahuje jednopísmenné příznaky:

- B: přijímá broadcast pakety,
- M: zapnut promiskuitní režim,
- R: rozhraní je používáno (běží, running)
- U: rozhraní je aktivní (up)
- L: loopback

Příkaz `netstat-nat` je zdánlivě podobný příkazu `netstat`, ale není tentýž:

`netstat-nat` výpis adres překládaných mechanismem NAT (pozor, před pomlčkou není mezera, opravdu se jedná o celý příkaz bez jakýchkoliv parametrů)

`netstat-nat -n` před druhou pomlčkou už mezera je, význam příkazu je stejný jako u předchozího, ale místo názvů počítačů se budou vypisovat IP adresy (tj. bez DNS překladu)

[12] *Inetd* má podobnou funkci jako ve Windows svchost.exe, může hostit démony. Na rozdíl od svchost je zde pro démony „pohostinství“ dobrovolné, mohou běžet samostatně. Inetd jednoduše naslouchá na příslušných portech a když zjistí pokus o komunikaci s démonem, jehož hostí, jeho kód spustí ve svém vlastním procesu, není třeba spouštět démona navíc (tj. místo více různých démonů běží jen jeden, inetd, a příslušné služby se spouštějí až na vyžádání).

B.6.2 Pokročilé testování

Pro pokročilé testování strojů a celé sítě existuje mnoho velmi užitečných nástrojů. V tomto předmětu je bohužel nestačíme probrat. Namátkou:

- tcpdump
- nmap
- ettercap

B.7 Firewall v Linuxu

V linuxových jádrech (od verze 2.4) najdeme firewall *NetFilter*, což je obousměrný stavový firewall (provádí funkce SPI).[13] *NetFilter* dokáže filtrovat pakety podle údajů v hlavičkách protokolů TCP, UDP, IP, ICMP a také dalších uvedených v souboru `/etc/protocols`, je použitelný pro mnoho činností souvisejících se zpracováním paketů, včetně mechanismu NAT.

Jedná se o modul jádra, ke kterému se nepřistupuje přímo, ale přes obslužný program `iptables`.[14] Takže pozor – firewall je NetFilter, obslužný program běžící v uživatelském režimu je `iptables`.

B.7.1 Princip firewallu

Základem je *tabulka* (table), přes tabulky prochází několik řetězců *chain* (čti: [čejn]), které již obsahují pravidla uplatňovaná na pakety (tato pravidla jsou zřetězená a na paket se uplatňují postupně, dokud se nenajde „pasující“ pravidlo, proto se tomu říká chain).

Tabulka obvykle určuje, co konkrétně se má v dané části řetězce dít (například jen filtrování, nebo překlad adres či změna některých dalších údajů v záhlaví procházejícího paketu), chain obvykle určuje, pro jaký typ paketů se to má dít (například pro příchozí pakety, odchozí, průchozí u routeru, připravené na směrování, zpracované směrováním, apod.).

Příklad

Pokud jsme na routeru se zapnutým přeposíláním (forwarding), je příchozí paket nejdřív zařazen do chainu *PREROUTING* (tj. pro činnosti, které se mají provést ještě před směrováním), v něm projde určenými tabulkami (například tabulkou *nat*, pokud se provádí překlad adres).

Pak proběhne směrování, které určí, jestli se tento paket má přeposílat (tedy je přeřazen do chainu *FORWARD* pro přeposílané pakety) nebo je určen přímo pro toto zařízení (půjde do chainu *INPUT* pro příchozí pakety), a v rámci daného chainu projde stanovenými tabulkami. Přeposílaný paket pak ještě projde chainem *OUTPUT*, protože se stává odchozím paketem.

Takže chain vlastně určuje „kategorii“ paketu, tabulky pravidla pro zacházení. Pokud přes tuto tabulku procházejí dva pakety takové, že každý je v jiném chainu, bude se s každým zacházet trochu jinak,

[13] Stavové firewally dokážou pracovat samozřejmě nejen se stavovými, ale i s nestavovými protokoly (jako je například UDP), přestože se v některých publikacích můžeme dočíst, že ne.

[14] Ve starších jádrech, se kterými se (doufejme) už nesetkáme, se používal firewall *IPChains* se stejnojmenným obslužným programem (`ipchains`).

třebaže typ prováděných operací bude stejný (filtrování je ve více chainech, ale jinak se filtrují příchozí, jinak průchozí a jinak odchozí pakety).

Standardně existují tyto tabulky (v příkazu `iptables` se před název tabulky dává přepínač `-t`):

- *filter* (výchozí)
- *mangle* (pro transformace – upravujeme záhlaví, hodnoty TTL, pole TOS/QoS, apod.)
- *nat* (pro mechanismus NAT)
- *raw* a *conntrack* jsou pomocné mechanismy označování (marking) paketů pro pozdější zpracování a stavového filtru (SPI), obvykle je třeba jejich funkcionalitu do jádra doplnit (načíst moduly)
- *security* je přidávána bezpečnostními moduly jádra jako je například SELinux.

Přes každou tabulku tedy vede několik *chainů* (řetězců), a to:

- Tabulka *filter* obsahuje standardně tyto chainy:
  - chain INPUT se uplatní na pakety, které do systému přicházejí,
  - chain OUTPUT se uplatní na pakety, které ze systému odcházejí,
  - chain FORWARD je určen pro pakety, které nejsou vytvořeny uvnitř systému a ani pro něj nejsou určeny, předávají se jen mezi rozhraními systému (průchozí pakety), typicky v zařízení, které slouží jako směrovač,

  pokud jde paket do chainu FORWARD, nepadne už do žádného jiného chainu.
- Tabulka *nat* obsahuje standardně tyto chainy:
  - chain PREROUTING pro příchozí pakety, na které se má použít DNAT (tj. má se přeložit cílová adresa, a to ještě před vlastním směrováním),
  - chain POSTROUTING pro odchozí pakety, na které se má použít SNAT (překládá se zdrojová adresa, až po směrování),
  - chain OUTPUT pro odchozí pakety, kdy se má modifikace provést ještě před dalším zpracováním.
- Tabulka *mangle* obsahuje standardně chainy pojmenované jako všechny, které jsou uvedeny u předchozích tabulek (INPUT, OUTPUT, FORWARD, PREROUTING, POSTROUTING) s tím rozdílem, že v nich najdeme pravidla modifikující i jiná pole záhlaví než jen ta adresová.
- Tabulka *security* může následovat po tabulce *filter*, obsahuje chainy INPUT, OUTPUT a FORWARD a dokáže podle potřeby pakety označovat a přesměrovávat k jiným modulům jádra.

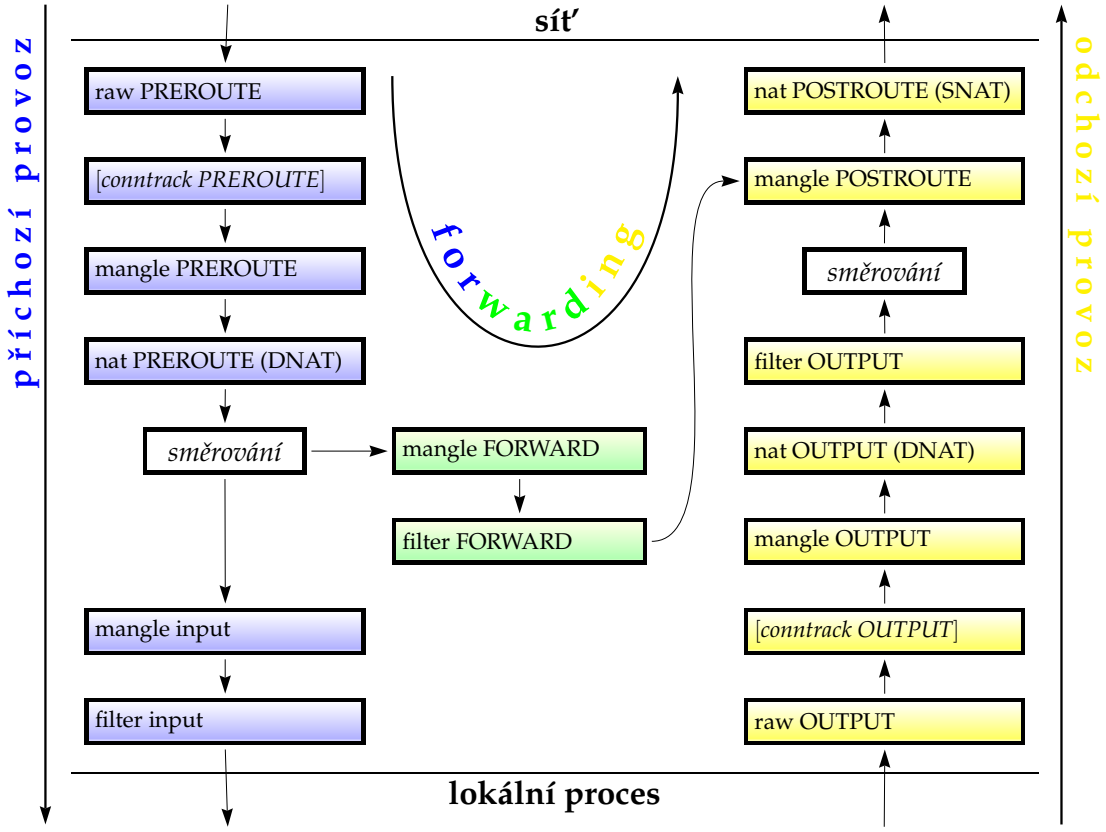
V tabulkách si můžeme vytvořit další chainy (uživatelsky definované), nemusíme zůstat jen u standardních, případně nové tabulky. Nové tabulky se však často přidávají jako další moduly jádra.

Na obrázku B.3 je znázorněna dráha paketu mezi jednotlivými tabulkami a chainy. Všimněte si, že přeposílané (forwarded) pakety nedorazí do žádného chainu typu INPUT ani OUTPUT.

U pravidel v chainu záleží na pořadí. Každé pravidlo má své pořadové číslo. Nová pravidla můžeme do chainu vkládat na začátek či na konec chainu, anebo na konkrétní pozici zadanou pořadovým číslem.

Program (ve skutečnosti služba, démon) `iptables` umožňuje pracovat s chainy (vytvořit nebo zrušit chain, nastavit výchozí politiku) a s jejich obsahem (přidávat a odstraňovat pravidla v chainu, vypsat seznam pravidel).

V různých distribucích se však můžeme setkat i s dalšími obslužnými programy, například `uwf` (Uncomplicated FireWall) s grafickým rozhraním *gufw*, `shorewall`, `sanewall`, `apf-firewall` a další. Často jde spíše o nástavby `iptables`.



Obrázek B.3: Dráha paketu mezi výchozími tabulkami a chainy v Netfilteru

B.7.2 Základní vnitřní příkazy a parametry pro filtrování

Ve vnitřních příkazech se používají tzv. *targety* určující akci, která se má s určeným paketem provést. Můžeme použít tyto targety:

- `ACCEPT` akceptuj, paket má povoleno projít, obvykle v tabulce *filter*,
- `DROP` zahoď bez odezvy (tiché zahození), taktéž v tabulce *filter*,
- `REJECT` odmítni (s odezvou), odešle ICMP zprávu o odmítnutí (přesněji – o nedostupnosti adresy) uzlu, od kterého paket přišel, pro tabulku *filter*,
- `DNAT` a `SNAT` patří k tabulce *nat*, účel je zřejmý – akcí je překlad cílové nebo zdrojové adresy na jinou IP adresu, používáme při překladu statických adres,
- `MASQUERADE` také patří k tabulce *nat* chainu POSTROUTING, u odchozích paketů překládá celý rozsah vnitřních (zdrojových) adres na jednu vnější (slouží ke skrývání lokálních adres), na rozdíl od SNAT bývá adres více a častěji se používá pro PAT (překlad portů), používáme pouze tehdy, když potřebujeme překládat dynamické soukromé adresy,
- `TOS` a `TTL` patří k tabulce *mangle*, mění pole TOS (type of service) a TTL v záhlaví paketu,
- `MARK` a `CONNMARK` se v tabulce *mangle* používají pro přidávání značek do záhlaví paketů,
- `LOG` umožňuje logovat informace ze záhlaví paketu, používá se obvykle s programem *syslog*,
- `RETURN` pokud jsme během vyhodnocování pravidel odskočili do jiného chainu, tento target nás vrátí zpět,

atd. Jednotlivé targety obvykle patří ke konkrétní tabulce, tedy když přidáme do jádra další modul s tabulkou, přidají se i další použitelné targety. Za targety se považují i uživatelsky definované chainy.

Příkaz `iptables` se používá s nejméně jedním parametrem (obvykle více), což bývá určení tabulky, vnitřní příkaz nebo parametr. Obvykle (ne však vždy) zadáváme tabulku, se kterou chceme v příkazu pracovat, k tomu použijeme přepínač `-t`, například `-t nat`. Zde se podíváme na vnitřní příkazy a parametry, které se používají při základním filtrování, v dalších podsekcích projdeme další (překlady adres, stavový firewall, značení paketů).

 Obvykle tedy používáme některý vnitřní příkaz (pozor, velká písmena) představující:

`-P` nebo `-policy` určuje výchozí zásadu (politiku) pro chain, ve kterém chceme pracovat (pokud na paket nebude pasovat žádné pravidlo chainu, použije se tato politika), za přepínačem následuje název chainu a dále *target* určující akci, která se má provést, výchozí politiku lze určit pouze pro předdefinované chainy, nikoliv pro uživatelsky definované; například
`iptables -P INPUT -j DROP`,

`-L` nebo `-list` vypisuje seznam všech pravidel v zadaném chainu, například
`iptables -L` (výpis bývá dlouhý dokonce i na desktopu, všechny tabulky)
`iptables -t filter -L -n`
`iptables -t filter -L -nv`
`iptables -t filter -L INPUT -n`
první příkaz vypíše všechna pravidla tabulky *filter*, nechává IP adresy (nepřekládá na doménové, díky poslednímu parametru), druhý příkaz udělá totéž, ale ve verbose („upovídaném“) módu, tudíž se dozvíme více, třetí příkaz vypíše pouze pravidla v chainu INPUT,

`-A` nebo `-append` připojí nové pravidlo na konec chainu, následuje název chainu a specifikace pravidla

`-I` nebo `-insert` připojí nové pravidlo na začátek chainu (pokud nezadáme žádný index) nebo na zadaný index (číslo pravidla můžeme napsat za název chainu), dto., první index je 1,

`-R` nebo `-replace` přepíše existující pravidlo na daném indexu,

`-D` nebo `-delete` odstraní pravidlo z chainu, následuje název chainu a buď číslo pravidla nebo jeho specifikace,

`-F` nebo `-flush` následované názvem chainu tento chain vyprázdní (vymaže všechna jeho pravidla)

`-Z` nebo `-zero` vynuluje všechny čítače v tabulce, obvykle se používá zároveň s příkazem `-L`, abychom viděli stav čítačů před jejich vynulováním, tedy například
`iptables -t filter -LZ`

`-N` nebo `-new-chain`, `-X` nebo `-delete-chain` pro vytvoření nebo odstranění chainu, následuje vždy název chainu,

`-E` nebo `-rename-chain` pro přejmenování chainu, následuje původní a nový název chainu.

 *Parametry* slouží k upřesnění příkazu (určují, podle čeho se pakety mají filtrovat), na rozdíl od příkazů jsou pro ně určena malá písmena, používáme především:

`-p` nebo `-protocol` určí protokol, například `-p tcp`, negace s vykřičníkem, např. `-p ! udp`,

`-dport` nebo `-sport` dovoluje k protokolu podle předchozí odrážky přidat konkrétní číslo portu (zdrojového nebo cílového, také se dá označit názvem příslušného aplikačního protokolu), například
`iptables -t filter -A OUTPUT -p tcp -dport 80 -j DROP`
`iptables -t filter -A OUTPUT -p tcp -dport http -j DROP`
(obojí: zablokovali jsme odchozí tcp spojení přes http port)

```
iptables -t filter -A FORWARD -p tcp -dport imap -j ACCEPT
iptables -t filter -A FORWARD -p tcp -dport pop3 -j ACCEPT
```

(povolili jsme přeposílání paketů se stahovanou poštou, aplikační protokol IMAP a POP3)

`-icmp-type` můžeme použít filtrování pro určitý typ ICMP zprávy, například

```
iptables -t filter -A FORWARD -p icmp -icmp-type 8 -j DROP
```

zahazujeme všechny zprávy ICMP Echo Request (tím zablokujeme odpovědi na `ping`)

`-syn` v TCP záhlaví je nastaven příznak SYN (samotný bez příznaku ACK), což znamená, že zřejmě jde o paket, který navazuje nové spojení (ne nutně, také je důležité otestovat stav spojení, viz dále o SPI)

`-s` nebo `-source` nebo `-src` a `-d` nebo `-destination` nebo `-dst` je zdrojová a cílová adresa, u sítě píšeme i délku prefixu, například `-s 192.89.120.0/24`, opět můžeme použít vykřičník pro negování (tj. všechny adresy kromě této),

`-i` nebo `-in-interface` zadává vstupní rozhraní, používá se v chainech INPUT, FORWARD a PREROUTING, například `-i eth0`, můžeme také zadávat negaci, například `-i ! eth4` znamená pakety ze všech síťových rozhraní kromě eth4,

`-o` nebo `-out-interface` podobně pro výstupní rozhraní, na které je paket směrován (v chainech OUTPUT, FORWARD a POSTROUTING), například

```
iptables -t filter -A OUTPUT -o eth1 -j ACCEPT
```

veškerý ochozí provoz (neodpovídající jiným pravidlům) odcházející přes port eth1 bude povolen

Další parametry slouží k upřesnění činnosti, která se má s paketem provést:

`-j` nebo `-jump` slouží k zadání provedení targetu (viz výše), který za tímto přepínačem následuje, znamená tedy odskok (jump) k provedení targetu, kterým může být některá akce (ACCEPT, DROP, SNAT apod.), a *nebo* název uživatelsky definovaného chainu, do kterého takto „odskočíme“ s tím, že se po procházení uživatelského chainu můžeme vrátit zpět (tj. odskok se zapamatováním adresy), například

```
iptables -t filter -A INPUT -p tcp -j tcppakety
```

(pokud jde o TCP paket, přesuneme se do chainu *tcppakety*; pokud v tom chainu nenajdeme žádné vyhovující pravidlo, vrátíme se zpět do chainu INPUT a pokračujeme následujícími pravidly, uživatelský chain *tcppakety* byl předem vytvořen příkazem `-N`)

`-g` nebo `-goto` je pro uživatelsky definované chainy podobné jako `-j`, s tím rozdílem, že se před odskokem jinam nezapamatuje adresa momentálního chainu (přesněji – pokud už byla předtím některá adresa zapamatována pomocí `-j`, nepřepíše se jinou adresou, tudíž při případném návratu během vyhodnocování jednoho paketu se nevrátíme do chainu, ze kterého odcházíme, ale bude přeskočen), například

```
iptables -t filter -A INPUT -p tcp -g tcppakety
```

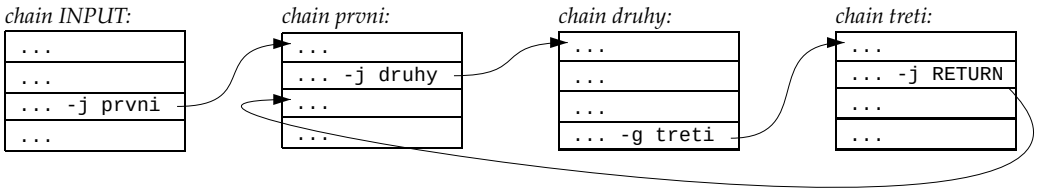
podobně jako předchozí, ale pokud v uživatelském chainu nenalezne vhodné pravidlo, už se do INPUT nevrací (ovšem v *tcppakety* může být také parametr `-j` nebo `-g` směrující vyhodnocování paketu jinam).

Parametr `-j` tedy lze použít pro odskok do jiného (obvykle uživatelsky definovaného) chainu. Zpět se vrátíme buď tehdy, když následný chain neobsahuje vhodné pravidlo, a nebo lze návrat vynutit

targetem RETURN (příkaz umístíme na konec vlastního chainu):

```
iptables -A vlastni_chain -j RETURN
```

(zde by bylo vhodné přidat ještě nějakou filtrovací podmínku, při jejímž splnění dojde k návratu, jinak pravidlo nemá moc smysl).



Obrázek B.4: Rozdíl mezi parametry `-j` a `-g` ve firewallu

Zbývá ještě několik pomocných přepínačů používaných u příkazu `-L`:

`-v` nebo `-verbose` zapíná verbose („ukecaný“) mód, chceme delší výpis s více informacemi,

`-n` nebo `-numeric` použijeme, když chceme ve výpisu IP adresy místo doménových, výpis je pak celkově rychlejší a snadněji se hromadně zpracovává,

další bychom našli v manuálové stránce příkazu: `man 8 iptables`

Výše uvedený výčet příkazů a parametrů není zdaleka úplný, podrobnější informaci bychom našli opět v manuálové stránce příkazu, případně na odkazech na konci této sekce.

Poznámka:

Měli bychom si uvědomit, že zpracovávání paketu v tabulce končí v okamžiku, kdy NetFilter najde první vyhovující pravidlo s cílovou akcí akceptující nebo zahazující (například ACCEPT nebo DROP). Pokud je v tomto pravidle akce DROP nebo jiná „zahazovací“, paket bude okamžitě zahozen, i kdyby třeba hned následující pravidlo tento paket povolovalo (k dalšímu pravidlu se nedostaneme).

Příklad

Pár ukázek práce s `iptables`:

`/sbin/iptables -t filter -L` výpis pravidel v tabulce `filter`

`/sbin/iptables -L -v -n -line-numbers` podrobný výpis (statistika) všech tabulek, s IP adresami (n), v chainech jsou řádky číslovány

`/sbin/iptables -P INPUT -j DROP` stanovíme výchozí politiku pro chain INPUT, ve které řekneme, že vše příchozí se má zahodit (samozřejmě musíme kromě jiného přidat před toto pravidlo také pravidla, která upřesní, co se zahazovat nemá)

`/sbin/iptables -P OUTPUT -j ACCEPT` všechno odchozí povolíme, propustíme dál (opět bychom měli uvažovat, jestli nepřidáme „výjimky“)

`/sbin/iptables -A INPUT -i lo -j ACCEPT` povolíme to, co probíhá mezi lokálními procesy (komunikují přes sockety na loopbacku)

`/sbin/iptables -A INPUT -p tcp -syn -j DROP` budou zahozeny všechny příchozí pakety, které mají v TCP záhlaví nastaven pouze SYN bit (to je vždy první paket spojení, tedy znemožníme navazování spojení zvnějšku)

`/sbin/iptables -A INPUT -p icmp -icmp-type echo-request -j ACCEPT` povolili jsme pakety s ICMP zprávou č. 8 (Echo Request) z vnějšku, můžeme použít číselné i slovní označení typu zprávy

`/sbin/iptables -A INPUT -p icmp -icmp-type time-exceeded -j ACCEPT` povolili jsme příchozí ICMP pakety Time Exceeded (vypršení času při navazování spojení)

`/sbin/iptables -A INPUT -p icmp -icmp-type destination-unreachable -j ACCEPT` povolili jsme příchozí ICMP pakety hlásící nedostupnost cíle

Úkoly

1. Nastavte výchozí politiku pro odchozí provoz na akceptování.
2. Sestavte sadu pravidel, která z příchozího provozu zakážou všechno kromě provozu protokolu HTTP, POP a IMAP.
3. Na koncové stanici chceme povolit u příchozích paketů pouze ty, které jdou na TCP port 80 nebo TCP port 8080, všechny ostatní příchozí zakážeme. Jak budou vypadat pravidla pro příslušný chain?
4. Sestavte pravidla, která povolí ICMP pakety (jakékoliv, všechny typy ICMP zpráv) v příchozím, odchozím i průchozím provozu.

B.7.3 SPI

Firewall v Linuxu s jádrem 2.4 a novějším podporuje také SPI (Stateful Packet Inspection). Aby bylo možné pracovat se stavy spojení, je třeba mít v jádře zaveden modul pro sledování stavu spojení (connection tracking). Dříve se používal modul `ip_conntrack` a další s ním související, nyní se spíše setkáme s bezpečnějším a funkčně širším `nf_conntrack` a moduly s ním souvisejícími. `nf_conntrack` plně podporuje IPv4 a IPv6 včetně jejich kombinování.

Příklad

Jak zjistíme, jestli je modul v jádře zaveden (výpis bývá dlouhý, přefiltrujeme si ho příkazem `grep`), výstup v SUSE Linuxu:

```
sarka@notebook:/home/sarka# /sbin/lsmod | grep conntr

nf_conntrack_ipv6        20196    4
nf_conntrack_netbios_ns   2152    0
nf_conntrack_ipv4        10480    4
nf_conntrack             67400    5 nf_conntrack_ipv6,xt_NOTRACK,xt_state,
nf_conntrack_netbios_ns,nf_conntrack_ipv4
ipv6                    242608    25 ip6t_REJECT,nf_conntrack_ipv6,ip6table_mangle
```

V prvním sloupci je název modulu, v druhém jeho velikost, následuje počet uživatelů tohoto modulu a případné závislosti (které zavedené moduly dotyčný modul vyžadují). Pokud výpis vypadal podobně jako ten výše, pak je vše OK.

Pokud potřebné moduly v jádře nejsou, tak je třeba je do jádra zavést. U každého potřebného modulu nejdřív ověříme, zda je dostupný:

```
/sbin/modeprobe -l *conntr*
```

(případně lze vypsat vše a pak to profiltrovat `grep`em). S dostupností obvykle nebývá problém. Zbývá modul dostat do jádra:

```
/sbin/modeprobe nf_conntrack
```

Pro další moduly je postup podobný.

Podrobný návod včetně ukázek využití modulů: http://www.abclinuxu.cz/blog/pb/2007/2/nf-conntrack

Pokud máme modul v jádře, můžeme ve firewallu filtrovat pakety podle *stavu spojení* (můžeme je považovat za stavy paketů ve vztahu ke spojení, ke kterému patří). Rozlišují se tyto stavy:

- NEW: klient přes firewall navazuje nové spojení (tj. první paket spojení), může být také u jednosměrných spojení
- ESTABLISHED: paket je součástí již navázaného spojení
- RELATED: paket sice navazuje nové spojení, ale zároveň je ve vztahu k některému existujícímu (typicky u FTP, kdy mohou být navázána dvě různá spojení na dvou portech)
- INVALID: paket nelze přiřadit k žádnému existujícímu ani navazovanému spojení

Pak v příkazech programu `iptables` můžeme používat filtrování podle stavů. Existují také virtuální stavy SNAT a DNAT pro pakety, kde byla přeložena zdrojová resp. cílová adresa.

Conntrack ve skutečnosti není samostatná tabulka, funkcionalita se používá v existujících tabulkách a chainech (na obrázku B.3 na straně 329 vidíme, kde konkrétně v datovém toku se vyskytuje).

Příklad

Příklady pravidel pro příchozí spojení:

`iptables -A INPUT -p tcp ! -syn -m state -state NEW -j LOG -log-prefix "Nenastaven SYN:"` protokolují se veškeré pakety navazující spojení zvnějšku, které nemají nastaven (samotný) příznak SYN, toto pravidlo je obvykle následováno tímto:

`iptables -A INPUT -p tcp ! -syn -m state -state NEW -j DROP` zajímáme se o pakety zapouzdřující TCP segment (protokol TCP), které *nemají* nastaven (samotný) příznak SYN (je tam negace) a zároveň stav spojení je nové (tj. navazuje se nové spojení a přitom není nastaven SYN bit), výslednou akcí je tiché zahození paketu

`iptables -A INPUT -m state -state NEW -j DROP` zdánlivě totéž, co předtím, ale pozor – pokud zvenku přijde paket navazující nové spojení a nebude mít (chybně) nastaven příznak SYN, tak projde (tudíž pokud chceme blokovat příchozí navazování spojení, uvedeme obě pravidla)

`iptables -A INPUT -m state -state RELATED, ESTABLISHED -j ACCEPT` necháme projít dovnitř pakety, které patří do existujícího spojení nebo s některým existujícím souvisejí

Pro práci s příchozími a odchozími spojeními a modulem `nf_conntrack` je možné použít také program `conntrack` z balíčku *conntrack-tools* (bývá běžně v repozitářích, můžeme doinstalovat).

Seznam aktivních spojení, která jsou povolena mechanismem conntrack, najdeme takto:

```
cat /proc/net/nf_conntrack
```

Příklad

Teď se podíváme na mezilehlé zařízení (třeba firewall), na kterém chceme nastavit port pro DMZ. Předpokládejme, že eth0 je port do lokální sítě, eth1 bude WAN rozhraní a eth2 povede do DMZ:

`iptables -A FORWARD -i eth0 -o eth2 -m state -state NEW, ESTABLISHED, RELATED -j ACCEPT` povolíme pakety patřící do existujícího spojení, vztažené k existujícímu i navazující spojení (provoz z LAN do DMZ), tento směr prakticky není omezován (až na INVALID pakety), výstup příkazu `iptables -L -v` bude:

```
Chain FORWARD (policy DROP)
target   prot opt  in    out   source      destination
...
ACCEPT   all  -    eth0  eth2  anywhere    anywhere     state ESTABLISHED,RELATED
```

`iptables -A FORWARD -i eth1 -o eth2 -m state -state NEW, ESTABLISHED, RELADED -j ACCEPT` podobně pro provoz vedoucí z vnější sítě do DMZ, tady je potřeba mít povoleno navazování spojení, protože v DMZ jsou často webové a další servery, se kterými je potřeba navazovat spojení i zvnějšku

`iptables -A FORWARD -i eth2 -o eth0 -m state -state ESTABLISHED, RELATED -j ACCEPT` zařízení umístěná v DMZ jsou obecně považována za méně důvěryhodná, tj. navazovaná spojení nepustíme do LAN (pouze pakety patřící do navázaných nebo vztažených spojení)

`iptables -A FORWARD -i eth2 -o eth1 -m state -state ESTABLISHED, RELATED -j ACCEPT` podobně pro provoz z DMZ do vnější sítě, servery obvykle nemají co navazovat nová spojení

Pak samozřejmě potřebujeme zbývající pravidla pro řízení provozu na portech eth0 a eth1. U chainu FORWARD předpokládáme jako výchozí zásadu „zahazovat vše“ (DROP), tedy co jsme explicitně v pravidlech nepovolili nebo neurčili k zahození s ICMP odezvou, bude zahozeno bez odezvy.

Poznámka:

Podpora conntrack je důležitá i pro funkčnost SNAT/DNAT/MASQUERADE, protože je třeba překládat podle jedné konkrétní dvojice adres v rámci jednoho spojení (v rámci dalšího spojení se může používat jiná soukromá adresa).

Úkoly

1. Zjistěte, zda máte v jádře načtené alespoň některé moduly pro SPI. Pokud ano, zjistěte, zda se sledování stavů spojení používá v aktuálních pravidlech firewallu (zjistíte z výpisu pravidel). Pokud máte dostatečná přístupová oprávnění, vypište obsah souboru, ve kterém jsou uchovávána momentálně navázaná spojení.
2. Sestavte pravidlo, které bude zakazovat příchozí spojení (na běžné pracovní stanici).
3. Sestavte pravidlo, které bude akceptovat pakety z již navázaných spojení.

B.7.4 NAT a maškaráda

Překlad adres se provádí vždy co nejblíže síťovému rozhraní, tj. těsně před odesláním paketu do sítě (všechny ostatní tabulky se procházejí předtím), resp. okamžitě po příchodu paketu ze sítě (a až potom se provádějí další tabulky).

Kromě targetů ACCEPT, DROP a jiných můžeme využívat také targety pro překlad adres:

- *SNAT* (Source NAT) – v odchozím provozu překládá statickou adresu odesílatele na jinou (statickou) adresu viditelnou ve vnější síti
- *DNAT* (Destination NAT) – v příchozím provozu překládá statickou adresu příjemce na jinou, obvykle soukromou statickou adresu viditelnou pouze ve vnitřní síti, používá se také pro přeposílání příchozích paketů na proces, který má tyto pakety dále zpracovat
- *MASQUERADE* (maškaráda) – je to obdoba SNAT, ale používáme tehdy, když překládáme *dynamické* adresy (v odchozím provozu překládá soukromé adresy z vnitřní sítě na jednu adresu viditelnou vně naší sítě, obvykle adresu toho zařízení, na kterém je toto pravidlo)

Nejdřív bychom měli nastavit výchozí politiky:

`iptables -t nat -P OUTPUT -j ACCEPT` pokud něco odchází přes tabulku nat, necháme to jít (to je obvyklá výchozí politika pro tabulku nat)

`iptables -t nat -P PREROUTING -j ACCEPT` podobně pro chain PREROUTING, případné zahození necháme na jiných tabulkách, totéž bychom mohli provést pro POSTROUTING

Nejdřív se podíváme na *překlad statické adresy*. Nemusí být zrovna veřejná, ale neměla by se měnit a měli bychom ji znát, protože tuto adresu napevno zadáváme do příkazu.

`iptables -t nat -A POSTROUTING -o eth2 -j SNAT -to 193.168.210.80` zajistíme SNAT na odchozím provozu odcházejícím přes eth2: zdrojová adresa v záhlaví odcházejícího paketu bude zaměněna za uvedenou (statickou viditelnou venku), do překladové tabulky se uloží údaj o této komunikaci (původní zdrojová plus cílová) pro zpětný překlad.

Pokud v síti používáme DHCP, použijeme maškarádu. Maškaráda funguje podobně jako SNAT, ale využívá se výhradně u dynamických adres (ostatně, dynamickou adresu bychom stejně nemohli do SNAT pravidla naklepat), údaje pro překlad se uchovávají v jednom ze souborů v souborovém systému `/proc` (pro každé spojení se ukládá dvojice soukromá adresa v LAN plus vnější adresa, se kterou se komunikuje).

`iptables -t nat -A POSTROUTING -o eth2 -j MASQUERADE` zajistili jsme maškarádu (masquerade) na portu eth2, tj. vše, co odchází ven ze sítě na portu eth2, bude mít přeloženu zdrojovou adresu (obvykle soukromou, přidělenou DHCP serverem) na adresu routeru, automaticky je zajištěn záznam s uvedením portu pro zajištění zpětné konverzace (tj. PAT)

`iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE` totéž, jen port je jinak pojmenován (to je velice častý název u routerů kombinovaných s DSL modemem)

Pokud má také náš router (firewall) IP adresu viditelnou vně naší sítě také dynamickou (od DHCP serveru našeho ISP), je třeba pozměnit hodnotu v jednom ze souborů virtuálního soub. systému `/proc`:
`echo 7 > /proc/sys/net/ipv4/ip_dynaddr`

To je pro případ, že „vnější“ adresa, která je při maškarádě použita pro překlad místních soukromých adres, „vypadne“, je potřeba požádat o novou, která je pravděpodobně jiná. Pokud je dotyčný parametr takto nastaven, pak se v takové situaci zamění stará dynamická adresa za nově získanou ve všech zaznamenaných právě probíhajících spojeních.

 Vrátíme se ke statickým adresám. Mechanismus NAT se používá také pro *přeposílání příchozího provozu*:

`iptables -t nat -A PREROUTING -i eth2 -p tcp -dport 80 -j REDIRECT -to-port 3128` pokud přes rozhraní eth2 přijde paket z TCP portu č. 80 (velmi pravděpodobně pro protokol HTTP), přesměruj ho na port 3128; to se používá tehdy, když na portu 3128 naslouchá Squid konfigurovaný jako transparentní proxy, tj. dále prověřuje a zpracovává tento typ paketů

Všimněte si, že filtrování a NAT lze jednoduše kombinovat s překladem portů.

 Mechanismus NAT může sloužit pro *překlad jedné statické adresy na jinou statickou*, pak je potřeba zajistit ručně překlad v obou směrech a u každého směru uvést obě adresy. Výhodou tohoto řešení oproti jednoduchému SNAT a maškarádě je rychlejší zpracování (provoz není zdržován vyhledáváním dynamicky přidělených adres v dočasných souborech). Předpokládejme, že v DMZ máme server se soukromou (uvnitř viditelnou) adresou 10.201.51.2, chceme, aby byl vně sítě viditelný pod adresou 195.201.51.2. Je třeba zadat následující dvě pravidla, jedno pro provoz směrem od serveru ven, druhé pro opačný směr (zpětný překlad):

`iptables -t nat -A PREROUTING -d 10.201.51.2 -j DNAT -to-destination 195.201.51.2` nastavení DNAT, tedy překladu cílové adresy v paketu (provoz směrem do sítě), zamění soukromou adresu za uvedenou 195.201.51.2 v záhlaví přicházejícího paketu

`iptables -t nat -A POSTROUTING -s 195.201.51.2 -j SNAT -to-source 10.201.51.2` nastavení mechanismu SNAT, tedy překladu zdrojové adresy (provoz směrem ven)

Poznámka:

Na jednu věc bychom si měli dát pozor: mechanismus NAT funguje tak, že se pravidlo s přesměrováním adresy použije pouze na první paket spojení, a pouze první paket spojení prochází tabulkou *nat*. Další pakety patřící do téhož spojení (tj. pakety, které nejsou „první“), vůbec do tabulky *nat* nepřijdou, proto se jich tato pravidla netýkají.

Úkoly

1. Ve výpisu pravidel firewallu zjistěte, zda je nastaven překlad adres.
2. Sestavte pravidlo pro SNAT odchozího provozu na portu wan0, překládá se na statickou adresu 195.49.88.1.
3. Sestavte pravidlo pro maškarádu odchozího provozu na portu wan0.

B.7.5 Označování paketů

 Firewall NetFilter může do záhlaví paketů přidávat *značky* (marks), kterým rozumí jak jeho pravidla, tak i některé další nástroje (včetně příkazu `ip`). Přidávání značek je jednou z funkcí souvisejících

s tabulkou *mangle* (protože jde o změnu neadresního údaje v záhlaví) nebo *nat*, značku pak můžeme využívat jinde (typicky v tabulce *filter*).

Rozlišujeme dva druhy značek:

- *NetFilter mark* se nastavuje targetem MARK, je viditelný i mimo firewall, touto značkou se označují pakety, v terminologii `iproute2` se také nazývá *fwmark* (firewall mark),
- *connection mark* se nastavuje targetem CONNMARK, slouží pouze pro vnitřní potřebu firewallu; touto značkou se označuje *spojení* (pozor, ne paket), značku lze přenést i do záhlaví paketů (pak jde o NetFilter značky), nebo naopak.

Značky NetFilter lze do záhlaví paketu uložit pomocí příkazu firewallu `iptables` targetem MARK, využívat je lze buď v tabulkách firewallu pro další filtrování či úpravy, a nebo mechanismem `iproute2` (příkazem `ip rule`).

Ukážeme si několik příkazů s využitím značek:

`iptables -t mangle -A PREROUTING -i eth2 -j MARK -set-mark 2` pokud paket přišel z rozhraní eth2, dostane značku 2 (použijeme target MARK, protože označení je důsledkem, nikoliv podmínkou), platí jak pro pakety, které půjdou do chainu INPUT, tak i pro pakety mířící do chainu FORWARD v jiných tabulkách (na obrázku B.3 můžeme vidět, že označujeme prakticky hned na vstupu firewallu, pokud nepoužíváme tabulku raw)

`iptables -t filter -A INPUT -m mark -mark 2 -p tcp -dport 80 -j ACCEPT` chceme filtrovat pakety s podmínkou „pokud má paket značku 2“ (znamená: pokud má paket značku 2 a zároveň jde na TCP port 80, akceptuj), všimněte si odlišnosti práce s označením oproti předchozímu příkazu (zde je targetem ACCEPT)

`iptables -t nat -A PREROUTING -p tcp -dport 80 -j CONNMARK -set-mark 2` u příchozích spojení mířících na TCP port 80 (http) nastavíme značku spojení na 2 (protože je pravidlo v tabulce *nat*, provede se vždy jen pro první paket spojení, proto se často řadí právě do tabulky *nat*)

`iptables -t mangle -A PREROUTING -j CONNMARK -save-mark` značku spojení můžeme nastavit také takto: předpokládejme, že značka paketu byla nastavena již předtím (v předchozí tabulce), pak hodnota této značky paketu se uloží do značky spojení (tj. connmark := fwmark)

`iptables -t mangle -A POSTROUTING -j CONNMARK -restore-mark` opačný přenos informace, značka spojení se uloží do značky paketu (tj. fwmark := connmark)

Pokud firewall označil paket značkou fwmark, lze tuto značku dále zpracovávat během směrování, například takto:

`ip rule add fwmark 4 table tabVen prio 2021` přidali jsme nové směrovací pravidlo: pakety označené značkou 4 jsou směrovány podle tabulky `tabVen` (na obrázku B.3 na straně 329 si všimněte, kdy ke směrování vzhledem k chainům firewallu dochází, a kdy tedy je třeba paket označit, aby se podle značky dalo směrovat)

Další informace:

Další informace o této problematice (včetně příkladů) najdeme na

- http://www.linuxexpres.cz/praxe/ipp2p-kladivo-na-stahovace
- http://blog.khax.net/2009/12/01/multi-gateway-balancing-with-iptables/

- http://nerdboys.com/2006/05/05/conning-the-mark-multiwan-connections-using-iptables-mark-connmark-and-iproute2/
- http://home.regit.org/netfilter-en/netfilter-connmark/
- http://ornellas.apanela.com/dokuwiki/pub:firewall_and_adv_routing

B.7.6 Skripty

Je vhodné si vytvořit jeden nebo více skriptů pro konfiguraci firewallu, a to z několika důvodů: předně potřebujeme, aby veškerá nastavení, která provedeme, byla platná neustále (i po případném restartu), abychom měli zálohu pro případ poruchy a aby bylo možné občas sadu pravidel měnit či někdy vypínat (mazat pravidla a pak podle potřeby obnovit).

Další informace:

Ukázky několika typických konfiguračních skriptů s pravidly `iptables` najdeme například na

- http://www.faqs.org/docs/iptables/examplecode.html (to je první skript, na další se dostaneme klepáním na odkaz *Next* dole).
- http://www.pmoghadam.com/blog/categories/Scripts/Round-robin%20load%20balancing%20NAT.txt
- http://tldp.org/HOWTO/IP-Masquerade-HOWTO/firewall-examples.html
- http://tldp.org/HOWTO/pdf/VPN-Masquerade-HOWTO.pdf
- http://jk.frozen-doe.net/hack/iptables/

B.7.7 Uložení změn

Aby byla konfigurace nejen platná, ale také uložena pro načtení po vypnutí a zapnutí počítače, je nutné tuto konfiguraci uložit. Je velmi pravděpodobné, že distribuce, kterou máme, podporuje příkazy `iptables-save` a `iptables-restore`, které nám mohou velmi usnadnit práci s „hromadným“ startováním či ukončováním firewallu.[15]

`iptables-save` uloží obsah všech tabulek a chainů (tj. všechna pravidla) na standardní výstup, tj. používáme přesměrování, například `iptables-save > /etc/firerules`

`iptables-save -c -t filter` při použití přepínače `-c` se uloží také stav čítačů paketů a oktetů, přepínač `-t` zase omezí uložení pouze na zadanou tabulku (zde tabulku filter)

`iptables-restore` načte pravidla (tabulky, chainy) ze standardního vstupu (tj. opět musíme použít přesměrování nebo třeba rouru s příkazem `cat`, například `iptables-restore < /etc/firerules`)

`iptables-restore -c` načte také hodnoty čítačů

`iptables-restore -n` pokud už předtím byla nějaká pravidla ve firewallu aktivována, nebudou nově načtenými přepsána, nová pravidla se pouze přidají (bez uvedení parametru `-n` by po vykonání příkazu byla původně aktivní pravidla přepsána, odstraněna – flushed)

[15] Pozor – tím, že ukončíme program `iptables`, rozhodně nevypneme firewall, ten je totiž modulem jádra. Vpodstatě je firewall zapnutý po celou dobu, co je jeho modul zaveden v jádře, obdobou vypnutí by bylo vyprázdnění všech tabulek vnitřním příkazem `-F`.

Pokud tyto příkazy nenajdeme, pak musíme použít postup typický pro danou distribuci. Například v distribucích založených na RedHat (včetně Mandrivy) je to příkaz

```
/sbin/service iptables save
```

V distribucích založených na Debianu v souboru `/etc/default/iptables` nastavíme řádek

```
enable_iptables_initd=true
```

čímž uložení povolíme, samotné uložení se provede příkazem

```
/etc/init.d/iptables save_active
```

Poznámka:

Pokud chceme firewall deaktivovat (pojem „vypnout“ není zrovna správný), vyprázdníme jeho chainy: `iptables -F`
(případně zadáme tabulku a chain). Pokud ale něco takového opravdu chceme udělat, měli bychom si předem provést zálohu pravidel, třeba výše popsaným způsobem.

Není vždy nutné provádět konfiguraci a její uložení v textovém režimu. Existuje dokonce několik různých grafických rozhraní k programu `iptables`. V desktopových distribucích je najdeme celkem běžně, případně je možné si některý z nich stáhnout z repozitáře a nainstalovat.

V UNIXových systémech založených na BSD (například OpenBSD) se často setkáme s firewallem *PacketFilter* (starší byl IPFilter), obslužný program je `pfctl`.

Další informace:

Problematika firewallu v Linuxu je velmi rozsáhlá, sekce v těchto skriptech zdaleka nepokrývá všechny možnosti, které NetFilter nabízí. Další informace o `iptables`:

- http://www.netfilter.org/documentation/index.html
- http://security.maruhn.com/iptables-tutorial/x9125.html
- http://lartc.org/lartc.html
- http://www.frozentux.net/iptables-tutorial/iptables-tutorial.html
- http://www.root.cz/serialy/vse-o-iptables/
- http://www.webstep.net/media/PDF/iptables.pdf
- http://www.faqs.org/docs/iptables/
- http://ornellas.apanela.com/dokuwiki/pub:firewall_and_adv_routing
- http://www.asia-oss.net/aoss25_slides/TH_Sawangpong_Kernel_Development_and_Embedded_System.pdf

B.8 Vzdálený přístup

Pokud chceme přistupovat k některému uzlu sítě s Linuxem vzdáleně (z jiného počítače), lze použít několik nástrojů. Předně všechny takové uzly určitě podporují příkaz `telnet`. Ovšem to, že tento příkaz podporují, ještě neznamená, že ho můžeme použít. Především protokol *telnet* je na mnoha počítačích už během instalace zakázán, a to z bezpečnostních důvodů. Pokud přesto chceme tímto způsobem na daný

počítač přistupovat, musíme *telnet* povolit. Postup je v různých distribucích odlišný, popis najdeme zde:

- Ubuntu: http://www.cyberciti.biz/faq/ubuntu-linux-enable-telnet-service/
- RedHat: http://aplawrence.com/Linux/enable_telnet.html

atd. (vždy bychom se měli raději podívat na stránky naší distribuce).

Nicméně používání protokolu *telnet* se moc nedoporučuje, místo něho bychom měli raději používat protokol *SSH* (příkaz `ssh` pro zajištění spojení a pak příkaz `scp` pro zabezpečené kopírování s využitím protokolu SSH), který je taktéž dostupný na prakticky všech uzlech sítě a klienti jsou dostupní i pro Windows. SSH má výhody především v oblasti zabezpečení, šifruje se celá komunikace včetně zasílání přihlašovacích informací.

Nejdřív je potřeba nainstalovat na „vzdálený stroj“ SSH server, pak na stroj, ze kterého budeme na ten vzdálený přistupovat, nainstalujeme SSH klienta. Jako SSH server se v Linuxu běžně používá *OpenSSH Server*, který najdeme v repozitáři obvykle pod názvem `openssh-server` (záleží na konkrétní distribuci, prostě si prohlédneme obsah repozitáře v některém vhodném programu, třeba i v grafickém prostředí). Taky je možné, že už máme tento balíček nainstalovaný.

Co se týče klientů, na strojích s Linuxem už obvykle bývá nainstalován, ve Windows si můžeme stáhnout třeba program *puTTY* nebo *TTSSH*.

Další informace:

Postup konfigurace a používání SSH můžeme najít na těchto odkazech:

- http://wiki.ubuntu.cz/SSH
- http://www.xenocafe.com/tutorials/openssh_linux/redhat/openssh_linux_redhat.php (oproti návodu lze OpenSSH instalovat i jednodušeji z repozitářů)
- http://www.debianhelp.co.uk/ssh.htm

Pokud chceme jen stáhnout soubor z některého FTP nebo WWW serveru, nepotřebujeme výše uvedené nástroje. Obvykle nám stačí webový prohlížeč nebo některý grafický klient, ale někdy se hodí i nástroj pro použití v textovém režimu. V tomto směru je k nezaplacení příkaz `wget`.

Tento příkaz tedy slouží ke stahování souborů z webu (podporuje protokoly FTP, HTTP, HTTPS včetně proxy), dokonce zvládá i rekurzivní stahování (například HTML stránky včetně vnitřní struktury, vytvoří se offline verze stránek) a dokáže navázat na dřívější přerušené stahování. Jednoduché příklady použití:

`wget http://ftp.edUNIX.cz/pub/danix/Evolution2/dvd/DANIX-evol2-dvd-2007-11-04.iso`
: stahujeme DVD Linuxu Danix (pozor, vždy je třeba si zjistit adresu pro nejnovější verzi)

`wget -c ftp://adresa/dokument.pripona` naváže na předchozí přerušené stahování (příkaz je spouštěn v tomtéž pracovním adresáři jako při předchozím stahování)

Úkol

Zobrazte si manuálovou stránku příkazu `wget` a zjistěte, jak se dá spustit

- ve „spider“ módu, tedy dokument, který má zadán ke stáhnutí, nestáhne, ale jen zkontroluje, zda je na svém místě,

- bezpečně, aby se stahovalo s využitím některého šifrování (SSL nebo TLS),
- s rekurzivním stahováním.

Vyzkoušejte si stáhnutí některého dokumentu s příponou PDF z webu (případně ISO obsaz některé distribuce Linuxu, pokud máte dostatečně rychlé připojení bez limitů).

B.9 Další příkazy

V Linuxu máme k dispozici velmi mnoho dalších příkazů, které souvisejí s adresami na síti, například

`mtr` nástroj pro aktivní diagnostiku sítě (pozor, generuje provoz na síti, podle něho pak vyhodnocuje parametry sítě)

`lft` další diagnostický nástroj

`firewalk` bezpečnostní testovací nástroj, který zasíláním TCP nebo UDP paketů (lze určit parametrem) zjišťuje, které porty jsou na přeposílajících (forwarding) zařízeních v síti otevřené

`brctl` program pro administraci zařízení sloužícího jako ethernetový bridge, jako první parametr se zadává některý z příkazů pro konfiguraci mostu (lze například zobrazovat informace, pracovat s porty či s nastavením protokolu STP, apod.)

`tcpdump` analyzátor paketů (podobně jako Wireshark)

`ipcalc` jednoduchý program, který nám může pomoci orientovat se v IP adresách sítí/podsítí/uzlů („IP kalkulačka“), ukázka použití je v příkladu níže

`host` používá se podobně jako `nslookup` v neinteraktivním režimu, tedy IP adresu přeloží na název a naopak (oproti `nslookup` bere přednostně údaje ze souboru `/etc/hosts`, až když tam údaj nenajde, táže se DNS serveru)

Pro konfiguraci bezdrátové síťové karty lze použít nástroje `iwlist` a `iwconfig`.

Příklad

Ukážeme si práci se zajímavým příkazem `ipcalc`, který podle zadané IP adresy podsítě vypíše masku, inverzní masku, rozsah adres uzlů v zadané podsíti, broadcast adresu a počet možných uzlů v síti. Příkaz chceme spustit na stanici s nainstalovaným Ubuntu. Tato distribuce sice dotyčný nástroj ve výchozí instalaci neobsahuje, ale dokáže poradit, jakým způsobem máme nástroj nainstalovat:

```
sarka@notebook:~$ ipcalc 10.1.0.0/16

The program 'ipcalc' is currently not installed.  You can install it by typing:
sudo apt-get install ipcalc
```

Takže program není nainstalován. Ale byli jsme upozorněni, že se nachází v repozitáři (jsme připojeni k Internetu, máme přístup do repozitářů Ubuntu), máme před očima příkaz, kterým program nainstalujeme. Protože známe administrátorské heslo, není to problém (všimněte si příkazu `sudo`, který v Ubuntu a některých dalších distribucích slouží k získání vyšších přístupových oprávnění).

```
sarka@notebook:~$ sudo apt-get install ipcalc
```

```
[sudo] password for sarka: **************
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  python-pylibacl python-brasero rdiff-backup python-pyxattr
Use 'apt-get autoremove' to remove them.
The following NEW packages will be installed:
  ipcalc
0 upgraded, 1 newly installed, 0 to remove and 378 not upgraded.
Need to get 26.6kB of archives.
After this operation, 131kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu/ maverick/universe ipcalc all 0.41-2 [26.6kB]
Fetched 26.6kB in 0s (72.0kB/s)
Selecting previously deselected package ipcalc.
(Reading database ... 162465 files and directories currently installed.)
Unpacking ipcalc (from .../archives/ipcalc_0.41-2_all.deb) ...
Processing triggers for man-db ...
Setting up ipcalc (0.41-2) ...
```

Instalace je hotova, restartovat netřeba (nejsme přece ve Windows), tedy konečně můžeme spustit příkaz, jako parametr zadáme IP adresu (pod)sítě včetně délky prefixu:

```
sarka@notebook:~$ ipcalc 10.1.0.0/16

Address:   10.1.0.0             00001010.00000001. 00000000.00000000
Netmask:   255.255.0.0 = 16     11111111.11111111. 00000000.00000000
Wildcard:  0.0.255.255          00000000.00000000. 11111111.11111111
=>
Network:   10.1.0.0/16          00001010.00000001. 00000000.00000000
HostMin:   10.1.0.1             00001010.00000001. 00000000.00000001
HostMax:   10.1.255.254         00001010.00000001. 11111111.11111110
Broadcast: 10.1.255.255         00001010.00000001. 11111111.11111111
Hosts/Net: 65534                 Class A, Private Internet
```

Existují různé nástroje pro testování zabezpečení sítě, které nejsou součástí běžných distribucí, ale rozhodně stojí za to si je stáhnout a používat. Například *SATAN*[16] (System Administrators Tool for Analyzing Networks) prochází celou síť a provádí bezpečnostní testy. Je oblíbený jak u administrátorů, tak i u hackerů, i když každý z nich má samozřejmě jinou motivaci.

Další informace:

- Zajímavý tutoriál o sítích v Linuxu najdeme na adrese http://linux-ip.net/html/part-concepts.html.
- http://www.linuxsoft.cz/wifi/
- http://www.root.cz/serialy/linux-jako-internetova-gateway/
- http://www.root.cz/clanky/luci-webove-rozhrani-pro-openwrt/
- Čečák, O. *Linux v příkazech – práce s Wi-Fi* [online]. Dostupné na http://www.linuxsoft.cz/article.php?id_article=1351

[16] Ke stažení na http://www.porcupine.org/satan/.