



Slezská univerzita v Opavě
Filozoficko-přírodovědecká fakulta v Opavě

Šárka Vavrečková

Skripta do předmětu

Počítačová síť a internet — cvičení —

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

Opava

Poslední aktualizace: 12. března 2020

Anotace: Tato skripta jsou určena pro studenty předmětu *Počítačová síť a internet* na Ústavu informatiky Slezské univerzity v Opavě. Obsahují pouze učební látku pro cvičení, pro přednášky najdete další skripta. V předmětu se zabýváme prostředky a postupy používanými v počítačových sítích.

Počítačová síť a internet, skripta pro cvičení

RNDr. Šárka Vavrečková, Ph.D.


Dostupné na: <http://vavreckova.zam.slu.cz/pocsit.html>

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

Sázeno v systému L^AT_EX

Předmluva

Co najdeme v těchto skriptech







 *Rychlý náhled:* Tato skripta jsou určena pro studenty informatických předmětů na Slezské univerzitě v Opavě. Obsahují látku vyučovanou na cvičeních předmětu *Počítačová síť a Internet*, ve kterém se zabýváme (jak název napovídá) počítačovými sítěmi. Tato skripta jsou doplněním skript pro přednášky z téhož předmětu (dostupná na stejném místě).



Probíraná témata s určitými výjimkami sledují obsahovou náplň přednášek. Ne zcela, protože každé téma probírané na přednáškách má trochu jiné množství souvisejících praktických úkolů. Tedy po procvičení si číselných soustav se budeme zabývat

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:

-  *Rychlý náhled* (skript, kapitoly), ve kterém se dozvíme, o čem to bude.
-  *Klíčová slova* kapitoly.
-  *Cíle studia* pro kapitolu nám řeknou, co nového se v dané kapitole naučíme.
-  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 0.1

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 0.1

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 Číselné soustavy	1
1.1 O číselných soustavách, zejména té dekadické	1
1.2 Binární soustava	2
1.3 Šestnáctková soustava	6
1.4 Bitová aritmetika	9
1.5 Jednotky a násobky	11
2 Počítačová síť na hostitelském zařízení	12
2.1 Síťová karta	12
2.2 Textový režim	13
2.3 Hostname	14
2.4 Adresy	15
2.4.1 Hardwarové adresy	15
2.4.2 IP adresy	20
3 Standardy	24
3.1 RFC dokumenty	24
3.2 ITU-T	27
4 Lokální sítě – Ethernet	30
4.1 Wireshark	30
4.2 Zařízení v lokální síti	35
4.3 Tabulka MAC adres na switchi	40
4.4 Kabely	43
4.4.1 Metalická kabeláž	43
4.4.2 Optická kabeláž	46
4.4.3 Další typy kabelů	49
4.5 Zakončení kabelu	50
4.5.1 Křížení	50
4.5.2 Krimpování konektoru na přímý UTP kabel	52
4.5.3 Testování	54
4.5.4 Krimpování konektoru na křížený UTP kabel	56
4.5.5 Kroucená dvojlinka se dvěma páry vodičů	57
4.5.6 Otočený a konzolový kabel	57
4.5.7 Osazení UTP kabelu do zásuvky	59
4.6 Strukturovaná kabeláž	60
4.6.1 Rozvaděč	60
4.6.2 Horizontální kabeláž	61


4.6.3	Pátevní síť	63
4.6.4	Strukturovaná kabeláž jako celek	65
5	Packet Tracer a konfigurace síťových zařízení Cisco	66
5.1	Instalace a zprovoznění	66
5.2	Úvod do práce v Packet Traceru	66
5.3	Konfigurace koncového zařízení	70
5.4	Monitorování paketů	71
5.5	Konfigurace zařízení Cisco	72
5.5.1	Jak se připojit	72
5.5.2	Pracovní módy	73
5.5.3	Konfigurace switche	75
5.5.4	Konfigurace routeru	78
5.5.5	Telnet a SSH	80
6	Co se děje na síťové a transportní vrstvě	82
6.1	Protokol IPv4	82
6.1.1	Maska a prefix adresy	82
6.1.2	Zjištění adresy, masky a brány	83
6.1.3	IPv4 pakety	84
6.1.4	Fragmentace paketu	85
6.1.5	Složení fragmentovaného paketu	88
6.2	Protokol IPv6	90
6.2.1	Adresy podle IPv6	90
6.2.2	IPv6 pakety	91
6.3	Protokol ICMP a příkazy, které ho používají	93
6.3.1	Testování dosažitelnosti	93
6.3.2	Zjišťování cesty	95
6.4	Protokoly na transportní vrstvě	96
6.4.1	Protokol TCP a navazování spojení	96
6.4.2	Protokol UDP a jednoduchá rychlá komunikace	97
7	Aplikační protokoly	98
7.1	Mechanismus DNS	98
7.2	Protokol HTTP	100
7.3	DHCP a získání IP adresy	100
7.4	Telnet a SSH	101
7.5	Statistiky síťových protokolů	101
8	Internetworking a průzkum paketů	103
8.1	Objevování sousedů	103
8.2	Práce s IPv4 adresami	104
8.2.1	Subnetting	104
8.2.2	VLSM	113
8.3	Směrování	117
9	Bezdrátové sítě	118
9.1	Wi-fi rámec	118
9.2	Průzkum frekvenčního spektra	120
	Literatura	123

Kapitola 1

Číselné soustavy

 *Rychlý náhled:* V oblasti počítačových sítí často používáme čísla v binární a hexadecimální soustavě, především při práci s nejrůznějšími typy adres. Proto bude naším prvním úkolem naučit se pracovat v binární a hexadecimální číselné soustavě.

 *Klíčová slova:* Dekadická (desítková) číselná soustava, binární (dvojková) číselná soustava, hexadecimální (šestnáctková) číselná soustava, osmičková číselná soustava, bitová aritmetika, byte, oktet, nibble

 *Cíle studia:* Po prostudování této kapitoly budete umět reprezentovat číslo v různých číselných soustavách a převádět čísla mezi různými číselnými soustavami.

1.1 O číselných soustavách, zejména té dekadické

Pro většinu lidí je přirozené počítat v dekadické (desítkové) soustavě, protože člověk má (většinou) deset prstů. Jenže v případě, že potřebujeme komunikovat se strojem sofistikovaněji než „běžný uživatel“, musíme svou práci s čísly alespoň částečně přizpůsobit tomuto stroji. Takže kde a jak se u počítačů a jiných výpočetních zařízení s jinými číselnými soustavami setkáváme:

- binární (dvojková) soustava – především při práci s adresami, budeme potřebovat zejména v tématu o internetworkingu,
- hexadecimální (šestnáctková) soustava – některé adresy se zapisují v této soustavě, například fyzické nebo adresy pro IPv6,
- oktálová (osmičková) soustava – například při práci s přístupovými oprávněními.

U těchto soustav probereme reprezentaci čísla v dané soustavě, převody do/z jiných soustav a v binární soustavě i některé operace.

Abychom měli na co navázat, vzpomeňme si, jak to funguje v dekadické soustavě.

Dekadická soustava je *poziční* (na rozdíl od zápisu římskými číslicemi), tedy konkrétní význam číslice záleží na pozici v čísle. Rozvinutý zápis dekadického čísla tuto vlastnost využívá, jak vidíme v následujícím příkladu.

**Příklad 1.1**

Vypíšeme rozvinutý zápis dekadického čísla 8 245:

$$\begin{aligned} 80\,245 &= 8 \times 10\,000 + 0 \times 1000 + 2 \times 100 + 4 \times 10 + 5 \times 1 \\ &= 8 \times 10^4 + 0 \times 10^3 + 2 \times 10^2 + 4 \times 10^1 + 5 \times 10^0 \end{aligned}$$

Takže především bychom měli zvládat mocniny čísla 10.



Všimněte si, že $10^0 = 1$. Obecně platí $\forall x > 0: x^0 = 1$.

**Úkol**

Následující dekadická čísla zapíšte v rozvinutém zápisu s využitím mocnin čísla 10:

72, 905, 15 550



Pokud potřebujeme zdůraznit, že dané číslo x je v dekadické (a ne jiné) soustavě, pak toto číslo zapíšeme takto: $(x)_{10}$, například $(80\,245)_{10}$.



Další „samozřejmou“ věcí, na kterou je však třeba zde upozornit, je, kdy vlastně dochází k navýšení řádu dekadického čísla (tedy kdy konkrétně se zvyšuje počet číslic): jestliže přičteme jedničku k číslu skládajícímu se pouze z číslic s nejvyšší hodnotou (v dekadické číslice 9), pak dochází k přetečení do vyššího řádu, přičemž se radikálně mění skladba číslic čísla. Například:

$$9 + 1 = 10, \quad 99 + 1 = 100, \quad 999 + 1 = 1000, \text{ atd.}$$

Tedy pokud k číslu, jehož všechny číslice jsou nastaveny na „maximum“ pro danou soustavu (u nás 9), přičteme jedničku, pak navýšíme řád (číslo bude o jednu číslici delší) a na začátku bude jednička následovaná samými nulami.

A opačně: pokud od čísla ve tvaru jednička následovaná nulami odečteme jedničku, snížíme řád a nastavíme všechny číslice na „maximum“. Toto pravidlo platí pro všechny soustavy, nejen desítkovou.

1.2 Binární soustava

V dekadické soustavě používáme mocniny čísla 10, v binární soustavě potřebujeme mocniny čísla 2. Protože je budeme používat poměrně často, měli bychom alespoň nežší mocniny čísla 2 znát nazpaměť.

**Úkol**

Zapamatujte si:

$2^0 = 1$	$2^3 = 8$	$2^6 = 64$	$2^9 = 512$
$2^1 = 2$	$2^4 = 16$	$2^7 = 128$	$2^{10} = 1024$
$2^2 = 4$	$2^5 = 32$	$2^8 = 256$	



Binární číslo je číslo složené z číslic nabývajících hodnoty 0 nebo 1. V zápisu je často třeba dát na vědomí, že se jedná o binární číslo a ne dekadické (protože i mnohá dekadická čísla se skládají pouze z číslic s hodnotou 0 nebo 1), takže pak píšeme $(x)_2$, například $(1001011)_2$.

Při převodu binárního čísla na dekadické jednoduše využijeme rozvinutý zápis binárního čísla.



Příklad 1.2 (Převod z binární soustavy do dekadické)

$$\begin{aligned}
 (10011101)_2 &= 1 \times 2^7 + 0 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 \\
 &= 1 \times 128 + 0 \times 64 + 0 \times 32 + 1 \times 16 + 1 \times 8 + 1 \times 4 + 0 \times 2 + 1 \times 1 \\
 &= 128 + 0 + 0 + 16 + 8 + 4 + 0 + 1 \\
 &= (157)_{10}
 \end{aligned}$$

$$\begin{aligned}
 (1110100)_2 &= 1 \times 2^6 + 1 \times 2^5 + 1 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 0 \times 2^0 \\
 &= 1 \times 64 + 1 \times 32 + 1 \times 16 + 0 \times 8 + 1 \times 4 + 0 \times 2 + 0 \times 1 \\
 &= 64 + 32 + 16 + 0 + 4 + 0 + 0 \\
 &= (116)_{10}
 \end{aligned}$$




Postup (Převod z binární soustavy do dekadické)

Je dáno číslo v binární soustavě.

- Spočítáme cifry tohoto čísla (počet pozic čísla) – tento počet označíme n .
V předchozím příkladu to je u prvního čísla 8, u druhého 7.
- Postupujeme zleva od nejvyššího řádu – číslici na daném místě (0 nebo 1) vynásobíme mocninou čísla 2 s exponentem o 1 menším než je řád této číslice. První číslici tedy vynásobíme číslem 2^{n-1} , druhou číslem 2^{n-2} , atd., poslední číslici vynásobíme číslem 2^0 , tedy jedničkou.
V předchozím příkladu u prvního čísla vynásobíme první číslici číslem 2^7 , u druhého čísla vynásobíme první číslici číslem 2^6 .
- Tuto řadu čísel sečteme, výsledek je totéž číslo, ale zapsané v dekadické soustavě.



 Zapamatujte si, že poslední číslice binárního čísla určuje, zda číslo v dekadickém zápisu bude sudé nebo liché.




Úkol

Následující binární čísla převed'te do dekadické soustavy.

$(10)_2$, $(11)_2$, $(1001)_2$, $(1111)_2$, $(10000)_2$, $(11111111)_2$, $(100000000)_2$.



 Ve výsledcích z předchozího úkolu si všimněte těchto věcí:

- čísla zapsaná binárně jako $(1111)_2$ a $(10000)_2$ se liší o 1, konkrétně platí $(1111)_2 + 1 = (10000)_2$
- čísla zapsaná binárně jako $(11111111)_2$ a $(100000000)_2$ se liší o 1, konkrétně platí $(11111111)_2 + 1 = (100000000)_2$

Je to tentýž princip jako u dekadické soustavy – pokud k číslu, jehož všechny číslice nabývají maximální hodnoty pro danou číselnou soustavu (zde 1), přičteme jedničku, získáme číslo začínající číslicí 1, za kterou následují jen nuly, přičemž řád čísla (počet číslic) se zvýší o 1.

**Postup**

Binární soustava se používá při uložení celého čísla do paměti počítače, protože počítač pracuje s bity (bit nabývá jedné ze dvou hodnot – 0 nebo 1). Takže když systém ukládá číslo, které mu zadáme v dekadickém tvaru, musí toto číslo nejdřív převést do binární soustavy a pak teprve uložit. Platí, že pro každý řád čísla potřebujeme jeden bit, takže například binární číslo o osmi rádech (tedy zapsané osmi binárními číslicemi) bude potřebovat minimálně osm pozic, a tedy minimálně osm bitů.



Každá proměnná či konfigurační hodnota (ať už při běžném programování nebo při konfiguraci síťových zařízení) má přiřazen svůj datový typ. Tento datový typ v paměti zabírá určitý počet bitů, který nás omezuje v maximálních hodnotách čísel, která do této proměnné či hodnoty budeme chtít uložit.

**Příklad 1.3**

1 Byte (oktet) je 8 bitů, a tedy maximální hodnota je následující:

$$(11111111)_2 = 128 + 64 + 32 + 16 + 8 + 4 + 2 + 1 = 255$$

Jednodušeji to vypočteme takto:

$$(100000000)_2 - 1 = 256 - 1 = 255$$

Z toho vyplývá, že do 8 bitů můžeme uložit číslo s hodnotou v rozsahu 0–255.

**Úkol**

Určete rozsah hodnot pro číslo uložené v 16 bitech.



Při převodu čísla z dekadické soustavy do binární můžeme použít přesně opačný postup – číslo rozložíme na součet mocnin čísla 2. Jenže „ruční“ vyhledávání použitelných mocnin čísla 2 je zbytečně pracné, proto tento úkol převedeme na dělení dvěma.

**Příklad 1.4 (Převod z dekadické soustavy do binární)**

Převedeme do binární soustavy číslo $(57)_{10}$. Budeme toto číslo opakovaně *celočíselně dělit dvěma* (tak dlouho, dokud výsledkem nebude 0), vždy si poznamenáme zbytek. Zbytek po dělení dvěma je vždy buď 0 nebo 1, což znamená, že u zbytků zůstáváme v binární soustavě.

$$\begin{array}{rcll} 57 & : & 2 & = 28 \text{ zbytek } 1 \\ 28 & : & 2 & = 14 \text{ zbytek } 0 \\ 14 & : & 2 & = 7 \text{ zbytek } 0 \\ 7 & : & 2 & = 3 \text{ zbytek } 1 \\ 3 & : & 2 & = 1 \text{ zbytek } 1 \\ 1 & : & 2 & = 0 \text{ zbytek } 1 \end{array}$$



Výsledky jednotlivých výpočtů pro nás nejsou důležité, zaměříme se na vypočtené zbytky a poskládáme je *od konce*, jak naznačuje šipka. Takže výsledek je $(111001)_2$.

Podobně zjistíme binární tvar pro dekadické číslo $(438)_{10}$:

438	:	2	=	219	zbytek 0	
219	:	2	=	109	zbytek 1	
109	:	2	=	54	zbytek 1	
54	:	2	=	27	zbytek 0	
27	:	2	=	13	zbytek 1	
13	:	2	=	6	zbytek 1	
6	:	2	=	3	zbytek 0	
3	:	2	=	1	zbytek 1	
1	:	2	=	0	zbytek 1	

Proto $(438)_{10} = (110110110)_2$.



Postup (Převod z dekadické soustavy do binární)

Je dáno číslo v dekadické soustavě.

- Toto číslo *celočíslně vydělíme* dvěma, zbytek po dělení zapíšeme do výsledku zcela vpravo.
- Výsledek dělení opět *celočíslně vydělíme* dvěma, zbytek po dělení zapíšeme vlevo od předchozího získaného výsledku (před něj).
- Pokračujeme stejně tak dlouho, dokud jako výsledek nevyjde nula. Zbytek po dělení pořád přepisujeme zleva k výsledku.



Příklad 1.5

Pokud číslo dokážeme z paměti rozložit na součet mocnin čísla 2, pak se samozřejmě nemusíme „mořit“ s dělením a přepisováním zbytků. Například

- $26 = 16 + 8 + 2 = 2^4 + 2^3 + 2^1$, takže $(26)_{10} = (11010)_2$
- $36 = 32 + 4 = 2^5 + 2^2$, takže $(36)_{10} = (10010)_2$
- $131 = 128 + 2 + 1 = 2^7 + 2^1 + 2^0$, takže $(131)_{10} = (10000011)_2$
- $192 = 128 + 64 = 2^7 + 2^6$, takže $(192)_{10} = (11000000)_2$



Úkol

Převed'te tato čísla z dekadické do binární soustavy:

$(12)_{10}$, $(25)_{10}$, $(34)_{10}$, $(65)_{10}$, $(160)_{10}$, $(255)_{10}$, $(400)_{10}$, $(937)_{10}$, $(1000)_{10}$.



Definice 1.1

Shrňme si pojmy označující oblasti paměti zabírající stanovený počet bitů:

- *Nibble* zabírá 4 bity
- *Word* (slovo) zabírá 16 bitů
- *Byte*, resp. *oktet* zabírá 8 bitů
- *Double Word* (dvojité slovo) má 32 bitů



1.3 Šestnáctková soustava

V hexadecimální (šestnáctkové) soustavě potřebujeme šestnáct různých číslic. Prvních deset 0–9 prostě využijeme z dekadické soustavy, s dalšími šesti si pomůžeme abecedou. Použijeme

A ... 10	D ... 13
B ... 11	E ... 14
C ... 12	F ... 15

Takže číslice pro vyjádření čísla v šestnáctkové soustavě jsou $0, \dots, 9, A, \dots, F$.

Převod reprezentace čísla z šestnáctkové soustavy do dekadické se moc často neprovádí, ale je dobré vědět, jak by to šlo – například podobně jako u jiných soustav, tedy pomocí rozvinutého zápisu, s využitím mocnin čísla 16.



Příklad 1.6 (Převod z šestnáctkové soustavy do dekadické)

$$\begin{aligned}
 (A2B)_{16} &= 10 \times 16^2 + 2 \times 16^1 + 11 \times 16^0 \\
 &= 10 \times 256 + 2 \times 16 + 11 \times 1 \\
 &= 2560 + 32 + 11 \\
 &= (2603)_{10}
 \end{aligned}$$



Také u opačného převodu (z šestnáctkové reprezentace na dekadickou) se dá inspirovat u binární soustavy (tj. opakovaně bychom celočíselně dělili číslem 16, zbytky po dělení bychom zapisovali zprava doleva). Později si ukážeme, že oba směry převodu jdou i jednodušeji, přes binární soustavu. Mocniny čísla 16 se totiž zase až tak dobře nepamatují...

Zaměříme se nyní na vztah mezi hexadecimální a binární soustavou.



Při těchto převodech využíváme velice důležitého pozičního vztahu – jedna hexadecimální číslice zcela přesně reprezentuje čtyři binární číslice. Proč tomu tak je? Protože $(F)_{16} = (1111)_2$ (maximální hodnota čísla vyjádřitelného jednou hexadecimální číslicí je rovna maximální hodnotě čísla vyjádřitelného čtyřmi binárními číslicemi).

Jak jsme se dozvěděli z výše uvedené definice, čtyři bity (tedy čtyři binární číslice) nazýváme *nibble*, a tedy jedna hexadecimální číslice představuje jeden nibble. Hodnoty, které se dají uložit do jednoho nibblu, bychom měli znát zpaměti ve všech třech soustavách.



Úkol

Zapamatujte si nebo se naučte rychle převést pomocí mocnin čísla 2:

$(0000)_2 = (0)_{16} = (0)_{10}$	$(1000)_2 = (8)_{16} = (8)_{10}$
$(0001)_2 = (1)_{16} = (1)_{10}$	$(1001)_2 = (9)_{16} = (9)_{10}$
$(0010)_2 = (2)_{16} = (2)_{10}$	$(1010)_2 = (A)_{16} = (10)_{10}$
$(0011)_2 = (3)_{16} = (3)_{10}$	$(1011)_2 = (B)_{16} = (11)_{10}$
$(0100)_2 = (4)_{16} = (4)_{10}$	$(1100)_2 = (C)_{16} = (12)_{10}$
$(0101)_2 = (5)_{16} = (5)_{10}$	$(1101)_2 = (D)_{16} = (13)_{10}$
$(0110)_2 = (6)_{16} = (6)_{10}$	$(1110)_2 = (E)_{16} = (14)_{10}$
$(0111)_2 = (7)_{16} = (7)_{10}$	$(1111)_2 = (F)_{16} = (15)_{10}$



**Příklad 1.7 (Převod z šestnáctkové soustavy do binární)**

Naším úkolem je převést číslo $(A2B)_{16}$ do binární soustavy. Je to totéž číslo jako v příkladu na začátku této sekce. Při převodu jednoduše využijeme vztahy z předchozího úkolu. Hexadecimální číslo rozložíme na jednotlivé číslice a každou číslici převedeme na binární nibble.

$$\begin{array}{c|c|c} A & 2 & B \\ \hline 1010 & 0010 & 1011 \end{array} \Rightarrow (A2B)_{16} = (1010\,0010\,1011)_2$$

Podobně převedeme číslo $(4F50D)_{16}$.

$$\begin{array}{c|c|c|c|c} 4 & F & 5 & 0 & D \\ \hline 0100 & 1111 & 0101 & 0000 & 1101 \end{array} \Rightarrow (4F50D)_{16} = (0100\,1111\,0101\,0000\,1101)_2$$

$$\text{nulu na začátku odstraníme:} \Rightarrow (4F50D)_{16} = (100\,1111\,0101\,0000\,1101)_2$$

A teď číslo $(A0D2EE)_{16}$.

$$\begin{array}{c|c|c|c|c|c} A & 0 & D & 2 & E & E \\ \hline 1010 & 0000 & 1101 & 0010 & 1110 & 1110 \end{array} \Rightarrow (A0D2EE)_{16} = (1010\,0000\,1101\,0010\,1110\,1110)_2$$

**Postup (Převod z šestnáctkové soustavy do binární)**

Je dáno číslo v šestnáctkové soustavě.

- Toto číslo rozdělíme na jednotlivé hexadecimální číslice.
- Číslice převedeme do binární soustavy, zachováme počet čtyř binárních číslic (nibble).
- Výsledek zřetězíme. Nuly na začátku výsledku můžeme ignorovat.

**Úkol**

Následující čísla v šestnáctkové soustavě převed'te na binární reprezentaci:

$$(2A)_{16}, (F77F)_{16}, (345)_{16}, (BB10)_{16}, (39AE4)_{16}.$$

**Příklad 1.8 (Převod z binární soustavy do šestnáctkové)**

Převedeme binární číslo $(11\,0011\,0100\,1100)_2$ do hexadecimálního tvaru. Použijeme přesně opačný postup k tomu z předchozího příkladu – číslo rozdělíme na čtveřice binárních číslic (od konce), skupinu nejvíc vlevo doplníme nulami na čtveřici a pak postupně převedeme na hexadecimální číslice.

$$\begin{array}{c|c|c|c} 11 & 0011 & 0100 & 1100 \\ \hline 0011 & 0011 & 0100 & 1100 \\ \hline 3 & 3 & 4 & C \end{array} \Rightarrow (11\,0011\,0100\,1100)_2 = (334C)_{16}$$

Podobně převedeme číslo $(110\,1001\,1110\,0000)_2$.

$$\begin{array}{c|c|c|c} 110 & 1001 & 1110 & 0000 \\ \hline 0110 & 1001 & 1110 & 0000 \\ \hline 6 & 9 & E & 0 \end{array} \Rightarrow (110\,1001\,1110\,0000)_2 = (69E0)_{16}$$



**Postup (Převod z binární soustavy do šestnáctkové)**

Je dáno číslo v binární soustavě.

- Toto číslo rozdělíme na čtveřice binárních číslic, postupujeme zprava (odzadu).
- Skupina nejvíc vlevo (na začátku) může mít méně než čtyři číslice – pak ji doplníme zleva nulami (nebo si je domyslíme).
- Jednotlivé čtveřice převedeme do šestnáctkové soustavy.
- Řadu hexadecimálních číslic zřetězíme.

**Úkol**

Převed'te do šestnáctkové soustavy tato čísla v binárním zápisu:

$(100001)_2$, $(11)_2$, $(11001101111)_2$, $(101010101010)_2$, $(110011001100000)_2$.



Jeden Byte (oktet), tedy osm bitů, zapíšeme buď osmi binárními číslicemi nebo dvěma hexadecimálními číslicemi. Toho se v technice využívá poměrně často.

A teď se vraťme k dekadické soustavě. Výše bylo poznamenáno, že existuje jednodušší způsob jak převádět číslo z hexadecimální soustavy do dekadické a naopak. Jde to přes binární soustavu.

**Příklad 1.9 (Převod z šestnáctkové soustavy do dekadické přes binární)**

Převedeme číslo $(42B)_{16}$ do dekadické soustavy, ale tentokrát půjdeme přes binární soustavu. Jednoduše jednotlivé hexadecimální číslice převedeme do binární soustavy, zřetězíme a pak binární číslo převedeme do dekadické (přičemž doufáme, že tam bude „co nejméně jedniček“).

$$\begin{array}{c|c|c} 4 & 2 & B \\ \hline 0100 & 0010 & 1011 \end{array} \Rightarrow (72B)_{16} = (100\,0010\,1011)_2$$

Teď binární číslo $(100\,0010\,1011)_2$ převedeme na dekadické (násobení nulou nebudeme psát):

$$\begin{aligned} (100\,0010\,1011)_2 &= 1 \times 2^{10} + 1 \times 2^5 + 1 \times 2^3 + 1 \times 2^1 + 1 \times 2^0 \\ &= 1 \times 1024 + 1 \times 32 + 1 \times 8 + 1 \times 2 + 1 \times 1 \\ &= (1067)_{10} \end{aligned}$$

Z toho vyplývá, že $(42B)_{16} = (1067)_{10}$.

**Úkol**

Převed'te do dekadické soustavy tato hexadecimální čísla (přes binární soustavu):

$(B)_{16}$, $(10)_{16}$, $(2F)_{16}$, $(11A)_{16}$.

**Příklad 1.10 (Převod z dekadické soustavy do šestnáctkové přes binární)**

Převedeme číslo $(892)_{10}$ do šestnáctkové soustavy, a to přes binární soustavu.

892	:	2	=	446	zbytek	0
446	:	2	=	223	zbytek	0
223	:	2	=	111	zbytek	1
111	:	2	=	55	zbytek	1
55	:	2	=	27	zbytek	1
27	:	2	=	13	zbytek	1
13	:	2	=	6	zbytek	1
6	:	2	=	3	zbytek	0
3	:	2	=	1	zbytek	1
1	:	2	=	0	zbytek	1



Takže mezistupněm je číslo $(11\ 0111\ 1100)_2$, které zbývá převést na šestnáctkové.

$$\begin{array}{c|c|c} 0011 & 0111 & 1100 \\ 3 & 7 & C \end{array} \Rightarrow (11\ 0111\ 1100)_2 = (37C)_{16}$$

Z toho vyplývá, že $(892)_{10} = (37C)_{16}$.



Úkol

Převeďte do šestnáctkové soustavy tato dekadická čísla, využijte převod přes binární soustavu: $(25)_{10}$, $(128)_{10}$, $(160)_{10}$.



Poznámka:


S šestnáctkovým zápisem čísla se setkáváme například:

- fyzické síťové adresy (MAC adresy), dále IPv6 adresy,
- (nejen) v HTML se takto často zapisuje reprezentace barvy.



1.4 Bitová aritmetika

Bitové (tedy binární) operace jsou v oblasti počítačových sítí a obecně v informatice velmi důležité. V praxi se s nimi setkáváme například při práci s adresami. Dále se budeme zabývat logickými operacemi na číslech v binární soustavě. Jejich specifikem je, že v podstatě zachovávají princip homomorfismu, a tedy je můžeme uplatňovat zvláště na jednotlivé bity daného binárního čísla.

 *Negace* je nejjednodušší operací na binárních číslech – u každé číslice převrátíme její hodnotu.




Příklad 1.11

Negujeme binární číslo 10011101.

$$\begin{array}{r} \text{Původní číslo:} \quad 1 \mid 0 \mid 0 \mid 1 \mid 1 \mid 1 \mid 0 \mid 1 \\ \text{Negované číslo:} \quad 0 \mid 1 \mid 1 \mid 0 \mid 0 \mid 0 \mid 1 \mid 0 \end{array}$$

$$\text{NOT } (10011101) = 01100010$$



 *Logický součet* je vlastně disjunkce. Opět provádíme pozičně po jednotlivých bitech (binárních číslicích).




Příklad 1.12

Jsou dána binární čísla 10110010 a 11110000. Vypočteme jejich logický součet.

První číslo:	1	0	1	1	0	0	1	0
Druhé číslo:	1	1	1	1	0	0	0	0
Logický součet (OR):	1	1	1	1	0	0	1	0

(10110010) OR (11110000) = 11110010



 *Logický součin* je konjunkce. Na dané pozici ve výsledku bude 1 právě tehdy, když na téže pozici je v obou původních číslech číslice 1.




Příklad 1.13

Jsou dána binární čísla 10110010 a 11110000. Vypočteme jejich logický součin.

První číslo:	1	0	1	1	0	0	1	0
Druhé číslo:	1	1	1	1	0	0	0	0
Logický součin (AND):	1	0	1	1	0	0	0	0

(10110010) AND (11110000) = 10110000



 *Logická nonekvivalence* je operace XOR. Na dané pozici ve výsledku bude 1 právě tehdy, když na téže pozici je v každém z původních čísel jiná hodnota.



Příklad 1.14

Jsou dána binární čísla 10110010 a 11110000. Vypočteme jejich logickou nonekvivalenci.

První číslo:	1	0	1	1	0	0	1	0
Druhé číslo:	1	1	1	1	0	0	0	0
Logická nonekvivalence (XOR):	0	1	0	0	0	0	1	0

(10110010) XOR (11110000) = 01000010



Úkoly

1. Proveďte logickou negaci těchto binárních čísel:
10110100, 11100001, 11111111
2. Proveďte logický součet této dvojice binárních čísel:
10110100, 11100001
3. Proveďte logický součin této dvojice binárních čísel:
10110100, 11100001
4. Proveďte logickou nonekvivalenci této dvojice binárních čísel:
10110100, 11100001



1.5 Jednotky a násobky

Víme, že jednotkou informace je jeden *bit*, který může mít hodnotu buď 0 nebo 1. Odvozenou jednotkou je jeden *Byte* nebo *oktet*. Jeden oktet představuje osm bitů, jeden Byte obvykle také.



Úkol

Připomeňte si:

- Jestliže do jednoho oktetu (Bytu) lze uložit 8 bitů, kolik různých čísel je možné do těchto 8 bitů uložit?
- Jaký rozsah hodnot lze do těchto 8 bitů uložit (tj. jaká je minimální a jaká maximální uložená hodnota)? Pozor, tato otázka je trochu jiná než předchozí!



Název	Značka	Hodnota v B
Kibibyte	KiB	$1 \text{ KiB} = 2^{10} \text{ B} = 1024 \text{ B}$
Mebibyte	MiB	$1 \text{ MiB} = 2^{10} \text{ KiB} = 1024 \text{ KiB} = 2^{20} \text{ B}$
Gibibyte	GiB	$1 \text{ GiB} = 2^{10} \text{ MiB} = 1024 \text{ MiB} = 2^{20} \text{ KiB} = 2^{30} \text{ B}$
Tebibyte	TiB	$1 \text{ TiB} = 2^{10} \text{ GiB} = 1024 \text{ GiB} = 2^{20} \text{ MiB} = 2^{30} \text{ KiB} = 2^{40} \text{ B}$
atd.		

Tabulka 1.1: Binární násobky

Pro odvozování také využíváme předpony, a to buď podle soustavy SI (násobnost $1000 = 10^3$), nebo podle ČSN IEC 60027-2 (binární násobky, násobnost $2^{10} = 1024$). V tabulce 1.1 na straně 11 je naznačeno, jak se binární násobky používají.



Úkoly


1. Vezměme si údaj 58 210 B. Tento údaj přepočtete na
 - kB,
 - KiB.
2. Vezměme si údaj 12 430 KB. Tento údaj přepočtete na
 - B,
 - MB,
 - KiB,
 - MiB.
3. Vezměme si údaj 12 430 KiB. Tento údaj přepočtete na
 - B,
 - MiB,
 - KB,
 - MB.
4. Vezměme si údaj 100 000 B. Přepočtete tento údaj do jednotek kB a KiB. Zjistěte, o kolik procent je údaj v kB vyšší než údaj v KiB.
5. Vezměme si údaj 100 000 b (bitů). Tento údaj přepočtete na
 - kb,
 - B,
 - kB.




Sestavte vzorec pro přepočet kB na KiB a pak vzorec pro opačný přepočet.




Počítačová síť na hostitelském zařízení

 *Rychlý náhled:* V této kapitole se podíváme na některé základní úlohy týkající se hostitelských (tj. koncových) zařízení. Nebude to vyčerpávající přehled, s dalšími úlohami (zjišťováním, konfigurováním, ochranou) se budeme setkávat během celého semestru.

Protože si tento předmět zapisují i studenti, kteří zatím neměli žádný předmět z oblasti operačních systémů, je zde také rychlý úvod do práce v textovém režimu. V kapitole najdete obvykle postupy jak pro Windows, tak i pro Linux (případně MacOS). Není nutné se učit obojí, volte postupy pro ten systém, který běžně používáte.

 *Klíčová slova:* Síťová karta, síťové rozhraní, textový režim, příkaz, hostname, MAC adresa (hardwarová adresa, fyzická adresa), IP adresa, IPv4, IPv6

 *Cíle studia:* Po prostudování této kapitoly budete umět aktivovat/deaktivovat síťové rozhraní, pracovat s MAC adresami a IP adresami na koncovém zařízení (počítači) – zjistit své adresy, nastavit IP adresu, a také zjistit a případně nastavit název svého zařízení.

2.1 Síťová karta

Z přednášek víme, že v síti najdeme:

- hostitelská (koncová) zařízení – počítače, servery, notebooky, atd.,
- aktivní síťové prvky – mezilehlá zařízení, která podle určitých pravidel přeposílají komunikaci mezi svými porty,
- pasivní síťové prvky – například kabely nebo zásuvky ve zdi.

Každé zařízení v síti (hostitelské zařízení nebo aktivní síťový prvek) potřebuje minimálně jedno *síťové rozhraní*, přes které je do sítě připojeno. Síťové rozhraní je v zařízení buď integrováno nebo se jedná o síťovou kartu.

Síťová karta je komponenta, která danému zařízení přidává síťové rozhraní a kterou k tomuto zařízení připojujeme obvykle přes vhodné rozhraní (většinou PCIe×1, PCIe×8, PCI nebo USB).



Úkol

V některém internetovém obchodě (například <http://www.lan-shop.cz>, <https://www.tsbohemia.cz>) nebo na <http://heureka.cz/> najděte kategorii pro síťové karty. Prohlédněte si filtr, zjistěte, podle kterých parametrů lze nabídku síťových karet filtrovat. Vyberte si postupně několik různých karet a v údajích se pokuste najít informaci o tom, který standard karta podporuje.



2.2 Textový režim

Většina lidí je zvyklá pracovat v grafickém režimu, tedy s okny, tlačítky, startovacím menu apod. Jenže ne vždy je to praktické. Některé úlohy se totiž dají provádět pouze v textovém režimu, nebo to sice jde oběma způsoby, ale v textovém režimu to může být méně pracné a méně časově náročné (opravdu!).

Textový režim se nám hodí například tehdy, když

- chceme určitou úlohu provést hromadně (pro velké množství dat/souborů/uživatelů/...), přičemž v grafickém režimu bychom se „uklikali“ myší (zkuste třeba přidat do systému tisíc nových uživatelů a pochopíte),
- opakovaně v čase nebo na mnoha počítačích (pak se vyplatí pro tuto činnost napsat skript, což je textový soubor s příkazy, které se spuštěním skriptu provedou),
- chceme provést úlohu, pro kterou v grafickém režimu nemáme nástroj nebo sice máme, ale zbytečně složitě dostupný (tak schválně – jak dlouho bude trvat, než v grafickém rozhraní najdete nástroj, kterým zjistíte dostupnost konkrétního počítače v síti, tedy pokud ho vůbec najdete?),
- chceme určitý systém konfigurovat vzdáleně přes síť.

Pokud jste ještě nezkoušeli pracovat v textovém režimu, nejdříve se naučte, jak se do tohoto režimu vlastně dostat.



Příklad 2.1 (Jak se dostat do textového režimu)

Ve Windows spustíme aplikaci Příkazový řádek – buď se k ní „doklepeme“ přes menu tlačítka *Start* (v Příslušenství), nebo spustíme příkazem `cmd`.

V Linuxu či jiném UNIX-like systému máme na výběr:

- Najdeme některou aplikaci, která bude fungovat jako terminál – většinou se jmenuje Terminál, xterm, Konsole nebo podobně.
- Použijeme některou textovou konzolu. Tyto konzoly jsou k dispozici i v případě, že není spuštěno grafické rozhraní. Pokud jsme zrovna v grafickém rozhraní a chceme použít první textovou konzolu, stiskneme `Ctrl+Alt+F1`, pro druhou `Ctrl+Alt+F2`, pro třetí `Ctrl+Alt+F3`, atd. Mezi konzolami se pak přepínáme stejnými zkratkami, ale už bez klávesy `Ctrl`, takže třeba `Alt+F2` pro druhou konzolu. V konzole se přihlásíme (zadáme své přihlašovací jméno a pak jsme dotázáni na heslo) a můžeme pracovat.

Pozor – textové konzoly obvykle nejsou k dispozici v systému běžícím ve virtuálním stroji.

Pokud pracujeme v textové konzoli a chceme se přesunout zpět do grafického režimu, jednoduše přejdeme na poslední (nebo v některých distribucích první) konzolu v pořadí, protože na ní je obvykle spuštěno grafické rozhraní (bývá to sedmá nebo osmá konzole, nebo zmíněná první, podle konkrétní distribuce).



Podíváme se na jednu ze základních úloh v oblasti sítí – deaktivaci a aktivaci síťového rozhraní.



Příklad 2.2 ((De-)aktivace síťového rozhraní)

V kterémkoliv operačním systému se to dá provést pravým tlačítkem myši (když ovšem najdete ikonu, na kterou je třeba takto klepnout), v textovém rozhraní se to dělá následovně.

Ve Windows:

```
ipconfig /release
```

```
ipconfig /renew
```

Správně bychom měli dodat i název síťového rozhraní, které chceme takto ovlivnit, ale vzhledem k tomu, že názvy síťových rozhraní ve Windows mají nepříjemně dlouhé názvy, se s tím většinou nikdo neobtěžuje a prostě takto deaktivujeme a aktivujeme všechna síťová rozhraní najednou.

V Linuxu je víc možností (zde budeme pracovat se síťovým rozhraním `eth0`, u jiného bychom tento název změnili nebo neuvedli vůbec):

```
ifconfig eth0 down
```

```
ifconfig eth0 up
```

Zkrácená verze, která však nemusí fungovat všude:

```
ifdown eth0
```

```
ifup eth0
```

S využitím novější sady příkazů pro práci se sítí:

```
ip link set dev eth0 down
```

```
ip link set dev eth0 up
```

Pokud nevíme, jak se naše síťové rozhraní nazývá, pak si nejdřív příkazem `ifconfig` nebo `ip link` necháme vypsát seznam všech síťových rozhraní s parametry; z výpisu bychom měli to správné rozhraní poznat.



2.3 Hostname

Každé hostitelské zařízení má svůj název (hostname), pod kterým vystupuje především v lokální síti. Jak ho zjistit:

- V grafickém rozhraní máme obvykle nějaký nástroj, který nám tento název sdělí. Například ve Windows spustíme nástroj *Systém* (klepneme pravým tlačítkem myši na ikonu *Počítač* a zvolíme *Vlastnosti*, případně najdeme v *Ovládacích panelech* nebo v nástroji *Nastavení*), v zobrazeném okně najdeme položku *Název počítače*.
- V textovém rozhraní použijeme příkaz `hostname`.

V grafickém rozhraní určitě nikdo nebude mít problém příslušnou informaci najít, podíváme se tedy na postup v textovém rozhraní.



Příklad 2.3 (Jak zjistit hostname, když jsme v textovém režimu)

V jakémkoliv operačním systému jednoduše v textovém režimu zadáme:

hostname

Vypíše se název zařízení, na kterém zrovna pracujeme.



2.4 Adresy

Kromě názvu bývá zařízení jednoznačně určeno také svými (číselnými) adresami. Na hostitelských zařízeních se setkáváme s těmito typy adres:

- MAC adresa (fyzická, hardwarová adresa),
- IP adresa.

Ke každému typu si něco řekneme a ukážeme si, jak se dá zjistit či ovlivnit.

2.4.1 Hardwarové adresy

Hardwarová (MAC, fyzická, EUI-48) adresa je vlastně nízkourovňovou adresou síťového rozhraní. Tuto adresu má každé síťové rozhraní, a to i tehdy, když zrovna není připojeno k síti. Síťové rozhraní má svou MAC adresu již přidělenou svým výrobcem, a tato adresa by správně měla být celosvětově jednoznačná. Jenže mnohá pravidla mají svou výjimku, toto pravidlo také.

Takže ve skutečnosti máme dva druhy MAC adres:

- adresa přidělená výrobcem (BIA – Burned-In-Address, „vypálená“), která je uložena „na-pevno“ v ROM 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. Protože jedna hexadecimální číslice reprezentuje čtyři binární číslice, znamená to, že 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 těchto hodnot, 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.

Fyzickou adresu obvykle zapisujeme tak, že dvojice hexadecimálních číslic oddělíme dvojtečkou nebo pomlčkou (dnes je běžnější pomlčka, ať se to neplete s IPv6 adresami), nebo po čtyřech číslicích tečkou. V některých konfiguračních prostředích je však zvykem „nezdržovat se“ s nadbytečnými oddělovači, tam najdeme prostě řetězec číslic. Takže například:

50:E5:49:A2:80:61

50-E5-49-A2-80-61

50E5.49A2.8061

50E549A28061

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

Tabulka 2.1: Struktura MAC adresy síťového rozhraní

Na jednom zařízení můžeme mít i víc než jen jednu hardwarovou adresu, a to z některého z těchto důvodů:

- Máme víc než jedno síťové rozhraní (třeba víc síťových karet); minimálně tehdy, když na počítači máme jak podporu ethernetové (drátové) sítě, tak i bezdrátové.
- Používáme některý virtualizační software (Virtual Box, VMWare Workstation, apod.) – každý virtualizovaný systém potřebuje přistupovat k síti nezávisle na ostatních, proto má k dispozici virtuální síťové rozhraní s vlastní vygenerovanou fyzickou adresou.



Příklad 2.4 (Jak zjistit hardwarovou adresu ve Windows)

Pokud chceme zjišťovat v grafickém režimu, je to v *Ovládacích panelech* panel *Centrum síťových připojení a sdílení*, vlevo najdeme *Změnit nastavení adaptéru*.

V okně pak máme seznam síťových rozhraní včetně těch virtuálních. Na vybrané rozhraní poklepeme a v okně (přibližně uprostřed) klepneme na tlačítko *Podrobnosti*.

Zobrazí se okno se seznamem vlastností síťového rozhraní, kde najdeme řádek *Fyzická adresa*.

V textovém rozhraní to jde (ve vyšších verzích) příkazem

```
getmac
```

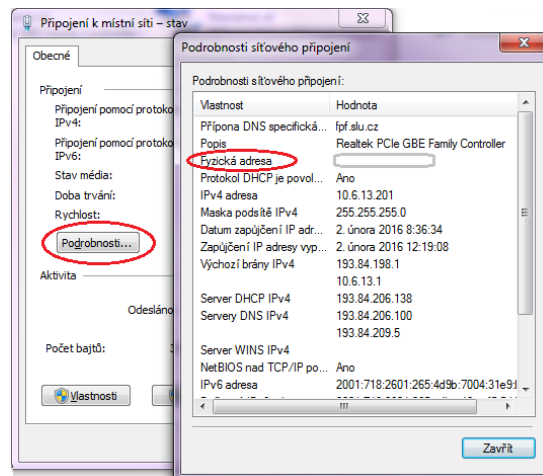
Vypíše se jedna nebo více fyzických adres. Pokud místo toho dostaneme chybové hlášení (třeba tehdy, když máme starší verzi Windows) nebo při více adresách chceme vědět, „která je která“, použijeme příkaz

```
ipconfig /all
```

Výpis je poměrně dlouhý, hledáme řádky začínající řetězcem „Fyzická adresa“.

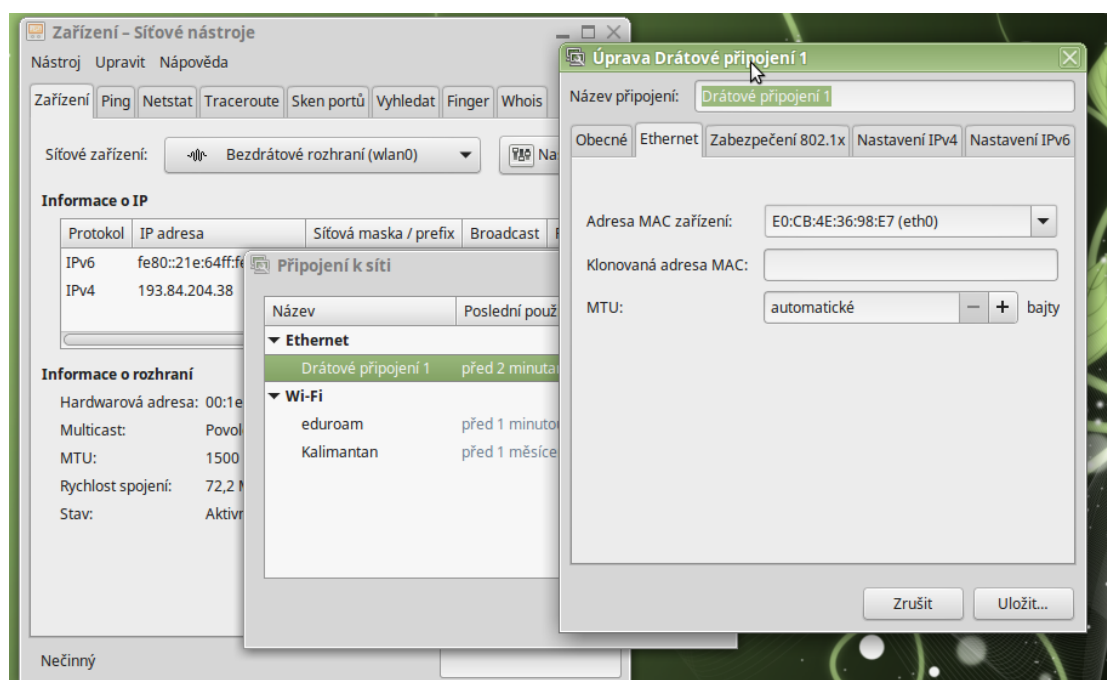
...

Fyzická adresa : 50-E5-49-A2-80-61



Příklad 2.5 (Jak zjistit hardwarovou adresu v Linuxu)

V Linuxu (a dalších UNIX-like systémech, včetně MacOS X) máme obvykle k dispozici také nástroj v grafickém rozhraní, ve kterém si tuto informaci zjistíme. Na obrázku 2.1 je například sada síťových nástrojů v Linux Mint, v rozhraní Mate. V okně, které na obrázku vidíme zcela vpravo, je MAC adresa na záložce *Ethernet*, v konfiguraci Wi-fi by se tato záložka jmenovala *Wi-fi* a vypadala by trochu jinak, protože každá z těchto sítí potřebuje trochu jiný typ parametrů.



Obrázek 2.1: Práce s adresami v Linuxu

Podívejme se však spíše na nástroje v textovém rozhraní. Pro zjištění fyzické adresy můžeme použít buď příkaz `ifconfig` nebo příkaz `ip link`. První z nich je starší a v některých distribucích přestává být podporován, spíše se prosazuje druhý způsob. V prvním případě napíšeme `ifconfig`

(případně můžeme jako parametr přidat název síťového rozhraní, například `eth0`, jinak se vypíší údaje postupně pro všechna síťová rozhraní). MAC adresu najdeme hned na prvním řádku výpisu pro dané rozhraní, za řetězcem `HWaddr`:

```
eth0  Link encap:Ethernet HWaddr 00:0c:29:92:58:67
      inet adr:192.168.126.136 ...
```

Druhá možnost je tato:

```
ip link show
```

Výpis bude celkově kratší a MAC adresa bude spíše ke konci údajů o tom síťovém rozhraní, které nás zajímá:

```
2: eth0: <BROADCAST,...
    link/ether 00:0c:29:92:58:67 ...
```



Úkol

Zjistěte hardwarové adresy síťových rozhraní na svém počítači.



Pokud se opravdu jedná o hardwarovou BIA adresu síťového rozhraní, pak podle její první poloviny můžeme zjistit výrobce (i když to samozřejmě jde i jinak).



Úkol

Na adrese <http://standards-oui.ieee.org/oui/oui.txt> najdeme seznam OUI přidělených různým výrobcům síťových zařízení. Najděte na této stránce výrobce síťových karet, jejichž MAC adresu jsme zjistili v příkladech v této sekci. Upozorňuji, že příklad v Linuxu proběhl ve virtualizovaném systému, přesto jde o unikátní (jednoznačnou) adresu.

Alternativou je vyhledávací nástroj na <http://www.miniwebtool.com/mac-address-lookup/>.



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

Ve Windows je postup změny hardwarové adresy náročný (je potřeba provést změnu v registru nebo si stáhnout speciální aplikaci) a ne vždy to funguje, ale v UNIX-like systémech to je vcelku jednoduchá věc, třebaže určité riziko, že síťové rozhraní odmítne fungovat, taky bude.



Postup (Změna hardwarové adresy)

Opět existují dvě možnosti, jak to provést – starší a novější.

```
ifconfig eth0 down
```

```
ifconfig eth0 hw ether 02:00:00:11:22:33
```

```
ifconfig eth0 up
```

Nejdřív jsme síťové rozhraní deaktivovali, pak jsme změnili MAC adresu a následně jsme rozhraní znovu aktivovali. Jde to i v jediném příkazu:

```
ifconfig eth0 down hw ether 02:00:00:11:22:33 up
```

Jiná možnost:

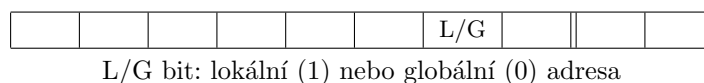
```
ip link set dev eth0 down
```

```
ip link set dev eth0 address 02:00:00:11:22:33
```

```
ip link set dev eth0 up
```



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 2.2.



Obrázek 2.2: Umístění L/G (local/global) bitu v MAC adrese

Takže pokud „autor změny“ zachovával pravidla, pak bychom podle tohoto bitu měli poznat, že jde o pozměněnou adresu a tudíž není zaručeno, že je celosvětově unikátní.

**Příklad 2.6 (Jak poznat lokální hardwarovou adresu)**

O následujících MAC adresách zjistíme, zda jsou BIA (od výrobce) nebo pozměněné lokálně platné.

MAC adresa	První oktet	... v binárním tvaru	Důsledek
00-11-21-30-A2-7C	00-...	0000 0000-...	BIA adresa
98-E7-9A-26-4B-55	98-...	1001 1000-...	BIA adresa
02-D3-2A-64-DD-20	02-...	0000 0010-...	lokální adresa
86-D3-2A-64-DD-20	86-...	1000 0110-...	lokální adresa

**Poznámka:**

V některých zdrojích se můžeme setkat s tvrzením, že L/G bit je sice v prvním oktetu, ale druhý zleva místo druhý zprava. Je to proto, že když zařízení odesílá rámec, odesílá ho po jednotlivých bitech. Existují konvence (bit-level endianness) říkájící, jestli se při odesílání jednoho oktetu má začít zprava (nejvíce významným bitem, MSB – most significant bit first) nebo zleva (nejméně významným bitem, LSB – low significant bit first). LSB používá například Ethernet, USB nebo RS-232 (servisní port), MSB například optické sítě SONET/SDH nebo teletext. Řazení bitů má na starosti fyzická vrstva. Když u Ethernetu použijeme na bitové úrovni místo LSB MSB, pořadí se obrátí a to, co správně má být v druhém bitu zprava, je najednou v druhém bitu zleva.

**Úkol**

U těchto adres zjistěte, zda jsou BIA nebo pozměněné lokálně platné.

- 00-90-CF-23-50-37 • 2C-60-0C-73-22-A5 • 4E-09-B4-A0-BF-33
- 36-09-A4-26-41-0C • E4-90-69-DD-82-E1 • FF-FF-FF-FF-FF-FF



Všimněte si poslední adresy v zadání předchozího úkolu. Pokud jsou všechny hexadecimální číslice nastaveny na *F*, znamená to, že v binárním tvaru jsou všechny číslice na maximum, tedy „samé jedničky“. Tato adresa je nejen lokálně platná (není globálně jednoznačná), ale navíc je *všeobecná* (broadcastová), tedy neoznačuje jedno konkrétní zařízení. Všeobecné MAC adresy se používají například tehdy, když je odeslán 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 2.3: Umístění I/G (individual/group) bitu v MAC adrese

**Příklad 2.7 (Rozpoznání všeobecné a skupinové MAC adresy)**

Skupinová adresa má nastaven I/G bit, a tedy první oktet je liché číslo. Takže:

- 86-D3-2A-64-DD-20 je individuální adresa, protože I/G bit je nastaven na 0 (ale zároveň lokální, protože L/G bit je nastaven na 1).
- 87-D3-2A-64-DD-20 je skupinová, protože I/G bit je nastaven na 1 (první oktet je lichý).

**Úkol**

Jsou následující adresy skupinové?

- 4E-09-B4-A0-BF-33
- 01-2D-63-BB-00-20
- 4F-00-21-7A-C0-59

**Poznámka:**

V praxi se však jako skupinové využívají pouze adresy začínající 01-00-5E-, a pak samozřejmě všeobecné (broadcastové) adresy.

**2.4.2 IP adresy**

Komunikující zařízení připojené do sítě (resp. každé jeho síťové rozhraní) obvykle potřebuje IP adresu. Tato adresa na rozdíl od MAC adresy není se síťovým rozhraním až tak pevně spojena a může být přidělována třeba pokaždé jiná, dokonce jedno síťové rozhraní může mít i víc než jednu IP adresu (v unixových systémech).

Rozlišujeme dva typy IP adres, podle verze protokolu, s nímž adresa souvisí:

- IPv4 adresa se skládá ze čtyř dekadických čísel v rozsahu 0..255 (tedy v rozsahu jednoho oktetu), čísla jsou oddělena tečkou. Délka IPv4 adresy je 4×8 bitů, tedy 32 bitů.
- IPv6 adresa se skládá z (maximálně) osmi šestnáctkových čísel, každé číslo je zapsáno (maximálně) čtyřmi hexadecimálními číslicemi, čísla jsou oddělena dvojtečkou. Délka IPv4 adresy je 8×16 bitů, tedy 128 bitů.

**Příklad 2.8**

Příklady IPv4 adres: 10.6.13.125 169.254.82.9 255.255.255.255

Příklad IPv6 adresy: 2001:718:2601:400:0:e5f8:31e8:b6da

Kdybychom u IPv6 adresy trvali na pevně daném počtu hexadecimálních číslic u jednotlivých čísel adresy, stačí doplnit nuly k číslu zleva, tedy ukázková adresa „v plném znění“ by byla

2001:0718:2601:0400:0000:e5f8:31e8:b6da



IPv4 adresy jsou celkem „uživatelsky přívětivé“, nejsou moc dlouhé a dekadický zápis je našemu myšlení bližší. Jenže jejich číselný rozsah (a tedy množství) už dávno nestačí požadavkům moderních technologií, proto se čím dál častěji setkáváme s IPv6 adresami, které jsou sice delší a hůře zapamatovatelné, nicméně jich je dostatek.

Lidé mají tendenci si složité věci zjednodušovat, u IPv6 to platí také. Pokud tedy v adrese máme sekvenci čísel s hodnotou nula, můžeme tento úsek adresy vynechat, ale musíme naznačit, kde konkrétně vynechaný úsek je (zdvojením symbolu dvojtečky).



Příklad 2.9

IPv6 adresu 2001:718:2601:0:0:e5f8:31e8:b6da zkrátíme takto:

2001:718:2601:0:0:e5f8:31e8:b6da \Rightarrow 2001:718:2601::e5f8:31e8:b6da

Podle umístění dvojsymbolu :: poznáme, kam při opačné úpravě umístit chybějící nuly. Taky je jasné, kolik nulových úseků bylo odstraněno, když víme, že celkový počet úseků je osm.

Jestliže je v adrese nulových úseků víc, krátíme vždy delší úsek. Pokud jsou dva stejně dlouhé úseky, volíme ten, co je víc vlevo. Například adresu 2001:718:2601:0:0:e5f8:0:0 krátíme takto:

- dobře: 2001:718:2601::e5f8:0:0
- špatně: ~~2001:718:2601:0:0:e5f8::~~ (ze dvou stejně dlouhých sekvencí nulových skupin jsme vybrali tu vpravo místo té vlevo)
- špatně: ~~2001:718:2601::e5f8::~~

Poslední „špatný“ příklad je chybně, protože by nebylo možné zpětně zjistit, kolik nulových úseků bylo vynecháno na prvním/druhém místě, adresa by nebyla jednoznačná, odpovídaly by adresy:

- 2001:718:2601:0:e5f8:0:0:0
- 2001:718:2601:0:0:e5f8:0:0
- 2001:718:2601:0:0:0:e5f8:0



Úkol

Následující IPv6 adresy запиšte ve zkráceném tvaru:

- 2001:05a0:0037:9b32:0000:0000:0000:564c
- 2001:05a0:0037:9b32:564c:0000:0000:0000
- 2001:05a0:0000:9b32:564c:0000:0000:27a3
- ff02:0000:0000:0000:0000:0000:0000:0001

Poslední z těchto adres je skupinová adresa představující všechna zařízení (jejich síťová rozhraní), která „rozumí“ IPv6. Do této skupiny patří každý počítač, server, notebook, tablet, router apod., který dokáže používat IPv6 adresy a přijmout paket vytvořený podle protokolu IPv6.



Příklad 2.10 (Zjištění IP adresy ve Windows)

Nejdřív se podíváme na Windows. V grafickém režimu zjistíme IP adresu stejně jako MAC adresu, tedy přes *Podrobnosti* u daného síťového adaptéru. V textovém režimu napíšeme

`ipconfig`

Výpis je kratší než u příkazu s parametrem (`ipconfig /all`). Najdeme odstavec pro to síťové rozhraní, o které nám jde (například „Adaptér sítě Ethernet Připojení k místní síti“ – první část je typ síťového rozhraní, druhá název rozhraní) a zaměříme se na řádky obsahující řetězec „Adresa

IPv4“ nebo „IPv6 adresa“. Řádků obsahujících adresy tam zřejmě bude více, na jejich význam se podíváme, až budeme probírat síťovou vrstvu.



Příklad 2.11 (Zjištění IP adresy v Linuxu)

Také v UNIX-like systémech můžeme obvykle použít některý nástroj v grafickém rozhraní. V textovém rozhraní se dají IP adresy zjistit takto:

```
ifconfig
```

(případně můžeme zadat i název síťového rozhraní). Ve výpisu hledáme u příslušného rozhraní řádky obsahující zkratku „inet“:

```
inet adr:10.0.0.2 ...
```

```
inet6-adr: fe80::21d:72ff:fe31:aa0 ...
```

Nebo použijeme „novější“ příkaz:

```
ip addr show
```

(případně přidáme `dev eth0`, když chceme výstup pouze pro toto jedno rozhraní). Opět u odstavce pro naše rozhraní hledáme řádky začínající řetězcem „inet“:

```
inet 10.0.0.2 ...
```

```
inet6 fe80::21d:72ff:fe31:aa0 ...
```



Poznámka:

Příkaz `ip` je specifický tím, že umožňuje krátit parametry. Například plný tvar výše použitého příkazu je `ip address show`

Ale díky možnosti krácení můžeme místo toho napsat třeba

```
ip addr show
```

```
ip a show
```

```
ip addr sh
```

```
ip a sh
```

```
apod.
```



Úkol

Zjistěte (především v textovém režimu) IP adresy svého hlavního síťového rozhraní (ethernetového nebo Wi-fi, případně obojí podle vybavení počítače).

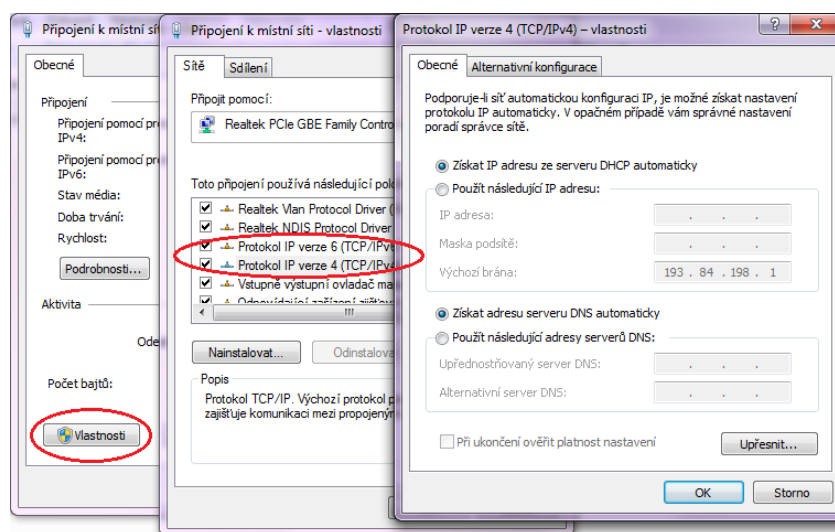


IP adresu máme buď *staticky* naklepanou v příslušném nástroji, nebo je vždy při připojování do sítě přidělována *dynamicky*. Měli bychom předně umět zjistit, která z těchto možností se nás týká, a dále umět nakonfigurovat (určit, zda má být adresa dynamicky požadována nebo jestli má být použita staticky přidělená adresa, a samozřejmě jak zadat tuto adresu).



Příklad 2.12 (Konfigurace IP adres ve Windows)

Ve Windows v grafickém režimu je toto nastavení opět v *Ovládacích panelech* je panel *Centrum síťových připojení a sdílení*, vlevo najdeme *Změnit nastavení adaptéru*. Klepneme na tlačítko *Vlastnosti* a v seznamu protokolů najdeme protokol IP (verze 4 nebo 6). Poklepáním získáme okno s pro nastavení IP adresy.



Obrázek 2.4: Konfigurace IP adresy ve Windows

Jak vidíme na obrázku 2.4, přepínačem se určuje, zda má být adresa získávána dynamicky (ze serveru DHCP), nebo jestli ji chceme „staticky“ ručně naklepat.



Příklad 2.13 (Konfigurace IP adres v Linuxu)

Konfiguraci můžeme provádět v grafickém režimu podobně jak bylo naznačeno u MAC adres, na obrázku je nástroj z Linux Mint s rozhraním Mate.

Jak vidíme získávání IPv4 adresy je zajišťováno dynamicky z DHCP serveru. Pokud bychom chtěli sami staticky zadat vlastní IPv4 adresu, v rozbalovacím boxu zvolíme místo *Automaticky (DHCP)* volbu *Ruční*, čímž se zpřístupní položky na zbytku záložky. Podobně na další záložce lze konfigurovat IPv6.

V textovém režimu máme víc možností (starší a novější způsob), například zadáme:

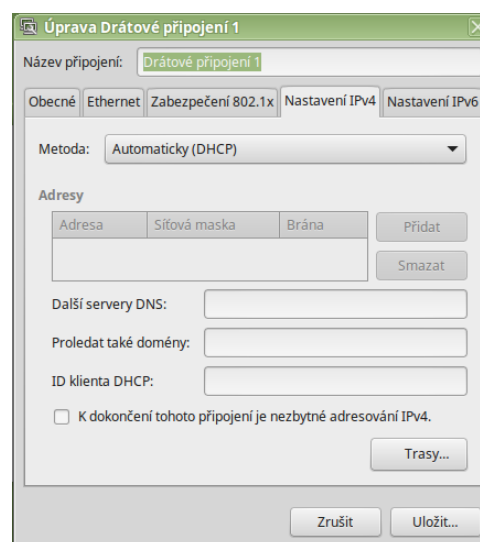
```
ip addr add 193.90.220.42/25 brd + dev eth0
```

čímž jsme přidali IPv4 adresu pro síťové rozhraní eth0.

V UNIX-like systémech mohou mít síťová rozhraní při používání této sady příkazů víc než jednu IP adresu, jedna z nich je pak primární. Proto v textovém rozhraní tuto adresu nepřepisujeme, ale přidáváme další. Ubrat adresu bychom mohli nahrazením parametru `add` parametrem `del`.


Pokud budeme chtít „pročistit“ sadu IPv4 adres pro dané síťové rozhraní, tedy odstranit všechny IPv4 adresy a nastavit získávání dynamicky přes DHCP, zadáme

```
ip -4 addr flush dynamic
```




Kapitola 3

Standardy

 *Rychlý náhled:* Cílem není zahloubat se do obsahu standardizačních dokumentů, ale spíše vypořizovat jejich strukturu a naučit se je alespoň trochu používat. V této kapitole se podíváme na některé veřejně dostupné dokumenty s popisem síťových standardů.

 *Klíčová slova:* RFC dokument, IETF, ITU-T

 *Cíle studia:* Po prostudování této kapitoly budete umět najít veřejně dostupné dokumenty vydané sdruženími IETF a ITU-T a orientovat se v těchto dokumentech.

3.1 RFC dokumenty

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

- 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.

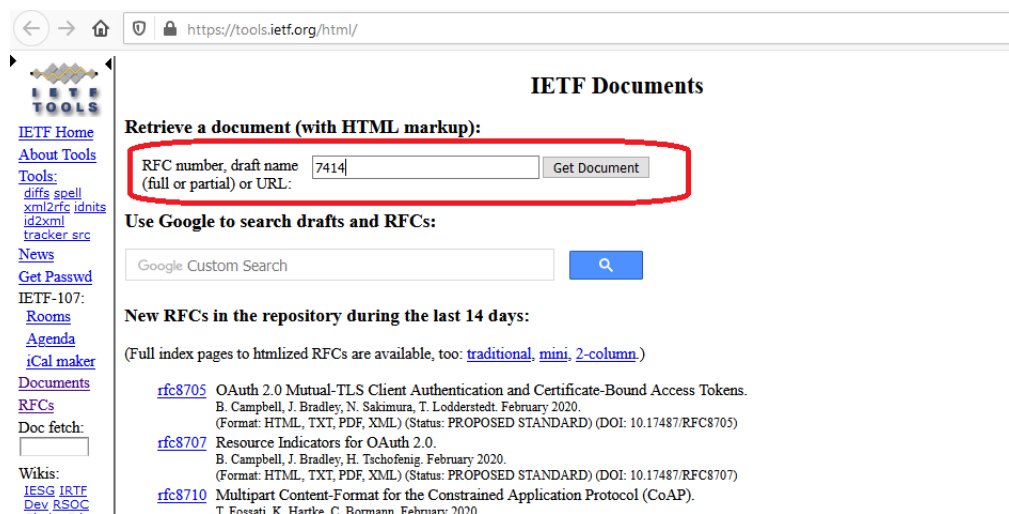


Příklad 3.1

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. Na obrázku 3.1 na straně 25 vidíme, kde je vyhledávací pole.

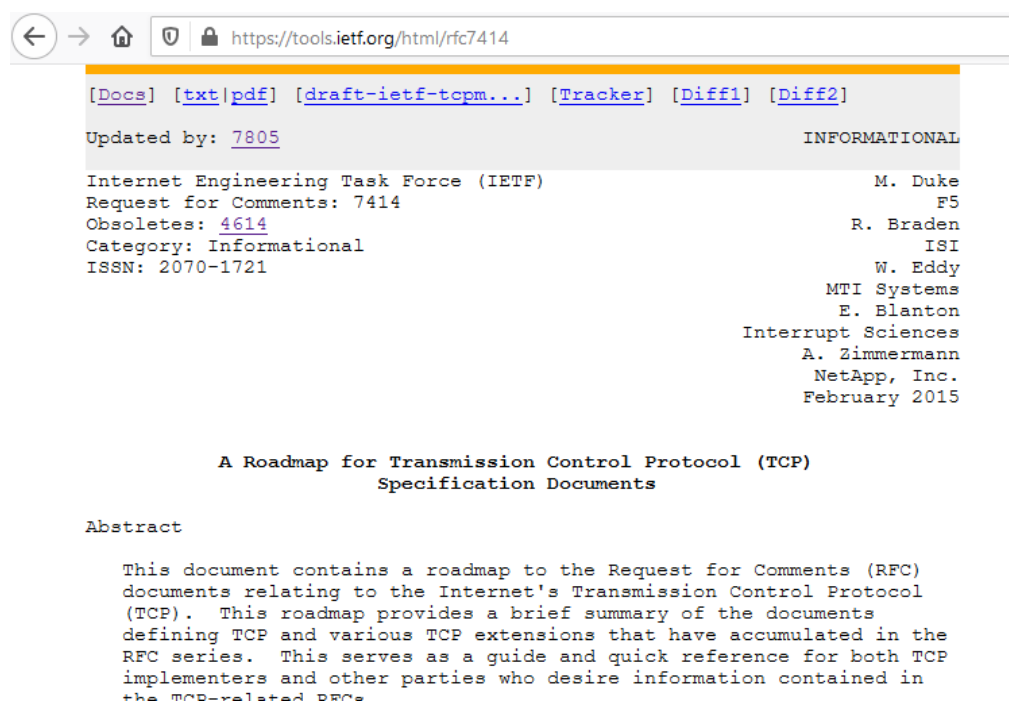
Pokud neznáme číslo RFC, dá se vyhledávat i podle klíčového slova, ale obvykle dostaneme příliš mnoho výsledků (v takovém případě je lepší se zeptat Googlu).



Obrázek 3.1: Cesta k RFC dokumentu 7414 ve vyhledávači IETF

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...“), dále 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 prohlédneme webovou stránku, je dokument rozčleněn na stejně dlouhé „stránky“ a každá má své záhlaví a zápatí.



Obrázek 3.2: Záhlaví RFC dokumentu 7414 A Roadmap to Transmission Control Protocol (TCP)

Zaměříme 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 3.2

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 TCP „odsunuta do historie“.



Příklad 3.3

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 nákrešů – 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.



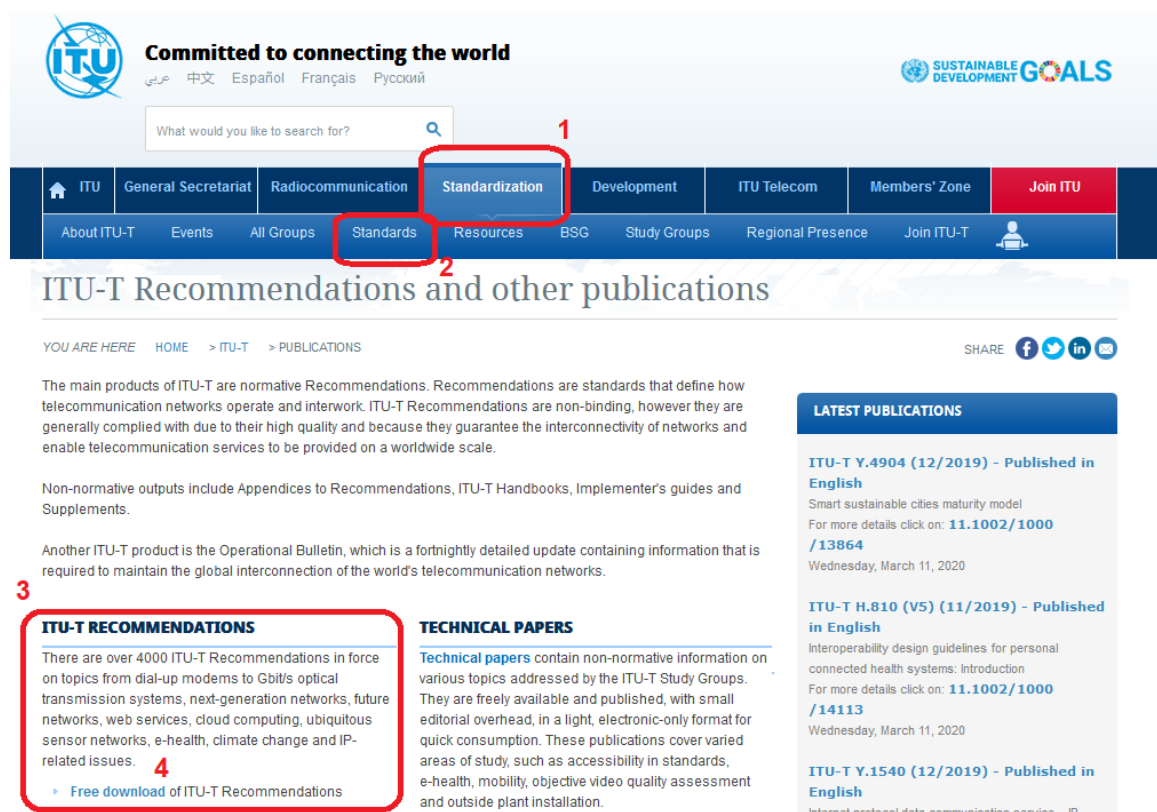
3.2 ITU-T

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>. Ve skutečnosti některé ITU-T standardy (přesněji doporučení, recommendations) nejsou veřejné – ty, na kterých spolupracuje s některou jinou standardizační institucí.



Příklad 3.4

Volně dostupný je například standard ITU-T G.993.2 (přístupová digitální síť VDSLv2, mnozí z nás mají doma VDSL modem/router). Na výše uvedené stránce organizace ITU-T můžeme v menu nahoře zvolit *Standardization – Standards*, pak v „dlaždicích bez rámečku“ najdeme *ITU-T Recommendations*, kde je odkaz *Free download*. Na obrázku 3.3 na straně 27 jsou jednotlivé kroky vyznačeny.



Obrázek 3.3: Stránka ke stažení ITU-T, doporučení G.993.2 (VDSLv2)

Tam hledáme doporučení podle počátečního písmene (tj. „G“) a postupně se „doklikáme“ k položce *G.993.2 (Very high speed digital subscriber line transceivers 2 (VDSL2))*. V tabulce máme různé verze, vybíráme samozřejmě co nejnovější.

Dokumentem se prokousávat nebudeme, nicméně jak vidíte, ke specifikaci se lze bez problémů (s trochou klikací námahy) opravdu dostat.



Příklad 3.5

Jedním ze standardů, které vznikly v ITU-T, ale „bohužel“ novější verze vznikla ve spolupráci s jinou standardizační institucí (zde ISO/IEC), 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 (OpenLDAP).

ITU-T Recommendations

YOU ARE HERE: HOME > ITU-T RECOMMENDATIONS > ITU-T X.500 (10/2019)

Search by number:

Search

Others:

Content search
Advanced search
Provisional name
ISO/IEC number
Formal description
Study Groups tree view ▶

ITU-T X.500 (10/2019) عربي | 中文 | English | Español | Français | Русский

Information technology – Open Systems Interconnection – The Directory: Overview of concepts, models and services

Recommendation ITU-T X.500 | International Standard ISO/IEC 9594-1 introduces the concepts of the Directory and the DIB (Directory Information Base) and overviews the services and capabilities which they provide.

Citation: <http://handle.itu.int/11.1002/1000/14031>
Approval date: 2019-10-14
Identical standard: ISO/IEC 9594-1 (Common)
Approval process: AAP
Status: In force
Observation: This text was produced through a joint activity with ISO and IEC. According to the agreement with our partners, this document is only available through payment.
Maintenance responsibility: ITU-T Study Group 17
Further details: [Patent statement\(s\)](#)
[Development history](#)
[\[16 related work items in progress\]](#)

Editions | **Related Supplement(s)** | Implementer's guide(s)

Ed.	ITU-T Recommendation	Status	Summary	Table of Contents	Download
9	X.500 (10/2019)	In force	here	here	here
8	X.500 (10/2016)	Superseded	here	here	here
7	X.500 (10/2012)	Superseded	here	here	here

Obrázek 3.4: Stránka ke stažení ITU-T, doporučení X.500

Na adrese <http://www.itu.int/itu-t/recommendations/rec.aspx?rec=X.500> najdeme základní informace o protokolu a také odkazy na PDF soubor s celým zněním. V tabulce tam najdeme různé verze. Pokud vybereme PDF soubor s verzí z roku 2019, jsme upozorněni, že přístup je pouze pro platící zájemce. Jestliže vybereme verzi z roku 2016, můžeme si PDF stáhnout.





Úkol


Prohlédněte si obsah PDF souboru se specifikací X.500 z roku 2016, o který se jednalo v předchozím příkladu, zejména:


- Srovnajte 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 (obdobu 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.




Kapitola 4


Lokální síť – Ethernet

 *Rychlý náhled:* Z přednášek bychom už měli vědět, co to jsou lokální sítě. Zde se seznámíme s nástroji, které slouží k jejímu monitorování (zejména s programem Wireshark), se zařízeními, která v lokálních sítích najdeme, dále si ukážeme, jak switch pracuje se svou tabulkou MAC adres. Další část kapitoly je věnována práci s kabely – seznámíme se s různými typy kabelů a naučíme se nasazovat koncovky na UTP kabely. Poslední sekce ukazuje, jak pracují VLAN sítě.

 *Klíčová slova:* Wireshark, repeater, hub, switch, bridge, router, gateway, segment sítě, kolizní doména, síť, broadcastová doména, tabulka MAC adres, kategorie kabelu, kroucená dvojlinka (twisted pair), UTP, STP, U/FTP, F/UTP, S/FTP, SF/UTP, optická kabeláž, optické vlákno, single-mode (jednovidové), multi-mode (vícevidové), pigtail, outdoor kabel, koaxiál, twinax, konektor, křížení, Tx/Rx, přímý kabel, křížený kabel, krimpování konektoru, RJ-45, 8p8c, krimpovací kleště, pin, tester (měřák), otočený kabel, konzolový kabel, telekomunikační zásuvka, svorkovnice, zařezávací nástroj, strukturovaná kabeláž, rozvaděč (rack), horizontální kabeláž, patch kabel, patch panel, vertikální kabeláž, páteřní síť, VLAN

 *Cíle studia:* Po prostudování této kapitoly budete umět zachytávat provoz na síti pomocí nástroje Wireshark, budete se orientovat v síťových zařízeních v lokální síti, budete rozumět tomu, jak switch zachází s tabulkou MAC adres, budete se orientovat v kabelech pro lokální síť, budete umět nasadit koncovku na UTP kabel a osadit kabel do zásuvky, také si ujasníte způsob komunikace při použití VLAN.

4.1 Wireshark

 *Sniffer* (síťový analyzátor, odchytač paketů, paketový odposlech) je takový nástroj, který umožňuje sledovat provoz na síti, zejména odposlouchávat posílané protokolové datové jednotky. Nejznámější sniffery jsou například:

- Wireshark,
- Cain & Abel,
- Kismet,
- tcpdump,
- Ettercap,
- MS Network Monitor.

Tento výčet rozhodně není konečný. Navíc každý z těchto programů má trochu jiné možnosti a určení. Obvykle jde především o to, aby správce sítě mohl otestovat provoz ve své síti, alespoň co se týče legálního využití.



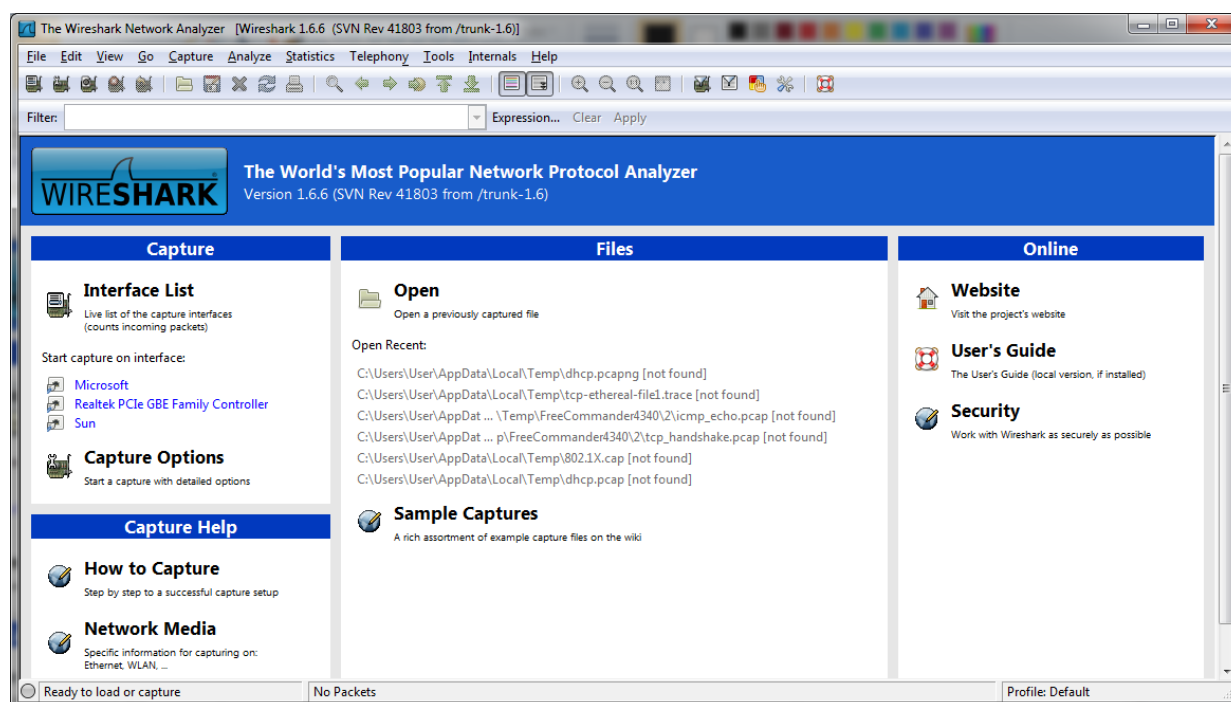
My se zde zaměříme na program *Wireshark*. Jeho výhodou je snadnost použití a přehlednost, podpora velkého množství formátů datových jednotek, a také volná dostupnost pro všechny běžné platformy (Windows, Linux, MacOS X, různé Unixy). Dá se říct, že Wireshark je jedním z nejpoužívanějších nástrojů tohoto typu. Wireshark se obvykle používá jako aplikace s grafickým rozhraním, ale pokud to někomu vadí, může používat i variantu pro textový režim – *tshark*.



Postup (Získání a instalace Wiresharku)

Pokud používáte Linux, obvykle stačí otevřít aplikaci, přes kterou se dostáváte do repozitáře softwaru, zadat klíčové slovo „wireshark“ a provést instalaci příslušného balíčku. Protože v unixových systémech je síť implementována v jádře systému, potřebujeme pro správné fungování programu (pro přístup k síťovým rozhraním) vyšší přístupová oprávnění.

V jiných operačních systémech je třeba najít instalační soubory na Internetu, například přímo na stránce projektu: <https://www.wireshark.org/download.html>. Na této stránce je seznam platforem, pro které existuje instalační soubor (vybereme tu svoji), a je tam taky návod k instalaci. Je možné, že během instalace je nám nabídnuto nainstalování i něčeho dalšího než jen samotného Wiresharku, u Windows je třeba souhlasit s instalací rozhraní *WinPcap* zprostředkovávajícího komunikaci Wiresharku se síťovým zásobníkem (v jiných systémech se toto rozhraní nazývá *Pcap* a je strukturálně jednodušší než to pro Windows).



Obrázek 4.1: Prostředí aplikace Wireshark

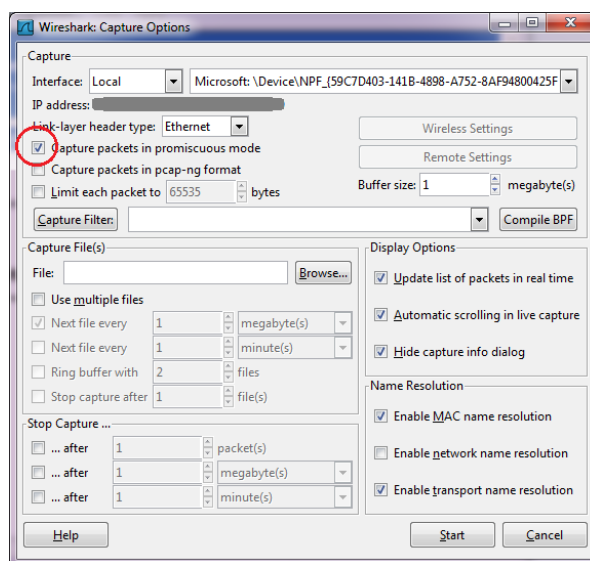
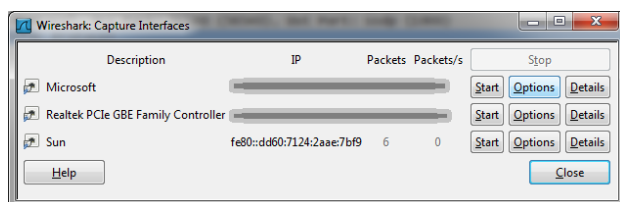


Postup (Používání Wiresharku)

Předpokládejme, že již máme Wireshark nainstalovaný a zprovozněný. Okno programu ve Windows je na obrázku 4.1.

Potřebné volby můžeme hledat v menu, ale obvykle si vystačíme s tím, co najdeme na titulní obrazovce. Dále záleží, co vlastně chceme dělat:

- Jestliže nás zajímá provoz na konkrétním síťovém rozhraní, který přímo s tímto rozhraním souvisí, pak v levém podokně najdeme příslušné síťové rozhraní (v části *Interface List*) a klepneme na ně.
- Jestliže nás zajímá obecně provoz na segmentu sítě, ke kterému jsme připojeni, musíme předem přepnout síťovou kartu do promiskuitního módu (to znamená, že přijmeme opravdu všechny pakety včetně těch, které nejsou přímo pro nás). To se dá provést i přímo ve Wiresharku: v menu *Capture – Interfaces*, u příslušného rozhraní



Obrázek 4.2: Nastavení síťové karty do promiskuitního módu

klepneme na *Options*. V okně zatrhneme volbu *Capture packets in promiscuous mode*.

- Zachycený provoz se dá taky uložit do souboru: *File – Save*.
- Další možností je prohlížení obsahu souboru se zachyceným provozem, a to buď na tomtéž počítači, nebo jiným. Pokud je již Wireshark nainstalován, pak stačí poklepat na soubor (obvyklá přípona je .PCAP nebo .CAP) a otevře se okno Wiresharku s obsahem tohoto souboru.



Příklad 4.1

Ukážeme si, jak Wireshark používat k zachycení a analýze reálného provozu. Předpokládejme, že umíme Wireshark spustit s takovými oprávněními, aby měl přístup k síťovým rozhraním, že víme, které rozhraní chceme sledovat, a že je toto rozhraní aktivní.

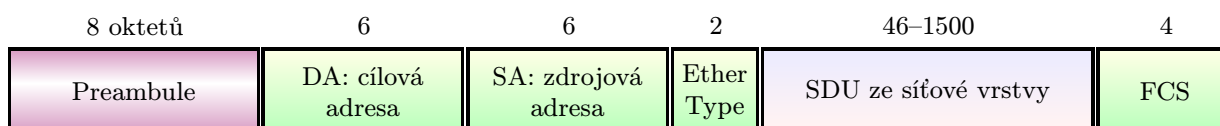
Spustíme Wireshark a volbou síťového rozhraní začneme zachytávat pakety. Teď pozor – předem bychom si měli zjistit, jak zachytávání zastavit, protože provoz bude pravděpodobně dost velký. Na obrázku vpravo vidíme ikonu, kterou to lze provést velmi rychle.



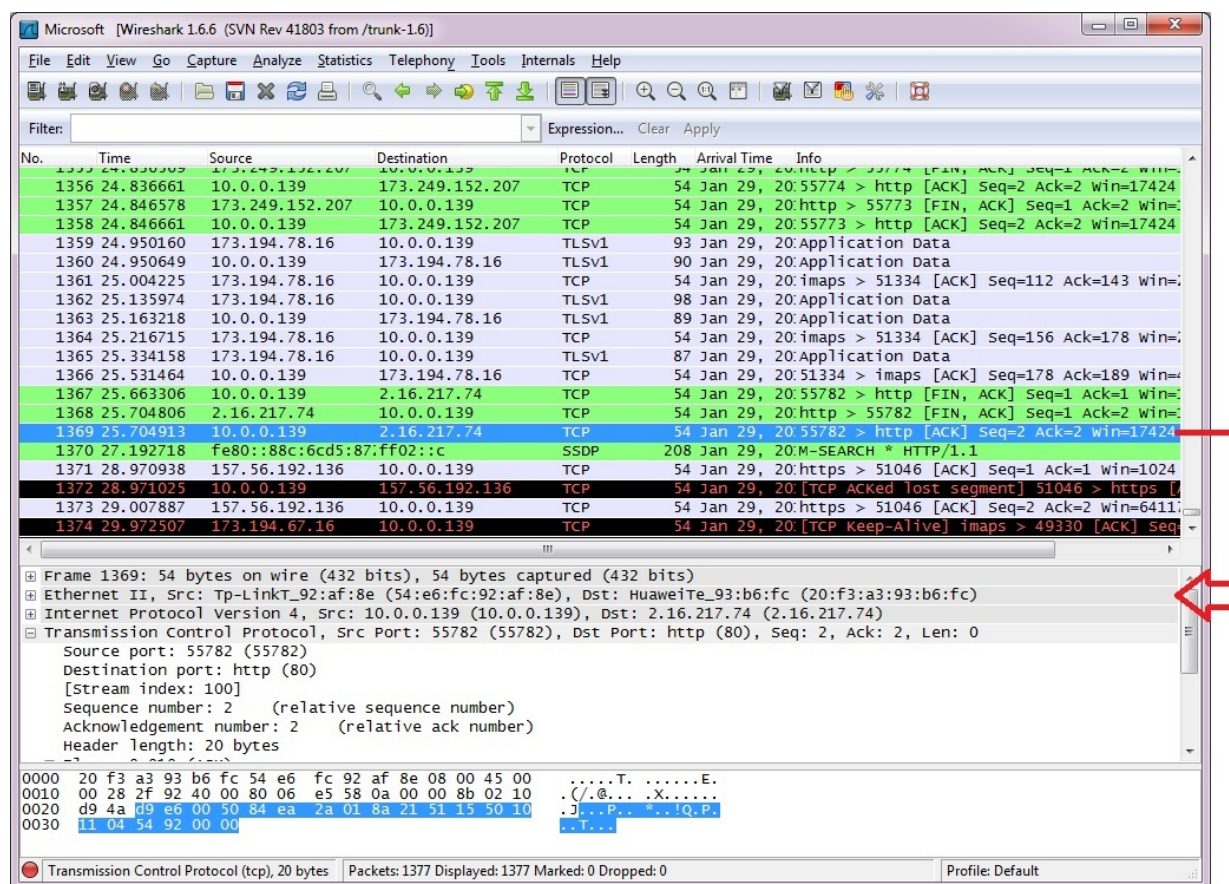
Na obrázku 4.3 vidíme okno po zastavení zachycování. V horní části okna je seznam zachycených paketů, z nichž jeden je vybrán (stačí na některý klepnout myší). Pod tímto podoknem je další část okna, ve které vidíme údaje o vybraném paketu. Ve spodní části okna je pak „rozbalená“ informace, která byla v paketu nalezena. Pokud je obsah šifrovaný nebo binární, nebude text srozumitelný.

Pokud byla některá položka vybrána, v prostřední části okna o ní zjistíme podrobnosti. Je tam několik „podstromů“ s údaji jednotlivých úrovní zapouzdřených PDU. První podstrom obsahuje provozní informace o rámci (časové údaje, délku rámce, které protokolové datové jednotky byly postupně zapouzdřeny) a pak informace od Wiresharku (například o orientačním zabarvení paketu v horním okně).

Půjde o ethernetový rámec, tedy další části budeme srovnávat s následujícím nákresem (informace o něm jsou ve skriptech pro přednášky):



Druhý podstrom obsahuje záhlaví vrstvy L2, tedy kromě preamble, ta byla oddělena už před analýzou, a kromě zápatí, které bylo taktéž předem odděleno (když proběhla kontrola neporušenosti rámce). Na následujícím obrázku je tento podstrom rozbalený:



Obrázek 4.3: Provoz zachycený Wiresharkem

```

Frame 8: 233 bytes on wire (1864 bits), 233 bytes captured (1864 bits)
Ethernet II, Src: Hewlett-5e:4d:26 (00:1f:29:5e:4d:26), Dst: Vmware-bb:3a:a0 (00:50:56:bb:3a:a0)
  Destination: Vmware-bb:3a:a0 (00:50:56:bb:3a:a0)
    Address: Vmware-bb:3a:a0 (00:50:56:bb:3a:a0)
    ....0 .... = IG bit: Individual address (unicast)
    ....0 .... = LG bit: Globally unique address (factory default)
  Source: Hewlett-5e:4d:26 (00:1f:29:5e:4d:26)
    Address: Hewlett-5e:4d:26 (00:1f:29:5e:4d:26)
    ....0 .... = IG bit: Individual address (unicast)
    ....0 .... = LG bit: Globally unique address (factory default)
  Type: IP (0x0800)
Internet Protocol Version 4, Src: 74.125.131.27 (74.125.131.27), Dst: 192.168.20.70 (192.168.20.70)
Transmission Control Protocol, Src Port: smtp (25), Dst Port: 54557 (54557), Seq: 52, Ack: 7, Len: 167
Simple Mail Transfer Protocol

```

Nejdřív je uvedena adresa cíle (Destination Address) a hned za ní adresa cíle (Source Address). Po rozbalení jejich vnitřních podstromů jsou přehledně zvýrazněny pozice speciálních bitů I/G (Individual/Group) a L/G (Local/Global). U obou adres jsou oba tyto bity nastaveny na nulu, tedy se jedná o individuální globální adresy. Zbývající bity jsou znázorněny tečkami (ve skupinách po čtyřech, tedy poloviny oktetu).

Poslední část záhlaví rámce je EtherType, zde je toto pole jednoduše nazvané Typ. Hodnota 0×0800 znamená, že uvnitř rámce najdeme IP paket. Rozbalováním dalších větví bychom se dostali dál „dovnitř“, zatím nám bude stačit vědět, kde najdeme údaje o rámci.



Úkol

Pokud máte možnost, nainstalujte si aplikaci Wireshark a vyzkoušejte ji. Upozornění: v těchto skriptech jsou screenshoty starší verze. Novější verze vypadá trochu jinak (především její úvodní okno), tedy se nelekejte – i v novější verzi najdete na titulní straně seznam síťových rozhraní. Stačí vybrat, klepnout či poklepat, a můžete zachytávat.



Další informace:

- <http://www.wiresharktraining.com/>
- https://www.wireshark.org/docs/wsug_html_chunked/



Poznámka:

Na webu je hodně stránek nabízejících ke stažení soubory se zachyceným síťovým provozem. Tyto soubory jsou výborné pro procvičení, můžeme na nich vidět typický provoz s využitím určitých konkrétních protokolů. Některým těmto stránkám je spíše lepší se vyhnout, ale jiné stojí za vyzkoušení, například:

- <https://wiki.wireshark.org/SampleCaptures>
- <http://packetlife.net/captures/>
- <http://www.netresec.com/?page=PcapFiles> (doporučuji spíše až po nabytí rozsáhlejších znalostí v oblasti počítačových sítí, hlavně u odposlechů škodlivého softwaru)
- <https://www.nostarch.com/packet2.htm> (na odkazu „Download the capture files for this book“) jsou soubory se zachyceným provozem jako příloha ke knize [2])





Úkol

Na stránce <https://wiki.wireshark.org/SampleCaptures> najdete odkaz na soubor `smtp.pcap` (použijte funkci vyhledávání na stránce, stránka je opravdu hodně dlouhá). Soubor stáhněte, otevřete (pokud máte nainstalován Wireshark, stačí poklepat na soubor) a prozkoumejte.

- V horním podokně se podívejte na význam jednotlivých sloupců. Také si zde všimněte, mezi kterými dvěma adresami komunikace probíhá (jsou to IPv4 adresy). Pamatujte si, že komunikaci s (jakýmkoliv) serverem vždy začíná klient. První dva řádky poněkud vybočují z následné komunikace, protože odpovídají zjišťování adresy serveru (klient má pouze jmenovou adresu serveru, potřebuje IP adresu). Zatím se můžeme zaměřit na pakety od třetího dále. Takže která adresa je od klienta a která od serveru?
- Ve sloupci *Protocol* je většinou protokol SMTP, který slouží ke komunikaci s e-mailovou schránkou, přičemž odesíláme e-mail. Dále zde najdeme protokol TCP, což znamená, že tyto pakety slouží k navázání, údržbě a ukončení TCP spojení (na transportní vrstvě). Jiných protokolů si zatím nemusíme všimnout.
- Projděte si alespoň část paketů (zejména na začátku, několik uprostřed a pak konec). Prohlédněte si údaje o rámci, ale také další části.
- Všimněte si, že některé pakety zřejmě došly poškozené (většinou červené písmo na černém pozadí). Některý z těchto paketů si vyberte a pokuste se zjistit, kde konkrétně nastala chyba (poznáte podle zvýraznění).



Určitě jste už pochopili, že Wireshark nám některé informace sděluje pomocí barev. Některé barevné kombinace je dobré si pamatovat, nebo alespoň vědět, kde je zjistíme.



Úkol

V menu Wiresharku si najdete *View – Coloring Rules*. Prohlédněte si momentální nastavení.



4.2 Zařízení v lokální síti

Na přednáškách jsme si říkali o různých typech aktivních síťových prvků (v terminologii Ethernetu DCE), nyní si je stručně zopakujeme.



Repeater (opakovač) dokáže zesílit příchozí signál (případně znovu vygeneruje) a poslat dál. Pracuje na vrstvě L1, pracuje pouze se signálem nebo nanejvýš proudem bitů, obvykle nepotřebuje příliš výkonný procesor (až tak moc toho nedělá).



Hub (rozbočovač) je v podstatě takový opakovač, který má více než dva porty. Cokoliv přijde na některý port, jen přepoše na všechny ostatní porty (případně signál zesílí nebo znovu vygeneruje). Stejně jako repeater, i hub pracuje pouze se signálem či proudem bitů, nedokáže se podívat „dovnitř“ paketu. Také pracuje na vrstvě L1.



Switch (přepínač) je aktivní síťový prvek pracující na vrstvě L2. Vede si tabulku adres uzlů svého segmentu (ke každému uzlu, který „zná“, má poznamenáno, přes který port je tento uzel dosažitelný). Switche jsou v lokálních (i jiných) sítích zřejmě nejběžnějšími DCE.



Switch s přidanou funkcionalitou vrstvy L3 je takový přepínač, kterému, jak název napovídá, byla přidána funkcionalita vrstvy L3 (tedy síťové). Znamená to, že se sice pořád jedná o switch, ale některé porty tohoto zařízení se mohou chovat jako porty routeru (což je typické zařízení pro L3) a bude možné se souvisejícím provozem zacházet na vyšší úrovni (rozbalit PDU až na úroveň vrstvy L3, pracovat s IP adresami apod.).



Bridge (most) je zařízení navzájem oddělující dva segmenty sítě, podobně jako switch, a také pracuje na L2. Vše, co umí bridge, umí i switch (ale naopak to až tak moc neplatí), algoritmy určené pro mosty bývají implementovány na přepínačích.



Router (směrovač) je aktivní síťový prvek pracující na vrstvě L3. Taký si vede tabulku (říkáme jí *směrovací tabulka*), v ní najdeme adresy sítí či podsítí, ve kterých se nacházejí dostupné uzly. Pro routery je typický více výpočetně náročný provoz než u switche, proto mají méně portů než switche, výkonnější procesor a více paměti, aby vyšší výpočetní zátěž unesly.

Routery sice implementují vrstvu L2, ale pouze v základu, aby dokázaly rozbalit rámec a dostat se k zapouzdřenému paketu. Pokročilejší možnosti protokolů vrstvy L2 zde nehledíme (v tom se liší od switchů s funkcionalitou vrstvy L3).



Brána (gateway) slouží k propojení dvou různých typů sítí, resp. slouží jako spojovací bod či „překladatel“. Brány pracují na vyšších vrstvách (podle konkrétního protokolu).

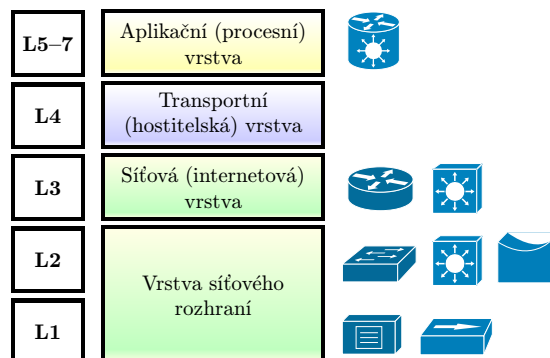
Abychom si ujasnili vztah různých DCE k vrstvám ISO/OSI nebo lépe síťového modelu TCP/IP (DoD), podívejme se na obrázek vpravo. Na vrstvě L1 pracují repeatery (opakovače) a huby (rozbočovače), na vrstvě L2 především switche (přepínače, včetně těch s funkcionalitou vrstvy L3) a bridge (mosty), na vrstvě L3 routery (směrovače) a switche s funkcionalitou vrstvy L3, na aplikační vrstvě pak brány (gateway).

Ujasníme si ještě následující pojmy:



Segment sítě je skupina propojených zařízení v síti takových, že pokud jedno z těchto zařízení odešle PDU, pak tento PDU obdrží všechna ostatní zařízení z této skupiny (jednotlivá zařízení pak PDU buď přijmou, když jsou adresáty, nebo zahodí, když je PDU určen pro jiné zařízení, ale pokud síťové rozhraní běží v promiskuitním módu, přijme cokoliv doručeného).

Pokud se jedná o všesměrové (broadcast) vysílání, PDU je odeslán i mimo segment, totéž (většinou) platí o PDU s neznámým adresátem.




Poznámka:

V segmentu jsou jednotlivá koncová zařízení (DTE) propojena pomocí rozbočovačů nebo opakovaců (dnes spíše jen opakovaců). Naopak přepínače (a případně směrovače) oddělují segmenty,

takže přepínač se může nacházet pouze „na hranici“ segmentu. Uvědomte si, že „na hranici“ segmentu jsou proto pouze buď přepínače nebo koncová zařízení, kdežto uvnitř pouze opakovače či rozbočovače.



 *Kolizní doména* je skupina takových zařízení v síti, která sdílejí tentýž segment. Přepínače oddělují kolizní domény.



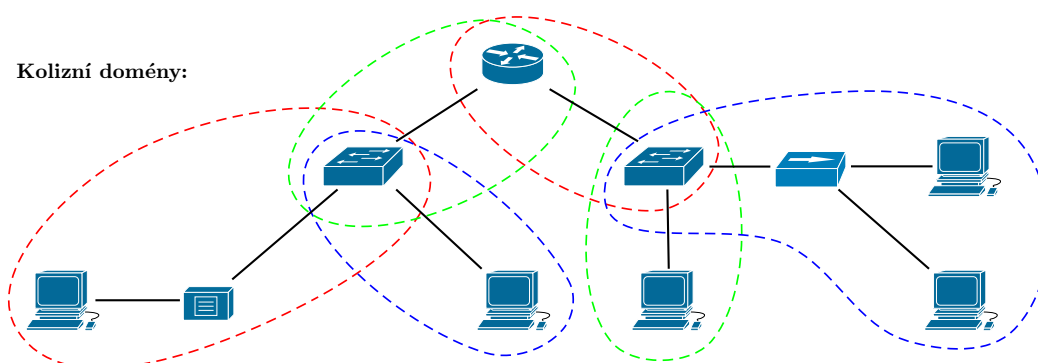
Poznámka:

To znamená, že pokud (hypoteticky) používáme rozbočovač, pak on i všechna k němu připojená zařízení jsou v téže kolizní doméně (stejném segmentu), ale když ho nahradíme přepínačem, rázem dostaneme tolik kolizních domén (segmentů), kolik zařízení je k přepínači připojeno.




Příklad 4.2

Prozkoumejte následující obrázek. Je na něm jeden router (nahore), dále dva switche, jeden hub, jeden repeater a několik počítačů. Barevně jsou vyznačeny různé kolizní domény, resp. segmenty.



Jak vidíme, switche i routery jsou vlastně na hranicích segmentů, kdežto hub i repeater jsou „uvnitř“. Switch a router oddělují kolizní domény, tedy pro každý port (a tedy každé připojené zařízení) u nich existuje samostatná kolizní doména. Router patří do dvou kolizních domén, protože jsou k němu připojena dvě zařízení. Levý switch patří do tří kolizních domén, protože jsou k němu připojena tři zařízení, o pravém switchi platí totéž.



 *Síť* je skupina fyzicky navzájem propojených zařízení navzájem přijímajících všesměrové vysílání, přičemž tato zařízení jsou členěna do síťových segmentů. Hovoříme také o *všesměrové (broadcast) doméně*.



Poznámka:

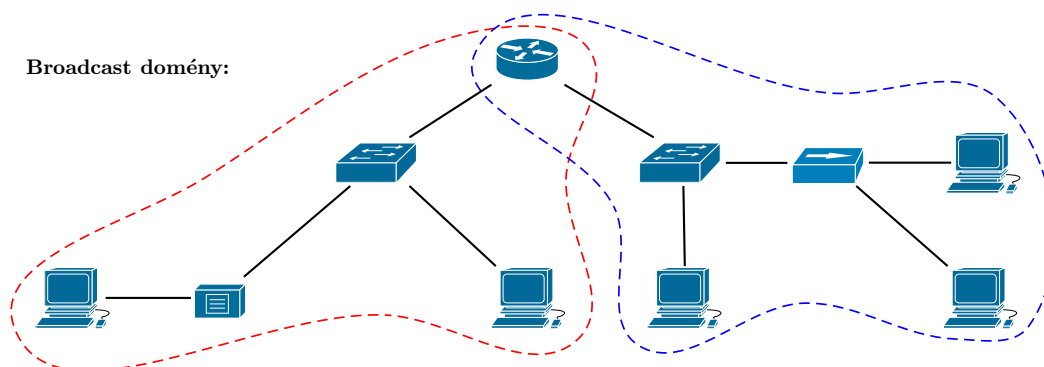
Na hranici sítě jsou buď koncová zařízení nebo routery, protože koncová zařízení jsou finálními příjemci všesměrového vysílání (nic nepřeposílají dál) a routery vlastně také (protože všesměrové vysílání zahazují, nepřeposílají). Routery propojují a zároveň oddělují různé sítě.





Příklad 4.3

Na následujícím obrázku jsou barevně znázorněny všesměrové (broadcast) domény.

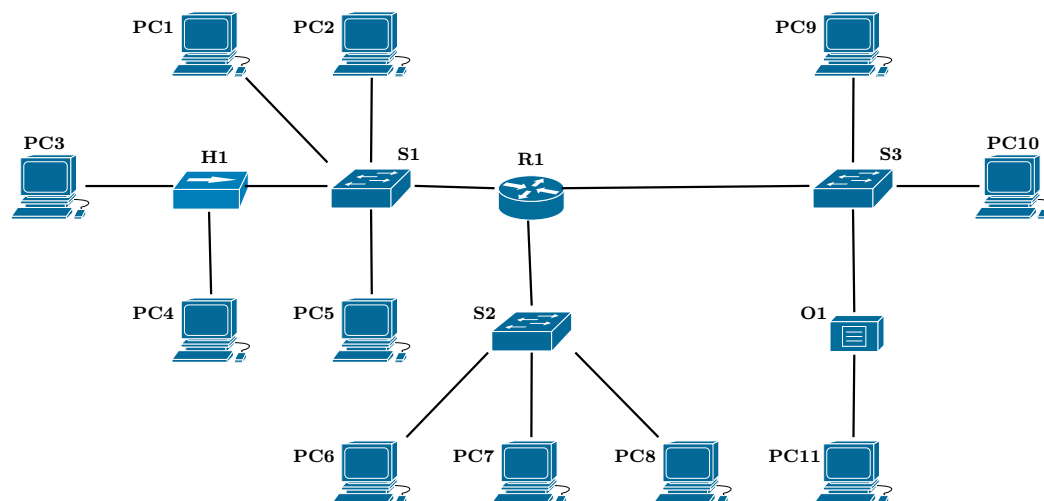


Všesměrové domény dokáže oddělit pouze router, který máme na obrázku jen jeden. K routeru jsou přímo připojena dvě zařízení, proto máme dvě všesměrové domény.



Úkol

Na následujícím obrázku zjistíte všechny kolizní a všesměrové domény – Ke každé kolizní doméně vypíšete seznam zařízení v ní, totéž pro všesměrové (nápopěda: kolizních je 13).



Příklad 4.4

Zaměříme se znovu na obrázek v předchozím úkolu. Pokud určité konkrétní zařízení vyšle unicast rámeček, která zařízení tento rámeček obdrží? Uvědomte si, že pokud běží síťové rozhraní zařízení v promiskuitním módu, pak toto zařízení přijme cokoliv, co k němu dojde (tedy cokoliv, co se „potuluje“ po segmentu), ať už je či není skutečným adresátem.

- odesílatel: PC1, adresát: PC5
 - ⇒ cesta je PC1 → S1 → PC5, kromě těchto zařízení k rámcu nikdo jiný nemá přístup,
 - předpokládá se, že S1 má v tabulce MAC adres adresu zařízení PC5,

- odesílatel: PC1, adresát: PC4
 - ⇒ cesta je $PC1 \rightarrow S1 \rightarrow H1 \rightarrow (PC3, PC4)$, z koncových zařízení (DTE) má k rámci přístup PC3, třebaže není adresátem,
 - předpokládá se, že S1 má v tabulce MAC adres adresu zařízení PC4,
- odesílatel: PC4, adresát: PC1
 - ⇒ cesta je $PC4 \rightarrow H1 \rightarrow (PC4, S1) \rightarrow PC1$, opět se k rámci dostane i PC3,
 - předpokládá se, že S1 má v tabulce MAC adres adresu zařízení PC1,
- odesílatel: PC2, adresát: PC8
 - ⇒ cesta je $PC2 \rightarrow S1 \rightarrow R1 \rightarrow S2 \rightarrow PC8$,
 - předpokládá se, že S1 má v tabulce MAC adres adresu zařízení R1 (ale ne PC8!), S2 má v tabulce MAC adres adresu zařízení PC8 (ale ne PC2), a dále router R1 (který rozbalil nejen rámec, ale i IP paket uvnitř) dokáže IP adresu zařízení PC8 zařadit do správné sítě a odeslat správným směrem,
- odesílatel: PC3, adresát: PC4
 - ⇒ cesta je $PC3 \rightarrow H1 \rightarrow (PC4, S1)$,
 - switch S1 tento rámec zahodí, protože přišel z portu, na kterém je cíl (zpět se nic neposílá), hub H1 nemá žádnou tabulku,
- odesílatel: PC10, adresát: PC11
 - ⇒ cesta je $PC10 \rightarrow S3 \rightarrow O1 \rightarrow PC11$,
 - předpokládá se, že S3 má v tabulce MAC adres adresu zařízení PC11.

A jak je to s všesměrovými rámci? Připomeňme si, že switch posílá všesměrové rámce na všechny porty kromě toho, ze kterého přišel, hub a repeater taktéž, ale router (směrovač) všesměrovou komunikaci zahazuje.

- odesílatel: PC1, adresát: FF – FF – FF – FF – FF – FF (všesměrová)
 - ⇒ cesta je $PC1 \rightarrow S1 \rightarrow (PC2, PC5, H1, R1) \rightarrow (PC3, PC4)$,
 - switch S1 odesílá rámec na všechny porty kromě toho, na kterém je PC1, hub H1 na všechny porty kromě toho od S1, router R1 rámec zahazuje a dál neposílá (nicméně pokud na něj má reagovat, pak reaguje),
- odesílatel: PC6, adresát: FF – FF – FF – FF – FF – FF (všesměrová)
 - ⇒ cesta je $PC6 \rightarrow S2 \rightarrow (PC7, PC8, R1)$,
 - switch S2 odesílá rámec na všechny porty kromě toho od PC6, router R1 rámec zahazuje a dál neposílá.



Úkol


Zjistěte, která zařízení (podle obrázku z předchozího úkolu) mohou zachytit ethernetový rámec, jestliže je odesílatel a adresát následující:

- odesílatel: PC2, adresát: PC1,
- odesílatel: PC4, adresát: PC5,
- odesílatel: PC9, adresát: PC10,

- odesílatel: PC7, adresát: PC1,
- odesílatel: PC2, adresát: broadcast,
- odesílatel: PC10, adresát: broadcast.



4.3 Tabulka MAC adres na switchi

 *Tabulka MAC adres* je tabulka, kterou si (nejméně jednu) vede každý switch a vlastně každé aktivní síťové zařízení s implementovanou vrstvou L2. V této tabulce jsou informace o dostupných uzlech v síti – tím je míněna síť až po nejbližší router včetně. Této tabulce 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 (přepínač) 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 broadcastová, je rámec také poslán na všechny porty kromě toho, ze kterého přišel.

Ale jak se vlastně dotyčný adresát do té tabulky dostane?

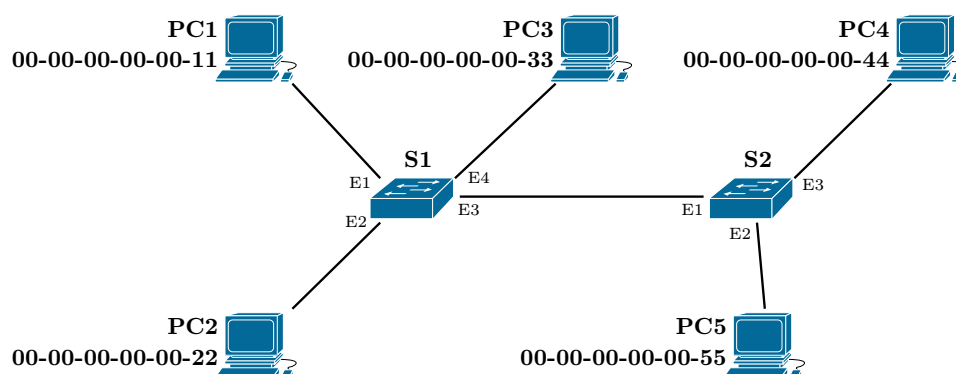
 Předpokládejme, že máme nový (nebo restartovaný) switch a poprvé ho zapojíme, tabulka je prázdná. V takovém případě jsou dvě možnosti, jak tabulku naplnit. Buď jsou do ní příslušné údaje „natvrdo“ přidány pomocí ruční konfigurace (či pomocí skriptu), nebo to jednoduše necháme na dotyčném switchi, ať 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 4.5

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	...

ovšem 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.



Poznámka:

Určitě jste si všimli, že k switchi S1 se sice přes port E3 dostal rámec od switche S2, ale adresa switche S2 v tabulce (zatím) není. Je to proto, že switch S2 zatím nebyl přímo odesílatelem žádného rámce procházejícího přes S1, pouze takové rámce přeposílal. Ovšem pokud by switch S2 sám odeslal rámec směrem k switchi S1, jeho adresa by se samozřejmě do tabulky taky dostala.

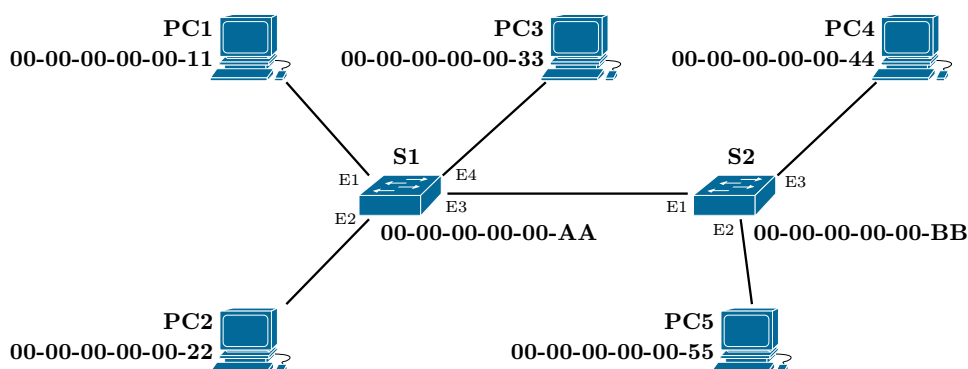
Switche mezi sebou komunikují například při použití protokolu STP, jehož účelem je zabránit logickým smyčkám v grafu switchů, aby nedocházelo k zacyklování provozu. STP je jedním z protokolů vrstvy L2, přičemž je zapouzdřován do LLC rámce (IEEE 802.2) a následně do MAC rámce IEEE 802.3.



Úkol

Zaměřte se na následující obrázek. Předpokládejte, že oba switche mají prázdné MAC tabulky a podle následujícího popisu provozu tyto tabulky postupně sestavte (vždy MAC adresu a číslo portu).

Ke všem níže uvedeným případům uveďte, jak který switch na daný rámec reaguje (kam ho přepošle). Dejte pozor na to, která adresa je pro který účel použita.



1. Odesílatel je PC3, cíl je PC2.
2. Odesílatel je PC4, cíl je PC1.
3. Odesílatel je PC5, cíl je PC4.
4. Odesílatel je PC1, cíl je PC2.
5. Odesílatel je PC2, cíl je PC1.
6. Odesílatel je S1, cíl je PC4.
7. Odesílatel je S2, cíl je PC4.
8. Odesílatel je S2, cíl je PC3.



**Poznámka:**

Může nastat situace (a taky běžně nastává), že v tabulce máme k jednomu portu víc MAC adres. Nezapomeňte, že za portem switchu může být buď koncové zařízení, nebo hub či switch, ke kterému bývá připojeno více dalších zařízení: právě jejich adresy budou k tomuto portu přiřazeny.



Doplňování záznamů do MAC tabulky se zdaleka neprovádí jen při „restartu“ switchu. Každý záznam (řádek) tabulky má totiž jen určitou dobu platnosti (proto bylo na začátku této sekce u struktury MAC tabulky zmíněno časové razítko). Konkrétní doba se liší podle výrobce, případně se dá konfigurovat. Může to být například 5 minut (což je typické pro síť s větším provozem), po uplynutí této doby je záznam odstraněn, případně může být časové razítko aktualizováno vždy, když z dané adresy přijde rámec – to je opatření, jehož účelem je pravidelně z tabulky odstraňovat neaktivní (například vypnutá či poškozená) zařízení.

Dalším důvodem odstranění záznamu z MAC tabulky je vyčerpání kapacity paměti určené pro tuto tabulku. Když dojde paměť, jsou novými záznamy přepisovány ty nejstarší.

**Poznámka:**

Pokud se switch příliš často chová jako hub (tedy provoz přeposílá na téměř všechny porty místo přeposílání na jediný), pak to samozřejmě škodí propustnosti sítě, síť je zbytečně zahlcována.

Tuto situaci úmyslně navozuje také jeden z typů útoků na síť – *MAC flooding*, kdy je switch „bombardován“ rámci s různými zdrojovými MAC adresami (včetně podvržených), což switch nutí k neustálému přidávání nových záznamů do MAC tabulky. Postupně jsou z tabulky vytěsnány všechny „správné“ záznamy, proto všechny posílané rámce mají vlastně neznámou adresu cíle (tabulka je přepsaná), veškerý provoz musí být posílán na všechny porty (kromě toho portu, ze kterého pochází).

**Další informace:**

<http://www.samuraj-cz.com/clanek/vite-jak-pracuje-switch/>




4.4 Kabely

4.4.1 Metalická kabeláž

V historii se pro Ethernet používal koaxiální kabel, ale v současné době je na vzdálenosti do 100 metrů používána kroucená dvojlinka (twisted pair). Je to kabel obsahující většinou 8 vodičů vždy po dvou zkroucených (4 dvojlinky). Je důležitá jeho kategorie – vyšší kategorie má lepší fyzikální vlastnosti nebo lepší stínění, což je důležité pro dosažení vyšších rychlostí a vyšších vzdáleností (nebo obojího najednou). Nižší kategorie mohou mít méně než 4 páry vodičů.


Novější standardy mají obvykle přísnější požadavky na kategorii použitého kabelu.

 Kabel bez stínění je označován jako *UTP* (Unshielded Twisted Pair), kabel se stíněním poněkud nepřesně jako *STP* (Shielded Twisted Pair). Ve skutečnosti může být stínění ve formě hliníkové fólie buď kolem jednotlivých dvojlinek (vnitřní) nebo kolem celého svazku dvojlinek (vnější), případně vnější stínění může být realizováno opletením.


V následující tabulce je přehled kategorií.

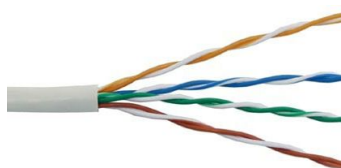
Kategorie	Použití	Stínění
Cat.1, Cat.2	V současné době se nepoužívá.	ne
Cat.3	Většinou telefonní rozvody. V počítačových sítích pro starý 10Mb Ethernet.	ne
Cat.4	Pro síť Token Ring. V současné době se tedy nepoužívá.	ne
Cat.5	Pro Fast Ethernet (100 Mb/s). Některé kvalitnější kabely se dají použít i pro Gigabit Ethernet, ale nedoporučuje se.	ne
Cat.5e	Pro Fast Ethernet a Gigabit Ethernet. V současné době nejpoužívanější kabeláž pro lokální síť.	ne
Cat.6	Pro Gigabit Ethernet a vyšší, ale pro 10G je omezen na cca 55 m.	ne
Cat.6A	Podobně jako Cat.6, ale má větší šířku pásma a i pro 10G přenáší až na 100 m.	ano
Cat.7	Pro 10G Ethernet a vyšší, na rozdíl od Cat.6A je plně stíněný a nabízí ještě vyšší šířku pásma.	ano
Cat.7A	Další vylepšení, ještě větší šířka pásma.	ano

V tabulce najdete na většině řádků informaci, že kabel dané kategorie není stíněný. Ve skutečnosti lze sehnat i stíněné kabely kategorie 5e a 6, ale moc se nepoužívají. Pokud někdo investuje do stíněného kabelu, obvykle raději vybere vyšší kategorii i kvůli případnému budoucímu přechodu sítě na vyšší standardy.

 Co se týče stínění, v praxi se nejčastěji setkáváme s typy uvedenými v tabulce vpravo (případně se objevuje nepřesné označení STP), označení podle standardu ISO/IEC 11801. Existují další možné kombinace stínění (po chvíli zkoumání určitě zjistíte, co znamená písmeno „F“ a „S“ v označení na konkrétním místě).

Označení	Stínění párů	Stínění svazku párů
UTP, U/UTP	žádné	žádné
U/FTP	hliníková fólie	žádné
F/UTP	žádné	hliníková fólie
S/FTP	hliníková fólie	kovové opletení
SF/UTP	žádné	hliníková fólie + kovové opletení

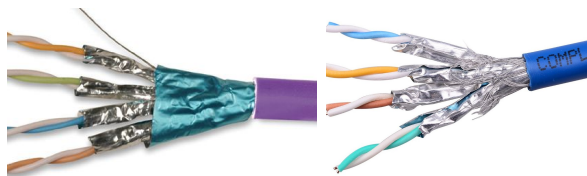
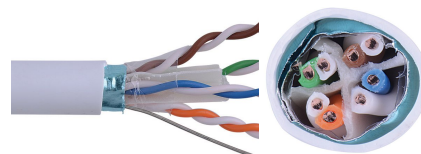
 Z výše uvedených je momentálně nejpoužívanější UTP (nestíněný) kabel kategorie 5e.




Vlevo je běžný kabel UTP Cat.5e. Vidíme 4 páry (tedy 8 vodičů), každý pár je do sebe zakroucený, díky čemuž je potlačeno vzájemné rušení. Pro odeslání signálu je určen vždy celý pár, signál je reprezentován rozdílem potenciálů obou vodičů.

Jak vidíme, každý vodič má izolaci v jiné barvě, přičemž v páru je vždy jeden vodič v plné barvě a druhý bílý s (někdy bohužel málo znatelným) proužkem v dané barvě – na obrázku to asi není moc dobře vidět, v reálu to bude patrnější.

Vpravo vidíme kabel F/UTP Cat.6. Páry nemají žádné zvláštní stínění, jen běžnou barevnou izolaci, kdežto celý svazek párů je zabalen do hliníkové fólie coby přidaného stínění. Kabel kategorie 6 a vyšší může mít uvnitř středový plastový oddělovač s průřezem „kříže“, který od sebe odděluje čtyři dvojlinky tak, aby se ještě více snížilo rušení, navíc je kabel celkově pevnější. Středový oddělovač může být i u Cat.5e, ale ne obvykle. Na obrázku ještě více vpravo je průřez celým kabelem, na kterém vidíme umístění jak dvojlinek, tak i středového oddělovače a hliníkového stínění.



Zcela vlevo vidíme kabel F/FTP Cat.7. Hliníková fólie je jak kolem jednotlivých párů, tak i kolem celého svazku. Vedle je kabel S/FTP, kde je hliníková fólie kolem párů, ale kolem celého svazku máme kovové opletení.

 Když kupujeme kabel typu kroucená dvojlinka, setkáme se s těmito parametry:

- kategorie,
- stínění – U/UTP, U/FTP, F/UTP, F/FTP, S/FTP, atd.,
- průměr měděného vodiče – většinou AWG23 nebo AWG24; menší číslo znamená větší průměr vodiče, silnější vodič znamená větší trvanlivost, ale taky o něco náročnější nasouvání konektoru,
- má/nemá středový oddělovač,
- drát/licna – kabely typu drát jsou určeny na pevné instalace (do zdi), kdežto licna (lanko) jsou pružnější kabely pro připojení přímo k zařízení,
- metráž/patch cord – metráž nemá konektory (často se prodává navinutá na cívce, typicky 305 nebo 500 m), patch kabel je opatřen konektory a připojuje se přímo k zařízení (bývá kratší).



Další informace:

Kabely se nejlépe srovnávají v internetových obchodech orientovaných na zboží z oblasti počítačových sítí. Můžete se podívat například na

- <http://www.intelek.cz>
- <http://www.softcom.cz>
- <http://www.lan-shop.cz>
- <http://www.itage.cz> (trochu chaotické, ale ve většině skladů něco najdete)



Úkol

Vyberte si jeden z výše uvedených internetových obchodů (třeba ten první) nebo i více a projděte si nabídku ethernetových kabelů kategorie 5e a 6, případně 6A. Všimněte si výše uvedených parametrů – stínění, průměr vodiče a dalších, podle toho, co je uvedeno. Také si všimněte ceny (u metráže případně přepočtete na délku 100 m, aby byla porovnatelná).



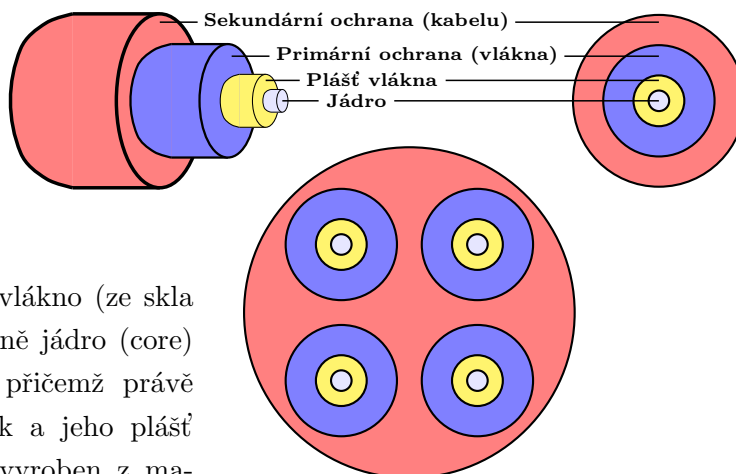
4.4.2 Optická kabeláž

Optické kabely (fiber optic cable) je potřeba použít především tam, kde nevystačíme se vzdáleností do 100 m. Optické kabely lze použít na vzdálenosti ve stovkách metrů nebo jednotkách či desítkách kilometrů (podle druhu).

Optický kabel se skládá z jednoho nebo více optických vláken obklopeného (obklopených) ochrannými vrstvami. Na obrázku vpravo je načrtnuta zjednodušená struktura optického kabelu obsahujícího jedno

vlákno a níže čtyři vlákna. Samotné vlákno (ze skla nebo plastu podobného sklu) je vlastně jádro (core) obklopené svým pláštěm (coating), přičemž právě jádrem se pohybuje světelný paprsek a jeho plášť slouží jako odrazové „zrcadlo“ – je vyroben z materiálu, který má podobné fyzikální vlastnosti jako

jádro, ale výrazně odlišný index lomu. Navíc zvyšuje pevnost vlákna a tím i jeho odolnost. Následuje primární ochrana vlákna (to často bývá speciální lak). Sekundární ochrana může být například kevlarová příze, některá vhodná plastická hmota, gel apod. Kolem pak bývá plášť kabelu případně kombinovaný s dalšími ochranami, záleží na požadavcích na zabezpečení kabelu proti mechanickému poškození, vodě, chemickému poškození, UV záření, hlodavcům apod. Nejsložitější ochranu mají samozřejmě podmořské kabely.



Podívejme se nyní na to, jak vlastně optická vlákna vedou signál. Optický signál o určité vlnové délce je vygenerován na jednom konci kabelu a musí být dovezen na jeho druhý konec. To není zrovna jednoduché, protože kdykoliv paprsek narazí na hranice mezi jádrem a jeho pláštěm, odrazí se pod (téměř) stejným úhlem,

pod jakým dopadl; zdroj světla vygeneruje samozřejmě více než jeden foton, a je možné, že různé fotony budou vyslány pod mírně odlišnými úhly. Tyto úhly se pak po cestě mohou násobnými odrazy ještě dále měnit, a platí, že čím větší úhel, tím déle fotonu daná cesta trvá.

Co z toho vyplývá? Čím delší cesta, tím větší je pravděpodobnost, že se svazky fotonů příslušejících jednotlivým posílaným bitům „pomíchají“ a v cíli nebude možné správně rozpoznat hranice mezi bity, ani správné hodnoty těchto bitů. Jsou dvě řešení tohoto problému: buď maximalizujeme přesnost nebo snížíme efektivní dosah (maximální vzdálenost kabelu).

Maximalizovat přesnost můžeme předně tak, že použijeme velmi kvalitní zdroj světla (laser místo LED diod) a zároveň co nejvíc zmenšíme průměr jádra vlákna, čímž zmenšíme a zpřesníme úhel odrazu, resp. zminimalizujeme množství odrazů (v ideálním případě poletí fotony pořád rovně bez odrazů).

Existují dva druhy optických vláken:

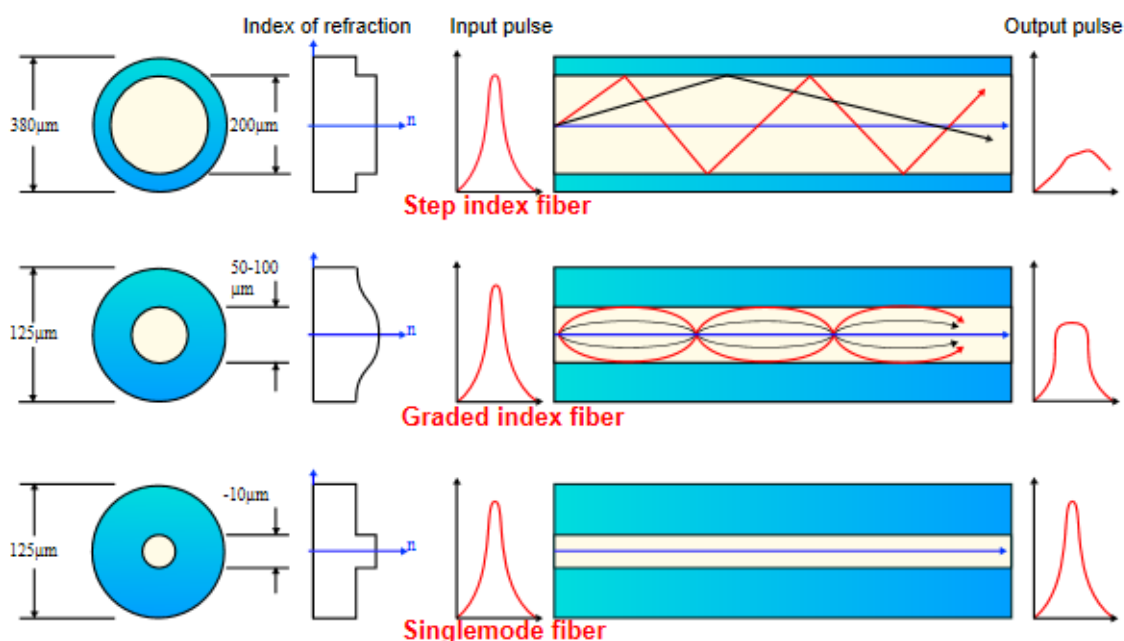
- *jednovidové vlákno* (single-mode fiber) má velmi malý průměr jádra a používá kvalitní zdroj světla, je dražší, ale má větší efektivní dosah (jednotky až desítky kilometrů),

• *vícevidové (mnohavidové) vlákno* (multi-mode fiber) má větší průměr jádra, na kvalitu zdroje světla jsou menší požadavky, je levnější, jeho efektivní dosah je spíše menší (stovky metrů). Liší se také rozsahy vlnových délek přenášeného signálu, jednovidová vlákna používají vyšší vlnové délky.

Ve vícevidových vláknech tedy dochází k odrazům signálu na rozhraní jádra a jeho pláště, rozeznáváme dvě provedení podle indexu lomu pláště:

- *vlákna se skokovou změnou indexu lomu* (step index fiber), kde index lomu v celém plášti stejný (a tedy úhel odrazu je pořád stejný),
- *vlákna s plynulou (gradientní) změnou indexu lomu* (graded index fiber), kde se index lomu pláště ve směru od jádra postupně snižuje.

Na obrázku 4.4 je naznačen průchod signálu v obou typech vícevidových vláken a v jednovidovém vláknu. Neoptimálněji je signál převáděn v jednovidovém vláknu, následuje vícevidové vlákno s gradientní změnou indexu lomu.



Obrázek 4.4: Různé druhy optických vláken¹

V optickém kabelu nemusí nutně být jen optická vlákna. Mohou jím být vedena také metalická vlákna (například kvůli napájení, to by přes samotnou optiku nešlo), případně různé výztuže ovlivňující pevnost kabelu.



Další informace:

- <http://www.chip.cz/novinky/trendy/opticke-site/>
- <http://www.root.cz/clanky/opticke-kabely-v-oceanu-tudy-dopravdy-tece-internet/>
- <http://fyzika.jreichl.com/main.article/view/557-opticka-vlakna>
- http://pctuning.tyden.cz/index.php?option=com_content&view=article&id=9994&catid=32&Itemid=72



¹Zdroj: <https://reggle.wordpress.com/2013/02/03/osi-layer-1-part-ii-fiber/>

**Poznámka:**

Srovnajme si vlastnosti metalických a optických kabelů:

- samozřejmě je odlišná reprezentace signálu,
- s tím souvisí fakt, že při použití optiky musí být na obou stranách prováděna transformace mezi optickým a elektrickým signálem,
- rušení – u metalických kabelů může docházet k elektromagnetickému rušení mezi jednotlivými páry i zvenku kabelu, u optického kabelu to nehrozí, rušení může být způsobeno pouze UV zářením (vlnová délka odpovídá přenášenému signálu), ale to se dá řešit optickou izolací,
- vzdálenost – metalické cca 100 m, optické stovky metrů až desítky kilometrů (tj. optické kabely mají menší útlum),
- napájení – přes metalický kabel lze (ne vždy) také napájet, u samotné optiky to není možné,
- cena – metalické kabely jsou značně levnější a totéž platí o koncovkách a propojovacích prvcích,
- fyzikální vlastnosti – metalické kabely jsou pružnější, ale snadněji se poškodí, na druhou stranu poškození se snadněji opraví (přinejhorším propojkou),
- krimpování (přípevňování koncovek) je u metalických kabelů jednodušší, atd.



Pro ilustraci si uvedeme několik konkrétních údajů:

- Gigabit Ethernet:
 - 1000Base-SX používá jako zdroj světla LED diodu nebo laser s vlnovou délkou 850 nm (770–860 nm), vícevidové kabely; pokud použijeme kabely s průměrem jádra 50 μm , je efektivní dosah 220 m, s průměrem jádra 62,5 μm to je 550 m,
 - 1000Base-LX používá laser o vlnové délce 1300 nm (1270–1355 nm), jednovidové kabely s průměrem jádra 10 μm pro dosah v jednotkách kilometrů (ale výrobci obvykle garantují až 20 km), nebo mnohavidové s průměrem jádra 50 nebo 62,5 μm pro menší vzdálenosti.
- 10Gbit Ethernet:
 - 10GBase-SR – laser 850 nm, vícevidový kabel, dosah 62 m,
 - 10GBase-LR – laser 1310 nm, jednovidový kabel, dosah 10 km (v praxi až 25 km),
 - 10GBase-LRM – laser 1310 nm, vícevidový kabel, dosah 220 m,
 - 10GBase-ER – laser 1550 nm, jednovidový kabel, dosah 40 km.




Vlákno je charakterizováno dvěma číselnými údaji: x/y, kde první údaj je průměr jádra a druhý průměr pláště jádra v mikrometrech. Například 50/125 je vlákno s průměrem jádra 50 μm a průměrem pláště 125 μm .



Podle ISO 11801 se pro optická vlákna používají tato označení:

- OM1 – vícevidové vlákno 62,5/125, obvykle pro LED diodové zdroje signálu,
- OM2 – vícevidové vlákno 50/125, obvykle pro LED diodové zdroje signálu,
- OM3 – vícevidové vlákno 50/125, obvykle pro laserové zdroje signálu,
- OS – jednovidové vlákno 9/125, zdroj je laser.



 *Pigtail* (doslova „prasečí ocásek“) je krátký kus kabelu (například 1,5 m) na jedné straně zakončený konektorem. Pigtaily se používají zejména u takových typů kabelů, kde je nasazení konektoru náročnější než osazení vodičů signálu do propojky, například u optických kabelů.




Úkol


Podobně jako jste v internetovém obchodě procházeli parametry metalických kabelů, podívejte se také na parametry a nabídku optických kabelů. Všimněte si výše uvedených parametrů (OM1, ..., OM3, OS, průměr jádra/průměr pláště apod.) a také počtu vláken v kabelu. Uvědomte si, že pro každý směr potřebujeme minimálně jedno vlákno (spíše více, pro větší propustnost).




4.4.3 Další typy kabelů


Kroucená dvojlinka nemusí mít jen dva nebo čtyři páry. V telekomunikacích (ale ne u koncového zákazníka) se používají kabely s 25, 100 nebo jiným počtem párů, čemuž odpovídají i konektory.

 Kabely označené jako *venkovní* (outdoor) jsou odolné proti kolísání teplot, vlhkosti, světlu, mohou mít vnitřní vyztužení ocelovým drátem (pro odolnost při zavěšení), případně vnější kovový plášť proti okusu (krysy a podobná zvířata).

 Existují také kabely určené pro speciální použití. Například twisted pair kabely označené jako *plenum* jsou určeny do ventilačních, klimatizačních a topných prostor. Jsou upraveny tak, aby byly odolnější proti ohni a aby v případě požáru nevedly nebezpečné plyny do nezasažených místností, což se projevuje i na jejich ceně. O něco levnější jsou kabely typu *riser*, které jsou odolnější proti ohni, ale už ne proti šíření plynů.

 *Koaxiální kabel* (co-axial cable) se v Ethernetu dnes už nepoužívá, setkáme se s ním pravděpodobněji buď při připojení externích antén nebo u rozvodů kabelové či satelitní televize používaných pro síťové rozvody. Je to souosý kabel (dva vodiče se společnou osou), kdy jeden vodič je středový drát, druhý vodič tvoří kovové opletení (jak je vidět na obrázku vpravo), mezi nimi je nevodivá izolace. Signál je určen rozdílem elektrických potenciálů těchto vodičů.



 *Twinax* (twin-axial cable) je obdoba koaxiálu s tím rozdílem, že oba vodiče jsou ve formě měděného drátu (ale také jsou vzájemně

izolované), žádné kovové opletení není použito. S twinax kabelem se setkáme u některých specifikací pro Ethernet, obvykle pro použití ve velkých datových centrech.




Další informace:

Pokud se zajímáte o kabely a obecně o síťový hardware, dobrým zdrojem informací je kniha [3] ze seznamu literatury na konci těchto skript. Je sice už trochu starší (u tištěné literatury se nedá nic dělat, přece jen nějakou dobu trvá, než je kniha vydána), ale přesto je v ní tato problematika vysvětlena velice podrobně a přehledně.



4.5 Zakončení kabelu

S kabely jsme se už seznámili, ale určitě je každému jasné, že každý kabel musí být nějak ukončen. Na obou koncích kabelu je buď konektor (kabel typu patch cord) nebo je osazen do zásuvek ve zdi (při instalaci do zdi), u některých typů kabelů se také používá typ pigtail (u optiky nebo twinaxu). V některých případech potřebujeme propojit dva konce kabelu (například tehdy, když máme dva krátké kabely a potřebujeme jeden dlouhý, tedy je nastavíme), pak můžeme použít speciální propojku.

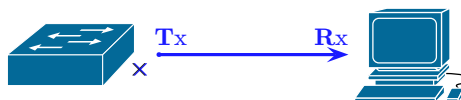
 Standardy rozlišují zakončení typu *konektor* (telecommunications connector) a zakončení typu *zásuvka* (telecommunications outlet). Dále se budeme zabývat jen kabely typu UTP, které jsou momentálně pro Ethernet nejpoužívanější.

4.5.1 Křížení

Zařízení na svém síťovém rozhraní vlastně provádí dvě základní operace:

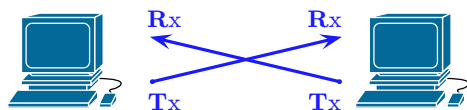
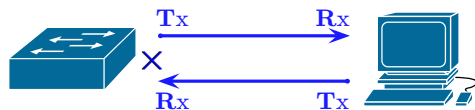
- Tx (Transmit) – odesílá,
- Rx (Receive) – přijímá (naslouchá a když zjistí, že druhá strana vysílá, vysílání přijme).

U kroucené dvojlinky jsou páry vodičů vyhrazené pro operaci Tx a páry vodičů vyhrazené pro operaci Rx, případně je toto přiřazení určováno dynamicky (ale v každém okamžiku je toto přiřazení jednoznačné). 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. 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 tímto 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 vedoucích z Tx jednoho zařízení musí být napojen za Rx druhého zařízení a naopak.

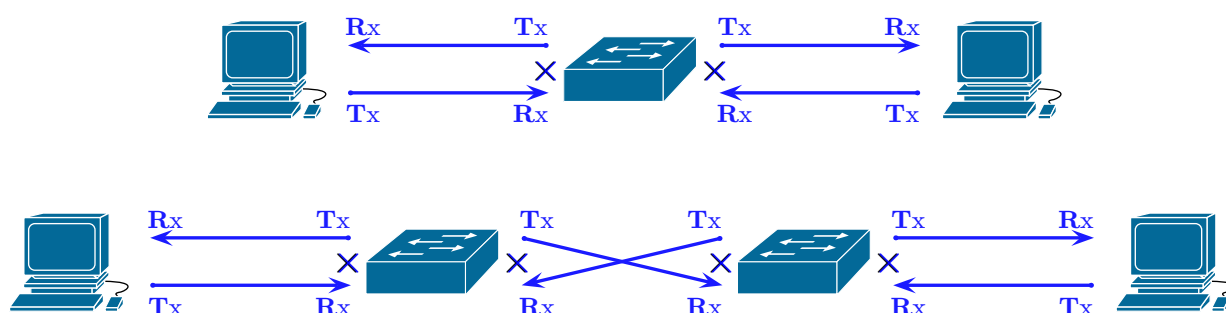


Na obrázku vlevo vidíme dvě řešení pro přímé propojení dvou zařízení – v prvním případě křížení zajistí jedno ze zařízení (většinou aktivní síťové prvky, zde switch), přičemž port zajišťující křížení je označen křížkem. V druhém případě je křížení zajištěno přímo v kabelu. Na obrázku 4.5 pak vidíme případy, kdy jsou na cestě mezilehlá zařízení, na portech těchto DCE je taktéž zajištěno křížení.



Poznámka:

Ta křížení, která vidíme na obrázku 4.5 (tj. u portu switche a v kabelu) nejsou všechna. Když se zaměříme na případy, kdy switch bere vstup (Rx) z jednoho portu a přeposílá ho na výstup (Tx)



Obrázek 4.5: Princip křížení na kroucené dvojince

jiného portu, dochází také ke křížení „uvnitř“ switche – například si můžeme na prvním řádku představit šipku od Rx na levé straně k Tx na pravé straně switche a od Rx na pravé straně k Tx na levé straně. Tyto hypotetické šipky (které jsem nezakreslila, ať to nevypadá jako přeškrtnutí) vlastně taky znamenají křížení.



Příklad 4.6

Spočítáme křížení na cestě mezi koncovými zařízeními v obou nakreslených případech.

- Na prvním řádku máme (1) jedno křížení na straně switche u levého portu, (2) jedno křížení uvnitř switche, (3) jedno křížení na straně switche u pravého portu \Rightarrow celkem tři křížení.
- Na druhém řádku jsou celkem (1) čtyři křížení na straně obou switchů u portů, (2) jedno křížení zajištěné kabelem, (3) dvě křížení uvnitř switchů \Rightarrow celkem sedm křížení.

Všimněte si, že vyšla lichá čísla – počet křížení na cestě mezi dvěma DTE (přes případná zařízení DCE na cestě) musí být vždy liché číslo. Totéž samozřejmě platí i pro přímé propojení dvou zařízení.



Rozlišujeme dva typy kabelů typu kroucená dvojlinka:

- *přímý kabel* – žádné křížení neobsahuje, používá se obvykle k propojení DTE a DCE (hostitel–switch, switch–router) – pozor, z pohledu routeru je switch vlastně DTE (proč?),
- *křížený kabel* – obsahuje křížení, používá se k propojení zařízení na stejné úrovni (DTE–DTE, DCE–DCE, například hostitel–hostitel, switch–switch) a také na připojení hostitele k směrovači (speciální případ, i když taky by se dal zdůvodnit).

Propojujeme:	Hub	Switch	Router	Hostitel
Hub	×	×	—	—
Switch	×	×	—	—
Router	—	—	×	×
Hostitel	—	—	×	×

— přímý kabel

× křížený kabel

Tabulka 4.1: Kdy použít přímý a 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 nasazený

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 (u Gigabit Ethernetu a rychlejšího) máme situaci trochu jednodušší – novější síťová rozhraní většinou dokážou automaticky rozpoznat, jestli mají nebo nemají zajistit vnitřní křížení, takže ve většině případů můžeme prostě použít přímý kabel.



4.5.2 Krimpování konektoru na přímý UTP kabel



Postup nasazení konektoru na kabel se nazývá *krimpování* nebo *konektorování*. Na kabel UTP obvykle krimpujeme konektor typu 8p8c (to znamená 8 pozic a 8 kontaktů), který se (ne zrovna přesně) označuje taky jako RJ-45.

Při krimpování musíme mimo jiné vodiče z kabelu správně uspořádat (umístit do kabelu ve správném pořadí). Momentálně se pro určení pořadí vodičů používá standard TIA/EIA-568-B z roku 2001, třebaže už existuje novější verze ANSI/TIA-568-C z roku 2014.



Standard 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!



Postup (Příprava na krimpování konektoru 8p8c)

Budeme potřebovat následující:

- dostatečně dlouhý UTP kabel, nejlépe kategorie 5e (10 cm *není* dostatečně dlouhý), měli bychom počítat s tím, že část kabelu budeme muset odstranit kvůli zarovnání vodičů a část zasuneme do konektoru,
- koncovky (konektory), mohou být i plastové kryty koncovek,
- krimpovací kleště (jsou nutné pro upevnění pinů – kontaktů – na vodiče kabelu), v obchodech taky mohou být pod názvem lisovací kleště, musí být pro konektor typu 8p8c,
- může se hodit ořezávací nástroj pro odstraňování plastové izolace z kabelu a vodičů, ale často bývá součástí krimpovacích kleští, případně se dá použít ostrý nožik,
- tester, který nám ukáže, jestli jsou všechny vodiče správně připojeny.

Vše potřebné je ve výuce a u vyučujícího k dispozici.

Krimpovací kleště se vyskytují v různé kvalitě a taky v různých cenových relacích. Většinou platí, že dražší kleště jsou kvalitnější, u méně kvalitních kleští je o něco vyšší riziko, že se nepovede správné napojení některého pinu na vodič.

Rozdíly mohou být také v tom, jaké konektory lze pomocí kleští krimpovat, což taky zvedá cenu – nás zajímá především 8p8c, ale kromě toho mohou mít pozici pro jiné počty pinů a vodičů – například 4p4c (pro telefonní/modemový RJ-11), 8p6c a další. Mnohé kleště mají také ořezávací nože pro odstraňování izolace.





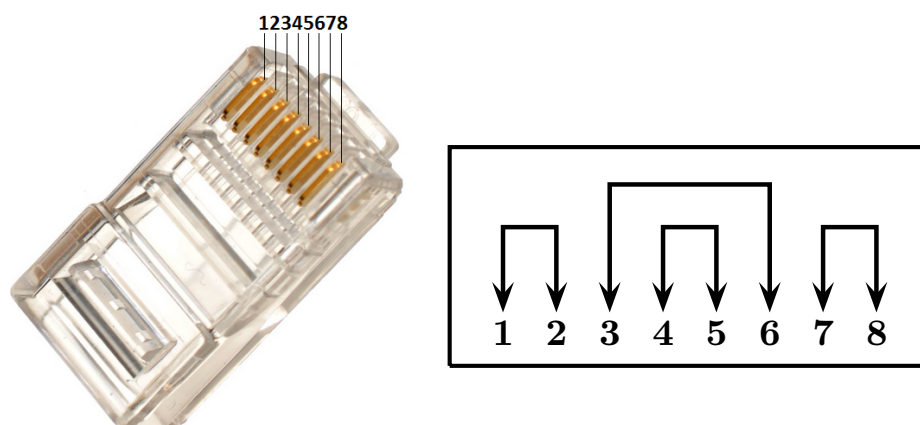
Úkol

V některém internetovém obchodě najdete krimpovací kleště (mohou být pod názvem „lisovací“) pro konektory typu 8p8c (někdy může být nutné hledat RJ-45, případně 8-pólové konektory apod.). Všimněte si jak vybavenosti kleští, tak i cenového rozpětí. Například zde:

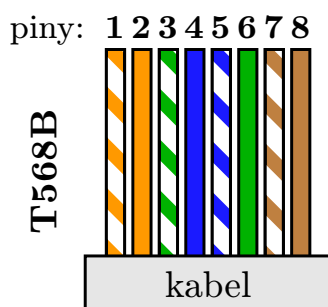
- <https://www.tsbohemia.cz>: PC příslušenství – Sítě a VoIP – Nářadí – Kleště
- <http://www.conrad.cz>: Kleště – Krimpovací kleště
- <http://www.intelek.cz>: Metalická kabeláž – Nářadí a vyvazování – Kleště



Piny jsou v konektoru v řadě, jejich čísla vidíme na obrázku 4.6 vlevo (konektor je natočen tak, že se díváme na piny). Vpravo je naznačeno, jak k sobě jednotlivé piny patří do dvojic. Ke každé dvojici pinů (podle obrázku) patří jeden pár vodičů, přičemž z těchto dvou vodičů je jeden v plné barvě, druhý v téže barvě, ale s bílým pruhem.



Obrázek 4.6: Pořadí a párování pinů v konektoru



Vodiče v konektoru mají barevnou izolaci, a právě podle těchto barev poznáme, v jakém pořadí mají být (barevný kód, color code). Pořadí podle T568B určeného v standardu TIA/EIA-568-B je naznačeno na obrázku vlevo. V naprosté většině případů volíme pro konektory na obou koncích kabelu právě toto pořadí.

Všimněte si, že párování vodičů určené barevným kódem odpovídá tomu, co je naznačeno na obrázku 4.6 (nahore) vpravo – nejdřív je oranžový pár (piny 1 a 2), na konci je hnědý pár (piny 7 a 8), pár pro piny 3 a 6 je zelený a pár pro piny 4 a 5 (přímo uprostřed) je modrý.



Poznámka:

Jak si pořadí barev zapamatovat? Předně si uvědomte, že se v pořadí střídají vodiče polobarevné (s bílou) a plnobarevné. Druhou indicií je způsob párování pinů (což pomůže pro zapamatování, kde konkrétně mají být zelené a modré vodiče). Někomu pomáhá i to, že nejvíc výrazná barva (modrá) je uprostřed a nejméně výrazné barvy (oranžová a hnědá), které se navíc liší jen sytostí barvy, jsou na okrajích.



Pokud pro patch kabel použijeme jiné pořadí než T568B (ale takové, aby byly v párech ty piny, které v párech být mají), v podstatě to taky bude fungovat. Přesto bychom si měli zvyknout standard dodržovat, protože pak při krimpování nemusíme zbytečně ztrácet čas zjišťováním, jaké že je pořadí na druhém konci kabelu, abychom to na tom „našem“ konci udělali stejně – obzvlášť nepříjemné je to v případě, kdy máme kabel protažený zdí nebo lištou a druhý konec není zrovna po ruce. Navíc bychom neměli komplikovat situaci svým případným následovníkům.



Postup (Krimpování 8p8c konektoru na UTP kabel)

Ted' už víme, co budeme potřebovat, zaměříme se na vlastní postup. V souhrnu je následující:

- pokud chceme nasadit krytku (která je pro cvičné účely nepovinná), musíme to udělat už ted' (pozor na směr – uvědomte si, co a jak vlastně má krýt),
- ke konci kabelu přiměříme konektor, ať víme, kolik izolace máme odstranit (cca půl centimetru, přiměřte koncovku) – vodiče by měly sahát až ke konci konektoru přes piny, ale zároveň izolace kabelu by měla zasahovat dovnitř konektoru,
pokud to ještě nemáte nacvičené, je lepší nechat konce delší a po urovnání nadbytek odcvaknout,
- odstraníme vnější izolaci kabelu podle toho, jak jsme v předchozím kroku naměřili, nesmíme porušit izolaci samotných vodičů (jinak to musíme celé odcvaknout a začít znovu),
- rozmotáme dvojlinky až k místu odstranění izolace, seřadíme vodiče podle T568B (seřazení by mělo „fungovat“ už od místa, na kterém jsme odřízli izolaci), co nejdůkladněji narovnáme,
- vodiče dáme těsně k sobě, tedy tak, jak budou v konektoru, a pokud jejich horní okraje společně netvoří rovinu nebo jsou příliš dlouhé, zarovnáme je odříznutím toho, co přechází (obvykle na to bývá břit přímo na krimpovacích kleštích),
- opatrně nasadíme konektor – tady pozor, tento krok je nejnáročnější, vodiče musí být přesně tam, kde mají být (správné pořadí a zasunuté až ke konci konektoru), konektor musí být správně natočený (koukáme na pozlacené piny),
- nasadíme krimpovací kleště a pořádně stiskneme,
- pokud jsme použili krytku, ted' ji nasuneme na konektor.

Pozor – po použití krimpovacích kleští už konektor nepůjde vysunout. Pokud dodatečně zjistíme, že je někde chyba, nezbývá než konektor ucvaknout a postupovat od začátku.

Na postup se můžete podívat i na videu <https://www.youtube.com/watch?v=VoHYaEi18qQ>



Úkol

Každý si na cvičení vyzkouší krimpování přímého UTP kabelu.



4.5.3 Testování

Správnost krimpování můžeme „nahrubo“ ověřit pohledem, ale pokud chceme, aby byl přenos opravdu bezproblémový, měli bychom použít tester. Testery mohou odhalit nejen špatné pořadí

vodičů, ale také například nedocvaknutí kontaktů – pak pin na konektoru nedoléhá až ke kovu vodiče a elektrony se nedostanou tam, kam se mají dostat.

Různé testery jsou různě složité a také jejich vybavenost funkcemi je odlišná. V každém případě mají (minimálně) dva plugy RJ-45, do kterých zapojíme oba konce testovaného kabelu. Od toho se také odvíjejí odlišnosti v ceně těchto produktů. Některé testery mají pouze jeden plug, přičemž jeden konec kabelu je zapojen právě sem, a druhý je zapojen do pracujícího zařízení (switche).



Obrázek 4.7: Testery pro ethernetové kabely (Zdroj: heureka.cz)

Na obrázku 4.7 vidíme několik dostupných testerů UTP kabelů. Jejich cena je v rozmezí několika set korun a několika desítek tisíc korun. Nejlevnější mají plugy pro RJ-45 konektory (případně taky telefonní RJ-11) a po připojení jednoduše postupným problikáváním dvou řad osmi (nebo jiného počtu, podle konektoru a kabelu) kontrolky ukáže, který vodič je ke kterému připojen – u přímého kabelu musí kontrolky v obou řadách bliknout ve stejném pořadí. Dražší modely mají buď zabudovanou podporu dalších typů konektorů (USB, koaxiál apod.), displej s přehlednějším zobrazením napojení vodičů nebo další funkce související s testováním. Také se dají pořídit kity (sady) obsahující krimpovací kleště, tester a další součásti.

Pro případ testování s napojením obou konců kabelu je praktické, když se tester skládá ze dvou oddělitelných částí – můžeme bez problémů testovat i tehdy, když například je každý konec kabelu v jiné místnosti.




Úkoly

1. Otestujte kabel, který jste nakrmpovali. Ve výuce jsou k dispozici testery.
2. Projděte si některý internetový obchod a udělejte si přehled o dostupných testerech včetně jejich rozhraní, funkcí, ceny. Například na:

- <http://www.suntech.cz>: Síťové prvky – Pasivní prvky – Nářadí – Testery
- <http://www.conrad.cz>: Měřicí technika – Měřiče a zkoušečky – Zkoušečky kabelů
- <http://www.intelek.cz>: Metalická kabeláž – Měřicí přístroje




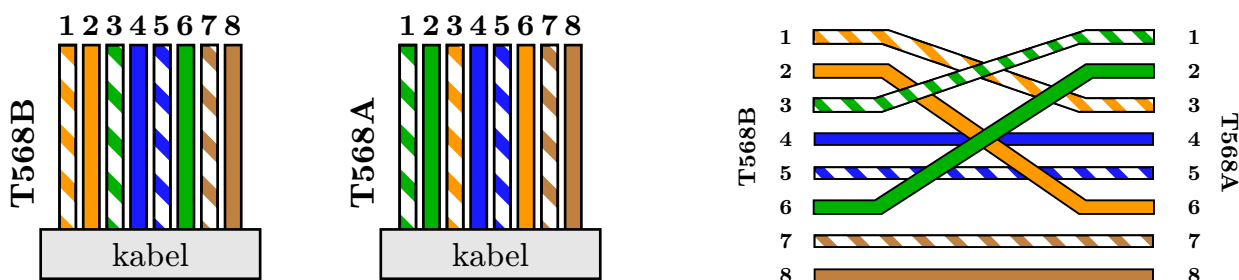
 Tester obvykle taky potřebuje napájení. Jestliže je jeden konec kabelu připojen ke spuštěnému switchi, pak může (nemusí) být napájení zajištěno switchem, ale pokud jsou oba konce kabelu v testeru, pak nutně potřebujeme baterii (typ baterie je uveden v dokumentaci). Typ a kvalita napájecího rozhraní taky rozhoduje o ceně testeru.

Pokud se tester skládá ze dvou částí, je baterie jen v jedné. Do druhé se proud přenáší kabelem.

4.5.4 Krimpování konektoru na křížený UTP kabel


Pokud krimpujeme křížený UTP kabel, musíme zajistit napojení páru Tx jednoho konce na pár Rx druhého konce a opačně.

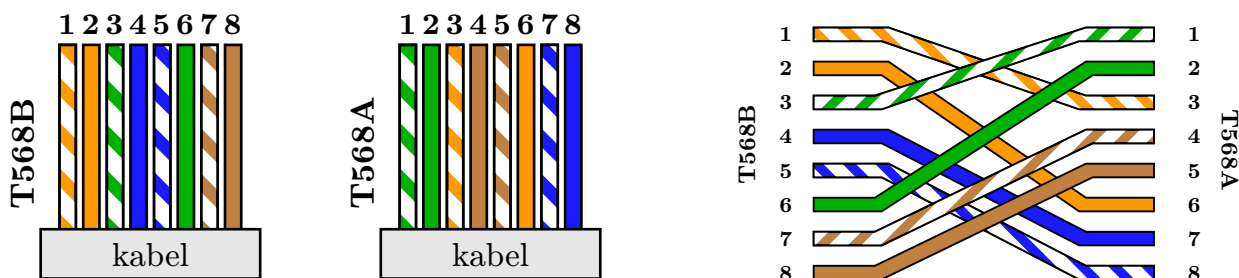
 Pokud má být kabel používán pouze pro Fast Ethernet (do rychlosti 100 Mb/s), využívají se pouze dva páry vodičů, přesněji piny 1, 2, 3 a 6, zbývající piny buď zůstávají prázdné (když použijeme kabel se dvěma páry místo čtyř) nebo se prostě nepoužívají. To znamená, že křížit budeme pouze páry využívající výše zmíněné piny a další dva páry necháme přímé.



Obrázek 4.8: Barevný kód pro křížený kabel – Fast Ethernet

Na obrázku 4.8 je pořadí vodičů předepsané standardem TIA/EIA-568-B (barevný kód), které je platné nejvýše pro rychlost přenosu 100 Mb/s. Ve směru zleva je naznačeno pořadí na jednom a druhém konci kabelu – to znamená, že jeden konec nakrmpujeme podle pořadí T568B a druhý podle pořadí T568A. Vpravo pak máme schéma napojení vodičů vzhledem k oběma koncům.

 Křížený UTP kabel pro Gigabit Ethernet už musí být krimpován podle novější verze standardu – ANSI/TIA-568-C, ve kterém je předepsáno křížení obou dvojic párů. Zde však už je porušena podmínka střídání plnobarevných a bílobarevných vodičů.



Obrázek 4.9: Barevný kód pro křížený kabel – Gigabit Ethernet



Další informace:

Nákresy pro různé barevné kódy: <http://www.flukenetworks.com/content/color-codes-dsx-cableanalyzer>

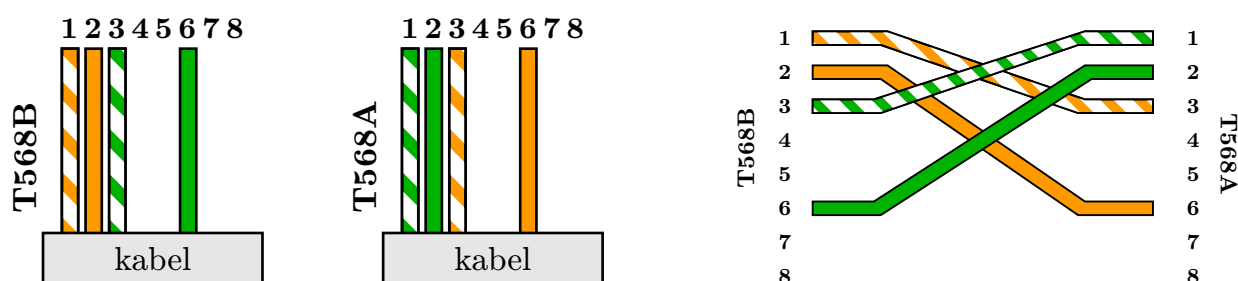


**Poznámka:**

Ve skutečnosti se naštěstí křížené kabely pro Gigabit Ethernet prakticky nepoužívají, protože GbE síťové rozhraní do dokáže „dohodnout“ se svým protějškem na protilehlém zařízení na tom, které páry se jak budou používat. Tedy pro Gigabit Ethernet se na všech typech propojení používá přímý kabel.

**4.5.5 Kroucená dvojlinka se dvěma páry vodičů**

Pokud náhodou budeme muset krimpovat „ošizený“ kabel s pouhými dvěma páry vodičů místo čtyř, pak osadíme piny 1, 2, 3 a 6 (tj. oranžový a zelený pár). Ale pozor – takový kabel je určen pouze pro Fast Ethernet, kdežto pro Gigabit Ethernet ho nelze použít.



Obrázek 4.10: Barevný kód pro křížený kabel se dvěma páry vodičů

Přímý kabel bude jednoduchý – oba konektory osadíme podle T568B (pouze oranžový a zelený pár), jak vidíme na obrázku 4.10 zcela vlevo. U kříženého kabelu pak použijeme na jednom konci T568B a na druhém T568A, jak je naznačeno na obrázku 4.10.

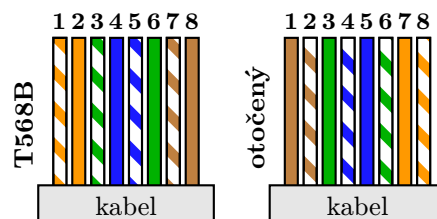
**Poznámka:**

U krimpování kříženého plnohodnotného kabelu se čtyřmi páry byl zvlášť postup pro Fast Ethernet a pro Gigabit Ethernet. Tady ne – proč myslíte?

**4.5.6 Otočený a konzolový kabel**

Otočený kabel (rollover cable) se používá jako součást připojení hostitele (počítače) ke konzoli aktivního síťového zařízení. Tento kabel má v konektorech vodiče na jednom konci v přesně opačném pořadí než na druhém konci, jak vidíme na obrázku vpravo.

Konzolový kabel je speciální kabel, kterým lze přímo propojit aktivní síťové zařízení (switch, router apod.) s počítačem za účelem přímé konfigurace (funguje i v případě, že zařízení nemá přidělenou IP adresu a nedostaneme se k němu přes síťové rozhraní). Na logické úrovni (tj. struktura komunikace) je tento typ připojení realizován přes sériové rozhraní (port RS-232, označení zakončení kabelu je DB-9 – 9pinové sériové rozhraní). Na



fyzické úrovni (tj. jak konkrétně vypadá kabel a koncovky) tomu tak často nebývá, výrobci se totiž postupně odklánějí od fyzického sériového portu.



Zdroj: Cisco.com

Aktivní síťové prvky u větších výrobců (včetně Cisco) už nemají plug typu DB-9 a místo toho je jako „console port“ označen port, který (fyzicky) vypadá jako běžný 8p8c (RJ-45). Takže je jasné, že na straně (třeba) switche musí být i konektor tohoto typu (obvykle bývá s nápisem „console“, označený světle modře).

Ovšem takový switch do tohoto konektoru posílá komunikaci, jejíž struktura (včetně způsobu využití jednotlivých vodičů) neodpovídá tomu, co je běžné pro ethernetové spoje, a tedy i na druhé straně (u počítače) je třeba zajistit, aby tato „neethernetová“ komunikace byla správně pochopena, a taktéž v opačném směru. Kdybychom použili běžný přímý nebo křížený kabel, nefungovalo by to – zařízení by se nedomluvila na logické úrovni. Proto také od počítače musí směřovat komunikace podle RS-232. Existuje víc možností:

- Na kabelu je na straně switche konektor 8p8c (RJ-45) a na straně počítače konektor DB-9 (sériové rozhraní RJ-232). Použijeme, pokud máme na počítači (notebooku) sériové rozhraní, což už není moc obvyklé. Tento kabel je na obrázku 4.11 vlevo.



Obrázek 4.11: Vlevo: konzolový kabel RJ-45/DB-9, vpravo: redukce DB-9 na USB-A (Zdroj: https://dcloud-cms.cisco.com/help/connect_console)

- Použijeme stejný konzolový kabel jako v předchozím případě, ale do konektoru DB-9 připojíme redukci z DB-9 na USB-A (na obrázku 4.11 vpravo), do počítače či notebooku zapojíme právě konektor USB. Na počítači by mělo fungovat emulování sériové linky do USB rozhraní, což je záležitost ovladače.
- Když nemáme konzolový kabel RJ-45/DB-9, nahradíme ho otočeným kabelem propojeným s redukcí RJ-45 na DB-9. Takže celkem bude použita tato posloupnost kabelů: RJ-45/RJ-45 otočený, redukce RJ-45/DB-9, redukce DB-9/USB.
- Některé aktivní síťové prvky mají fyzický sériový port, takže je třeba použít kabel s tímto portem na straně síťového zařízení.
- Někteří výrobci pro tento účel používají micro-USB port či jiné.




Další informace:

- <http://www.websitea.com/croftcom/Cisco/LaptopSerialPort.htm>
- <http://blog.router-switch.com/2013/05/how-to-connect-to-a-cisco-standard-console-port-rj-45/>




4.5.7 Osazení UTP kabelu do zásuvky

 Zásuvka ve zdi, do které zapojujeme konektory RJ-45, se podle standardu označuje jako *telecommunications outlet/connector*, čímž je míněno, že ve skutečnosti je součástí zásuvky i obdoba toho, co je u kabelu typu patch cord samotný konektor RJ-45, tedy interní konektor, do kterého je nutné osadit konce vodičů kabelu vedeného (obvykle zdí) k zásuvce.

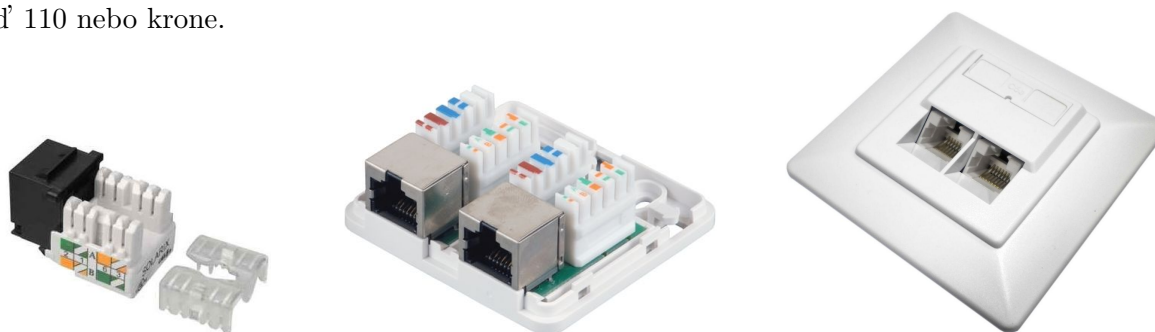
Podle standardu ANSI/TIA-568-C má být na každém pracovním místě k dispozici nejméně dvojice zdírek (plugů), z nichž jedna je pro čtyřpárovou kroucenou dvojlinku kategorie minimálně 5e, ať už stíněnou nebo nestíněnou. Druhá může být stejná nebo třeba pro optický kabel. Starší standard TIA/EIA-568-B vyžadoval minimálně kategorii 3, což už dnes samozřejmě nestačí.

Zásuvky jsou buď podomítkové (používají se tehdy, když kabel vede zdí a zásuvka má být také zanořená) nebo nadomítkové (u nich je výhodou, že při instalaci nemusíme sekát do zdi, kabel je v liště). Princip je vlastně velmi podobný tomu, jak to funguje u zásuvek pro elektrickou síť, jen jsou předpisy trochu mírnější – například kabely nemusí být v trubkách (v kabelech máme jen nízké napětí a proud).

 Zásuvka bývá často modulární, skládá se z několika částí:

- *interní konektor* se svorkovnicí, do které se zařezávají konce vodičů, obvykle bývají štěrbiny označeny podle barevného schématu, taky se můžeme setkat s označením „keystone“,
- kryt (rámeček) se zdírkou, ke které se po zařezání vodičů připevní interní konektor.

Konkrétní rozvržení a provedení se liší podle výrobce, dnes se často setkáváme s tzv. beznástrojovými řešeními, kdy je osazení kabelu velice jednoduché. U svorkovnice bývá označení většinou buď 110 nebo krone.



Obrázek 4.12: Svorkovnice do zásuvky s jednou a dvěma zdírkami a zásuvka (Zdroj: solarix.cz)

Postup (Osazení UTP kabelu do zásuvky)

Postup a potřebné nástroje se mírně liší podle konkrétní zásuvky, obvykle si vystačíme s tímto:

- zásuvka včetně interního konektoru,
- kabel (obvykle vedený zdí),
- štípací kleště nebo jiný nástroj pro přeštípnutí nadbytečného kusu vodiče,
- zařezávací nástroj (zařezávací nůž, impact tool, punch down tool), pro svorkovnici typu 110 nebo krone (pozor – typ musí souhlasit), ale u některých zásuvek není potřeba.

Dáváme si pozor na to, ať kabel moc neohýváme, mohli bychom poškodit vodiče. Měl by mít určitou vůli (i kvůli tepelné roztažnosti), část může zůstat v zásuvce smotaná.



Obrázek 4.13: Zařezávací nástroje (Zdroj: aspa.cz, alza.cz, tsbohemia.cz)

Cílem je dostat vodiče kabelu na správná místa se svorkovnicí a pak je galvanicky propojit s piny v konektoru. Postup:

- Připravíme si interní konektor a zkontrolujeme barevné značení (většinou použijeme to podle T568B). Na svorkovnici je obvykle naznačeno obojí, neměli bychom si je poplést.
- Odstraníme vnější izolaci z kousku kabelu (max. 13 mm) a protáhneme či nasadíme tento kousek do konektoru (podle konstrukce). Jednotlivé páry opatrně oddělíme od sebe.
- Páry alespoň částečně rozmotáme a rozdělíme do štěrbin podle barevného kódu.
- Pokud zásuvka není beznástrojová, pak musíme vodiče do svorkovnice zasunout zařezávacím nožem. V každém případě odstraníme přečnívající části vodičů (štípacími kleštěmi).

U beznástrojových zásuvek může být potřeba na konektor nacvaknout speciální kryt, podle konstrukce zásuvky, viz obrázek 4.12 vlevo).

- Teď stačí nacvaknout konektor do krytu a připevnit na zeď.

Postup pro beznástrojovou zásuvku je ukázán na <https://www.youtube.com/watch?v=AelPDxMRpcw>.



Další informace:

- Osazení UTP Cat.6 do beznástrojové zásuvky: <https://www.youtube.com/watch?v=2uShSEQ4u2E>
- Osazení STP Cat.6A (F/FTP): <https://www.youtube.com/watch?v=DbMTF8UmEoE>
- Náskres pro kabel kategorie 6A: <http://www.krugel.cz/photos/fotografie/anoniekompaktcz.jpg>



4.6 Strukturovaná kabeláž

4.6.1 Rozvaděč

Kam se switchem, routerem, serverem a podobnými zařízeními? Pokud se jedná o výkonná zařízení pro korporátní sféru, umísťujeme je do rozvaděče (racku), což je skříň poskytující zařízením upevnění, řízené chlazení, elektroinstalaci a konektivitu. Existují různé druhy rozvaděčů – nástěnné, stojanové, plně kryté nebo otevřené rámy.

Rozměry jsou standardizované, resp. existuje několik možných rozměrů (šířka a hloubka). Udává se vždy vnitřní šířka – 10", 19", 21", 23", nejběžnější je 19". Vnitřní hloubka může být taky různá, běžné je 600 nebo 800 mm. Pokud je rack hlubší než je nutné, nevádí to, obvykle se dá vnitřní upínací mechanismus přizpůsobit (naopak to už samozřejmě vadí).

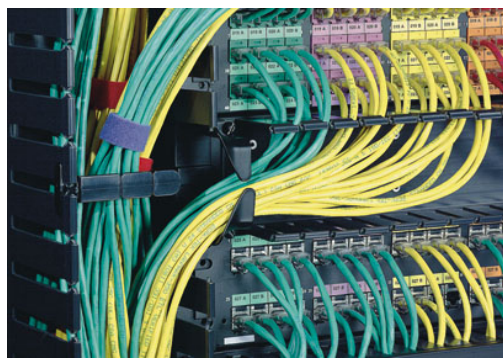
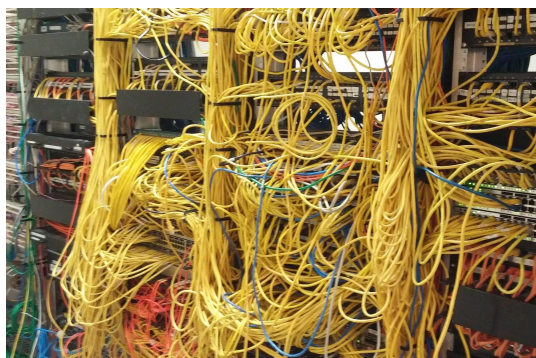
Výška zařízení umísťovaných do rozvaděče se udává v standardizovaných jednotkách označovaných U (unit). Platí vztah $1 \text{ U} = 1,75'' = 4,4 \text{ cm}$. Switche do LAN mají obvykle výšku 1U, u jiných zařízení je již běžnější větší výška. Například zařízení s výškou 2U je vysoké téměř 9 cm.

Výška rozvaděče se také udává v těchto jednotkách, například do racku o výšce 18U je možné umístit různá zařízení o výškách, jejichž součet je právě 18U. Ovšem není vhodné rack příliš „přepřát“, především tehdy, když máme uvnitř výkonná zařízení vyžadující dobré chlazení.



Úkoly


1. Na <http://www.czc.cz/rack-server/produkty> použijte filtr pro formát zařízení a zjistěte, jestli je právě v prodeji některý server s výškou 4U.
2. Podívejte se na následující dva obrázky. Na jednom z nich je něco špatně. Co to je?

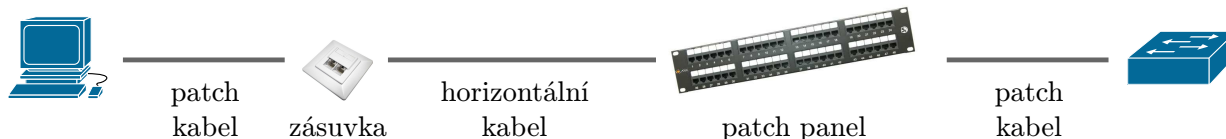


(nápopěda: v barvách to není, i když barvy mohou s přehledností velmi pomoci)



4.6.2 Horizontální kabeláž

 *Horizontální rozvody* jsou pojmenovány podle toho, že vedou víceméně vodorovně (horizontálně); takto nazýváme rozvody propojující zařízení v rámci jednoho patra budovy. Obvyklá struktura horizontální kabeláže je následující:



 Co se týče délky jednotlivých částí trasy, platí:

- Celá fyzická přenosová cesta od hostitele na levé straně ke switchi (či jinému aktivnímu síťovému prvku od vrstvy L2 výše) na pravé straně 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.

Kabeláž. O kabelech jsme se toho naučili už celkem hodně, zbývá jen rozdíl mezi patch kabely (patch cord) a kabely používanými do zdi pro horizontální kabeláž. Ve většině případů se pro obojí používá kroucená dvojlinka, momentálně nestíněná UTP, ale pro každý z těchto dvou případů potřebuje kabel trochu jiné vlastnosti:

- *Kabel typu lanko* (licna, stranded) – pro patch kabely. Vodivé jádro vodičů je ze svazku smotaných tenkých měděných drátů obalených izolací, má větší elektrický odpor. Kabel je pružnější, dá se lépe ohýbat, ale je větší riziko poškození až zlomení vodičů.
- *Kabel typu drát* (solid) – pro horizontální kabeláž (do zdi). Vodivé jádro vodičů je z plného silnějšího drátu. Kabel se hůře ohýbá, ale zato je trvanlivější.



Poznámka:

Pokud ve specifikaci či přímo na kabelu objevíte zkratku CCA (Copper Clad Aluminium), radši se takovému kabelu vyhněte. Vodiče v tomto kabelu jsou vyrobeny z hliníku a potaženy tenkou vrstvou mědi. Jsou sice levnější (a to o dost), ale mají horší vodivost a nemohou být ani zdaleka nataženy na takové vzdálenosti, jaké jsou běžné pro celoměděné kabely.



Příklad 4.7

Typ kabelu a další údaje najdeme přímo natištěné na vnější izolaci kabelu. Například:

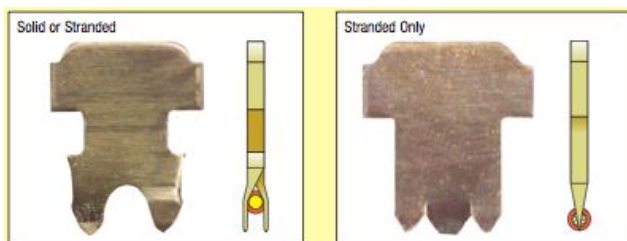
„...4P 24AWG UTP Patch Cable ISO/IEC 11801 and TIA/EIA 568 ... verified CAT5e...“

což znamená, že jde o 4párový kabel s vodiči o síle 24AWG (tj. spíše slabšími – čím menší číslo před AWG, tím větší průměr), nestíněný patch kabel (pro připojení k hostiteli nebo switchi) splňující dané standardy, kategorie 5e.



Úkol

Pro firmy, kde jsou lidé schopní krimpování kabelů, se kabely kupují jako metráž, „klubko“ apod. – bez konektorů a v délce několika set metrů. V některém internetovém obchodě (například na <http://www.intelek.cz> nebo <http://www.czc.cz>) najdete kategorii pro takové kabely a ověřte si, kolik přibližně stojí 305 m nestíněné kroucené dvojlinky kategorie 5e jednoho i druhého typu.



Ve skutečnosti se tyto dva typy kabelů taky trochu jinak krimpují, resp. existují konektory 8p8c typu drát a typu lanko (licna). Liší se v tom, jakým způsobem se propojí pin v konektoru s vodičem v kabelu.

Zdroj: <https://www.brucetambling.com>

V konektoru *typu drát* nože pinu vodič „obejmou“, přičemž se ostrím nožů naruší izolace vodiče. V konektoru *typu lanko* se nůž pinu do vodiče zařízne (zapíchne), což není problém, protože vodič v takovém kabelu je vlastně spletenec tenkých měděných lanek. Nejběžnější nože v konektorech typu drát a lanko jsou na obrázku vlevo.

**Poznámka:**

Z toho vyplývá, že pokud se pokusíme připevnit konektor typu lanko na kabel typu drát, nože u pinů se buď zlomí nebo zkroutí. Buď kontakt nebude fungovat vůbec nebo přestane fungovat po určité době (až se zkroutený nůž vyviklá). Opačně to nebývá problém (konektor typu drát na kabel typu lanko). Abychom to nepopletli, je lepší vždy mít konektor stejného typu jako kabel.



Kabely typu drát se používají do zdi, tedy běžně minimálně jeden jejich konec není osazen konektorem RJ-45, ale osazen do zásuvky. Nicméně na druhý konec často používáme konektor a takto připojujeme do patch panelu.



Patch panel. Jak bylo vidět výše, důležitou součástí horizontálního vedení je patch panel. Můžeme si ho představit jako předsunutou sadu portů pro „schovaný“ switch, přičemž patch panel taky obvykle připevňujeme do racku, typická výška je 1U.



Zdroj: <http://www.excel-networking.com>

Vlastně bychom mohli horizontální kabel připojit rovnou do switchu, ale použití patch panelu nám zpřehledňuje a zjednodušuje přístup k portům switchu (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 (ale zase pozor na přehlednost). Pokud je třeba změnit zapojení pro konkrétní horizontální kabel vedoucí z určité zásuvky, nemusíme provádět změnu na switchi, stačí přehodit kabel na patch panelu.

Jak vidíme na obrázku, většinou jsou na čelní straně patch panelu porty pro zasunutí konektorů 8p8c horizontálních kabelů, kdežto z druhé strany najdeme svorkovnice podobné těm v zásuvkách. Takže patch kabel vedoucí ze switchu neosazujeme konektorem, ale zasuneme vodiče do zdírek na svorkovnici u konkrétního portu patch panelu. To je výhoda, protože zaříznutí velkého množství vodičů do svorkovnice trvá kratší dobu než nasazování konektorů.

Existují také patch panely, které umožňují používat konektory na obou stranách. Další variantou jsou neosazené patch panely, do kterých je třeba dodat moduly, do nichž se osazují konektory či vodiče (jeden modul je tedy spojení RJ-45 zásuvky na jedné straně a svorkovnice na druhé straně). Jejich výhodou je, že si osadíme jen tolik pozic, kolik doopravdy potřebujeme, a zbývajícím prostorem může proudit vzduch při chlazení.

**Úkol**

Na <http://www.intelek.cz/> najdete u metalické kabeláže kategorii Patch panely, pro kategorii 5e. Zjistěte, kolik portů obvykle mají a kolik stojí.



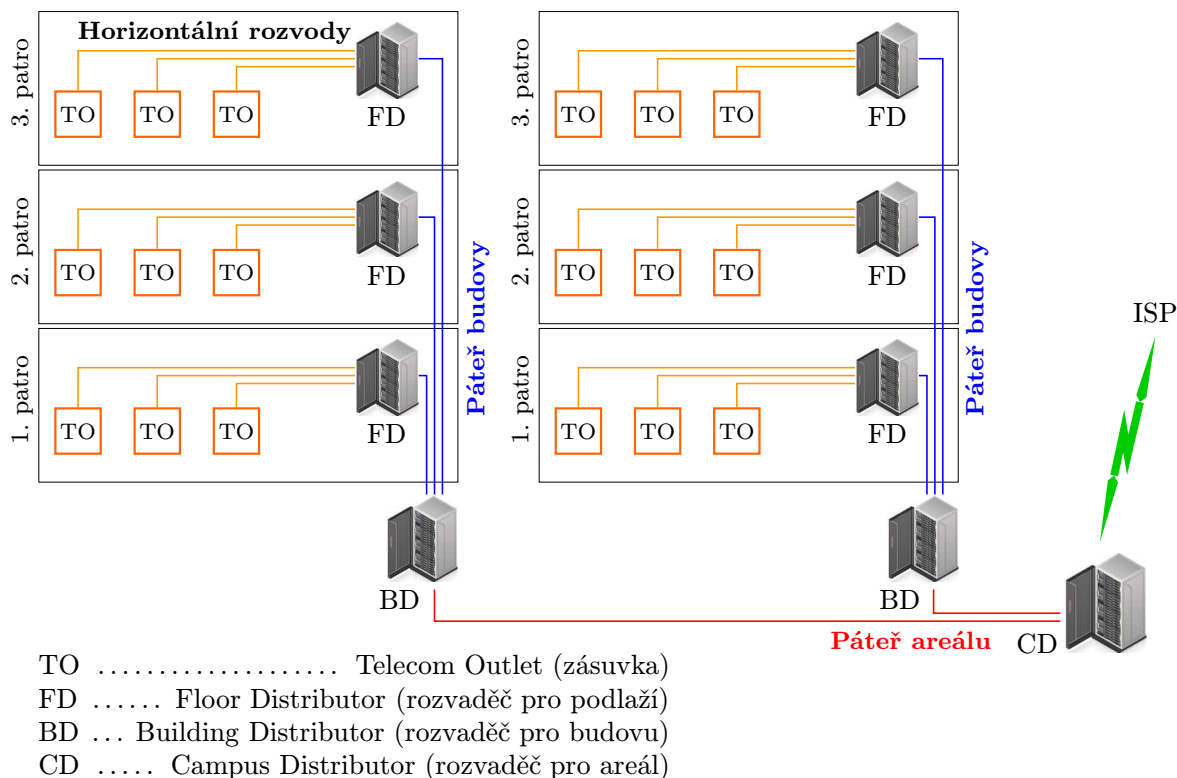
4.6.3 Páteřní síť

Zatím máme vyřešenou horizontální kabeláž – v rámci jednoho patra budovy. Jenže to nestačí, musíme propojit patra z celé budovy a taky jednotlivé budovy areálu.

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

- *floor distributor* (FD) – rozvaděč, do kterého ústí horizontální rozvody z celého patra,
- další prostředky sloužící potřebám zaměstnanců na daném patře (například lokální server).

Horizontální rozvody víceméně zachovávají topologii typu hvězda, přičemž FD je v centru této hvězdy.



Obrázek 4.14: Schéma rozvodů strukturované kabeláže

Páteřní síť budovy (building backbone) propojuje všechna patra budovy, resp. všechny FD. Pokud od FD odsekneme horizontální rozvody, pak páteřní síť budovy má taky v podstatě fyzickou topologii typu hvězda, v jejímž středu je *building distributor* (BD) – rozvaděč pro celou budovu. Když přidáme i horizontální rozvody, získáme topologii typu strom (zatím se dvěma patry), v kořeni stromu je BD.

Pojem *campus* (areál) označuje skupinu budov, které určitým způsobem patří sobě (například areál patřící jedné firmě), a tedy je třeba tento areál společně zasilovat. *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.

Celá struktura sítě je naznačena na obrázku 4.14.

Zatímco pro horizontální rozvody se obvykle používají metalické kabely (kroucená dvojlinka), u páteřních rozvodů se již pravděpodobněji setkáváme s optikou (čím vyšší úroveň, tím pravděpodobněji). Především pro páteř areálu je optika vhodná i proto, že optický kabel můžeme natáhnout na mnohem větší vzdálenosti, nejsme omezeni pouhými 100 metry pro celé vedení.

Taky je třeba si uvědomit, že i na aktivní síťová zařízení na nejvyšší úrovni existují určité požadavky – ve stromové topologii jde přes kořenové zařízení poměrně velký datový tok, takže se u tohoto zařízení očekává vysoká výkonnost, co nejméně zdržování provozu filtrováním (to zvládnou aktivní síťová zařízení hlouběji ve stromě) a taky zabezpečení proti výpadku – redundantní (sekundární, náhradní) zařízení, které v případě výpadku primárního zařízení převeze jeho roli.

**Poznámka:**

Pokud se jedná o menší prostor, pak mohou být některé části sdružovány. Například když máme sesíťovat jen jedinou budovu, není nutné rozlišovat BD a CD, popřípadě u domku s několika zásuvkami na patro a několika patry nemusí nutně být rack pro každé patro.




4.6.4 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,
- 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>



Packet Tracer a konfigurace síťových zařízení Cisco

 *Rychlý náhled:* V této kapitole se již zaměříme na aktivní síťové prvky. Nejdříve se seznámíme s nástrojem Packet Tracer, ve kterém lze virtuálně vytvářet, konfigurovat a testovat počítačové sítě, a následně se naučíme základům konfigurace síťových zařízení společnosti Cisco.

 *Klíčová slova:* Packet Tracer, topologie, kabel, switch, router, monitorování, paket, mód (režim), user exec, privileged exec (enable), globální konfigurační mód, subkonfigurační mód, rozhraní, linka, konzola (console), virtuální terminál (VTY), running-config, startup-config, flash, RAM, NVRAM, Telnet, SSH

 *Cíle studia:* Po prostudování této kapitoly budete umět používat nástroj Packet Tracer, porozumíte různým módům v zařízeních Cisco a dokážete provést základní konfiguraci switchu a routeru.

5.1 Instalace a zprovoznění

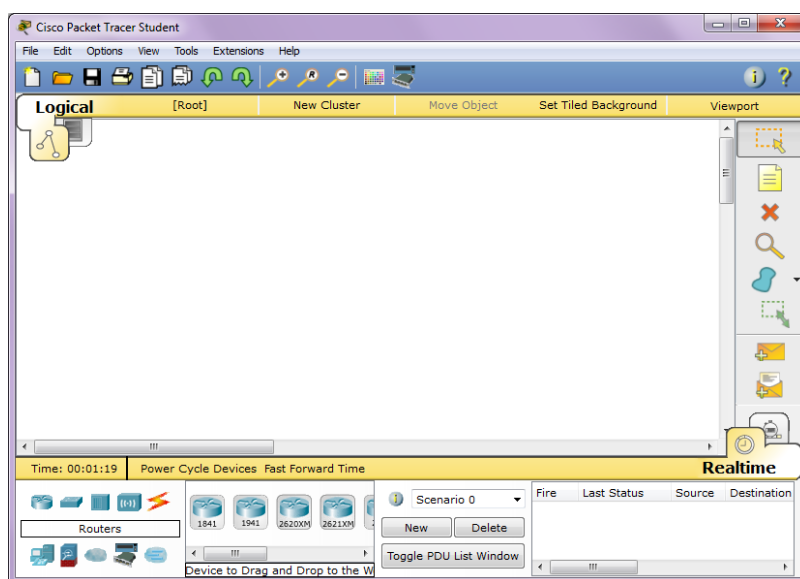
Program Cisco Packet Tracer je jednoduchá aplikace, ve které můžeme navrhovat a konfigurovat síť bez toho, abychom opravdu „fyzicky“ příslušná zařízení a kabely měli v ruce.

Cisco Packet Tracer je pro studijní účely zdarma ke stažení, verzi 6.2 najdeme na adrese <https://arief-jr.blogspot.com/2016/01/download-cisco-packet-tracer-62-for.html>. Vyšší verze mají licenční podmínky, které nemusí každému vyhovovat, navíc vyžadují autorizaci.

Instalace je jednoduchá a po spuštění programu pracujeme v grafickém prostředí. Po spuštění se objeví okno podle obrázku 5.1.

5.2 Úvod do práce v Packet Traceru

Většinu okna zabírá pracovní plocha, na které budeme vytvářet topologii své sítě. Dole najdeme síťové prvky, které při tom budeme používat – zcela vlevo je přepínač kategorií prvků (routery, switchy, koncová zařízení, . . .), po klepnutí na kategorii se vedle objeví seznam konkrétních zařízení,



Obrázek 5.1: Prostředí aplikace Cisco Packet Tracer

ze kterých si vybíráme. Je zde taky kategorie *Connections*, ve které najdeme různé typy kabelů a jiných spojů.



Postup (Vytváření topologie sítě na pracovní ploše)

S pracovní plochou se pracuje následovně:

- Pokud chceme na plochu umístit nový prvek, vybereme si nejdřív kategorii (vlevo dole), pak hned vedle konkrétní prvek, *neklepeme na něj*, jen ho „chytíme“ myší a přetáhneme na místo, kde ho chceme mít.
- Pokud chceme stejných zařízení umístit na plochu víc, pak *klepneme* na ikonu zařízení a pak na ploše klepneme na jednotlivá zamýšlená umístění.
- Když jsme se zmýlili a chceme prvek odstranit, pak v panelu nástrojů u pravého okraje okna klepneme na ✖ (čímž se přesuneme do „mazacího módu“) a pak klepneme na dotyčný prvek. Až zmizí, klepneme na tomtéž panelu vpravo na 🗑 (běžný mód) a můžeme dál pracovat.
- Jednotlivé prvky propojujeme spojem (třeba kabelem) následovně:
 - dole zvolíme kategorii *Connections* (ikona ⚡),
 - vybereme si typ spoje (například přímý kabel) a *klepneme* na jeho ikonu,
 - klepneme na první propojované zařízení – zobrazí se seznam volných portů, z něhož si klepnutím vybereme port, čímž do něj „zasuneme“ konektor spoje, pak totéž uděláme s druhým propojovaným zařízením.



Další informace:

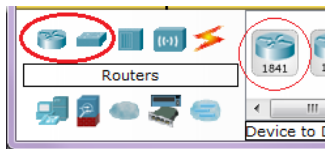
Pokud si s něčím nevíte rady, zvolte v nápovědě *Help – Contents*. Nápověda je velice dobře zpracovaná a obvykle tam najdete odpovědi na své otázky.





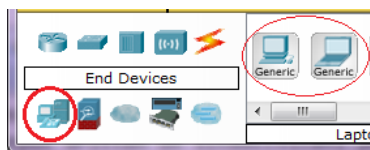
Příklad 5.1 (Vytvoření první topologie)

Naším prvním úkolem je vytvořit jednoduchou topologii s jedním routerem, ke kterému budou připojeny dva switche, ke každému switchi připojíme dvě koncová zařízení.



Zatím je naše pracovní plocha prázdná. Nejdřív umístíme router. V seznamu kategorií zařízení (viz vlevo) vybereme routery (první ikona) a v seznamu konkrétních zařízení si zvolíme hned ten první (1841). Označení teď nemusíme řešit, jsou to prostě označení konkrétních produktů od společnosti Cisco. *Chytneme myši a přetáhneme* nahoru na pracovní plochu okna.

Dále přidáme dva switche. Vlevo vybereme kategorii pro switche a hned první switch v seznamu myši postupně dvakrát *chytneme a přetáhneme* na pracovní plochu.



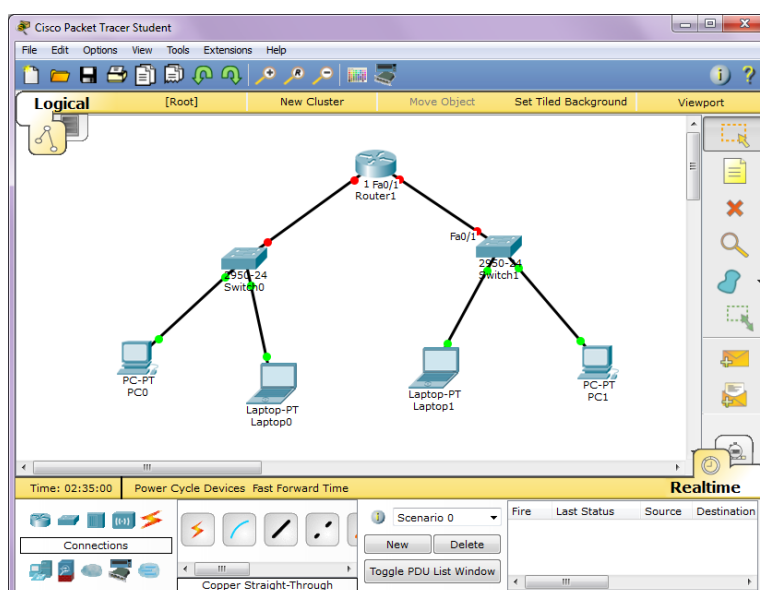
Zbývají koncová zařízení – *End Devices*, viz obrázek vlevo. Nějaká vybereme (je jedno jaká, můžeme klidně nakombinovat počítače a notebooky), a tím jsme vytvořili základ naší topologie.

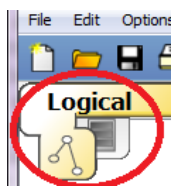
Topologii dokončíme správným propojením uzlů sítě. Dole najdeme kategorii *Connections* (ikona). Jak jistě všichni vědí, pro všechny spoje budeme potřebovat *přímý ethernetový kabel*. Takže *klepneme* na spoj označený černou čarou (ikona), *Copper Straight-Through*). Kurzor by se nám měl změnit na „zástrčku“. Dále:

- klepneme na první propojované zařízení (třeba počítač nejvíc vlevo), čímž se nám zobrazí seznam volných portů tohoto zařízení, jeden volný port vybereme (volte „Fast Ethernet“ nebo tak nějak) klepnutím na něj \Rightarrow zasunuli jsme jeden konec kabelu do portu,
- klepneme na druhé propojované zařízení (switch) a v seznamu volných portů taky jeden vybereme.

U počítače máme použitelný obvykle jenom jeden port (nevšímejte si portů typu RS-232 nebo Console), u aktivních síťových prvků mnohem více.

Takto postupně propojíme vše, co má být propojeno. Na obou stranách kabelu se objeví barevné kolečko indikující stav portu. Zelená znamená, že je port aktivní. Na následujícím obrázku vidíme momentální stav naší topologie.



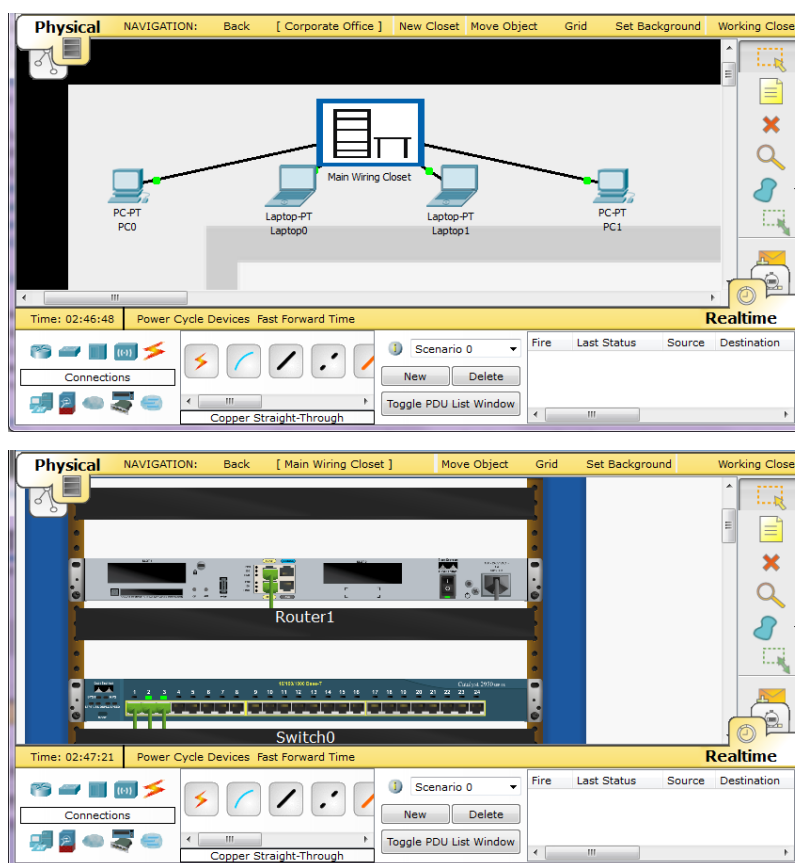


To, co jsme právě navrhli, se v terminologii Packet Traceru nazývá *logická topologie*, třebaže ve skutečnosti jsme používali také prvky fyzické topologie. *Fyzickou topologií* v terminologii Packet Traceru nazýváme návrh konkrétního umístění v místnostech a racku. Přepínač mezi logickou a fyzickou topologií je vlevo nahoře (viz obrázek vlevo), tedy klepneme na „upozaďené“ ouško přepínače a dostaneme se do okna pro fyzickou topologii.



Příklad 5.2 (Jak se dostat k fyzické topologii)

V okně pro fyzickou topologii se nám nejdřív zobrazí něco na způsob mapy. Po klepnutí na mapu se dostaneme k budově, pak klepnutím na obrázek racku až k samotnému racku. V našem případě by v něm byly dva switche a jeden router.



Tady například vidíme, které porty jsme použili při zapojování kabelů, a dokonce tu máme blikající kontrolky značící aktivitu. Nahoře je odkaz pro přesun o úroveň výše (Physical NAVIGATION: Back). Klepnutím na levé ouško přepínače nahoře vlevo se dostaneme zpět do logické topologie.



Úkol

Pokud máte možnost, nainstalujte si Packet Tracer. Podle příkladů si vyzkoušejte vytvoření jednoduché topologie a projděte si prostředí programu včetně vestavěné nabídky zařízení a spojů. To, co vytvoříte, se dá taky uložit (přes ikonu nebo v menu, jako v kterémkoliv jiném editoru).



5.3 Konfigurace koncového zařízení

Každé zařízení je třeba předem nakonfigurovat. V Packet Traceru stačí na zařízení klepnout a v dialogovém okně, které se zobrazí, zadáme všechny parametry.

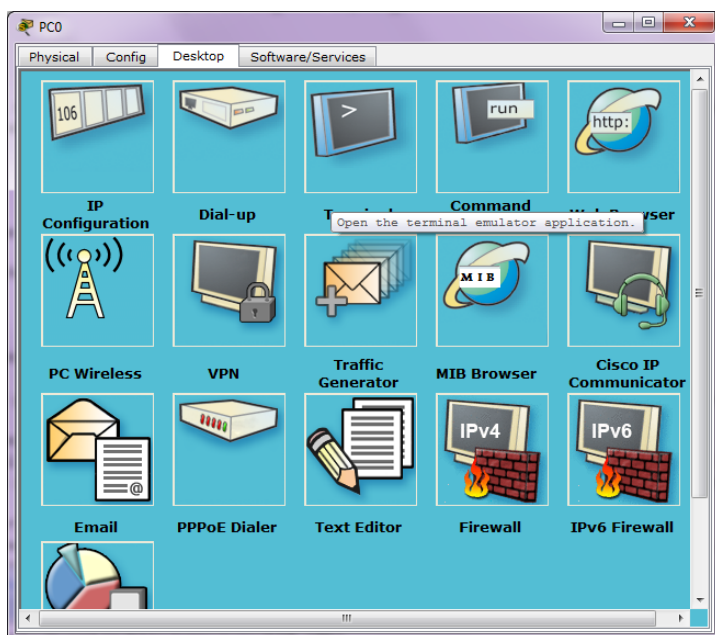
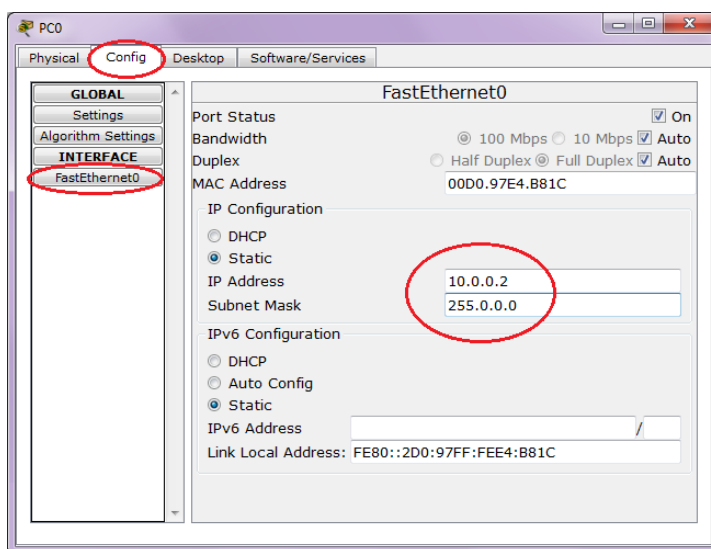


Příklad 5.3 (Konfigurace IP adresy počítače)

Předpokládejme, že na pracovní ploše Packet Traceru máme již několik zařízení, třeba v topologii podle předchozích příkladů. Vybereme si některý počítač a klepneme na jeho ikonu. Objeví se dialogové okno s několika záložkami.

Na první záložce můžeme například zadat jméno zařízení nebo se dají procházet jednotlivá rozhraní v zařízení, ale momentálně nás nejvíc zajímá záložka *Config*. Zde si v levém panelu najdeme příslušné síťové rozhraní (v našem případě FastEthernet0) a pak zadáme zvolenou IP adresu tak, jak vidíme na obrázku.

O IP adresách se budeme podrobněji učit později, teď nám stačí vědět, že našemu zařízení přiřazujeme adresu 10.0.0.2 a maska podsítě je 255.0.0.0 (doplní se automaticky). Zavřením okna svou volbu uložíme.



Zajímavá je také záložka *Desktop* – nabízí nám podobné možnosti, jako bychom u dotyčného zařízení přímo seděli a používali jeho nástroje. I zde můžeme nastavit IP adresu, otevřít terminál či příkazový řádek (podle toho, v jakém si přejeme být operačním systémem), můžeme otevřít „jako“ webový prohlížeč a generovat HTTP provoz, atd.

Základní konfiguraci máme za sebou, ale to není zdaleka všechno, co se dá ovlivňovat. Někdy potřebujeme na simulovaném zařízení rozjet i něco dalšího – například potřebujeme, aby se nějakým způsobem projevoval operační

systém (ne že by tam nebyl, ostatně u IP adresy se předpokládá, že je definována v operačním systému, ale zatím jsme nic takového nemuseli zohledňovat). Taky se dají nainstalovat další typické aplikace, to vše na poslední záložce dialogového okna – *Software/Services*.



Úkol

Projděte si konfiguraci koncového zařízení (umístěte některé koncové zařízení na pracovní plochu Packet Traceru, klepněte na ně). Koncovému zařízení přiřadíte IP adresu, třeba tu podle příkladu. Všimněte si, jaké jsou MAC adresy zařízení.



5.4 Monitorování paketů

Packet Tracer se dá používat i jinak než jen k návrhu topologie sítě, můžeme v něm také pozorovat procházení paketů sítě. K tomu ale kromě nadefinování topologie potřebujeme zprovoznit (nakonfigurovat) jednotlivá zařízení a dále generovat nějaký provoz (simulovat činnost).



Příklad 5.4 (Monitorování provozu na ethernetové síti)

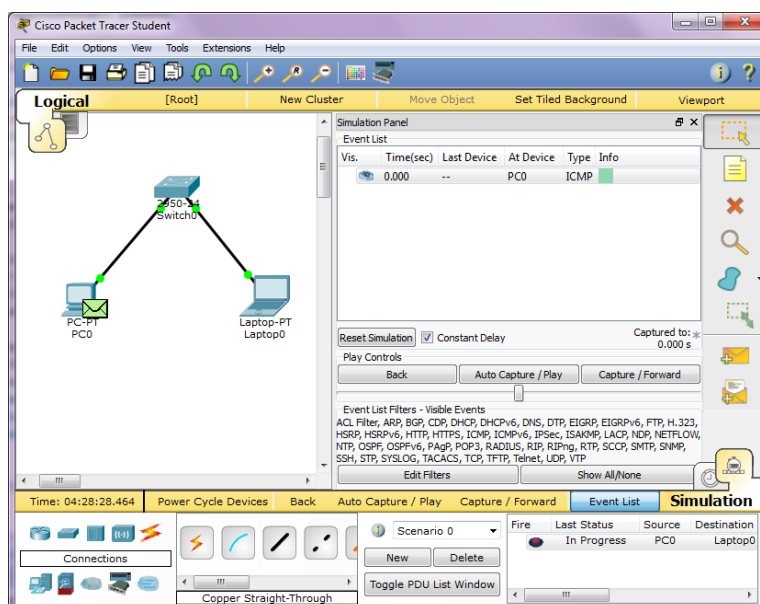
Na pracovní ploše si vytvoříme logickou topologii sítě s jedním switchem a dvěma počítači (či počítačem a notebookem, dle vlastní fantazie), přičemž obě koncová zařízení propojíme se switchem. Postupujeme stejně jako v předchozích příkladech. Jednomu koncovému zařízení přiřadíme IP adresu 10.0.0.2, druhému adresu 10.0.0.3. Masky bude v obou případech 255.0.0.0.



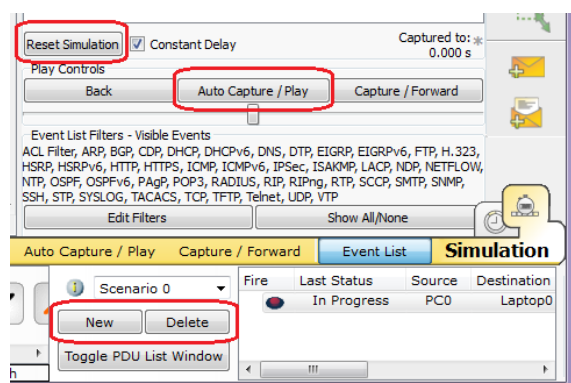
Abychom mohli sledovat provoz na síti, musíme se přepnout do *simulačního režimu*. To provedeme tak, že klepneme na šedé ouško přepínače vpravo dole, viz obrázek.

V simulačním režimu se zaměříme na jedno ze dvou tlačítek na panelu vpravo (s obálkami, viz obrázek). Ta horní (*Add Simple PDU*) je pro nás momentálně naprosto dostačující.

Takže klepneme na ikonu horní obálky a pak klepneme postupně na jeden počítač a pak na druhý. Tím dáme najevo, odkud kam má paket jít, a automaticky se v něm nastaví zdrojová a cílová IP adresa (které jsme předem pro zařízení nakonfigurovali). Okno Packet Traceru by teď mělo vypadat zhruba tak jak na následujícím obrázku.



Kdybychom chtěli profiltrovat seznam protokolů, které nás zajímají, pak pod seznamem protokolů v pravé spodní čtvrtině okna klepneme na *Edit Filters*.



Pak spustíme provoz – klepneme na tlačítko *Auto Capture/Play* (máme je tam dokonce dvě), a můžeme sledovat, kde se momentálně paket nachází a které protokoly vstupují do komunikace. Komunikaci ukončíme klepnutím na tlačítko *Reset Simulation*.

Výchozí typ komunikace při vytváření PDU je ping, tedy jednoduše na jednom zařízení žádáme o odezvu od druhého zařízení. Pokud bychom chtěli


jiný typ komunikace, museli bychom použít volbu *Add Complex PDU* (to je ta druhá obálka).

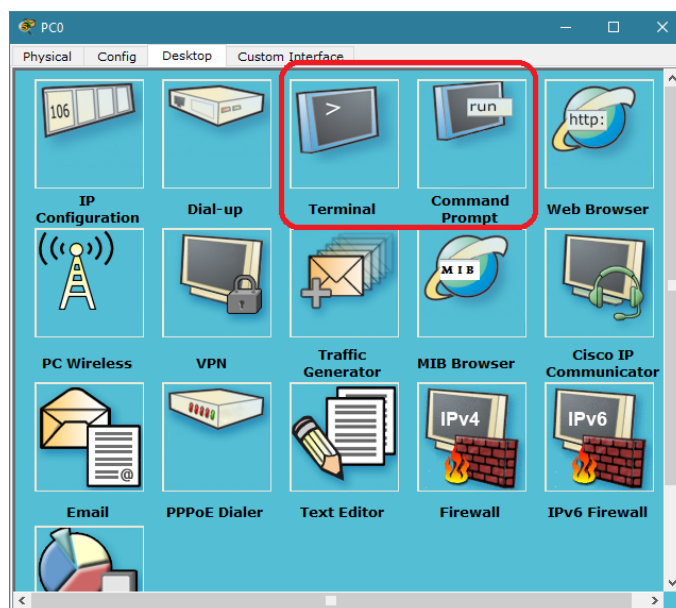
Právě jsme vytvořili první *scénář* (scenario). Ke správě scénářů používáme ovládací prvky v okně dole. Můžeme mít vytvořeno víc scénářů (víc druhů komunikace) tlačítkem *New* a přepínat se mezi nimi. Scénář vymažeme (i v případě, že se nám něco nepovedlo) klepnutím na tlačítko *Delete*.




5.5 Konfigurace zařízení Cisco

5.5.1 Jak se připojit

 Pokud klepneme na ikonu počítače, kterou jsme si „přetáhli“ na plochu, dostaneme konfigurační okno s několika záložkami. Na kartě *Config* zadáváme IP adresu apod., na kartě *Desktop* máme přístup k nástrojům daného počítače. Obsah této karty vidíme na obrázku 5.2. Momentálně nás zajímají nástroje *Terminal* a *Command Prompt*.



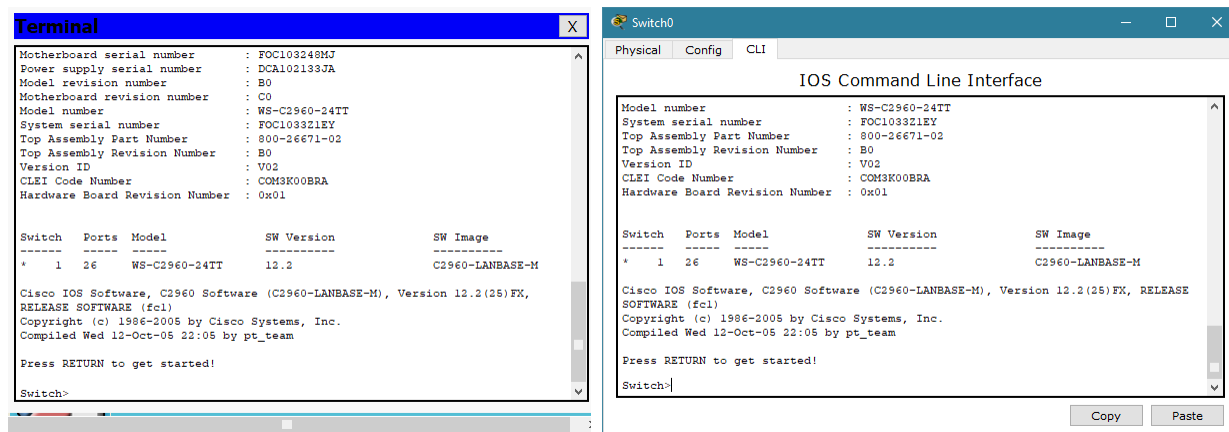
Obrázek 5.2: Nástroje počítače v Packet Traceru

 Pokud chceme konfigurovat síťové zařízení (switch nebo router), musíme se k němu připojit buď přes konzolový kabel, nebo přes Telnet či SSH. Ovšem Telnet a SSH je možné použít až tehdy, kdy má zařízení přiřazenu IP adresu a je alespoň v základu zabezpečeno, tedy konzolovému kabelu se minimálně na začátku práce nevyhne.

V Packet Traceru se ke konfiguraci přes konzolu dostaneme jedním z těchto způsobů:


- stejně jako ve „fyzickém“ případě, tj. vybereme konzolový kabel (modrý), na počítači zapojíme do portu RS-232, na síťovém zařízení do konzolového portu (Console), na počítači přejdeme do nástroje *Terminal* (potvrdíme parametry, pak se nám zobrazí CLI),
- klepneme na síťové zařízení a přejdeme na záložku CLI.

Druhý způsob jde samozřejmě jenom v Packet Traceru a nepotřebujeme k němu na pracovní ploše nic dalšího. V každém případě se dostaneme do *rozhraní CLI* (Command Line Interface) operačního systému IOS, který běží na Cisco zařízeních. Na obrázku 5.3 vidíme rozhraní CLI (vlevo podle prvního způsobu, vpravo podle druhého způsobu).




Obrázek 5.3: Záložka CLI při konfiguraci switche (vlevo: přes počítač, vpravo: přímo)

V rozhraní CLI už můžeme zadávat příkazy.

 Ať už jsme se připojili jakkoliv, vždy se napojujeme na některou *linku*. Ta linka je buď konzola (console) v případě použití konzolového portu, nebo virtuální terminál (VTY) v případě telnetu a SSH. Použitá linka je aktivována a propojena s určitým rozhraním (interface): konzola se propojuje vždy s konzolovým portem, VTY se propojují se síťovými porty (FastEthernetxxx, GigabitEthernetxxx apod.), podle toho, kam jsme se fyzicky připojili.

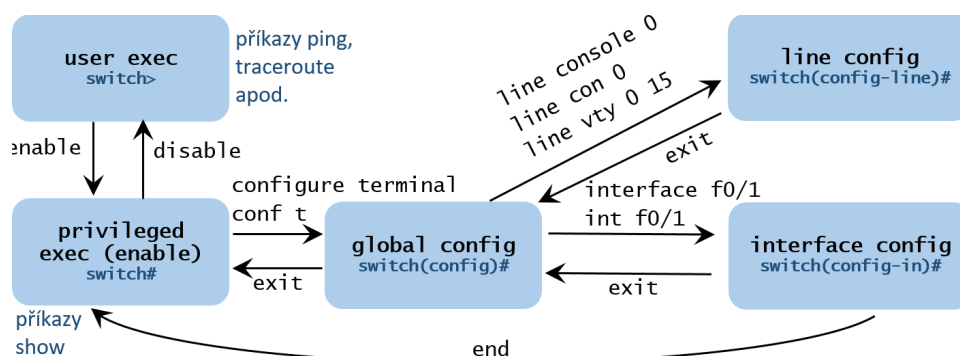
5.5.2 Pracovní módy

 Na zařízeních Cisco rozlišujeme několik různých pracovních módů (režimů). Na obrázku 5.4 jsou jak jednotlivé módy, tak i možnosti přechodů mezi nimi.


- *User exec* – základní režim, do kterého se dostaneme po připojení k zařízení za účelem konfigurace. Prompt má formu názvu zařízení následovaného symbolem „větší“ (>). V tomto módu lze nanejvýš poslat ping či traceroute.

Tento mód může být chráněn přístupovým heslem (které je nastaveno na konzoli a/nebo na virtuálních terminálech pro Telnet či SSH, jak zjistíme na dalších stranách).

- *Privileged exec* (také mu říkáme enabled) – vyšší režim, ve kterém typicky zadáváme příkazy pro výpis (například výpis tabulky MAC adres, výpis nastavení SSH, atd.). Tento mód bývá chráněn tzv. „enabled“ heslem. To, že jsme v enabled režimu, poznáme podle promptu – je to název zařízení následovaný symbolem hash (#).
- *Globální konfigurační mód* – v tomto režimu zadáváme takové konfigurační příkazy, které mají vliv na celé zařízení, například nastavujeme název zařízení nebo heslo do enabled módu či přidáváme řádek do směrovací tabulky. V promptu se nám po přechodu do tohoto režimu objeví mezi názvem zařízení a symbolem # řetězec „config“.
- *Subkonfigurační módy* – konfigurujeme něco konkrétního, například rozhraní (port) nebo linku:
 - subkonfigurační mód pro linku (buď konzolu nebo virtuální terminál) slouží pro nastavení dané linky, například pro danou linku nastavujeme heslo nebo v případě konzoly způsob zobrazování výstupů syslogu,
 - subkonfigurační mód pro rozhraní (některý port) slouží pro nastavení rozhraní, například u routeru nastavujeme na rozhraní IP adresu.



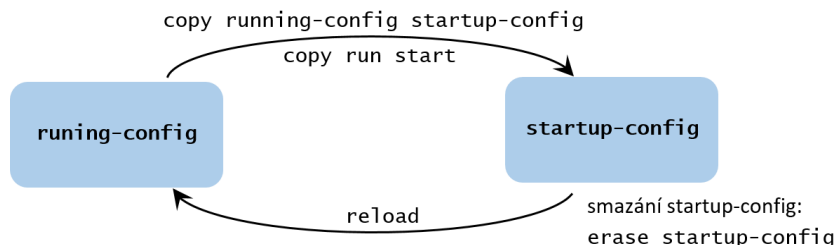
Obrázek 5.4: Módy na zařízeních Cisco

 Když provádíme změny, tyto změny se projeví pouze v běhové konfiguraci (*running-config*). To je RAM čip, tedy po restartu zařízení by byly veškeré změny zahozeny. Proto se po provedení změn doporučuje uložit běhovou konfiguraci do místa, ze kterého se změny jen tak neztratí, do *startup-config*. Startup-config si svůj obsah udrží i po restartu (tomu typu čipu říkáme NVRAM = Non-volatile RAM, v reálu jde o flash čip), a během startu zařízení je právě z něj konfigurace načítána.

Oficiálně máme v síťovém zařízení tyto druhy pamětí:

- ROM (v reálu NOR FLASH): firmware zařízení,
- flash: obsahuje obraz operačního systému IOS, který se používá na Cisco zařízeních (jednoduše soubor, ze kterého startuje systém),
- running-config (soubor v RAM): konfigurace, která se právě používá; pokud děláme změny pomocí příkazů (například nastavujeme IP adresu, heslo apod.), tyto změny se projeví zde, ale jde o RAM, tedy po restartu je smazaná,
- startup-config (soubor v čipu NVRAM/flash): odtud se načítá konfigurace při startu systému IOS, sem můžeme uložit running-config po provedení změn, aby se neztratily.

Jak running-config, tak i startup-config jsou ve skutečnosti soubory a jako se soubory se s nimi může zacházet (například můžeme zálohovat konfiguraci prostým zkopírováním souboru třeba přes síť). Na obrázku 5.5 máme místa pro uložení konfigurace a přechody mezi nimi.



Obrázek 5.5: Uložení konfigurace na zařízeních Cisco

Jsou tam tři základní příkazy:

- `copy running-config startup-config` (nebo zkráceně `copy run start`) uloží běhovou konfiguraci z RAM paměti do startup-config, kde zůstane i po restartu,
- `reload` – pokud se nám nelíbí obsah running-config, můžeme softwarově „restartovat“ zařízení a natáhnout do RAM obsah startup-config (tj. smažeme všechny změny, které jsme provedli různými konfiguračními příkazy),
- `erase startup-config` (nebo zkráceně `erase start`) smaže uloženou konfiguraci, tedy jakýsi restart do továrního nastavení, ovšem running-config se pořád používá, dokud nezadáme `reload` (pozor, při reloadu jsme dotázáni, jestli nechceme uložit running-config do startupu; pokud chceme opravdu smazat konfiguraci, odmítneme).

5.5.3 Konfigurace switche

Pokud se nám dostane do rukou switch bez konfigurace (nově koupený nebo se smazanou konfigurací), v podstatě i tak bude fungovat. Ovšem nikoliv bezpečně, protože jeho konfigurace není nijak chráněna, nejsou nastavena žádná hesla a není možné se na něj dostat přes Telnet ani SSH.

 Takže takový switch bychom měli nakonfigurovat minimálně takto:

- nastavit název zařízení, ať ho při vzdáleném přístupu snadno najdeme a poznáme,
- nastavit hesla do enabled módu a na linky, také je zaheslovat,
- můžeme nastavit banner (hlášku, která se objeví každému, kdo se pokusí k zařízení připojit),
- zprovoznit vzdálený přístup (přiřadit IP adresu na VTY a nastavit adresu brány).

Následuje okomentovaný seznam příkazů. Prompt (tedy výzva, kterou vypisuje zařízení) je zabarven modře, příkaz zůstává černou barvou.

Nejdřív přejdeme do enabled módu (zatím není nastaveno heslo, takže se rovnou zobrazí „hash“ prompt), pak do globálního konfiguračního módu, a zadáme název zařízení.

`Switch>enable` přejdeme do módu „enabled“ (privileged exec)

`Switch#configure terminal` přejdeme do globálního konfiguračního módu

`Switch(config)#hostname S1` nastavíme název switche

V promptu se automaticky změnil název zařízení.

Dále nastavíme heslo nejdřív do enabled módu, a pak na konzoli, plus další parametry. Heslo do enabled módu se týká celého zařízení (proto bude zadáno v globálním konfiguračním módu, kdežto heslo na linky (konzolu a terminály) se týká pouze těchto linek, tedy bude zadáno v příslušném subkonfiguračním módu. V případě konzoly přejdeme do subkonfiguračního módu, zadáme heslo a použijeme příkaz `login`, který slouží k vynucení používání hesla na konzoli (jinak bychom ho sice měli nastavené, ale nebylo by vyžadováno).

```
S1(config)#enable secret heslo    nastavíme heslo do enabled módu
S1(config)#line console 0         přejdeme do konfigurace konzole
S1(config-line)#password heslo    nastavíme heslo pro přístup do konzole
S1(config-line)#login             vynutíme si, že na konzoli musí uživatel heslo opravdu zadat, bez toho
                                  není na konzoli vůbec puštěn
S1(config-line)#logging synchronous  ať na konzoli syslog moc neotravuje
S1(config-line)#exit              ukončíme konfiguraci konzole, přejdeme do globálního konfiguračního
                                  módu (tento příkaz není nutný, pokud pak budeme přecházet do jiného subkonfigu)
```

Syslog (služba, která má na starosti logování událostí) ve výchozím nastavení posílá na konzoli hlášení o téměř veškerých událostech v systému, což je sice užitečné, při větším množství událostí to nepříjemně zasahuje do zadávání příkazů (hlášení se míchají do zadávaných příkazů). Příkaz `logging synchronous` způsobí, že sice se hlášení vypisují, ale ne tak „agresivně“. Příkaz `exit` nás posune do nadřazeného módu, tedy globálního konfigu.

Konzola by byla, teď nastavíme virtuální terminály. Běžné switche mají linky VTY očíslované od 0 do 15, tedy při zadání subkonfiguračního módu bychom na žádnou neměli zapomenout (rozsah se zadává oddělený mezerou). Konzola je jenom jedna, s číslem 0 (proto jsme u předchozího subkonfigu zadávali za `line console` tu nulu). takže přejdeme do subkonfiguračního módu pro (všechny) virtuální terminály a nastavíme heslo stejným způsobem jako u konzoly. Syslog nemusíme řešit, na VTY se standardně nic nevypisuje.

```
S1(config)#line vty 0 15          přejdeme do konfigurace virtuálních terminálů (telnet, SSH)
S1(config-line)#password heslo    nastavíme heslo na terminály
S1(config-line)#login             vynutíme si, že v terminálech je nutné heslo zadat
S1(config-line)#exit              ukončíme konfiguraci virtuálních terminálů
```

Všimněte si, že pro zadání hesla do enabled módu používáme klíčové slovo `secret`, kdežto pro zadání hesel na linky je to `password`.

Dalším krokem je dodatečné zabezpečení hesel. Ve výchozím nastavení je totiž řada hesel (na konzoli a VTY) uložena v souboru `running-config` v textovém formátu bez šifrování a kdokoli, kdo může vypsat konfiguraci, se k nim dostane. Následující příkaz zadaný v globálním konfiguračním módu zajistí, že hesla jsou v konfiguraci uložena ve formě nečitelných hash řetězců. Další příkaz nastaví banner (varovné hlášení), který se zobrazí při přístupu na zařízení přes konzolu a VTY.

```
S1(config)#service password-encryption  heslo se ukládá do running-config, kde je „viditelné“,
                                          ale po tomto příkazu tam bude místo hesla jen jeho hash = bezpečnější
S1(config)#banner motd #neautorizovany pristup je prisne zakazan#  nastavíme „varovací in-
formaci, aby hacker věděl, že v konfiguraci nemá co dělat
```

Proč ten banner? V mnoha zemích totiž platí, že pokud hacker není „informován“, že něco nesmí, tak u soudu nebo jakéhokoli jiného podobného orgánu může použít výmluvu „ale já jsem nevěděl(a), že to nesmím...“

Abychom mohli k zařízení přistupovat vzdáleně přes Telnet nebo SSH, musí mít přiřazenou IP adresu (protože ta se zadává jako parametr dotyčného příkazu). Switch pracuje na vrstvě L2, takže IP adresu nemůžeme přiřadit na „fyzické“ rozhraní, ale někam to jít musí... od toho máme virtuální rozhraní (pozor, není to totéž jako virtuální terminál). Použijeme virtuální rozhraní VLAN 1 (ano, je to opravdu virtuální lokální síť, ale to teď nehraje roli). Takže přejdeme do subkonfiguračního módu pro dané rozhraní (nikoliv linku), nastavíme adresu a zapneme příkazem `no shutdown`. Taky budeme potřebovat bránu (nebyla by nutná, pokud bychom zůstali v lokální síti), tu nastavíme v globálním konfiguračním módu, protože se týká všech potenciálně používaných rozhraní.

```
S1(config)#interface vlan 1    přejdeme do konfigurace pro virtuální rozhraní VLAN 1
S1(config-if)#ip address 10.0.10.2 255.255.255.0    nastavíme mu IP adresu (zadáme IP ad-
    resu a masku)
S1(config-if)#no shutdown    zapneme (tj. negace vypnutí)
S1(config-if)#exit    ukončíme konfiguraci rozhraní
S1(config)#ip default-gateway 10.0.10.1    nastavíme výchozí bránu
```

U Cisca se často setkáváme s tím, že popření některého příkazu se udělá prostě tak, že před původní příkaz přidáme popírající `no`, jako to je u výše uvedeného `no shutdown`. Pro vypnutí rozhraní máme příkaz `shutdown`, pro zapnutí jeho popření.

Ještě poznámka – L2 rozhraní jsou ve výchozím stavu zapnutá, kdežto L3 rozhraní jsou ve výchozím stavu vypnutá. Z bezpečnostních důvodů budeme chtít, aby nepoužívaná rozhraní byla vypnutá i na vrstvě L2. Switch má rozhraní poměrně hodně, a abychom nemuseli postupně procházet jedno po druhém, použijeme příkaz `range` pro rozsah:


```
S1(config)#interface range f0/4,8-12    můžeme pracovat s více rozhraními najednou (rozsah)
S1(config-in)#shutdown    vypneme nepoužívaná rozhraní
```

Ted' by bylo fajn se podívat, jak jsme to všechno nakonfigurovali. K tomu slouží různé příkazy `show`. Pozor, `show` příkazy se zadávají v enabled módu, nikoliv v konfiguračních módech, tedy použijeme příkaz `end` (nebo několikrát příkaz `exit`, abychom se dostali do enabled módu. Následuje seznam několika nejužitečnějších `show` příkazů.

```
S1(config-in)#end    přeskočíme rovnou do enabled módu
S1#show ip interface brief    zobrazíme tabulku rozhraní (každé rozhraní na jednom řádku,
    včetně toho virtuálního, které má přiřazenou IP adresu)
S1#show interfaces    zobrazíme podrobné informace o rozhraních, hodně dlouhý výpis (me-
    zerníkem se dostáváme na další stránky výpisu)
S1#show interface f0/1    zobrazíme podrobné informace o rozhraní FastEthernet 0/1 (tedy f0/1)
S1#show mac address-table    zobrazíme tabulku MAC adres
S1#show running-config    zobrazíme obsah souboru running-config
S1#show startup-config    zobrazíme obsah souboru startup-config
S1#copy running-config startup-config    uložíme running-config do startup-config
```

Pokud jsme v některém konfiguračním módu (včetně subkonfiguračních) a nechce se nám kvůli show příkazům přecházet do enabled, přidáme před příkaz klíčové slovo `do`:

`S1(config-in)#do show running-config` zadáme show příkaz v jiném než enabled módu

 Komu by se chtělo psát tak dlouhé příkazy, když to jde krátit? Obecně můžeme krátit všechny příkazy, klíčová slova a označení linek a rozhraní, pokud zadání zůstává jednoznačné. Například:

Plná verze příkazu	Zkrácená verze příkazu
<code>enable</code>	<code>ena</code>
<code>configure terminal</code>	<code>conf t</code>
<code>show ip interface brief</code>	<code>sh ip int br</code>
<code>show running-config</code>	<code>sh run</code>
<code>show startup-config</code>	<code>sh start</code>
<code>copy running-config startup-config</code>	<code>copy run start</code>

Tabulka 5.1: Příklady zkracování příkazů

Další příkazy show: například `sh arp`, `sh version`, `sh protocols`, `sh cdp`, `sh cdp neighbors`




Poznámka:

Někdy se stane, že se to „sekne“ nebo je výpis příliš dlouhý a nám už se to nechce dál procházet. V některých případech (příliš dlouhý výpis) stačí použít klávesovou zkratku `Ctrl+C`, a pokud se to „sekne“, tak `Ctrl+Shift+6`.

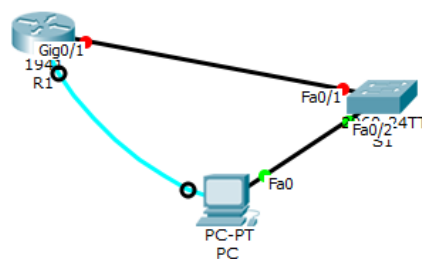


5.5.4 Konfigurace routeru

Zabezpečení routeru (hesla do enabled módu, na konzoli a VTY, banner apod.) se dělá úplně stejně jako na switchi. Navíc jsou postupy, které se vztahují k vrstvě L3. Zatímco na switchi jsme nastavovali IP adresu na virtuálním rozhraní, na routeru budeme nastavovat IP adresy na „fyzických“ rozhraních.

 Takže zařízení pojmenujeme, nastavíme hesla do enabled módu a na VTY (routery mají méně virtuálních terminálů, obvykle 0 až 4), zašifrujeme hesla v souboru s konfigurací a nastavíme banner:

```
Router>enable
Router#configure terminal
Router(config)#hostname R1
R1(config)#enable secret heslo
R1(config)#line console 0
R1(config-line)#password heslo
R1(config-line)#login
R1(config-line)#logging synchronous
```




Obrázek 5.6: Počítač připojený konzolovým kabelem k routeru


```
R1(config-line)#exit
R1(config)#line vty 0 4
R1(config-line)#password heslo
R1(config-line)#login
R1(config-line)#exit
R1(config)#service password-encryption
R1(config)#banner motd #Neautorizovany pristup prisne zakazan#
```

Tedy na rozhraních GigabitEthernet 0/0 (tedy g0/0) a 0/1 nastavíme popisek (description) a IP adresu, nesmíme zapomenout je zapnout (tady je to opravdu důležité, na routeru jsou L3 rozhraní ve výchozím nastavení vypnutá). Výhodou je, že nemusíme řešit vypínání nepoužívaných portů.

```
R1(config)#interface g0/0    přecházíme do konfigurace rozhraní
R1(config-if)#description linka do site 10    nastavíme popisek rozhraní
R1(config-if)#ip address 10.0.10.1 255.255.255.0    přidělíme rozhraní IP adresu
R1(config-if)#no shutdown    zapnutí rozhraní
R1(config-if)#interface g0/1    další rozhraní
R1(config-if)#description linka do site 20
R1(config-if)#ip address 10.0.20.1 255.255.255.0
R1(config-if)#no shutdown
```


 Pro ověření, zda jsme postupovali správně, můžeme použít show příkazy (stejně jako na switchech), ale také můžeme zkusit ping a traceroute. Oba tyto příkazy se zadávají buď v user exec módu, nebo v enabled módu, tedy musíme přejít alespoň do enabled:

```
R1(config-if)#end    přejdeme do enabled módu
R1#ping 10.0.10.2    pingneme zadanou adresu (tuto adresu jsme v předchozí sekci přidělili switchi, předpokládáme, že tato zařízení jsou propojena)
R1#traceroute 10.0.10.2    vypíšeme trasu, v tomto případě asi bude krátká
```

 Zatím jsme pracovali jen s IPv4 adresami. S IPv6 adresami se zachází podobně, jen místo příkazu ip zadáváme ipv6, místo masky zadáváme délku prefixu a pak jsou tu další drobné odlišnosti (například na každém rozhraní obvykle míváme kromě globálně platné adresy nastavenou i link-local adresu, dále IPv6 směrování je nutné zapnout, atd.).

Takže na rozhraní g0/1 nastavíme jednu globální IPv6 adresu a jednu link-local (ona by se nastavila sama, ale „škaredá“, a my chceme krátkou adresu, která se nám bude snadno zadávat), a ověříme nastavení pomocí show příkazů.

```
R1(config-if)#interface g0/1    chceme konfigurovat rozhraní GigabitEthernet 0/1
R1(config-if)#ipv6 address 2001:db8:acad:1::1/64    nastavíme IPv6 adresu (globálně platnou)
R1(config-if)#ipv6 address fe80::1 link-local    nastavíme link-local adresu (tu můžeme dát stejnou na všechna rozhraní téhož routeru, protože platí jen v dané síti, takže klidně souhrnně přes int range)
R1(config-if)#no shutdown    zapneme
R1(config-if)#end    přechod do enabled režimu
R1#sh ipv6 int brief    zobrazíme si tabulku rozhraní, měly by tam být IPv6 adresy
R1#sh ipv6 int g0/1    vlastnosti konfigurovaného rozhraní
```


 Na routeru nás zajímá ještě jedna věc – směrovací tabulka. Následující příkazy nám ukazují, jak pracujeme se směrovací tabulkou (jenom základ, dynamické směrování je nad rámec tohoto předmětu):

```
R1#sh ip route    zobrazí směrovací tabulku pro IPv4
R1#sh ipv6 route  zobrazí směrovací tabulku pro IPv6
R1(config)#ipv6 unicast-routing  zapnutí IPv6 směrování
R1(config)#ip route 192.168.20.0 255.255.255.0 209.165.200.225    přidá do směrovací tabulky
    statický záznam (zadáváme IP adresu, masku a IP adresu sousedního zařízení, přes které
    vede cesta)
R1(config)#ip route 192.168.20.0 255.255.255.0 g0/0    podobně, ale místo souseda zadáváme
    rozhraní směřující k dané síti (naše)
R1(config)#ip route 0.0.0.0 0.0.0.0 172.16.64.2    default route (cesta k našemu poskytovateli
    internetu) pro IPv4
R1(config)#ipv6 route ::/0 2001:db8:acad:4::2    default route pro IPv6
```

5.5.5 Telnet a SSH

Pokud chceme na zařízení přistupovat vzdáleně (což asi chtít budeme, ne každý může kdykoliv zaskočit do datového centra), musíme zařízení přidělit alespoň jednu IP adresu (switchi na virtuální rozhraní některé VLAN sítě, která bude přes daný fyzický port dostupná, routeru na ten port, přes který budeme vzdáleně přistupovat), a taky musí být nastavená hesla. Obvykle nenastavujeme pouze heslo, ale radši nastavíme kompletní přihlašovací údaje (jméno + heslo pro nejméně jeden účet), a to i pro Telnet. Pro Telnet by to stačilo, SSH navíc ještě potřebuje vygenerovat šifrovací klíč.

Ukážeme si postup na switchi, na routeru by to bylo podobné (až na to, že IP adresy bychom samozřejmě nastavovali na fyzických rozhraních).

 Takže nejdřív zabezpečit enabled režim a zajištit šifrování hesel (je jedno, v jakém pořadí):

```
Switch>enable    přechod do enabled módu
Switch#conf t    přechod do globálního konfiguračního módu
Switch(config)#hostname S1    nastavení názvu
S1(config)#service password-encryption    šifrování hesel v konfiguraci
S1(config)#enable secret heslo    nastavení hesla do enabled módu
S1(config)#banner motd #neautorizovany pristup je prisne zakazan#
```

Dále budeme předpokládat, že chceme používat SSH. Nastavíme verzi SSH na 2 (je možné, že bude nastavena automaticky, záleží na verzi systému IOS). Pak nastavíme doménové jméno (je jedno, jakou doménu zadáme, může být vymyšlená, je tu kvůli tomu, že se použije při generování klíčů), následně vygenerujeme RSA klíč.

```
S1(config)#ip ssh version 2    používání verze 2
S1(config)#ip domain-name firma.com    pouze pro účely generování klíčů, klidně nesmysl
S1(config)#crypto key generate rsa    vygenerujeme klíče; spustí se průvodce, délku klíče zadáme
    ideálně 1024 (příliš krátký klíč by způsobil, že budeme přepnuti na SSH verze 1)
```

Na jednotlivých linkách (VTY a pro jistotu i konzola) si vynutíme používání SSH, následující příkaz určí, že uživatelé mají být autentizováni vůči lokální databázi, v reálu to znamená, že místo prostého zadávání hesla budou vytvořeny opravdové uživatelské účty, každý se jménem a heslem.


```
S1(config)#line vty 0 15    (na routeru bude linek méně)
S1(config-line)#transport input ssh    zakážeme telnet, přes síť pouze SSH (pokud bychom
    chtěli naopak jen telnet, zadali bychom transport input telnet, kdybychom chtěli obojí,
    pak transport input all)
S1(config-line)#login local    autentizace proti lokální databázi (na linkách nebude jen heslo,
    ale budou se používat plnohodnotné uživatelské účty)
S1(config-line)#exit
S1(config)#username uzivatel1 secret heslo    vytvoříme uživatele (zde dva)
S1(config)#username uzivatel2 secret heslo
S1(config)#line console 0    podobně pro konzolu
S1(config-line)#login local
S1(config-line)#exit
```

A pak stejně jako předtím – nastavíme IP adresu na virtuálním rozhraní (u routeru na fyzických rozhraních) a pak nastavíme bránu

```
S1(config)#interface vlan 1
S1(config-if)#ip address 10.0.10.2 255.255.255.0
S1(config-if)#no shutdown
S1(config-if)#exit
S1(config)#ip default-gateway 10.0.10.1
S1(config)#exit
```

Také pro SSH máme show příkazy:


```
S1#show ssh
S1#show ip ssh
```


 Další zabezpečení, a abychom to trochu vystřídali, teď budeme na routeru (na switchi to je stejné):


```
R1(config)#no ip domain-lookup    aby se router nepokoušel chybně zadaný příkaz interpretovat
    jako doménové jméno
R1(config)#security passwords min-length 10    při zadávání a změně hesla je vynucována min.
    délka hesel 10
R1(config-line)#exec timeout 5    pokud se na této lince 5 minut nic neděje, je uživatel odhlášen
    (nastavujeme typicky na virtuálních terminálech, tedy předem přesunout do subkonfigurač-
    ního módu pro VTY)
R1#copy start tftp    pokud máme na některém počítači rozběhnutý TFTP server, můžeme si na
    něj stáhnout do zálohy startup-config tohoto zařízení (nebo running-config, je to jednoduché
    kopírování); spustí se průvodce, ve kterém zadáme například IP adresu TFTP serveru a
    cílové umístění na něm)
```


Kapitola 6

Co se děje na síťové a transportní vrstvě

 *Rychlý náhled:* V této kapitole si zopakujeme IP adresy a prozkoumáme formát IPv4 a IPv6 paketů. Následně se podíváme na možnosti využití protokolu ICMP při testování dosažitelnosti a zjišťování cesty k zařízení v síti. Nakonec prozkoumáme TCP a UDP segmenty.

 *Klíčová slova:* Adresa, IPv4, IPv6, paket, Wireshark, ICMP, ping, traceroute, TCP, UDP

 *Cíle studia:* Po prostudování této kapitoly se zorientujete v IPv4/IPv6 paketech a TCP/UDP segmentech, také porozumíte mechanismům testování dosažitelnosti a zjišťování cesty k zařízení.

6.1 Protokol IPv4

6.1.1 Maska a prefix adresy

Jak víme, IP adresu mají nejen konkrétní zařízení, ale také sítě a podsítě, přičemž adresu (pod)sítě je možné zjistit, pokud máme adresu kteréhokoliv zařízení v dané (pod)síti a buď masku nebo číslo označující délku prefixu. IP adresa se tedy skládá ze dvou částí – síťové a hostitelské, přičemž hranice mezi těmito dvěma částmi je dána právě buď maskou nebo délkou prefixu. Ukážeme si, jak z adresy zařízení v síti zjistíme adresu sítě.



Příklad 6.1

Zařízení má přidělenou IP adresu 169.254.237.18 a masku 255.255.128.0. Totéž pomocí délky prefixu: 169.254.237.18/17 (číslo 17 je počet jedniček zleva v binárním zápisu masky). Obě adresy převedeme do binárního tvaru a provedeme na nich operaci logického součinu (AND):

IP adresa zařízení:	10101001.11111110.11101101.00010010
maska:	11111111.11111111.10000000.00000000
operace AND:	10101001.11111110.10000000.00000000

Po převodu do desítkové soustavy získáme adresu sítě: 169.254.128.0/17. Všechna zařízení v této síti mají prvních 17 bitů shodných jak s touto adresou, tak i s ostatními zařízeními v téže síti.





Úkoly

- Pro následující adresy zařízení zapsané s maskou sítě zjistěte délku prefixu a adresu sítě.
 - adresa 10.182.1.240, maska 255.0.0.0
 - adresa 132.197.54.85, maska 255.255.0.0
 - adresa 204.122.129.32, maska 255.255.128.0
 - adresa 172.204.36.64, maska 255.255.192.0
 - adresa 172.204.56.200, maska 255.255.255.128
- Pro následující adresy zařízení zapsané s délkou prefixu určete masku a adresu sítě.
 - adresa 10.0.48.132/8
 - adresa 242.35.131.48/17
 - adresa 132.146.64.7/16
 - adresa 192.168.198.32/18
- Následující adresa není platnou IPv4 adresou. Proč?
192.264.84.0



6.1.2 Zjištění adresy, masky a brány

O IPv4 adresách jsme se již bavili a víme, že pro zjištění své adresy můžeme ve Windows použít příkaz `ipconfig`, v Linuxu pak `ifconfig` nebo `ip address show` (příp. zkrácený tvar).



Příklad 6.2

V Příkazovém řádku ve Windows jsme použili příkaz `ipconfig` s tímto výsledkem (je zobrazen jenom začátek, nám to bude stačit):

Adaptér bezdrátové sítě LAN Bezdrátové připojení k síti 2:

```
Přípona DNS podle připojení . . . :
Místní IPv6 adresa v rámci propojení . . . : fe80::88c:6cd5:872d:3b28%14
Adresa IPv4 . . . . . : 10.0.0.140
Maska podsítě . . . . . : 255.255.255.0
Výchozí brána . . . . . : 10.0.0.138
```

Nás momentálně zajímají řádky **Adresa IPv4**, **Maska podsítě** a **Výchozí brána**. Masku podsítě nám říká, že ze čtyř oktetů (32 bitů) adresy jsou první tři oktety (24 bitů) síťovou částí adresy a zbývající jeden oktet (8 bitů) je hostitelská část adresy. Jiný zápis (pomocí délky prefixu) by byl 10.0.0.140/24. Adresu sítě získáme tak, že všechny bity v hostitelské části adresy nastavíme na 0, tedy 10.0.0.0/24.

Výchozí brána, tedy zařízení, na které budeme směřovat veškerý provoz určený pro cokoli, co není přímo v síti s adresou 10.0.0.0/24, má adresu 10.0.0.138/24 (musí samozřejmě také patřit do téže sítě a také se na ni vztahuje rozdělení na síťovou a hostitelskou část).

Na jiném počítači může výpis vypadat třeba takto:

Adaptér sítě Ethernet Připojení k místní síti:

```
Přípona DNS podle připojení . . . : fpf.slu.cz
IPv6 adresa . . . . . : 2001:718:2601:265:4d9b:7004:31e9:b6df
IPv6 adresa . . . . . : 2001:718:2601:265:ffff:f982:b769:2182
Dočasná IPv6 adresa . . . . . : 2001:718:2601:265:e040:72c1:7a2c:7210
Místní IPv6 adresa v rámci propojení . . . : fe80::4d9b:7004:31e9:b6df%10
```

```

Adresa IPv4 . . . . . : 10.6.13.220
Maska podsítě . . . . . : 255.255.255.0
Výchozí brána . . . . . : 193.84.198.1
                        10.6.13.1

```

Opět se soustředíme až na řádky od **Adresa IPv4**. Vlastně tato adresa vypadá celkem podobně, dokonce je stejná maska. Zápis pomocí délky prefixu je 10.6.13.220/24, adresa sítě je 10.6.13.0/24. Máme tady dvě výchozí brány – ovšem pro nás je relevantní ta druhá, protože patří do naší sítě a můžeme s ní komunikovat: 10.6.13.1/24.



Úkol

Zjistěte svou IPv4 adresu, masku a adresu brány.

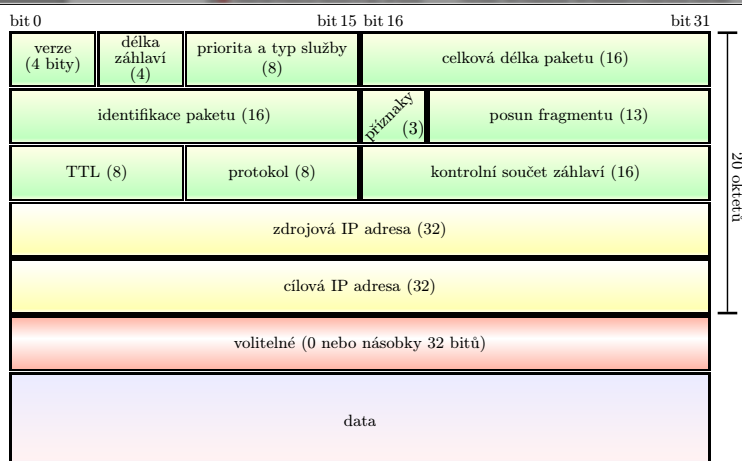
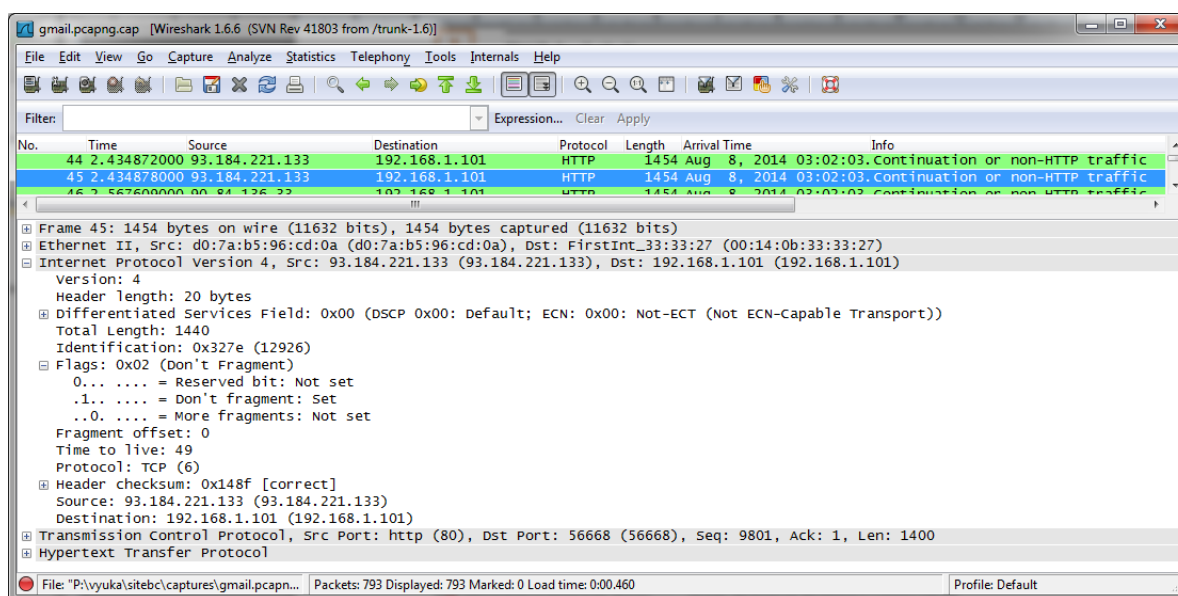


6.1.3 IPv4 pakety



Příklad 6.3

Program Wireshark už známe, tedy se můžeme prozkoumat některý skutečný IPv4 paket.




Když srovnáme snímek z Wiresharku s výše umístěnou strukturou IPv4 paketu, zjistíme, že:

- opravdu se jedná o verzi 4 protokolu IP,
- pole *Header Length* obsahuje údaj pro 20 oktetů (to odpovídá běžné velikosti IPv4 záhlaví),
- pole pro prioritu a typ služby je celé na staveno na 0, což je také běžné,
- pole *Total Length* (celková délka paketu) je 1440 oktetů, a protože je záhlaví dlouhé 20 oktetů, máme v paketu zapouzdřeno 1420 oktetů dat,
- přecházíme do „druhého řádku“ – pole *Identification* obsahuje identifikátor paketu, čtyři hexadecimální číslice představují 16 binárních číslic, resp. dva oktety,
- v příznacích (pole *Flags*) je druhý bit (Do not Fragment) nastaven na 1, tedy je zakázáno paket po cestě fragmentovat,
- *Fragment Offset* je nastaven na 0 (ovšem, při zákazu fragmentace nemohou být žádné fragmenty, které by měly jiný offset),
- hodnota TTL je 49 – pokud například byla původně 64, pak paket prošel přes pět směrovačů nebo L3 switchů, tímto polem jsme se dostali do „třetího řádku“,
- v poli *Protocol* je číslo 6, což znamená, že uvnitř je zapouzdřen TCP segment (o délce 1420 oktetů),
- následuje kontrolní součet záhlaví, který byl ověřen a souhlasí,
- pak tu máme zdrojovou a cílovou adresu.

Podle údajů dále vidíme, že uvnitř je opravdu TCP segment a v něm je zapouzdřena PDU protokolu HTTP.



 Všimněte si, že adresy jsou v IP paketu v opačném pořadí než v rámci vrstvy L2 – tedy máme nejdřív adresu zdroje a až potom adresu cíle.



Úkol

Chvilí odchytávejte provoz Wiresharkem (například při surfování na webu nebo při kontrole e-mailové schránky) a projděte si několik IPv4 paketů. Zjistěte, jaká je délka záhlaví a celková délka paketu, jak jsou nastaveny příznaky, jaká je hodnota TTL, co je uvnitř zapouzdřeno, zda sedí kontrolní součet záhlaví, jaké jsou IP adresy komunikujících stran.

Všimněte si, že hodnoty TTL mohou být v opačných směrech komunikace i hodně odlišné (ovšem záleží na konkrétním provozu, který jste odchyťovali). Hodnota TTL v paketu závisí nejen na konkrétní zvolené cestě a tedy počtu routerů na cestě, ale také na výchozí hodnotě nastavené v odesílajícím operačním systému.



6.1.4 Fragmentace paketu

Ukážeme si, jak funguje fragmentování paketu podle hodnoty MTU na cestě a jeho následné složení. Víme, že MTU (Maximum Transmission Unit) nám říká, jaké je maximum pro velikost paketu, který má být přenesen po dané přenosové cestě. Pokud chceme přenést paket, který je větší než jak dovoluje hodnota MTU, musíme tento paket buď fragmentovat (tedy rozdrobit na menší části a ke každé přidat záhlaví) nebo zahodit.



Postup (Fragmentace IPv4 paketu)

Předně zkontrolujeme, jestli lze provést fragmentaci (příznak DF musí být nastaven na 0). Pokud je fragmentace povolena, postupujeme takto:

1. Vypočteme délku dat pro fragment a počet fragmentů:
 - od MTU odečteme velikost záhlaví paketu (obvykle 20 oktetů),
 - vzniklé číslo vydělíme celočíselně osmi, toto číslo si označme X ,
 - maximální možná velikost dat zapouzdřených do paketu je $X * 8$,
 - pokud je původní délka dat *po odečtení záhlaví* paketu M , pak počet fragmentů je $M/(X * 8) + 1$.
2. Vytvoříme první fragment:
 - většinu záhlaví zkopírujeme z původního paketu,
 - příznak MF (More Fragments) nastavíme na 1,
 - do pole *Posun fragmentu* uložíme číslo 0,
 - jako data vložíme do prvního fragmentu oktety s adresami $0 \dots (X * 8 - 1)$.
3. Vytvoříme další fragmenty kromě posledního (n je pořadí paketu, který zpracováváme):
 - opět přejmeme většinu záhlaví,
 - příznak MF nastavíme na 1,
 - do pole *Posun fragmentu* uložíme číslo $X * (n - 1)$,
 - jako data vložíme oktety s adresami $X * 8 * (n - 1) \dots X * 8 * n - 1$.
4. Vytvoříme poslední fragment ($n = M/(X * 8) + 1$):
 - přejmeme většinu záhlaví,
 - příznak MF nastavíme na 0 (další fragmenty už nebudou),
 - do pole *Posun fragmentu* uložíme číslo $X * (n - 1)$,
 - jako data vložíme oktety s adresami $X * 8 * (n - 1) \dots M - 1$.



Příklad 6.4

Předpokládejme, že chceme poslat (či přeposlat) IPv4 paket s těmito informacemi v záhlaví:

- *Header Length* = 20, *Total Length* = 2657,
- *Identifier* = $0 \times 42B7$,
- *Flags*: DF = 0, MF = 0, *Fragment Offset* = 0.

Na routeru je však pro port, kterým chceme paket poslat, definováno MTU = 865, což znamená, že se paket „nevejde“. Protože je příznak DF (Do not Fragment) nastaven na 0, je možné provést fragmentaci.

Uvnitř paketu je zapouzdřeno $2657 - 20 = 2637$ oktetů dat (odečetli jsme záhlaví). U dat zapouzdřených uvnitř paketu si můžeme představit, že jsou jednotlivé oktety očíslovány „adresami“ $0 \dots 2636$ (nezapomeňte, že i nultý oktet je součástí celku, proto od počtu oktetů odečteme 1).

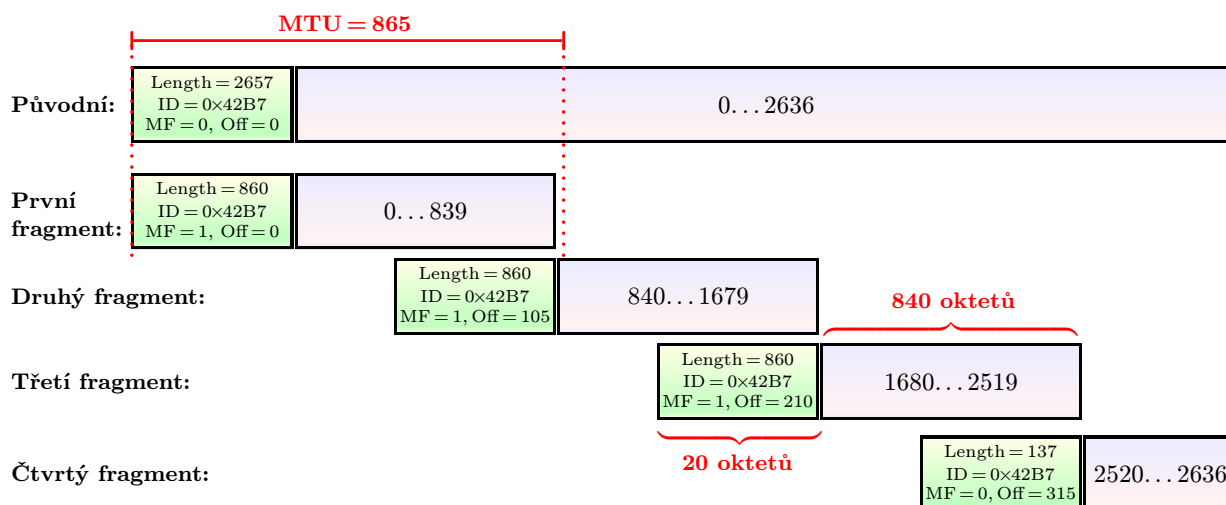
Podle prvního bodu postupu vypočteme délku dat zapouzdřovaných do fragmentu:

- od MTU odečteme záhlaví: $865 - 20 = 845$,
- výsledek celočíselně vydělíme osmi: $845 : 8 = 105$, tedy položíme $X = 105$,
- zapouzdřená data rozdělíme na úseky o délce $X * 8 = 105 * 8 = 840$ oktetů,
- počet fragmentů bude (dělíme celočíselně) $2637 : 840 + 1 = 3 + 1 = 4$ fragmenty.

Do pole *Fragment Offset* uložíme vždy adresu začátku dat fragmentu vydělenou 8. Shrňme si údaje pro jednotlivé fragmenty:

	1. fragment	2. fragment	3. fragment	4. fragment
Počet oktetů ve fragmentu	840	840	840	117
Adresy oktetů	0...839	840...1679	1680...2519	2520...2636
Offset v původních datech	0	840	1680	2520
Pole Fragment Offset	0	105	210	315

Některá pole budou v záhlaví všech fragmentů stejná, především identifikátor paketu a adresy budou přejaty. Z tabulky plyne, co bude v poli *Fragment Offset* a která data konkrétně do jednotlivých fragmentů umístíme. Příznak MF (More Fragments) bude u prvních tří nastaven na 1, u posledního na 0.



Na obrázku vidíme, jak budou fragmenty ve skutečnosti vypadat, včetně některých údajů v záhlaví.



Úkol

Potřebujeme přes cesty s různými hodnotami MTU přenést následující pakety s těmito údaji v záhlaví:

1. Na portu s definovaným MTU = 820:
 - Header Length = 20, Total Length = 920,
 - Identifier = $0 \times 3A82$,
 - DF = 0, MF = 0, Fragment Offset = 0.

2. Na portu s definovaným MTU = 820:
 - Header Length = 24, Total Length = 1424,
 - Identifier = 0×59FB,
 - DF = 0, MF = 0, Fragment Offset = 0.
3. Na portu s definovaným MTU = 520:
 - Header Length = 20, Total Length = 460,
 - Identifier = 0×2B47,
 - DF = 0, MF = 0, Fragment Offset = 0.
4. Na portu s definovaným MTU = 520:
 - Header Length = 20, Total Length = 1610,
 - Identifier = 0×6A18,
 - DF = 0, MF = 0, Fragment Offset = 0.
5. Na portu s definovaným MTU = 520:
 - Header Length = 20, Total Length = 525,
 - Identifier = 0×394C,
 - DF = 1, MF = 0, Fragment Offset = 0.

Ke každému tomuto paketu zjištěte, jak konkrétně s ním bude zacházeno – podle předchozího příkladu.



6.1.5 Složení fragmentovaného paketu

Fragmentovaný paket je třeba taky někdy složit do původní podoby, resp. zkompletovat data ve fragmentech přenášena. Zde si musíme uvědomit, že protokol IP poskytuje obvykle službu datagramového typu, tedy nejen není zaručeno, že všechny fragmenty dojdou do cíle, ale fragmenty vlastně mohou jít různými cestami a mohou do cíle dorazit v jiném pořadí než v jakém byly vyslány.

Z toho vyplývá, že nemá smysl fragmenty skládat po cestě, provádí to až cílové zařízení (zatímco fragmentovat může kterékoliv zařízení vrstvy L3 na cestě, pokud je DF = 0).



Příklad 6.5

Budeme pokračovat v předchozím příkladu, tentokrát z pohledu cílového zařízení.

Cílové zařízení přijme čtyři pakety, které vidíme vpravo. Všechny mají stejný identifikátor, takže je jasné, že patří k sobě.

Takže naším úkolem je uspořádat pakety tak, aby bylo zřejmé, že máme opravdu všechny, a zároveň ve správném pořadí. Především si všímáme pole *Fragment Offset* (na obrázku označeno jako „Off“). Na začátku musí být paket s touto hodnotou 0, takže poslední došlý paket bude právě prvním v pořadí podle dat. Na druhé místo dáme paket s off-setem 105, pak 210 a následně 315.

Length = 860 ID = 0×42B7 MF = 1, Off = 210	
Length = 860 ID = 0×42B7 MF = 1, Off = 105	
Length = 137 ID = 0×42B7 MF = 0, Off = 315	
Length = 860 ID = 0×42B7 MF = 1, Off = 0	

Ale to není všechno. Musíme zkontrolovat:

- zda je u posledního paketu nastaven příznak MF = 0 (ostatní pakety musí mít MF = 1),
- jestli nám nechybí některý paket mezi prvním a posledním.

Length = 860 ID = 0x42B7 MF = 1, Off = 0	
Length = 860 ID = 0x42B7 MF = 1, Off = 105	
Length = 860 ID = 0x42B7 MF = 1, Off = 210	
Length = 137 ID = 0x42B7 MF = 0, Off = 315	

Paket, který jsme zařadili na konec (s offsetem 315), opravdu má MF = 0, takže první podmínka je splněna. Druhou podmínku ověříme tak, že vypočteme hodnotu, o kterou se mají lišit offsety v po sobě následujících fragmentech, a ověříme, zda tomu tak opravdu je.

V paketu, který jsme zařadili na začátek, je *Total Length* = 860, *Header Length* = 20, tedy payload zabírá $860 - 20 = 840$ oktetů. Z toho vyplývá, že offsety v jednotlivých fragmentech mají být násobky

tohoto čísla (a následně je musíme vydělit osmi), tedy

Adresa prvního oktetu	0	840	1680	2520	...
Offset po vydělení osmi	0	105	210	315	...

Naše čtyři pakety po seřazení obsahují v poli *Fragment Offset* právě ta čísla, která jsme vypočetli do druhého řádku tabulky, takže je vše v pořádku a můžeme dokončit kompletaci.



Takže shrňme postup:

- První bude ten paket, jehož offset je 0.
- Poslední bude ten paket, který má MF = 0.
- Pořadí je dáno číslem v poli pro offset.
- Offsety mají být v odstupech, které zjistíme z délkových polí prvního paketu.



Úkol

Do cílového zařízení byly doručeny čtyři pakety v tom pořadí, které je naznačeno vpravo. Délka záhlaví je 20. Seřaďte pakety a ověřte, zda se některý fragment neztratil.

Pro každý paket vypočtete adresy prvního a posledního oktetu v původní posloupnosti dat (nezapomeňte na násobení osmi), také zjistěte, kolik dat bylo celkem v původním paketu přenášeno a jak velký byl původní paket včetně záhlaví (tedy údaj z pole *Total Length*).


Poznámka: všimněte si, že tyto fragmenty vznikly podle úkolu 4 na straně 88.


Length = 516 ID = 0x6A18 MF = 1, Off = 62	
Length = 122 ID = 0x6A18 MF = 0, Off = 186	
Length = 516 ID = 0x6A18 MF = 1, Off = 124	
Length = 516 ID = 0x6A18 MF = 1, Off = 0	



6.2 Protokol IPv6

6.2.1 Adresy podle 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* (adresy uzlu v rámci jednoho prefixu). Zapisujeme je hexadecimálními číslicemi ve *skupinách* po čtyřech číslicích (tj. po dvou oktetech) oddělené dvojtečkou. Z toho vyplývá, že adresa bude mít osm skupin oddělených dvojtečkami.

 Části adresy mezi dvojtečkami budeme dále nazývat *skupiny*. Pokud jsou ve skupině všechny bity nastaveny na 0, budeme o ní hovořit jako o *nulové skupině*.

Aby adresa nebyla až tak dlouhá, je zvykem používat zkrácenou variantu, ve které je vynechána (maximálně) jedna souvislá posloupnost nulových skupin. Krátíme vždy pouze jednu jedinou posloupnost nulových skupin. Vybereme buď tu nejdelší, nebo když je ve více posloupnostech stejný počet nulových skupin, pak tu první zleva. V rámci skupiny odstraňujeme nuly zleva.



Příklad 6.6

Na příkladech si ukážeme sestrojení zkrácené formy IPv6 adresy.

- Původní adresa: 2001:0db8:3c4d:0000:0000:0000:a011:0000
Po zkrácení: 2001:db8:3c4d::a011:0
(vybrali jsme posloupnost tří nulových skupin uprostřed adresy, odstranili jsme ji a na místě nechali dvojici dvojteček)
- Původní adresa: 2001:0db8:3c00:0000:a011:0000:0000:0000
Po zkrácení: 2001:db8:3c00:0:a011::
(vybrali jsme posloupnost tří nulových skupin na konci)
- Původní adresa: 2001:0db8:3c4d:0000:0000:a011:0000:0000
Po zkrácení: 2001:db8:3c4d::a011:0:0
(vybrali jsme posloupnost dvou nulových skupin uprostřed, protože máme dvě takové posloupnosti a tato je víc vlevo)
- Původní adresa: 2001:0db8:3c4d:0025:0000:03a0:a011:0000
Po zkrácení: 2001:db8:3c4d:25:0:3a0:a011:0
(žádnou nulovou skupinu nemažeme, protože obě takové posloupnosti obsahují jen jednu nulovou skupinu, těch si nevšímáme)

Protože víme, že celkový počet skupin je osm, je jasné, jakým způsobem se adresa rekonstruuje do původního formátu. Například v prvním případě je v upravené adrese pět skupin, tedy víme, že chybí tři (nulové) skupiny, a taky víme, kam je vložit – na místo dvojité dvojtečky.

Uvědomte si, že opravdu musíme krátit vždy jen jednu posloupnost nulových skupin, aby byla adresa jednoznačná. Kdybychom například v třetím případě zkrátili obě nulové posloupnosti, dostali bychom adresu 2001:db8:3c4d::a011::, podle které bychom sice poznali, že „někam“ je třeba doplnit celkem čtyři nulové skupiny, ale nevěděli bychom, kolik na které místo.



**Definice 6.1 (Kanonický tvar IPv6 adresy)**

Kanonický tvar IPv6 adresy předepisuje tyto podmínky:

- hexadecimální číslice se mají zapisovat malými písmeny,
- vynechávání počátečních nul ve skupině je povinné,
- 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.



Takže v předchozím příkladu jsme IPv6 adresy převáděli do kanonického tvaru. Postup:

- máme adresu s 8 skupinami, v každé skupině odstraníme nuly zleva,
- najdeme posloupnost nulových skupin, která je buď nejdelší nebo z nejdelších nejvíc vlevo,
- pokud je v posloupnosti víc než jedna nulová skupina, pak tyto nulové skupiny odstraníme a na místě krácení necháme dvojitou dvojtečku.

**Úkol**

Následující IPv6 adresy převed'te do kanonického tvaru.

- ff02:0000:0000:0000:0000:0001:0002
- 2001:4f30:0054:0000:0000:0000:01b6:0001
- 2001:4f30:0000:0000:0000:01b6:0000:0001
- fe92:056a:0000:0000:01b6:c200:0000:0000
- fe92:0000:056a:0000:01b6:c200:0022:0000



Jak bylo napsáno o několik stránek výše, pokud je uživateli přidělena (statická) adresa, potřebuje samotnou adresu, masku nebo délku prefixu a adresu brány. Kam toto všechno zadáme? Vlastně jsme se tam už dívali – v kapitole 2.4.2 na straně 22 je to ukázáno na příkladu.

6.2.2 IPv6 pakety

IPv6 paket má vždy jedno *povinné záhlaví* s nejdůležitějšími informacemi (nejvíc místa zabírají adresy zdroje a cíle, každá je dlouhá 128 oktětů), a dále následuje buď přímo SDU od jiného protokolu nebo některé *volitelné záhlaví*.

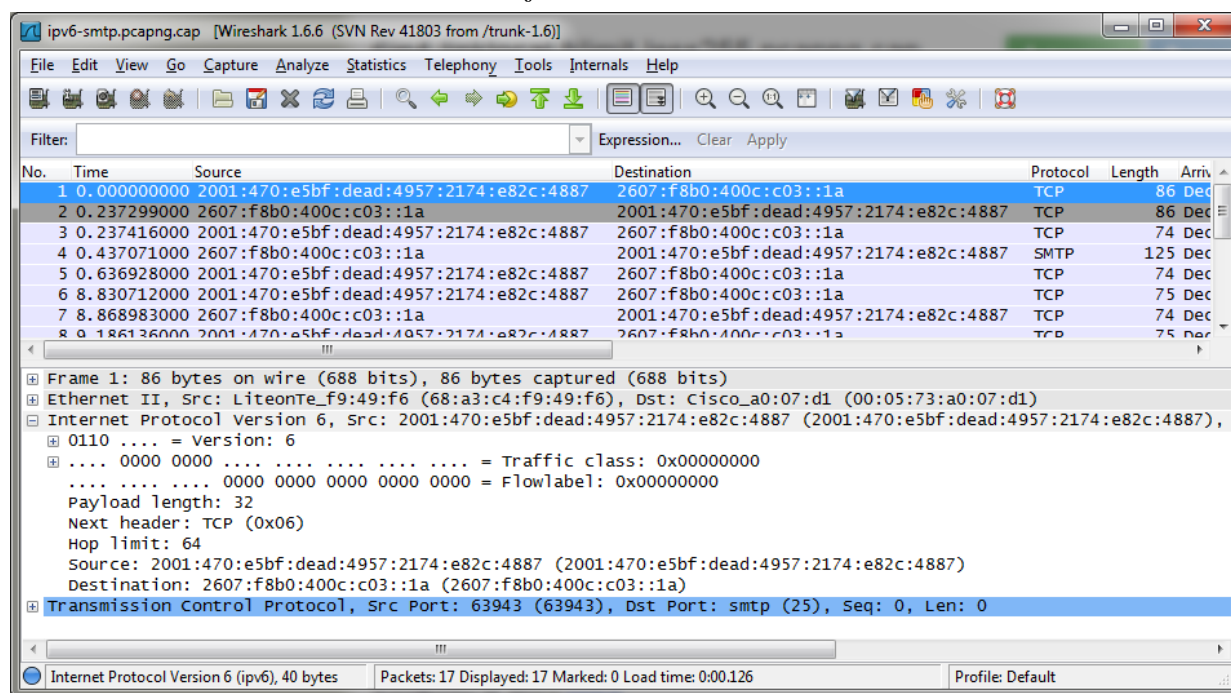
Následuje struktura IPv6 paketu (řádky jsou zarovnané na 64 bitů):

verze (4)	priorita (8)	označení datového toku (20)	délka dat (16)	další záhlaví (8)	hop limit (8)
<div> <div>IP adresa zdroje (128)</div> <div>IP adresa cíle (128)</div> </div>					



Příklad 6.7

Podíváme se na odpovídající paket ve Wiresharku. Na následujícím obrázku jsou zobrazeny podrobnosti o IPv6 paketu (prvním v seznamu). Je zřejmé, že paket byl zapouzdřen v rámci Ethernet II. Zaměříme se na informace související s vrstvou L3.



Větev pro vrstvu L3 je rozbalená, a hned v prvním prvku tohoto podstromu vidíme, že jde o protokol IP verze 6 (protože v prvním poli *Version* je číslo 6). Další pole:

- Pole *Traffic Class* a *Flow Label* mají všechny bity nastaveny na 0, což je vcelku běžné. Znamená to, že odesílatel nechce nijak ovlivňovat cestu paketu sítě.
- Pole *Payload Length* (délka dat) obsahuje číslo 32, je to jen krátký paket s žádostí mířící na server.
- *Next Header* určuje, že za povinným záhlavím najdeme TCP segment (hodnota 6).
- *Hop Limit* (obdoba TTL v IPv4) je 64, což zřejmě bude výchozí hodnota pro odesílající zařízení (paket prohlížíme na tom zařízení, ze kterého je odeslán, takže ještě z této hodnoty nebylo nic odečteno).
- Následuje zdrojová a cílová adresa.



Úkol

Na stránce <http://packetlife.net/captures/protocol/udp/> najděte soubor DHCPv6.cap a prohlédněte si jeho obsah, resp. především první paket. Všimněte si, že

- V poli *Hop Limit* je hodnota 1 (a byla tam při odeslání, protože jde o odchozí paket). To znamená, že paket půjde nejdál k nejbližšímu routeru, za něj už ne (router na něj případně reaguje, ale už dál ho nepřešle). Tato hodnota se dává do paketů, které mají zůstat v místní síti.

- Pole *Next Header* obsahuje informaci o tom, že za povinným záhlavím následuje volitelné záhlaví typu 0 (Hop-by-hop Option). Až u volitelného záhlaví máme v poli *Next Header* informaci o tom, že uvnitř je zapouzdřena datová jednotka protokolu ICMP.

Volitelné záhlaví typu 0 čte každý router na cestě, tedy vždy obsahuje sdělení pro všechny routery, přes které komunikace půjde. Obvykle jde o upozornění, že je třeba změnit určité nastavení.

V tomto případě je routerům sdělena informace, která je sice i ve vnořeném ICMP paketu, ale kdyby nebyla ve volitelném záhlaví, routery by ji ignorovaly (protože standardně se dovnitř nedívají). Jde o informaci, že adresa daného zařízení má být vyjmuta ze skupiny se zadanou multicast adresou. Multicast komunikace totiž funguje tak, že při přihlašování do skupiny se zařízení „registruje“ do skupiny na routerech, přes které mají být doručovány multicast pakety, aby bylo jasné, že tyto pakety mají být zasílány i dotyčnému zařízení, a při odhlašování se zase na routerech odhlašuje.



6.3 Protokol ICMP a příkazy, které ho používají

6.3.1 Testování dosažitelnosti

Protokol ICMP používáme, pokud se chceme ujistit, že konkrétní zařízení (typicky server) je dosažitelné. Jak ve Windows, tak i v unixových systémech pro tento účel existuje příkaz `ping`, používá se prakticky stejně, jen jsou drobné odlišnosti v parametrech. Kromě tohoto účelu jednoduše použijeme tento příkaz, když chceme otestovat svou vlastní síťovou konektivitu.

Tazatel odesílá ICMP zprávy *Echo Request*, dotazovaný odpovídá ICMP zprávami *Echo Reply*.



Příklad 6.8 (Příkaz `ping`)

Ve Windows příkaz `ping` standardně pošle **čtyři** ICMP zprávy *Echo Request*, pokud nestanovíme jinak. Na příkazovém řádku zadáme

<code>ping www.google.com</code>	chceme zjistit, zda je (pro nás) dostupný server s danou doménovou adresou
<code>ping 193.84.206.76</code>	chceme zjistit, zda je (pro nás) dostupný server s danou IP adresou
<code>ping -n 2 www.google.com</code>	totéž co v prvním případě, ale pošleme jen dva ICMP pakety, ne čtyři
<code>ping -t www.google.com</code>	tak dlouho se odesílají echo žádosti, dokud nestiskneme Ctrl+C
<code>ping 127.0.0.1</code>	otestujeme své vlastní síťové rozhraní
<code>ping loopback</code>	totéž

V Linuxu to funguje naprosto stejně, s tím rozdílem, že výchozí chování je jiné – posílají se žádosti až do chvíle, kdy stiskneme Ctrl+C. Navíc můžeme použít ještě jeden parametr:

<code>ping -c 1 -R www.google.com</code>	odešle se jen jedna žádost, ale díky přepínači <code>-R</code> se zobrazí i cesta k cíli
------------------------------------------	------------------------------------------------------------------------------------------



Ve Windows i v Linuxu máme ještě další možnosti, například se dá vynutit používání protokolu IPv4 nebo IPv6 nebo určit hodnotu TTL.

Ve výpisu nás zajímá, jestli na všechny testovací pakety došla odpověď, ale také jak dlouho trvalo, než odpověď došla (získáme hrubou představu o propustnosti cesty), případně zda tato doba nějak výrazněji kolísá (to taky není v pořádku).



Úkol

Vyzkoušejte ve svém systému různé varianty příkazu `ping` – především odeslání konkrétního počtu dotazů nebo dotazování „do nekonečna“. Otestujte také svou vlastní IP adresu a následně loopback adresu, srovnajte obě hodnoty. Náповědu k příkazu získáte následovně:

- ve Windows: `ping -?` nebo `ping /?`
- v Linuxu: `man ping` nebo `ping -?`



Poznámka:

Některé servery (nebo firewally na cestě k nim) jsou nakonfigurovány tím způsobem, že na ICMP Echo Request nereagují vůbec, případně sice ano, ale upřednostňují jiné druhy paketů – v takovém případě získáme podezřele vysoká nebo proměnlivá čísla u čekání na odpověď. Z toho vyplývá, že podle výstupu příkazu `ping` nelze v takových případech určit problém.



Příkaz `ping` používáme i tehdy, když provádíme rekonfiguraci sítě či serveru, aktualizujeme, virtualizujeme, replikujeme server apod., a potřebujeme sledovat, zda nedochází k výpadkům. Pak zvolíme variantu „s nekonečnem“ a ukončíme až ve chvíli, kdy odezvu nepotřebujeme.

Podívejme se nyní na pakety, které v komunikaci příkazu `ping` procházejí sítí.



Úkol

Spusťte Wireshark a následně v Příkazovém řádku nebo v `bash` zadejte příkaz `ping` na nějaký „vhodný“ server. Doporučuji si obojí nachystat předem (i včetně vypsání příkazu, pak stačí jen klepnout na Enter), aby Wireshark nezachycoval zbytečně moc paketů.

V rozhraní Wiresharku si do filtru zadejte buď `icmp` nebo `icmpv6` (podle toho, na jakou adresu jste v reálu „pingali“), ať si dlouhý seznam trochu protřídíte, a podívejte se na strukturu paketu. Pokud se nezobrazuje pole pro filtr, zapněte jeho zobrazování (*View – Filter Toolbar*). Zjistěte:

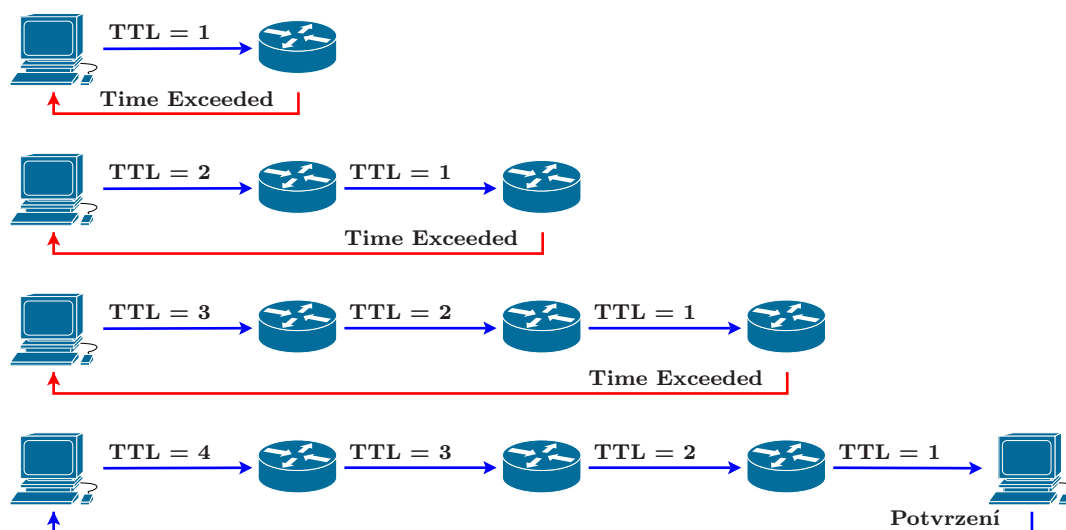
- do čeho jsou ICMP pakety zapouzdřovány,
- jaké číslo ICMP zprávy je používáno pro dotaz a odpověď,
- jaké hodnoty jsou v polích *Identifier* a *Sequence* pro dané dotazy a odpovědi,
- jaká data jsou za záhlavím zapouzdřena (podívejte se úplně dolů do podokna *Packet Bytes*, případně si zapněte jeho zobrazování).



6.3.2 Zjišťování cesty

Občas potřebujeme pro dané zařízení zjistit, přes které routery k němu vede cesta. Hodí se to například tehdy, když se na cestě objevují problémy, chceme nastavit statickou cestu či prostě chceme zjistit, zda nejsou směrovací tabulky špatně nastaveny.

Ve Windows používáme příkaz `tracert`, v Linuxu příkaz `traceroute`. Tento příkaz postupně odesílá IP pakety s TTL nastaveným na 1, 2, 3, atd., což znamená, že tyto pakety se na cestě zastaví na prvním, druhém, třetím, atd. routeru (každý router na cestě sníží hodnotu TTL o 1). Dotyčné routery, na kterých hodnota TTL vyprší (je snížena na 0), se zpět odesílateli (tedy nám) ozvou s ICMP zprávou *Time Exceeded* (čas vypršel). Tímto způsobem si zmapujeme postupně celou cestu k cíli (nebo k firewallu, který naše pokusy „zařízne“). Postup je naznačen na následujícím obrázku.



Poslední uzel na cestě (tj. cíl) nám neodešle ICMP zprávu *Time Exceeded*, ale příslušnou odpověď, která se liší podle toho, v jakém systému pracujeme a jaké PDU příkaz použil.

- 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*.
- V Linuxu to jsou ve výchozím nastavení 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 Unreachable* s kódem *Port Unreachable* (nedostupný port). V parametrech příkazu však můžeme zvolit jiný druh testovacích paketů – buď ICMP pakety jako ve Windows nebo TCP segmenty.

Takže ve Windows dostáváme od routerů na cestě ICMP zprávy *Time Exceeded* a od cíle dostaneme ICMP zprávu *Echo Reply*. V Linuxu dostáváme od routerů na cestě taky ICMP zprávy *Time Exceeded*, od cíle dostaneme ICMP zprávu *Destination Unreachable* s kódem *Port Unreachable*. Když příkaz přijme „ukončující“ typ zprávy, přestane vysílat testovací pakety.

Použití příkazů `tracert` ve Windows a `traceroute` v unixových systémech je v podstatě podobné (ostatně, je jasné, kde se Microsoft inspiruje), většinou si vystačíme bez parametrů, případně si můžeme vynutit IPv4 či IPv6. V unixových systémech máme možnost v parametrech zvolit jiný typ testovacích paketů.

**Příklad 6.9 (Příkaz tracert a traceroute)**

Pokud chceme vypsat cestu k serveru `www.seznam.cz`, napíšeme:

`tracert www.seznam.cz` ve Windows

`traceroute www.seznam.cz` v Linuxu či jiném unixovém systému

**Úkol**

Otestujte trasu k následujícím serverům/adresám:

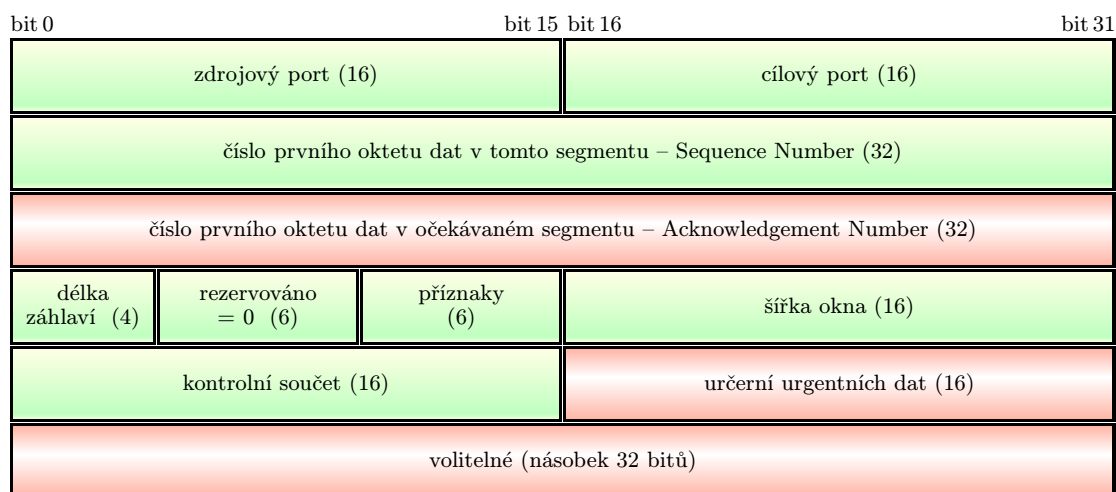
- www.slu.cz
- www.google.com
- 127.0.0.1



6.4 Protokoly na transportní vrstvě

6.4.1 Protokol TCP a navazování spojení

Protokol TCP se používá v případech, kdy je třeba mezi klientem a serverem navázat spojení a v rámci tohoto spojení poslat větší množství dat. Jeho záhlaví je poněkud složitější, protože v něm kromě polí s čísly portů potřebujeme také mechanismus, který nám zajistí kontrolu nad posloupností posílaných segmentů (pro správné seřazení a zjištění chybějících segmentů), potvrzování,...

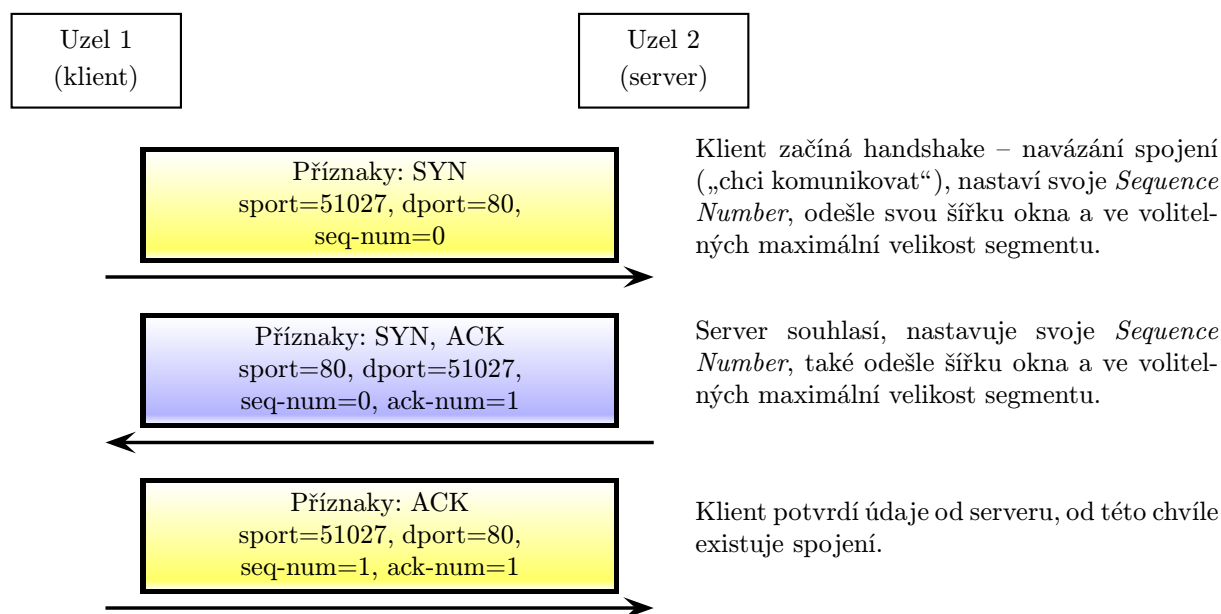
**Úkol**

Ve Wiresharku si odchyťte jakoukoliv HTTP komunikaci (například při spuštění Wiresharku si ve webovém prohlížeči otevřete nějakou webovou stránku, klidně Facebook nebo stránku univerzity), alternativně si na webu <https://wiki.wireshark.org/SampleCaptures> najdete nějakou HTTP komunikaci (například soubor `http.cap`). Projděte si PDU v této komunikaci (při vlastním odchytu především najdete její začátek). Zjistěte:

- jak vypadá záhlaví TCP segmentu, zda tedy odpovídá tomu, co se učíme (obrázek nahoře), a to zvlášť pro první TCP segment konverzace a pro ty následující,

- jak vypadají segmenty patřící do TCP Three-way-handshake (první tři segmenty), jak jsou v nich nastaveny příznaky,
- co konkrétně je uloženo v polích pro čísla portů segmentů a jak se mění v příchozí a odchozí komunikaci, jak je nastavena šířka okna a jak jsou nastaveny příznaky.

Obrázek níže naznačuje, jak vypadá TCP handshake.



6.4.2 Protokol UDP a jednoduchá rychlá komunikace

Protokol UDP používáme, když posíláme natolik malé množství dat, že se vejde do jediného segmentu, nepotřebujeme potvrzování a naopak potřebujeme, aby byl přenos rychlý a nezahlcoval síť, případně když odesíláme broadcast či multicast komunikaci. Typicky protokol UDP používají aplikační protokoly DNS, DHCP nebo SNMP.



Úkol


Odchyťte si jakoukoliv DNS komunikaci nebo na <https://wiki.wireshark.org/SampleCaptures> vyberte něco použitelného, třeba `dns.cap` nebo `dhcp.cap`.





Najděte v podokně se strukturou PDU UDP segment a srovnajte s nákresem. Podívejte se na čísla portů.



Aplikační protokoly

 *Rychlý náhled:* Zde se zaměříme na aplikační protokoly. Nejdřív se podíváme na DNS překlad a na způsob, jak se k němu dostat jako uživatelé na koncovém zařízení, a naučíme se, jak se dostat do databáze WHOIS. Následují sekce o HTTP, DHCP, Telnetu a SSH, ve kterých jde především o to jak vypadá PDU příslušného protokolu. Poslední sekce je o získávání statistických informací na koncovém zařízení pomocí příkazu `netstat`.

 *Klíčová slova:* DNS, WHOIS, HTTP, DHCP, Telnet, SSH, netstat

 *Cíle studia:* Po prostudování této kapitoly budete vědět, jak vypadají PDU běžných aplikačních protokolů, a také budete umět získat statistické informace pro síťové rozhraní na koncovém zařízení; například dokážete zjistit, které procesy právě komunikují se sítí nebo naslouchají na některém portu.

7.1 Mechanismus DNS

DNS (Domain Name System) je protokol, který zajišťuje (kromě jiného) překlad doménových adres na IP adresy. Ve všech operačních systémech máme k dispozici nejen DNS resolver, který používají aplikace (například webový prohlížeč), ale také program, ve kterém můžeme provádět tento překlad „ručně“ – je to příkaz `nslookup` (ve Windows i v unixových systémech), v unixových systémech máme navíc příkaz `dig`, který nám nabízí pokročilé možnosti (nejen) testování DNS serverů.



Příklad 7.1

Když chceme zjistit IP adresu k některé jmenné adrese (například `www.google.com`), jednoduše napíšeme:

```
nslookup www.google.com
```

Pro opačný překlad (známe IP adresu a chceme jmennou adresu) se to dělá stejně, jen jako parametr bude IP adresa.





Úkol

Vyzkoušejte si použití příkazu `nslookup` na některém serveru.



DNS pakety se většinou zapouzdřují do UDP segmentů. Podíváme se na jejich strukturu.



Úkol

Spusťte si Wireshark a zároveň proveďte to, co je v předchozím příkladu (tím si vynutíte odeslání DNS paketu). Prohlédněte si komunikaci s DNS serverem. Pokud žádná nenastala, může to být tím, že váš DNS resolver už má zadaný server ve své cache paměti, tedy proveďte vše znovu, ale zvolte jiný server (takový, se kterým jste v uplynulých minutách nekomunikovali). Pokud se přesto nedaří, najděte si soubor s komunikací na Internetu.

Prohlédněte si DNS dotaz a DNS odpověď (pokud je tam příliš mnoho řádků, použijte filtr). Všimněte si čísel portů v UDP záhlaví (pro dotaz a odpověď), a dále v DNS packagech:

- jak je formulován dotaz (tedy na co všechno se ptáme),
- co všechno je v odpovědi,
- jak je v odpovědi nastavena hodnota TTL (tedy jak dlouho bude mít náš resolver tento záznam v cache paměti).



Poznámka:

Pokud si takto chcete „pohrát“ s určitou adresou a vadí vám, že ji má DNS resolver ve své cache paměti (tedy se neposílají DNS pakety), můžete vyprázdnit jeho cache paměť. To například ve Windows provedete příkazem

```
ipconfig /flushdns
```



WHOIS databáze obsahují informace o doménách. Přístup k nim se nám může hodit v případě, že z některé domény se k nám dostává škodlivý software (malware) či směřují útoky, nebo chceme danou doménu koupit a potřebujeme ověřit, zda má majitele.



Úkol

Na adrese <http://www.whois.com/whois/> nebo <http://www.whois-search.com/> zjistíte podrobnosti o některé vybrané doméně (například `slu.cz`) – například kdy byla registrována, dokdy registrace platí (je pravidelně prodlužována), sídlo společnosti, kontaktní osobu, DNS server (nserver) apod.

Srovnejte s výstupem na adrese <http://whois.domaintools.com>, také co se týče způsobu, jakým se k výsledku dostanete.



7.2 Protokol HTTP

Zprávy protokolu HTTP jsou celkem jednoduché – klient pošle HTTP dotaz (většinou typu GET nebo POST) a server odpoví HTTP odpovědí (většinou typu OK s daty).



Úkol

Ve Wiresharku buď zachyťte vlastní HTTP komunikaci nebo si najdete soubor s komunikací na Internetu. Všimněte si, že:

- protokol HTTP používá spojitou komunikaci, zapouzdřuje se do protokolu TCP, a tedy nejdřív proběhne TCP handshake,
- na straně serveru se používá port 80, na straně klienta některý dynamický port, obojí je pro celou komunikaci zachováno,
- jak je formulován HTTP dotaz (je velmi krátký, najdeme v něm především adresu žádané stránky či objektu),
- co všechno je v dotazu na nás prozrazeno (operační systém, webový prohlížeč apod.),
- jak vypadá odpověď.



Poznámka:

Pokud se jedná o zabezpečenou komunikaci, tedy přes HTTPS (tj. HTTP zapouzdřené v SSL nebo TLS), pak toho ve Wiresharku moc neuvidíme. V TCP segmentu bude SSL/TLS paket s velmi krátkým SSL záhlavím (typ zapouzdřeného protokolu, MIME typ, verze SSL/TLS, délka zapouzdřených dat), samotná data jsou šifrovaná.



7.3 DHCP a získání IP adresy

Protokol DHCP umožňuje získat IP adresu dynamicky. DHCPv4 nemá vlastní PDU, využívá PDU protokolu Bootstrap, naproti tomu DHCPv6 využívá ICMPv6 zprávy.



Úkol

Na webu <http://packetlife.net/captures/protocol/bootp/> najdete soubor `dhcp.cap` (zřejmě bude až dole) a prohlédněte si v něm komunikaci podle DHCPv4. V tomto souboru je poněkud rozsáhlejší komunikace – první čtyři zprávy doopravdy představují mechanismus dynamického získání IP adresy, další se již vztahují k žádostem o znovupřidělení adresy (tj. žadatel má přidělenou IP adresu, tedy komunikace je unicastová). Tento soubor vznikl z testovacích důvodů, *Lease Time* (tedy doba platnosti přidělení adresy) je pouhou jednu minutu, proto tak brzká žádost o znovupřidělení. Všimněte si, že

- v prvních čtyřech paketech (žádost o novou adresu) žadatel odesílá z adresy 0.0.0.0, kdežto server odesílá ze své IP adresy, cílová adresa je vždy broadcastová,
- DHCP pakety (Bootp) jsou zapouzdřeny v UDP segmentech a používají se porty 67 a 68,

- v prvním paketu je jediná konkrétní adresa – MAC adresa žadatele (žádnou jinou MAC ani IP adresu ještě nezná),
- když si rozbalíme větev *Option: . . . Parameter Request List*, zjistíme, o jaké údaje klient žádá, přičemž obdobné pole najdeme i v dalších DHCP paketech jdoucích ve stejném směru,
- DHCP server už v první odpovědi sděluje v podstatě všechno, co klient potřebuje, ale IP adresa ještě není definitivně přidělena.



7.4 Telnet a SSH

Protokol Telnet slouží ke vzdálenému přístupu na zařízení. Je textově orientovaný (jako většina aplikačních protokolů), což by až tak nevadilo, horší je, že se sítí přenášejí nešifrovaně jak data, tak i autentizační informace (credentials). Proto je vhodnější místo Telnetu používat protokol SSH.



Úkol

Ze stránky <http://packetlife.net/captures/protocol/telnet/> si stáhněte soubor `telnet.cap` a otevřete ho ve Wiresharku.

Telnetové zprávy se zapouzdřují do TCP segmentů, tedy nejdřív proběhne TCP handshake. Následuje dojednání parametrů telnetového terminálu, pak vzájemný pozdrav (echo) a klient je dotázán na jméno a heslo (v 16. paketu). Následuje jméno a heslo po jednotlivých písmenech (no, trochu se to protahuje). Jak vidíme, pokud by někdo chtěl odposlechnout přihlašovací údaje, nic mu v tom nebrání. Dokonce i následující komunikace je „po písmenkách“.



Úkol

Ze stránky <http://packetlife.net/captures/protocol/ssh/> si stáhněte soubor `SSHv2.cap` a otevřete ho ve Wiresharku. Začíná to podobně (TCP handshake), ale vzájemné poznávání komunikujících stran se omezuje jen na oznámení podporované verze protokolu SSH a výměnu veřejných klíčů. K přihlašovacím údajům a posílaným datům se už tak lehce jako u Telnetu nedostaneme, vše je šifrováno.



7.5 Statistiky síťových protokolů

Příkaz `netstat` vypisuje statistiky týkající se síťových protokolů, například můžeme zjistit, kdo naslouchá na konkrétním TCP portu či právě komunikuje s konkrétní adresou. Ke každému záznamu zjistí sockety, tedy komunikaci IP adresy a čísla portu pro každou stranu.

Tento příkaz pochází z unixových systémů, ale byl „portován“ také do Windows. Převod nebyl proveden úplně ideálně, například některé přepínače mají ve Windows jiný význam než v unixových systémech.



Příklad 7.2 (Program `netstat` ve Windows)

Ve Windows jsou nejpoužívanější tyto kombinace přepínačů:

- `netstat -an` vypíše plnou statistiku pro všechny protokoly,
- `netstat -ano` přidá sloupeček s PID procesů, které zrovna takto komunikují,
- `netstat -anb` místo PID budeme mít názvy procesů,
- `netstat -sbp tcp` vypíše všechna TCP spojení včetně názvů procesů (přepínač „p“ by měl být za pomlčkou jako poslední),
- `netstat -sp ip` vypíše statistiku protokolu IP,
- `netstat -sp icmp` vypíše statistiku pro protokol ICMP (obecně: přepínač „p“ následovaný názvem protokolu znamená, že bude vypsáno všechno, co se týká zadaného protokolu),
- `netstat -es` vypíše plnou statistiku pro síť Ethernet (v reálu se vypíše statistika pro všechno, co se zapouzdřuje do ethernetových či jiných rámců, Microsoft to prostě nazval jinak než měl).



Příklad 7.3 (Program `netstat` v Linuxu)

V Linuxu se hodně používají následující kombinace:

- `netstat -an` vypíše plnou statistiku pro všechny protokoly,
- `netstat -anp` totéž co v předchozím případě, navíc je sloupeček s PID a názvy komunikujících procesů (podobně jako ve Windows `netstat -ano`),
- `netstat -ant` seznam TCP spojení,
- `netstat -anu` seznam UDP spojení (spíše naslouchání),
- `netstat -antp` totéž, ale navíc přidáme PID a názvy komunikujících procesů,
- `netstat -tnpl` podobně, ale zobrazí se pouze naslouchání na portech (listening),
- `netstat -l` seznam portů, na kterých někdo naslouchá,
- `netstat -s` zobrazí statistiku pro jednotlivé protokoly,
- `netstat -antp | grep ESTA` zobrazí všechna aktivní připojení.



Ve Windows máme pro `netstat` deset přepínačů, v Linuxu je jich téměř dvakrát tolik. Většina administrátorů má své „oblíbené kombinace“, ostatní si v případě potřeby najdou v nápovědě.

Internetworking a průzkum paketů

Rychlý náhled: Vracíme se na vrstvu L3. Nejdřív se podíváme na překlad mezi L3 a L2 adresami a zjištění sousedů v síti, dále si procvičíme podsítování (subnetting) a VLSM. Poslední krátká sekce uvádí způsob, jak si nechat vypsat směrovací tabulku na koncovém zařízení.

Klíčová slova: IPv4, IPv6, MAC, ARP, neighbor, podsítování, subnetting, VLSM, směrování

Cíle studia: Po prostudování této kapitoly budete umět zjistit IP adresy sousedů v síti, zvládnete rozdělení sítě na podsítě a budete umět vypsat směrovací tabulku na koncovém zařízení.

8.1 Objevování sousedů

Proč nás zajímají sousedé? Od DHCP serveru zjistíme základní údaje pro fungování v síti – vlastní adresu, masku nebo délku prefixu, adresu brány, adresy DNS serverů, ... Jenže když chceme odeslat paket, s IP adresami si nevystačíme, protože na vrstvě L2 musíme paket zapouzdřit do rámce s MAC adresami. Svou známe, ale co ta od adresáta?

- Pokud posíláme paket do místní sítě, musíme znát MAC adresu cíle.
- Pokud posíláme paket mimo místní síť, potřebujeme MAC adresu brány.

Takže potřebujeme protokol, který bude zjišťovat MAC adresy sousedů a pak si je evidovat v tabulce sousedů, aby to zjišťování nemusel provádět moc často. V tabulce sousedů máme (kromě jiného) tyto údaje:

- IP adresu souseda,
- jeho MAC adresu,
- informaci, zda byl tento údaj staticky vložen nebo dynamicky získán pomocí protokolu.

IPv4 používá protokol ARP, IPv6 používá protokol NDP. Oba protokoly pracují takto:

- Je vyslán ARP/NDP dotaz „Kdo má IP adresu xxx? Sdělte prosím na adresu yyy.“ (Who has xxx? Tell yyy). Obě tyto adresy jsou IP, žádáme, aby se majitel adresy xxx ozval se svou MAC adresou.
- Následuje odpověď „Adresa yyy je na MAC adrese zzz.“ (yyy is at zzz).

Dotaz i odpověď jsou stejně formulovány, v odpovědi jsou „vyplněna“ prázdná pole.



Příklad 8.1

Také v operačním systému si můžeme vypsát tabulku sousedů. Jak se to dělá:

- ve Windows napíšeme:
 - `arp -a` pro ARP tabulku v IPv4,
 - `netsh interface ipv6 show neighbors` pro NDP tabulku v IPv6,
 - V Powershellu (spustíme ho zadáním „powershell“ přes Start, klepneme na PowerShell) můžeme použít příkaz `get-NetNeighbor`, který umí zobrazit IPv4 i IPv6 sousedy,
- v Linuxu napíšeme
 - `ip neighbor show` (případně můžeme krátit řetězcem parametrů, například `ip n sh`).



Úkoly

1. Na <http://packetlife.net/captures/protocol/arp/> najdete soubor `gmail.pcapng.cap`. Otevřete ho ve Wiresharku a vyfiltrujte si pouze protokol `arp`. Podívejte se na formát dotazu a odpovědi. Jedná se o ARP a IPv4, takže ARP paket je zapouzdřen přímo v rámci vrstvy L2.
2. Vyzkoušejte některé z příkazů pro výpis tabulky sousedů pro IPv4 i IPv6, podle vašeho operačního systému.



8.2 Práce s IPv4 adresami

8.2.1 Subnetting

Podsítování (subnetting) spočívá v tom, že jednu síť rozdělíme na několik dílů – podsítí (subnetů), přičemž příslušnost k určité podsíti se dá poznat podle konkrétních bitů adresy. Jde o to, že z hostitelské části adresy si „vypůjčíme“ pár bitů a použijeme je pro odlišení jednotlivých podsítí. Tím zmenšíme počet bitů využitelných pro hostitele (proto v rámci podsítě existuje méně adres než v původní síti).

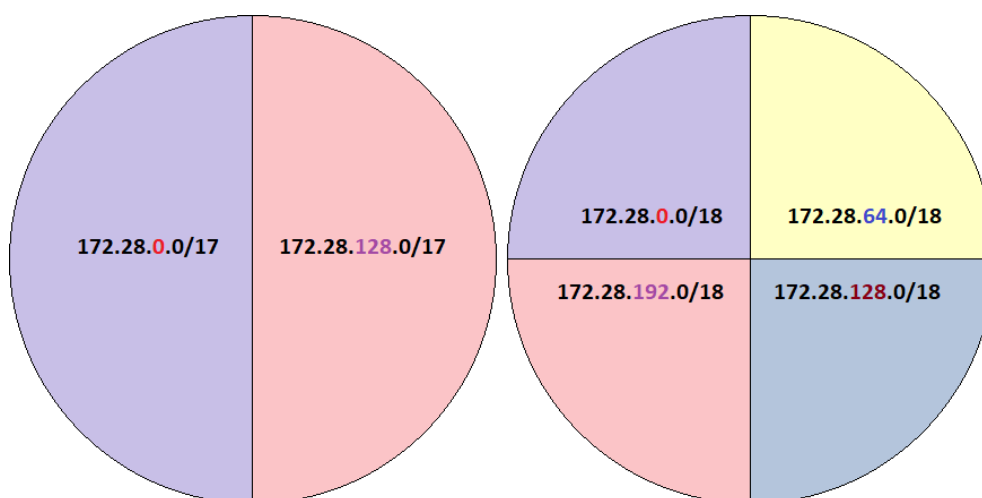
 Takže adresa se při podsítování skládá ze tří částí:

- síťová část adresy,
- podsíťová část adresy,
- hostitelská část adresy.

Součet bitů všech tří částí samozřejmě musí dávat 32.

Původní:	síť	hostitelská část	
Po podsítování:	síť	podsíť	hostitelská část

Obrázek 8.1: Struktura IP adresy při použití podsítování



Obrázek 8.2: Rozdělení sítě 172.28.0.0/16 na dvě a čtyři podsítě

Na obrázku 8.2 je graficky znázorněno rozdělení sítě 172.28.0.0/16 na dvě (vlevo) a čtyři (vpravo) podsítě. Tyto podsítě musí být disjunktní (nesmí existovat adresa, která by patřila do obou) a zároveň chceme, aby co nejlépe „vytěžovaly“ adresový prostor (žádná adresa nesmí zůstat „mimo“).



Příklad 8.2

Vezměme síť 172.28.0.0/16, o které byla zmínka dříve a platí pro ni obrázek 8.2 Pro původní síť platí:

- adresa sítě je 172.28.0.0/16,
- první použitelná adresa je 172.28.0.1,
- poslední použitelná adresa je 172.28.255.254,
- broadcast pro tuto podsít' je 172.28.255.255.

Při rozdělení na dvě podsítě pro první z nich platí:

- adresa podsítě je 172.28.0.0/17,
- první použitelná adresa je 172.28.0.1,
- poslední použitelná adresa je 172.28.127.254,
- broadcast pro tuto podsít' je 172.28.127.255.

Pro druhou podsít' platí:

- adresa podsítě je 172.28.128.0/17,
- první použitelná adresa je 172.28.128.1,
- poslední použitelná adresa je 172.28.255.254,
- broadcast pro tuto podsít' je 172.28.255.255.

Všimněte si, že první dva parametry se shodují u původní sítě a první podsítě (až na délku prefixu), druhé dva parametry se shodují u původní sítě a druhé podsítě. První a druhá podsít' na sebe navazují (broadcast první podsítě se liší o 1 od adresy následující podsítě).



Jak to vypadá s počtem adres? Absolutní počet adres samozřejmě zůstává, ale musíme brát na vědomí, že krajní dvě adresy z rozsahu (tj. adresa sítě a broadcastová adresa) se nedají použít pro hostitele. Pokud v síti vytvoříme dvě podsítě, pak právě z tohoto důvodu o dvě adresy přijdeme; pokud vytvoříme čtyři podsítě, přijdeme o šest adres, atd.



Příklad 8.3

Máme adresu sítě třídy C ve tvaru 192.168.48.0 (tj. tři oktety jsou adresou sítě) a chceme tuto síť rozdělit na podsítě tak, že pro podsíťovou část vyhradíme dva bity (tj. maximálně $2^2 = 4$ podsítě). Je to adresa třídy C, tedy adresu sítě tvoří tři oktety (24 bitů), a pokud z druhé části adresy máme přidat dva bity pro podsítě, bude adresa podsítě dlouhá 26 bitů (to je délka prefixu).

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.

Všimněte si, že adresa podsítě je o 1 vyšší než broadcastová adresa předchozí podsítě. Masku podsítě bude 255.255.255.192, protože poslední oktet masky binárně je 11000000 (tj. první dva bity ještě patří k podsíti), což je desítkově 192.



**Příklad 8.4**

Postup ukázaný v předchozím příkladu 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 hostitelské části adresy jsou 0),
3. broadcastová adresa v podsíti (tj. všechny bity v hostitelské 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

Tabulka 8.1: Vytvoření čtyř podsítí pro síť 192.168.48.0/24

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, v podstatě bychom ji nemuseli všude psát. Binární forma posledního oktetu, která je připsaná v druhém a třetím sloupci, tam taky nemusí být, pokud to umíme z paměti.

Jak vidíte, sloupce pro adresu podsítě a broadcast v podsíti není těžké vyplnit, když zvládneme 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.



Při použití uvedeného postupu jsou všechny podsítě stejně velké, což usnadňuje určování adres podsítí. Ve výše uvedeném příkladu jsou čísla pro čtvrtý oktet v aritmetické posloupnosti: 0, 64, 128, 192, a nedosažené maximum 256.

**Příklad 8.5**

Vyzkoušejme vytvoření větších podsítí. Základní adresa je 10.0.0.0/8, potřebujeme minimálně 10 podsítí. Ovšem reálný počet podsítí je vždy mocnina čísla dvě (protože tolik možností nám dává určitý počet bitů), tedy si najdeme nejbližší vyšší (nebo rovnou) hodnotu, což je $16 = 2^4$. To znamená, že pro podsíťování využijeme čtyři bity, jejichž hodnoty budou 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111. Celkem 16 možností.

Takže síťová část zabírá osm bitů, podsítě čtyři bity, celkem dvanáct bitů. Zbývá 20 bitů pro hostitelskou část. Nebudeme si vypisovat všechny podsítě, podíváme se na několik prvních a pár posledních:

LAN1: adresa podsítě: 10.0.0.0/12 • první hostitel: 10.0.0.1 • poslední hostitel: 10.15.255.254 • broadcast: 10.15.255.255	LAN2: adresa podsítě: 10.16.0.0/12 • první hostitel: 10.16.0.1 • poslední hostitel: 10.31.255.254 • broadcast: 10.31.255.255
LAN3: adresa podsítě: 10.32.0.0/12 • první hostitel: 10.32.0.1 • poslední hostitel: 10.47.255.254 • broadcast: 10.47.255.255	LAN4: adresa podsítě: 10.48.0.0/12 • první hostitel: 10.48.0.1 • poslední hostitel: 10.63.255.254 • broadcast: 10.63.255.255
LAN5: adresa podsítě: 10.64.0.0/12 • první hostitel: 10.64.0.1 • poslední hostitel: 10.79.255.254 • broadcast: 10.79.255.255	LAN6: adresa podsítě: 10.80.0.0/12 • první hostitel: 10.80.0.1 • poslední hostitel: 10.95.255.254 • broadcast: 10.95.255.255
LAN7: adresa podsítě: 10.96.0.0/12 • první hostitel: 10.96.0.1 • poslední hostitel: 10.111.255.254 • broadcast: 10.111.255.255	LAN8: adresa podsítě: 10.112.0.0/12 • první hostitel: 10.112.0.1 • poslední hostitel: 10.127.255.254 • broadcast: 10.127.255.255
. . .	
LAN15: adresa podsítě: 10.224.0.0/12 • první hostitel: 10.224.0.1 • poslední hostitel: 10.239.255.254 • broadcast: 10.239.255.255	LAN16: adresa podsítě: 10.240.0.0/12 • první hostitel: 10.240.0.1 • poslední hostitel: 10.255.255.254 • broadcast: 10.255.255.255

Tabulka 8.2: Rozdělení sítě 10.0.0.0/8 na šestnáct podsítí

V druhém oktetu adresy máme „postupku“ (aritmetickou posloupnost) 0, 16, 32, 48, 64, 80, 96, 112, 128, 144, 160, 176, 192, 208, 224, 240 (a nepoužité 256). Adresní prostory podsítí nám pokrývají celý adresní prostor původní sítě, navazují na sebe, přičemž některé adresy jsou nepoužitelné pro hostitelská zařízení. V každé podsíti musíme mít dvě vyhrazené adresy (adresa podsítě a broadcast), přičemž v původní síti to byly celkem 2 adresy, kdežto po podsítování to je $2 * 16 = 32$ adres, tedy podsítováním jsme přišli o $32 - 2 = 30$ adres.

Prostorově úspornější zápis pomocí tabulky je níže v tabulce 8.3. Nenajdete tam všech 16 podsítí, jen čtyři první a dvě poslední.



Příklad 8.6

Zkusme to z druhé strany. V předchozím příkladu jsme měli určeno, kolik musí být podsítí, teď budeme mít určeno, kolik hostitelských zařízení se do podsítí musí vejít.

Máme k dispozici adresový prostor sítě 172.16.0.0/16 (tj. hranice mezi síťovou a hostitelskou částí je přesně v polovině, $16 + 16 = 32$) a potřebujeme vytvořit podsítě takové, že do každé se musí

Název	Adresa podsítě	Rozsah pro klienty	Broadcast podsítě
LAN1	10.0.0.0/12 druhý oktet: .00000000	od 10.0.0.1 do 10.15.255.254	10.15.255.255 druhý oktet: .00001111
LAN2	10.16.0.0/12 druhý oktet: .00010000	od 10.16.0.1 do 10.15.255.254	10.31.255.255 druhý oktet: .00011111
LAN3	10.32.0.0/12 druhý oktet: .00100000	od 10.32.0.1 do 10.47.255.254	10.47.255.255 druhý oktet: .00101111
LAN4	10.48.0.0/12 druhý oktet: .00110000	od 10.48.0.1 do 10.63.255.254	10.63.255.255 druhý oktet: .00111111
• • •			
LAN15	10.224.0.0/12 druhý oktet: .11100000	od 10.224.0.1 do 10.239.255.254	10.239.255.255 druhý oktet: .11101111
LAN16	10.240.0.0/12 druhý oktet: .11110000	od 10.240.0.1 do 10.255.255.254	10.255.255.255 druhý oktet: .11111111

Tabulka 8.3: Vytvoření šestnácti podsítí pro síť 10.0.0.0/8

vejít nejméně 511 hostitelských zařízení (počítačů, serverů, routerů, atd., jakékoliv přidělitelné adresy).

Tady pozor – potřebujeme 511 přidělitelných adres, ale víme, že v daném rozsahu jsou vždy dvě adresy nepoužitelné. Tedy potřebujeme adresový rozsah s nejméně $511 + 2 = 513$ adresami. Pokud bychom pro podsítě použili 9 bitů, získali bychom $2^9 = 512$ adres, tj. pro 510 zařízení, což je málo. Musíme tedy použít 10 bitů, čímž dostaneme $2^{10} = 1024$ možných adres, tj. pro $1024 - 2 = 1022$ zařízení.

Takže pro hostitelskou část necháme 10 bitů. Původně to bylo 16, takže 6 bitů zbývá pro podsítě. Délka prefixu je $16 + 6 = 22$ (síťová plus podsíťová část). Jaký bude počet podsítí? Můžeme vytvořit celkem $2^6 = 64$ podsítí.

U velkých podsítí (navíc s větším počtem těchto podsítí) je dobré si nějak zjednodušit práci. Mohli bychom sice adresy podsítí tvořit tak, že bychom vždy k předchozí podsíti přičítali počet adres podsítě, ale pak bychom museli přičítat přes hranici mezi oktety, což je poněkud komplikované. Nebo bychom mohli počítat v binární soustavě a pak převést do desítkové, což je taky zbytečně pracné. Takže jak na to?

Délka nového prefixu pro podsítě je 22, maska pro všechny podsítě je 255.255.252.0, třetí oktet je binárně 1111 1100. Zaměříme se na poslední jedničku z masky. Při převodu do desítkové soustavy bychom ji převedli na $2^2 = 4$. Takže si shrňme, co jsme získali z masky: jsme v třetím oktetu a máme číslo 4.

Přesuňme se k adresám podsítí. Začínáme vždy s tou adresou, která patří celé síti (tj. u nás 172.16.0.0), jen délka prefixu bude 22 místo původních 16. V této adrese se (stejně jako u masky) přesuneme do třetího oktetu, a budeme postupně přičítat číslo 4. Takže pro třetí oktet postupně dostaneme čísla 0, 4, 8, ...:

172.16.0.0/22

172.16.4.0/22

172.16.8.0/22

172.16.12.0/22...

Údaje o prvních a posledních dvou podsítích máme níže v tabulce 8.4. V následující tabulce je pak zápis po řádcích.

LAN1: adresa podsítě: 172.16.0.0/22 • první hostitel: 172.16.0.1 • poslední hostitel: 172.16.3.254 • broadcast: 172.16.3.255	LAN2: adresa podsítě: 172.16.4.0/22 • první hostitel: 172.16.4.1 • poslední hostitel: 172.16.7.254 • broadcast: 172.16.7.255
• • •	
LAN63: adresa podsítě: 172.16.248.0/22 • první hostitel: 172.16.248.1 • poslední hostitel: 172.16.251.254 • broadcast: 172.16.251.255	LAN64: adresa podsítě: 172.16.252.0/22 • první hostitel: 172.16.252.1 • poslední hostitel: 172.16.255.254 • broadcast: 172.255.255.255

Tabulka 8.4: Rozdělení sítě 172.16.0.0/16 na podsítě pro min. 511 zařízení

Název	Adresa podsítě	Rozsah pro klienty	Broadcast podsítě
LAN1	172.16.0.0/22 třetí oktet: .00000000	od 172.16.0.1 do 172.16.3.254	172.16.3.255 třetí oktet: .00000011
LAN2	172.16.4.0/22 třetí oktet: .00000100	od 172.16.4.1 do 172.16.7.254	172.16.7.255 třetí oktet: .00000111
LAN3	172.16.8.0/22 třetí oktet: .00001000	od 172.16.8.1 do 172.16.11.254	172.16.11.255 třetí oktet: .00001011
LAN4	172.16.12.0/22 třetí oktet: .00001100	od 172.16.12.1 do 172.16.15.254	172.16.15.255 třetí oktet: .00001111
• • •			
LAN63	172.16.248.0/22 třetí oktet: .11111000	od 172.16.248.1 do 172.16.251.254	172.16.251.255 třetí oktet: .11111011
LAN64	172.16.252.0/22 třetí oktet: .11111100	od 172.16.252.1 do 172.16.255.254	172.16.255.255 třetí oktet: .11111111

Tabulka 8.5: Vytvoření 64 podsítí pro síť 172.16.0.0/16



Zatím jsme se pohybovali v rámci jednoho oktetu adresy (adresy podsítí se lišily v jediném oktetu). Stačilo zjistit, o kolik se budou adresy podsítí lišit, a pak jsme jen přičítali. Ale co když se bity pro podsítování rozprostírají přes hranici mezi dvěma oktety?



Příklad 8.7

Větší firma s cca 800 pobočkami po celém světě používá síť 10.0.0.0/8, přičemž každá pobočka potřebuje vlastní podsíť. V každé pobočce se předpokládá až 30 000 zařízení s IP adresou (od mobilů přes servery až po stroje ve výrobě). Naším úkolem je vytvořit podsítě tak, aby tyto požadavky byly splněny:

- $800 \text{ podsítí} - 2^9 = 512 < 800 < 1024 = 2^{10} \Rightarrow$ pro podsítovou část adresy musíme zvolit nejméně 10 bitů,
- $30\,000 \text{ zařízení} - 2^{13} = 28\,561 < 30\,000 < 57\,122 = 2^{14} \Rightarrow$ v části adresy pro hostitele potřebujeme nejméně 14 bitů.

Síťová část adresy zabírá 8 bitů, a když vše sečteme, získáme $8 + 10 + 14 = 32$. Takže nám to vychází úplně přesně, jiná možnost ani neexistuje (kdyby vyšlo menší číslo než 32, mohli bychom zvolit buď víc podsítí nebo víc klientů v podsíti).

Délka prefixu podsítí bude 18 (sečteme bity pro síťovou a podsíťovou část), třetí oktet masky je binárně **1100 0000**, maska je 255.255.192.0. Stejně jako v předchozím příkladu, zaměříme se na poslední jedničku masky. Nachází se v třetím oktetu, je to druhá binární číslice zleva a tedy po převodu do desítkové soustavy získáme $2^6 = 64$. Takže jsme v třetím oktetu a máme číslo 64 \Rightarrow v třetím oktetu adres podsítí budeme přičítat číslo 64.

10.0.0.0/18

10.0.64.0/18

10.0.128.0/18

10.0.196.0/18

Ale co teď? Vždyť $196 + 64 = 256$, ale to už je mimo povolený rozsah oktetu. Je to stejná situace, jako když například v desítkové soustavě postupně přičítáme číslo 20. Až do 80 není problém, ale když chceme přičíst 20 ještě jednou, musíme vynulovat tu číslici, u které jsme prováděli změny, a přičíst jedničku k vyššímu řádu. Totéž provedeme zde, jen místo číslic máme oktety.

...

10.1.0.0/18

10.1.64.0/18

...

10.1.196.0/18

10.2.0.0/18

...

10.255.196.0/18

LAN1: adresa podsítě: 10.0.0.0/18 <ul style="list-style-type: none"> • první hostitel: 10.0.0.1 • poslední hostitel: 10.0.63.254 • broadcast: 10.0.63.255 	LAN2: adresa podsítě: 10.0.64.0/18 <ul style="list-style-type: none"> • první hostitel: 10.0.64.1 • poslední hostitel: 10.0.127.254 • broadcast: 10.0.127.255
LAN3: adresa podsítě: 10.0.128.0/18 <ul style="list-style-type: none"> • první hostitel: 10.0.128.1 • poslední hostitel: 10.0.195.254 • broadcast: 10.0.195.255 	LAN4: adresa podsítě: 10.0.196.0/18 <ul style="list-style-type: none"> • první hostitel: 10.0.196.1 • poslední hostitel: 10.0.255.254 • broadcast: 10.0.255.255
LAN5: adresa podsítě: 10.1.0.0/18 <ul style="list-style-type: none"> • první hostitel: 10.1.0.1 • poslední hostitel: 10.1.63.254 • broadcast: 10.1.63.255 	LAN6: adresa podsítě: 10.1.64.0/18 <ul style="list-style-type: none"> • první hostitel: 10.1.64.1 • poslední hostitel: 10.1.127.254 • broadcast: 10.1.127.255
• • •	
LAN1023: adresa podsítě: 10.255.128.0/18 <ul style="list-style-type: none"> • první hostitel: 10.255.128.1 • poslední hostitel: 10.255.195.254 • broadcast: 10.255.195.255 	LAN1024: adresa podsítě: 10.255.196.0/18 <ul style="list-style-type: none"> • první hostitel: 10.255.196.1 • poslední hostitel: 10.255.255.254 • broadcast: 10.255.255.255

Tabulka 8.6: Rozdělení sítě 10.0.0.0/8 na podsítě s využitím 10 bitů

Údaje o několika prvních a dvou posledních podsítích jsou v tabulce 8.6, následuje zobrazení tabulka s podsítěmi po řádcích.

Název	Adresa podsítě	Rozsah pro klienty	Broadcast podsítě
LAN1	10.0.0.0/18 druhý a třetí oktet: .00000000.00000000	od 10.0.0.1 do 10.0.63.254	10.0.63.255 druhý a třetí oktet: .00000000.00111111
LAN2	10.0.64.0/18 druhý a třetí oktet: .00000000.01000000	od 10.0.64.1 do 10.0.127.254	10.0.127.255 druhý a třetí oktet: .00000000.01111111
LAN3	10.0.128.0/18 druhý a třetí oktet: .00000000.10000000	od 10.0.128.1 do 10.0.195.254	10.0.195.255 druhý a třetí oktet: .00000000.10111111
LAN4	10.0.196.0/18 druhý a třetí oktet: .00000000.11000000	od 10.0.196.1 do 10.0.255.254	10.0.255.255 druhý a třetí oktet: .00000000.11111111
LAN5	10.1.0.0/18 druhý a třetí oktet: .00000001.00000000	od 10.1.0.1 do 10.1.63.254	10.1.63.255 druhý a třetí oktet: .00000001.00111111
LAN6	10.1.64.0/18 druhý a třetí oktet: .00000001.01000000	od 10.1.64.1 do 10.1.127.254	10.1.127.255 druhý a třetí oktet: .00000001.01111111
• • •			
LAN1023	10.255.128.0/18 druhý a třetí oktet: .11111111.10000000	od 10.255.128.1 do 10.255.195.254	10.255.195.255 druhý a třetí oktet: .11111111.10111111
LAN1024	10.255.196.0/18 druhý a třetí oktet: .11111111.11000000	od 10.255.196.1 do 10.255.255.254	10.255.255.255 druhý a třetí oktet: .11111111.11111111

Tabulka 8.7: Vytvoření 1024 podsítí pro síť 10.0.0.0/8



Poznámka:

Proč to všechno?

- Podsítě nám zjednodušují filtrování provozu v návaznosti na další služby, také jeho monitorování, příp. napojení na mechanismus VLAN (každá VLAN musí odpovídat jedné podsíti).
- Broadcastové domény můžeme omezovat i na podsítě, takže broadcasty nám nebloudí v celé síti, což opět znamená snížení provozu v síti.



Úkoly

1. Vezměte si adresu sítě stejnou jako v prvním příkladu – 192.168.48.0. Vyzkoušejte výpočet pro případ, že pro podsíťovou část vyhradíte pouze jeden bit. Pak vyzkoušejte případ, kdy pro podsíťovou část adresy vyhradíte tři bity.

2. Síť 10.0.0.0/8 rozdělte na 32 podsítí. Napište masku pro vytvořené podsítě a vypište údaje o prvních čtyřech a posledních dvou podsítích.
3. Síť 172.20.0.0/16 rozdělte na podsítě tak, abyste dostali nejméně 50 podsítí a v každé podsíti aby bylo nejméně 1000 použitelných adres. Napište masku pro vytvořené podsítě a vypište údaje o prvních čtyřech a posledních dvou podsítích.



Úkol

Na webu existují IP kalkulačky (subnet calculators), které nám mohou zjednodušit práci. Na testu jsou nepřipustné, ale v praxi – proč ne.

Na <http://ip-kalkulacka.nmonitoring.com/> nebo <http://www.subnet-calculator.com/> si vyzkoušejte, jak se IP kalkulačka používá. Především je třeba zadat adresu sítě. Vyzkoušejte třeba 10.0.0.0/8 (to znamená soukromé adresy třídy A). Zadejte délku prefixu 10 (nebo jiné číslo větší než 8), případně ve formě masky. Zobrazte a zkontrolujte výsledek. Pak vyzkoušejte pro větší délku prefixu.

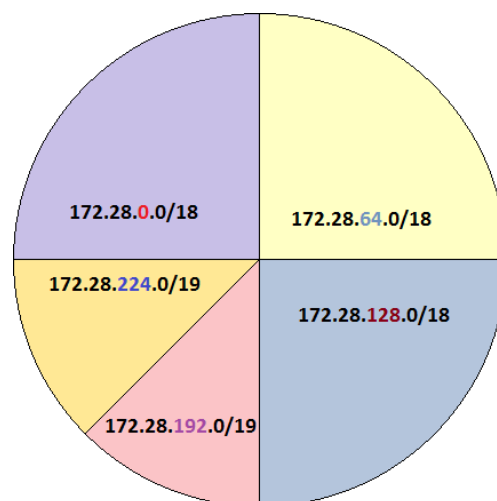


8.2.2 VLMS

Masky podsítí s proměnnou délkou – VLMS (Variable Length Subnet Mask) – jsou mechanismem rozšiřujícím možnosti podsítování. Zatímco u podsítování jsme si pevně stanovili, kolik bitů bude pro adresu podsítě, u VLMS se tento počet bitů může u různých podsítí měnit.

Při podsítování máme pro každou podsít tentýž maximální počet hostitelů, při použití VLMS 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.

Dělení zachovává binárnost. Jednu větší (pod)sít můžeme rozdělit na dvě, čtyři, osm, ... podsítí. Na obrázku 8.3 vidíme, že síť 172.28.0.0/16 byla rozdělena nejdříve na čtyři stejně velké podsítě, a pak ta poslední na dvě menší podsítě. V reálu tedy máme pět podsítí, z toho tři větší, dvě menší. Pokud bychom chtěli dvě větší a čtyři menší sítě, taky by to bylo možné (rozdělili bychom na poloviny také podsít 172.28.128.0/18). Důležitým pravidlem je, že dělíme sítě na konci rozsahu.



Obrázek 8.3: Rozdělení sítě 172.28.0.0/16 na tři větší a dvě menší podsítě



Příklad 8.8

Navážeme na příklad na straně 106. Tam jsme dělali síť 192.168.48.0/24 na čtyři stejně velké podsítě, tedy vytvořili jsme podsítě s délkou prefixu 26 a na podsítování byly použity dva bity.

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 podsít' a dvě menší. Pro první podsít' si zvolíme délku prefixu 25 (tedy pro podsít' vyhradíme jeden bit), pro druhou a třetí podsít' pak 26 (dva bity pro podsít').

- Podsít' 1: první bit posledního oktetu bude 0, tedy:
 - adresa podsítě je 192.168.48.0/25 (poslední oktet binárně 00000000),
 - broadcast adresa pro tuto podsít' je 192.168.48.127/25 (poslední oktet 01111111),
 - hostitelé mají adresy z rozsahu 192.168.48.1/25 až 192.168.48.126/25.
- Podsít' 2: první dva bity posledního oktetu budou 10, tedy:
 - adresa podsítě je 192.168.48.128/26 (poslední oktet binárně 10000000),
 - broadcast adresa pro tuto podsít' 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.
- Podsít' 3: první dva bity posledního oktetu budou 11, tedy:
 - adresa podsítě je 192.168.48.192/26 (poslední oktet binárně 11000000),
 - broadcast adresa pro tuto podsít' 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.

Srovnajte s příkladem na straně 106. Všimněte si, že v tomto příkladu nám první podsít' vlastně „spolkla“ původní první a druhou podsít' 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 ($2^6 - 2$).

Adresa podsítě	Délka pref.	Hostitelé	Broadcast	Maska
192.168.48.0 posl. oktet: .00000000	/25	od 192.168.48.1 do 192.168.48.126	192.168.48.127 posl. oktet: .01111111	255.255.255.128 posl. oktet: .10000000
192.168.48.128 posl. oktet: .10000000	/26	od 192.168.48.129 do 192.168.48.190	192.168.48.191 posl. oktet: .10111111	255.255.255.192 posl. oktet: .11000000
192.168.48.192 posl. oktet: .11000000	/26	od 192.168.48.193 do 192.168.48.254	192.168.48.255 posl. oktet: .11111111	255.255.255.192 posl. oktet: .11000000

Tabulka 8.8: Vytvoření tří různě velkých podsítí pro síť 192.168.48.0/24


V první podsíti je celkem 128 adres, z nich 126 je možné přidělit hostitelským zařízením (poslední oktet 1 až 126). V druhé a třetí podsíti je 64 adres, z nich 62 lze přidělit.



Poznámka:

V příkladu jsme postupovali tím způsobem, že jsme nejdřív určili počty bitů pro jednotlivé podsítě (z čehož vyplynulo, kolik klientských zařízení maximálně se do podsítě vejde), ale v reálné situaci bychom museli postupovat naopak – nejdřív je třeba si rozmyslet, kolik místa potřebujeme v jednotlivých podsítích (může vyplývat z rozsahu VLAN nebo jednoduše maximálního počtu klientů připojených k síťovým zařízením plus rezerva) a pak bychom rozvrhli délky prefixů pro jednotlivé podsítě.



 Ujasníme si ještě něco, co jsme ve skutečnosti už v předchozích příkladech používali, a to je vztah mezi délkou prefixu a počtem adres v podsíti.

...	/21	/22	/23	/24	/25	/26	/27	/28	/29	/30
...	2048	1024	512	256	128	64	32	16	8	4

Tabulka 8.9: Vztah mezi délkou prefixu a počtem adres v podsíti

Z tabulky 8.9 bychom si měli pamatovat alespoň sloupec pro /24, ostatní sloupce už nebude problém doplnit. Takto velmi rychle určíme, jak velká která podsít vlastně bude. Ovšem nesmíme zapomenout, že od údaje z druhého řádku tabulky ještě musíme odečíst 2, abychom získali počet přidělitelných adres.



Příklad 8.9

V praxi bude častější jiný typ zadání, například:

Adresa sítě je 172.16.0.0/20. Rozdělte 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 síť.

Každá podsít 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 podsít volíme délku prefixu tak, aby se tam klienti „vešli“, zaokrouhlujeme na číslo $2^n - 2$ směrem nahoru, délka prefixu pak bude $32 - n$. Pro pobočky:

- 400 klientů \Rightarrow potřebujeme 9 bitů v hostitelské části adresy, protože $2^8 = 256$ je málo, $2^9 = 512$ dostačuje, tj. podsít zabere 3 bity, délka prefixu bude $20 + 3 = 23$,
- 220 klientů \Rightarrow 8 bitů v hostitelské části ($2^8 = 256$), zbývají 4 bity pro podsít, délka prefixu je 24,
- 72 klientů \Rightarrow 7 bitů v hostitelské části, 5 bitů pro podsít, délka prefixu je 25,
- 45 klientů \Rightarrow 6 bitů v hostitelské části, 6 bitů pro podsít, délka prefixu je 26,
- 10 klientů \Rightarrow 4 bity v hostitelské části, 8 bitů pro podsít, délka prefixu je 28,
- pro WAN 2 klienti \Rightarrow 2 bity v hostitelské části, 10 bitů pro podsít, délka prefixu je 30.

Nejdřív vytvoříme sloupce s adresou sítě, délkou prefixu a maskou. Masku (resp. poslední jedničku v masce) opět použijeme pro určení adresy následující podsítě. První řádek (pro největší podsít) vypadá takto:

Název	Zaříz. /adres	Adresa podsítě	Délka pref.	Hostitelské adresy	Broadcast v podsíti	Maska
VLAN10	400 512	172.16.0.0 ...0000.0000 0000	/23			.254.0 ...1110.0000 0000

Adresu následující podsítě určíme jednoduše tak, aby se do ní vešel objednaný počet hostitelských zařízení. Podíváme se na masku – poslední jednička je v třetím oktetu, a její pozice odpovídá $2^1 = 2$. Tedy vezmeme adresu první podsítě a k jejímu třetímu oktetu přičteme číslo 2.

Název	Zaříz. /adres	Adresa podsítě	Délka pref.	Hostitelské adresy	Broadcast v podsíti	Maska
VLAN10	400 512	172.16.0.0 ...0000.0000 0000	/23			.254.0 ...1110.0000 0000
VLAN40	220 256	172.16.2.0 ...0010.0000 0000	/24			.254.0 ...1111.0000 0000

Od tohoto řádku dále již budeme pracovat v posledním oktetu, což je jednodušší. Stejným způsobem (zjištění hodnoty pro poslední jedničku v masce, automatické přičtení druhé mocniny do konkrétního oktetu) pokračujeme na dalších řádcích:

Název	Zaříz. /adres	Adresa podsítě	Délka pref.	Hostitelské adresy	Broadcast v podsíti	Maska
VLAN10	400 512	172.16.0.0 ...0000.0000 0000	/23			.254.0 ...1110.0000 0000
VLAN40	220 256	172.16.2.0 ...0010.0000 0000	/24			.255.0 ...1111.0000 0000
VLAN30	72 128	172.16.3.0 ...0011.0000 0000	/25			.255.128 ...1111.1000 0000
VLAN20	45 64	172.16.3.128 ...0011.1000 0000	/26			.255.192 ...1111.1100 0000
VLAN50	10 16	172.16.3.192 ...0011.1100 0000	/28			.255.240 ...1111.1111 0000
WAN1	2 4	172.16.3.208 ...0011.1101 0000	/30			.255.252 ...1111.1111 1100
WAN2	2 4	172.16.3.212 ...0011.1101 0100	/30			.255.252 ...1111.1111 1100
WAN3	2 4	172.16.3.216 ...0011.1101 1000	/30			.255.252 ...1111.1111 1100

Broadcasty pro podsítě vypočteme jednoduše – podíváme se na adresu podsítě z následujícího řádku a odečteme jedničku (protože broadcastová adresa z jedné podsítě je těsně před adresou následující podsítě). Takže pro první podsít je broadcast 172.16.2.0 mínus jedna (odečítáme v posledním oktetu, a protože je nulový, je to podobné, jako když v desítkové soustavě od 20 odečteme jedničku (vyšší řád snížíme o 1, nižší řád nastavíme na maximum): 172.16.1.255. Podobně ostatní řádky. Pro výpočet broadcastu v posledním řádku použijeme „virtuální“ následující neexistující podsít (172.16.3.220), takže tam bude výsledek 172.16.3.219.

Další věc, kterou je třeba dopočítat, jsou adresy přidělitelných adres. To jsou všechny mezi adresou podsítě a broadcastem pro danou podsít. Výsledek je v tabulce 8.10.



Úkoly

1. Máme přidělenou adresu 172.16.0.0/20. Navrhněte adresaci sítě s těmito podsítěmi:

- LAN1: 500 zařízení
- LAN2: 40 zařízení
- LAN3: 700 zařízení
- LAN4: 820 zařízení
- WAN1, WAN2, WAN3: 2 zařízení

Název	Zaříz. /adres	Adresa podsítě	Délka pref.	Hostitelské adresy	Broadcast v podsíti	Maska
VLAN10	400 512	172.16.0.0 ...0000.0000 0000	/23	172.16.0.1 až 172.16.1.254	172.16.1.255	255.255.254.0 ...1110.0000 0000
VLAN40	220 256	172.16.2.0 ...0010.0000 0000	/24	172.16.2.1 až 172.16.2.254	172.16.2.255	255.255.255.0 ...1111.0000 0000
VLAN30	72 128	172.16.3.0 ...0011.0000 0000	/25	172.16.3.1 až 172.16.3.126	172.16.3.127	255.255.255.128 ...1111.1000 0000
VLAN20	45 64	172.16.3.128 ...0011.1000 0000	/26	172.16.3.129 až 172.16.3.190	172.16.3.191	255.255.255.192 ...1111.1100 0000
VLAN50	10 16	172.16.3.192 ...0011.1100 0000	/28	172.16.3.193 až 172.16.3.206	172.16.3.207	255.255.255.240 ...1111.1111 0000
WAN1	2 4	172.16.3.208 ...0011.1101 0000	/30	172.16.3.209 až 172.16.3.210	172.16.3.211	255.255.255.252 ...1111.1111 1100
WAN2	2 4	172.16.3.212 ...0011.1101 0100	/30	172.16.3.213 až 172.16.3.214	172.16.3.215	.255.252 ...1111.1111 1100
WAN3	2 4	172.16.3.216 ...0011.1101 1000	/30	172.16.3.217 až 172.16.3.218	172.16.3.219	255.255.255.252 ...1111.1111 1100

Tabulka 8.10: VLSM pro síť 172.16.0.0/20

2. Máme přidělenou adresu 172.16.0.0/20. Navrhněte adresaci sítě s těmito podsítěmi:

- LAN1: 120 zařízení
- LAN2: 65 zařízení
- LAN3: 430 zařízení
- LAN4: 26 zařízení
- LAN5: 34 zařízení
- WAN1, WAN2, WAN3: 2 zařízení



8.3 Směrování

V operačním systému máme k vypsání směrovací tabulky příkaz `route`.




Úkol


Vypište svou směrovací tabulku. Jak se to dělá:


- ve Windows: `route print`
- v Linuxu: `route` nebo `ip route show`



Bezdrátové sítě

 *Rychlý náhled:* V této kapitole se podíváme na formát Wi-fi rámce a následně na nástroje pro průzkum frekvenčního spektra. Na cvičeních si vyzkoušíme také hardwarový analyzátor frekvenčního spektra.


 *Klíčová slova:* Wi-fi rámec, frekvenční spektrum, InSSIDer, Acrylic WiFi, NetSpot, WifiInfo View, LinSSIDer, iwScanner, Wifi Analyzer, HeatMapper

 *Cíle studia:* Po prostudování této kapitoly se zorientujete v tom, které adresy jsou umístěny do Wi-fi rámce, a také se naučíte prozkoumávat frekvenční spektrum používané Wi-fi sítěmi.

9.1 Wi-fi rámec

Z přednášek víme, že ve Wi-fi síti typu infrastruktura (kde tedy používáme AP) se v rámci používá poněkud více MAC adres než jsme zvyklí z Ethernetu, a navíc tam máme dva speciální příznaky určující, jestli je či není momentální zdroj/cíl na daném spoji AP.

Shrňme si nejdřív pojmy a postupy:

 V záhlaví Wi-fi rámce máme tyto dva příznaky:

- FromDS – je nastaven na 1, pokud na daném spoji je momentálním odesílatelem AP,
- ToDS – je nastaven na 1, pokud na daném spoji je momentálním příjemcem AP.

Zkratka „DS“ v názvech příznaků je *distribuční systém*.

 Z toho vyplývá, že tyto příznaky bývají 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 zaměříme na *adresy*.

1. V ad-hoc síti je to stejně jako u Ethernetu, máme MAC adresu cíle a MAC adresu zdroje.
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 zachovááno – 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).

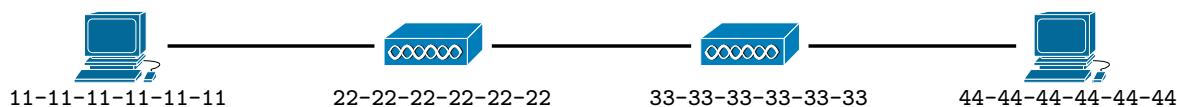
Opět se první dvě adresy vztahují k aktuálnímu spoji, ale samozřejmě musí být v záhlaví uloženo i to, kdo je zdrojem a cílem rámce. V první i druhé dvojici je zachovááno dané pořadí.

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).



Příklad 9.1

Na obrázku je naznačena jednoduchá topologie se dvěma AP. Předpokládejme, že klient s MAC adresou 11-11-11-11-11-11 posílá rámec přes Wi-fi síť klientovi s MAC adresou 44-44-44-44-44-44.



Podle topologie vidíme, že rámec půjde postupně po třech spojkách. *Na prvním spoji* budou údaje v rámci následující:

- příznaky FromDS=0 (protože *zdroj* na spoji není AP), ToDS=1 (protože *cíl* na spoji je AP).
- adresy:
 - 22-22-22-22-22-22 (příjemce na spoji je první AP)
 - 11-11-11-11-11-11 (odesílatel na spoji je zdrojové zařízení)
 - 44-44-44-44-44-44 (přidáme adresu cíle)



Úkol

Pokračujte v řešení z předchozího příkladu – doplňte pro druhý a třetí spoj na cestě hodnoty příznaků FromDS a ToDS a všechny adresy. Postupujte podle soupisu uvedeného před příkladem.




**Poznámka:**

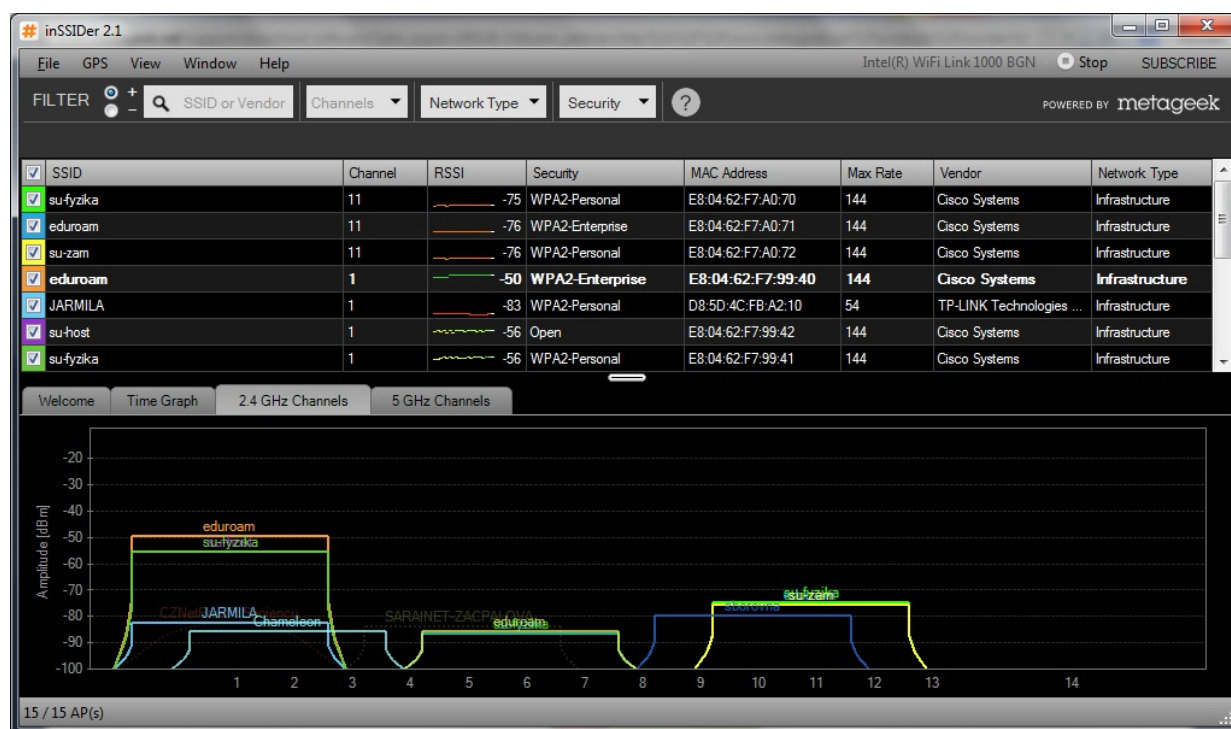
Uvědomte si, že to vše platí v případě, že komunikujete s AP pracujícím na vrstvě L2. Jestliže však jste připojeni do buňky, kde AP pracuje na vrstvě L3 (takže Wi-fi router), pak na vrstvě L2 je cílem právě ten AP, a v rámci budou opravdu jen dvě MAC adresy. Proč? Protože router (jakýkoliv) odděluje sítě, a MAC adresami se zabýváme pouze v rámci sítě. Jiné MAC adresy vlastně ani nemůžeme znát, než jen tu od našeho Wi-fi routeru, a (každá) síť nám degraduje na typ ad-hoc.



9.2 Průzkum frekvenčního spektra


Existují různé možnosti, jak si ověřit vlastnosti Wi-fi sítí, které jsou v dosahu. Můžeme použít specializované zařízení (síťový analyzátor) nebo jednoduše aplikaci.


 Aplikace *inSSIDer* pro Windows a MacOSX od společnosti MetaGeek byla původně volně ke stažení na webu <http://www.inssider.com/downloads/>, teď už se bohužel jedná o shareware.

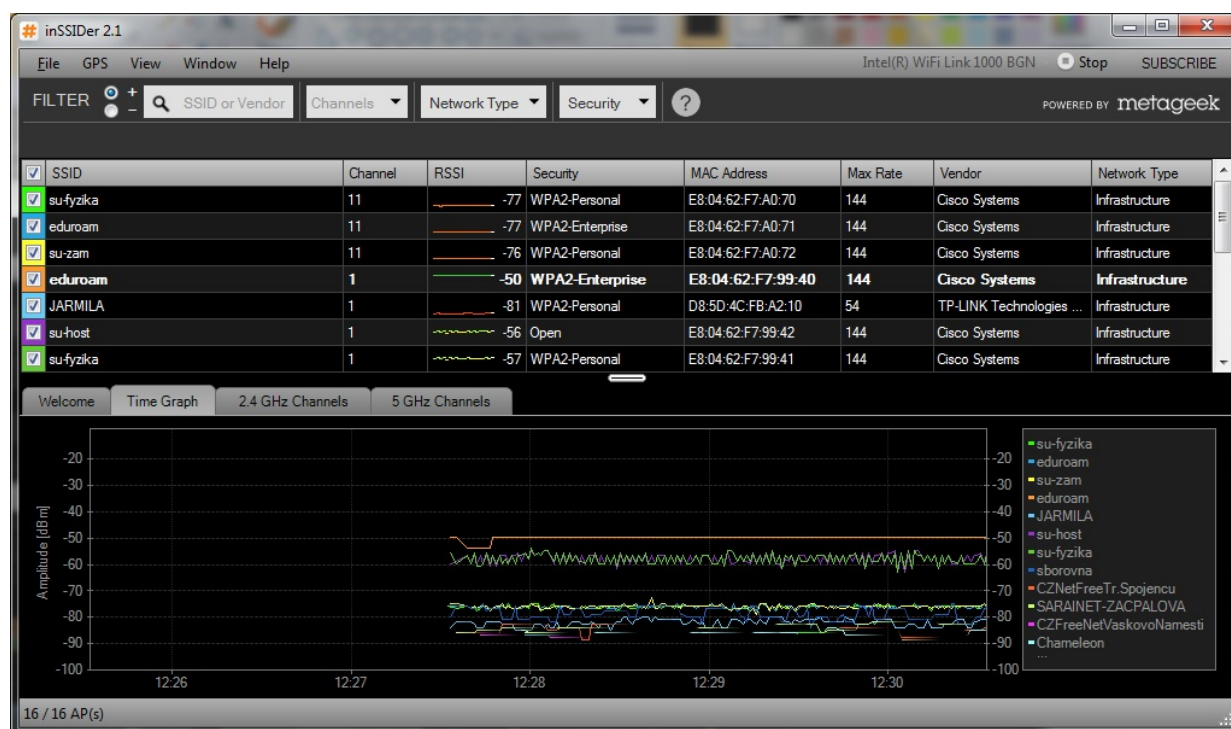


Obrázek 9.1: Program InSSIDer ve Windows


Volně k dispozici jsou však jiné podobné nástroje.

 Velice se InSSIDeru podobá nástroj *Acrylic WiFi*, který ve volně stažitelné verzi Home (pouze pro domácí použití) umí v podstatě 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 9.2: Program InSSIDer, záložka Time Graph

 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.

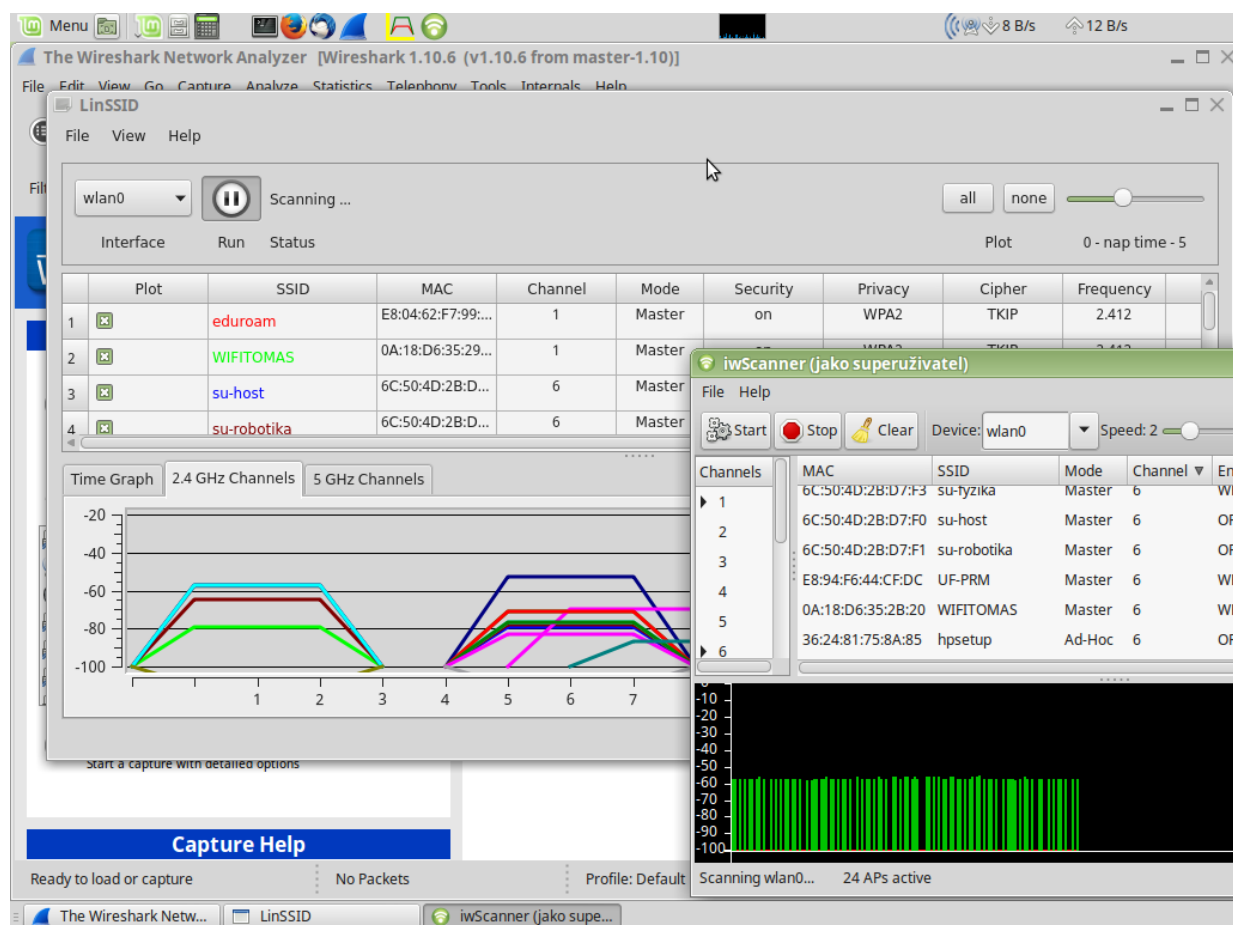


Další informace:

- <https://www.acrylicwifi.com/en/wlan-software/wlan-scanner-acrylic-wifi-free/>
- <https://www.netspotapp.com/>
- http://www.nirsoft.net/utis/wifi_information_view.html
- <http://alternativeto.net/software/inssider/>



 Programy typu *EkaHau HeatMapper* (<http://www.ekahau.com/wifidesign/ekahau-heatmapper>, po registraci ke stažení) dokážou vytvořit mapu pokrytí a dokážou tak pomoci s optimalizací rozmístění síťových prvků.



Obrázek 9.3: Programy LinSSIDer a iwScanner



Úkol

Vyberte si některou z výše jmenovaných aplikací pro průzkum frekvenčního spektra bezdrátové sítě (InSSIDer, LinSSIDer, Acrylic WiFi, NetSpot, atd.) a vyzkoušejte, prohlédněte si, co vše taková aplikace dokáže zjistit.



Literatura

- [1] OLIVIERO, Andrew – Bill WOODWARD. *Cabling: The Complete Guide to Copper and Fiber-Optic Networking*. John Wiley & Sons, 2014. ISBN 1118807383. Část dostupná také na: <https://books.google.cz/books?id=BjYKAwAAQBAJ&printsec=frontcover>
- [2] SANDERS, Chris. *Analýza sítí a řešení problémů v programu Wireshark*. 1. vyd. Brno: Computer Press, 2012, 288 s. ISBN 978-80-251-3718-5.
Ukázkové soubory se zachycenými pakety: <https://www.nostarch.com/packet2.htm> (klepněte na odkaz „Download the capture files for this book“)
- [3] TRULOVE, James. *Sítě LAN: hardware, instalace a zapojení*. 1. vyd. Praha: Grada, 2009. ISBN 978-80-247-2098-2