

**SLEZSKÁ
UNIVERZITA**

FILOZOFICKO-
PŘÍRODOVĚDECKÁ
FAKULTA V OPAVĚ



Šárka Vavrečková

Skripta do předmětu

Počítačová síť a internet — cvičení —

Ústav informatiky
Filozoficko-přírodovědecká fakulta v Opavě
Slezská univerzita

Opava

Poslední aktualizace: 9. května 2025

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>

Není dovoleno tento soubor zveřejňovat jinde než na této adrese.

Ústav informatiky
Filozoficko-přírodovědecká fakulta v Opavě
Slezská univerzita
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 Síť na hostitelském zařízení a adresace	12
2.1 Síťová karta	12
2.2 Textový režim	13
2.3 Hostname	14
2.4 Hardwarové adresy	15
2.5 IP adresy	20
2.5.1 Adresy podle IPv4	23
2.5.2 Adresy podle IPv6	24
3 Standardy	28
3.1 RFC dokumenty	28
3.2 ITU-T	31
4 Úvod do konfigurace síťových zařízení	34
4.1 Aktivní síťová zařízení	34
4.2 Jak se dostat ke konfiguraci zařízení	35
4.2.1 Připojení k zařízení pomocí konzolového kabelu	36
4.2.2 Použití konzolového serveru	37
4.2.3 Program Packet Tracer	37
4.3 Úvod do práce v Packet Traceru	39
4.3.1 Pracovní plocha a vytvoření topologie sítě	39
4.3.2 Konfigurace koncového zařízení	44
4.3.3 Generování a monitorování provozu	45
4.3.4 Jak na konfiguraci switchu a routeru	47
4.4 Konfigurace zařízení Cisco	48

4.4.1	Linky console a VTY	48
4.4.2	Pracovní módy	49
4.4.3	Základní konfigurace switche	51
4.4.4	Základní konfigurace routeru	55
5	Lokální sítě – Ethernet	58
5.1	Wireshark	58
5.2	Tabulka MAC adres na switchi	65
5.3	Kabely	68
5.3.1	Metalická kabeláž – kroucená dvojlinka	68
5.3.2	Vodiče v TP kabelech	70
5.3.3	Optická kabeláž	73
5.3.4	Další typy kabelů	77
5.3.5	Křížení	77
5.4	Zakončení kabelu	80
5.4.1	Krimpování konektoru na přímý UTP kabel	80
5.4.2	Testování	83
5.4.3	Krimpování konektoru na křížený UTP kabel	84
5.4.4	Kroucená dvojlinka se dvěma páry vodičů	85
5.4.5	Otočený a konzolový kabel	86
5.4.6	Osazení UTP kabelu do zásuvky	86
5.4.7	Optické koncovky	88
5.4.8	Další ethernetové konektory a moduly	90
5.5	Strukturovaná kabeláž	91
5.5.1	Rozvaděč	91
5.5.2	Horizontální kabeláž	92
5.5.3	Páteřní sítě	93
5.5.4	Strukturovaná kabeláž jako celek	95
6	Pokračujeme v konfiguraci	96
6.1	Konfigurace VLAN	96
6.1.1	Oddělené VLANy na switchi	96
6.2	Směrování mezi VLAN sítěmi	99
6.2.1	Router-on-a-Stick	99
6.2.2	Využití multilayer switche	100
6.3	Telnet a SSH	103
7	Co se děje na síťové a transportní vrstvě	106
7.1	Struktura paketu protokolu IP	106
7.1.1	Pakety protokolu IPv4	106
7.1.2	Pakety protokolu IPv6	108
7.2	Protokol ICMP a příkazy, které ho používají	110
7.2.1	Testování dosažitelnosti	110
7.2.2	Zjišťování cesty	112
7.3	Protokoly na transportní vrstvě	113
7.3.1	Protokol TCP a průzkum parametrů spojení	113

7.3.2	Protokol UDP a jednoduchá rychlá komunikace	117
7.4	Průzkum provozu ve Wiresharku	117
7.4.1	Filtry zobrazení	118
7.4.2	Filtry zachytávání	120
7.4.3	Analýza a statistika ve Wiresharku	122
8	Aplikační protokoly	125
8.1	Mechanismus DNS	125
8.2	Protokoly Telnet a SSH	129
8.3	Statistiky síťových protokolů	129
8.4	DHCP a možnosti získání IP adresy	130
8.4.1	Konfigurace DHCPv4	130
9	Internetworking a průzkum paketů	134
9.1	Objevování sousedů	134
9.2	Práce s IPv4 adresami	135
9.2.1	Subnetting	135
9.2.2	VLSM	144
9.3	Směrování	148
9.3.1	Směrovací tabulka	148
9.3.2	Směrovací protokol RIP pro IPv4	150
9.3.3	Směrovací protokol OSPF pro IPv4 v použití single area	151
10	Bezdrátové sítě	154
10.1	Wi-fi rámec	154
10.2	Průzkum frekvenčního spektra	156
	Literatura	159

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 запиšte 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 nejší mocniny čísla 2 znát na paměť.

**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 převedte 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 řádech (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 *celočíslně 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.

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



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 připsujeme 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* (dvojitě 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 převezech 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 z paměti 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 převedte 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.

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

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

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



**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**

Převedte 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**

Převedte 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.

0011		0111		1100	
3		7		C	$\Rightarrow (11\ 0111\ 1100)_2 = (37C)_{16}$

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



Úkol

Převed'te 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.

Původní číslo:	1	0	0	1	1	1	0	1
Negované číslo:	0	1	1	0	0	0	1	0

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.

$$\begin{array}{r} \text{První číslo:} \quad 1 \mid 0 \mid 1 \mid 1 \mid 0 \mid 0 \mid 1 \mid 0 \\ \text{Druhé číslo:} \quad 1 \mid 1 \mid 1 \mid 1 \mid 0 \mid 0 \mid 0 \mid 0 \\ \hline \text{Logický součet (OR):} \quad 1 \mid 1 \mid 1 \mid 1 \mid 0 \mid 0 \mid 1 \mid 0 \\ (10110010) \text{ OR } (11110000) = 11110010 \end{array}$$



 *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.

$$\begin{array}{r} \text{První číslo:} \quad 1 \mid 0 \mid 1 \mid 1 \mid 0 \mid 0 \mid 1 \mid 0 \\ \text{Druhé číslo:} \quad 1 \mid 1 \mid 1 \mid 1 \mid 0 \mid 0 \mid 0 \mid 0 \\ \hline \text{Logický součin (AND):} \quad 1 \mid 0 \mid 1 \mid 1 \mid 0 \mid 0 \mid 0 \mid 0 \\ (10110010) \text{ AND } (11110000) = 10110000 \end{array}$$



 *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.

$$\begin{array}{r} \text{První číslo:} \quad 1 \mid 0 \mid 1 \mid 1 \mid 0 \mid 0 \mid 1 \mid 0 \\ \text{Druhé číslo:} \quad 1 \mid 1 \mid 1 \mid 1 \mid 0 \mid 0 \mid 0 \mid 0 \\ \hline \text{Logická nonekvivalence (XOR):} \quad 0 \mid 1 \mid 0 \mid 0 \mid 0 \mid 0 \mid 1 \mid 0 \\ (10110010) \text{ XOR } (11110000) = 01000010 \end{array}$$



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.



Kapitola 2

Síť na hostitelském zařízení a adresace

 *Rychlý náhled:* V této kapitole se podíváme na některé základní úlohy týkající se hostitelských (tj. koncových) zařízení. Většina z těchto úloh se bude týkat adres. 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, maska, prefix, kanonický tvar IP adresy.

 *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, podle IP adresy zjistit adresu sítě.

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 obchodě dle vlastního uvážení (např. <https://www.alza.cz/>, <https://www.tsbohemia.cz>, <https://www.mironet.cz/>) nebo na <http://heureka.cz/> či <https://www.zbozi.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` (klepneme na tlačítko  a napíšeme `cmd`).

V Linuxu, MacOS 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.

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 vypsat 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 Hardwarové adresy



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

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

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

**Poznámka:**

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*. Nebo se tam dostaneme jednodušeji přes ikonu sítě v pravé části hlavního panelu.

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

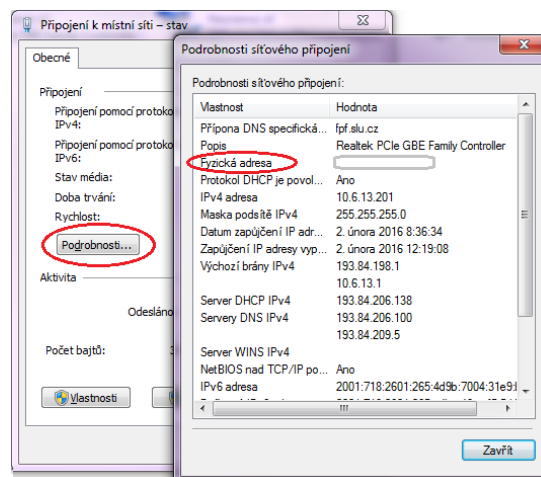
```
getmac
```

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“.

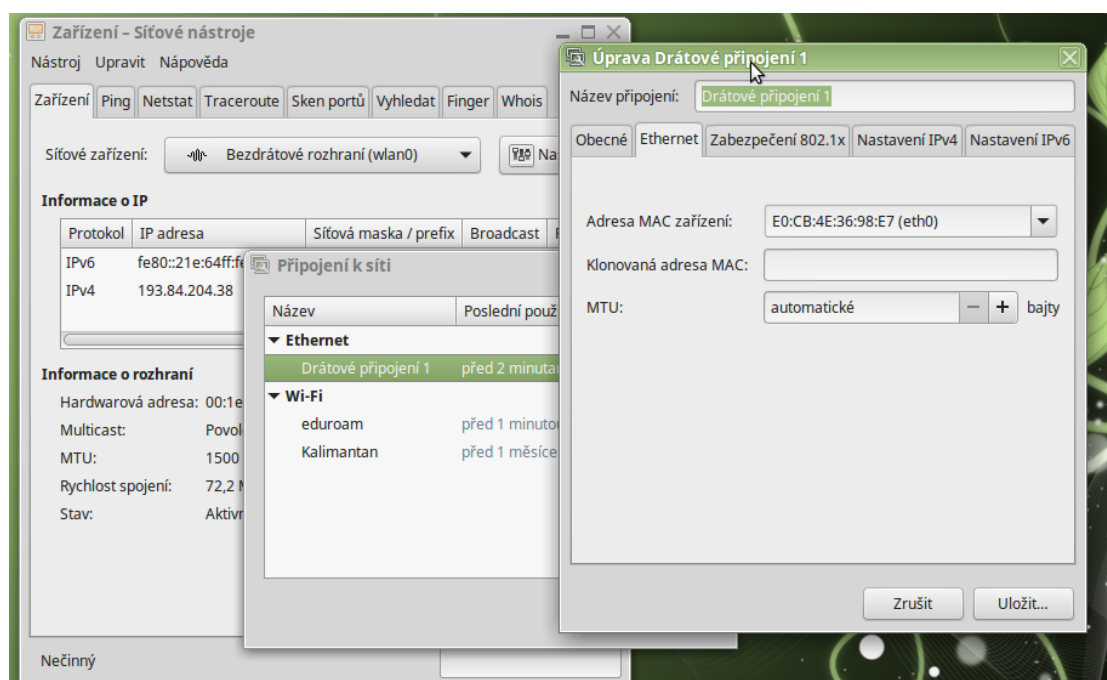
...

```
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) 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.

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`



Obrázek 2.1: Práce s adresami v Linuxu

(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 `HWadr`:

```
eth0  Link encap:Ethernet HWadr 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 ...
```



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/>.





Úkol

Zjistěte hardwarové adresy síťových rozhraní na svém počítači. Zjistěte také, kdo je výrobcem síťového rozhraní, kterému je daná adresa přiřazena.



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:

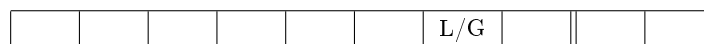
```
ip link set dev eth0 down
```

```
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.



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

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 odesílán rámec určený pro všechna zařízení v lokální síti.

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



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

Obrázek 2.3: 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.



2.5 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



**Příklad 2.9 (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 2.10 (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).

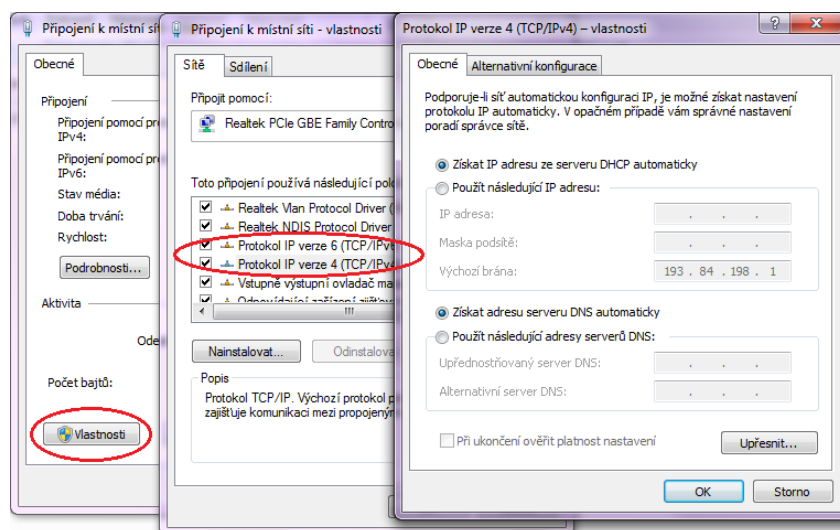


IP adresu máme buď *staticky* naklepanou v příslušném nástroji, nebo je vždy při připojování do sítě přidělována *dynamicky*. Měli bychom předně umět zjistit, která z těchto možností se nás týká, a dále umět nakonfigurovat (určit, zda má být adresa dynamicky požadována nebo jestli má být použita staticky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



Příklad 2.11 (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 2.12 (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.

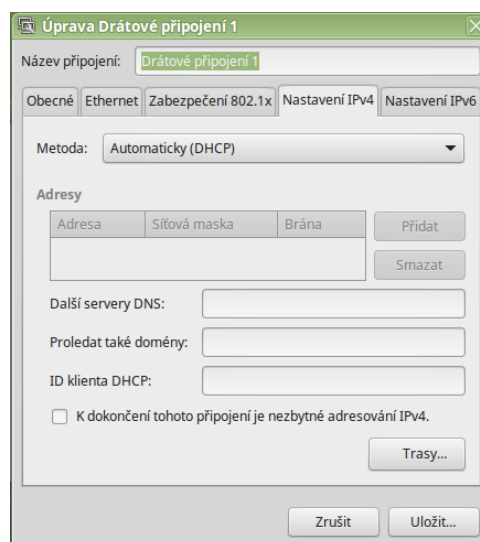
Jak vidíme získávání IPv4 adresy je zajišťováno dynamicky z DHCP serveru. Pokud bychom chtěli sami staticky zadat vlastní IPv4 adresu, v rozbalovacím boxu zvolíme místo *Automaticky (DHCP)* volbu *Ruční*, čímž se zpřístupní položky na zbytku záložky. Podobně na další záložce lze konfigurovat IPv6.

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
```



2.5.1 Adresy podle IPv4

Jak víme, IP adresu mají nejen konkrétní zařízení, ale také síť 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 2.13

Zařízení má IP adresu 170.254.237.18 a masku 255.255.128.0. Zápis s délkou prefixu: 170.254.237.18/17 (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:	10101010.11111110.11101101.00010010
maska:	11111111.11111111.10000000.00000000
operace AND:	10101010.11111110.10000000.00000000

Po převodu do desítkové soustavy získáme adresu sítě: 170.254.128.0/17. 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



Příklad 2.14

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**. Maska podsítě nám říká, že ze čtyř oktetů (32 bitů) adresy jsou první tři oktety (24 bitů) síťovou částí adresy a zbývajících 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 cokoliv, 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.



2.5.2 Adresy podle IPv6

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.



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, skupině 16 bitů taky říkáme *hextet*. Z toho vyplývá, že adresa bude mít osm skupin/hextetů oddělených dvojtečkami.

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. Postupujeme takto:

1. Pokud jsou na začátku skupiny číslice 0, odstraníme je. Například $0025 \rightarrow 25$. Pokud je celá skupina nulová, odstraníme tři nuly a jednu necháme.
2. Zaměříme se na posloupnosti nulových skupin.
 - Pokud je v adrese jediná, odstraníme ji a necháme dvě dvojtečky na místě krácení:
 $2001:54:b:0:0:0:4fa:8 \rightarrow 2001:54:b::4fa:8$
 - Pokud je v adrese několik posloupností nulových skupin, volíme tu nejdelší. Pokud jsou dvě stejně dlouhé, volíme tu víc vlevo.

Pokud by vybraná nulová posloupnost obsahovala jen jednu skupinu, necháme ji bez krácení:
 $2001:54:b:0:a40:0:4fa:8 \rightarrow$ necháme bez změny, obě nulové posloupnosti obsahují jen jednu skupinu

Krátíme vždy pouze jednu jedinou posloupnost nulových skupin, protože musíme zajistit jednoznačnost.



Příklad 2.15

Na příkladech si ukážeme sestavení 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 nulové 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 2.1 (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 pohltnout 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,
- takovou posloupnost nulových skupin, pokud obsahuje víc než jednu skupinu, odstraníme a na místě krácení necháme dvojitou dvojtečku.



Příklad 2.16

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

`2001:718:2601:0:0:e5f8:31e8:b6da` $\implies$ `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.

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 v kanonickém (tj. 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
- ff02:0000:0000:0000:0000:0000:0001:0002
- 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
- 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.



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 nej-důležitější 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.

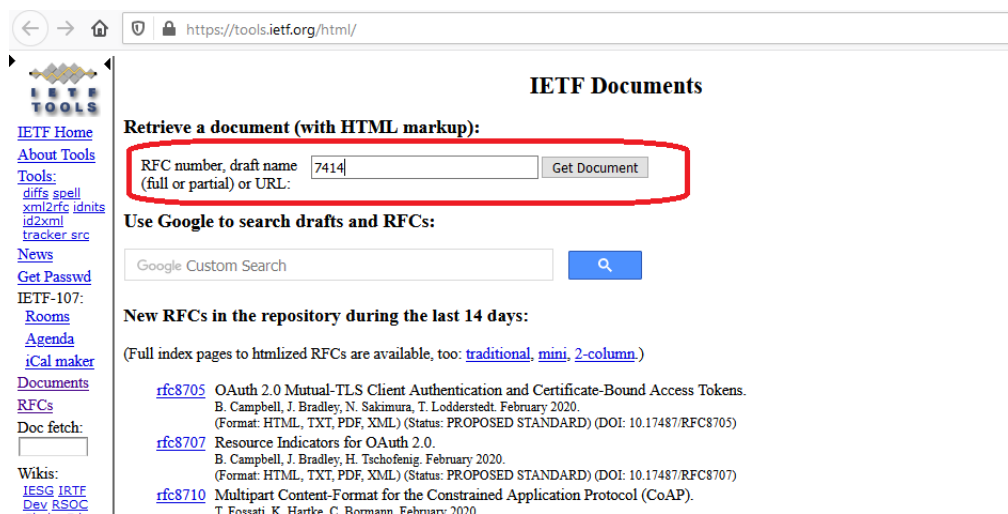


Příklad 3.1

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 29 vidíme, kde je vyhledávací pole.

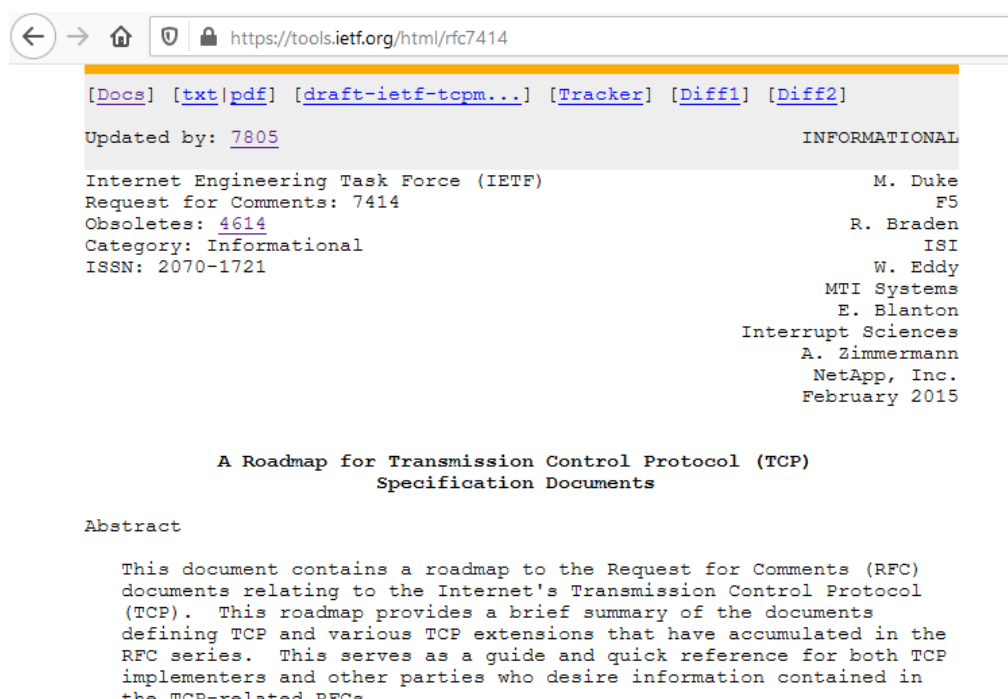
Pokud neznáme číslo RFC, dá se vyhledávat i podle klíčového slova, ale obvykle dostaneme příliš mnoho výsledků (v takovém případě je lepší se zeptat Googlu).



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.



Příklad 3.4

Chceme RFC dokument, který popisuje formát IPv4 paketu. Stačí se zeptat Googlu: „rfc ipv4 packet“. Měli bychom se takto dostat k dokumentu RFC 791, který právě popisuje protokol IP.

V sekci 2.1 je nákres vztahů mezi protokoly, kde jasně vidíme, že na nižší vrstvě bude některý protokol pro místní síť, kdežto na nadřazené vrstvě bude některý z protokolů TCP, UDP nebo jiný

na transportní vrstvě, a nad nimi se očekávají zase další (aplikační) protokoly.

Na další stránce vidíme jednoduchý náčrtek přenosové cesty v případě, že provoz jde přes router. Cesta začíná i končí v aplikaci, Internet module je prostě modul zvládající protokolový zásobník TCP/IP, dále zkratka LNI znamená „Local Network Interface“.

Na následující stránce se například dozvíme, že adresy zabírají 4 oktety, tedy 32 bitů.

O několik stránek dál v sekci 3.1 najdeme konečně náčrtek IP paketu s vysvětlením všech položek.



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 náčrsky 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 3.5

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 32 jsou jednotlivé kroky vyznačeny.

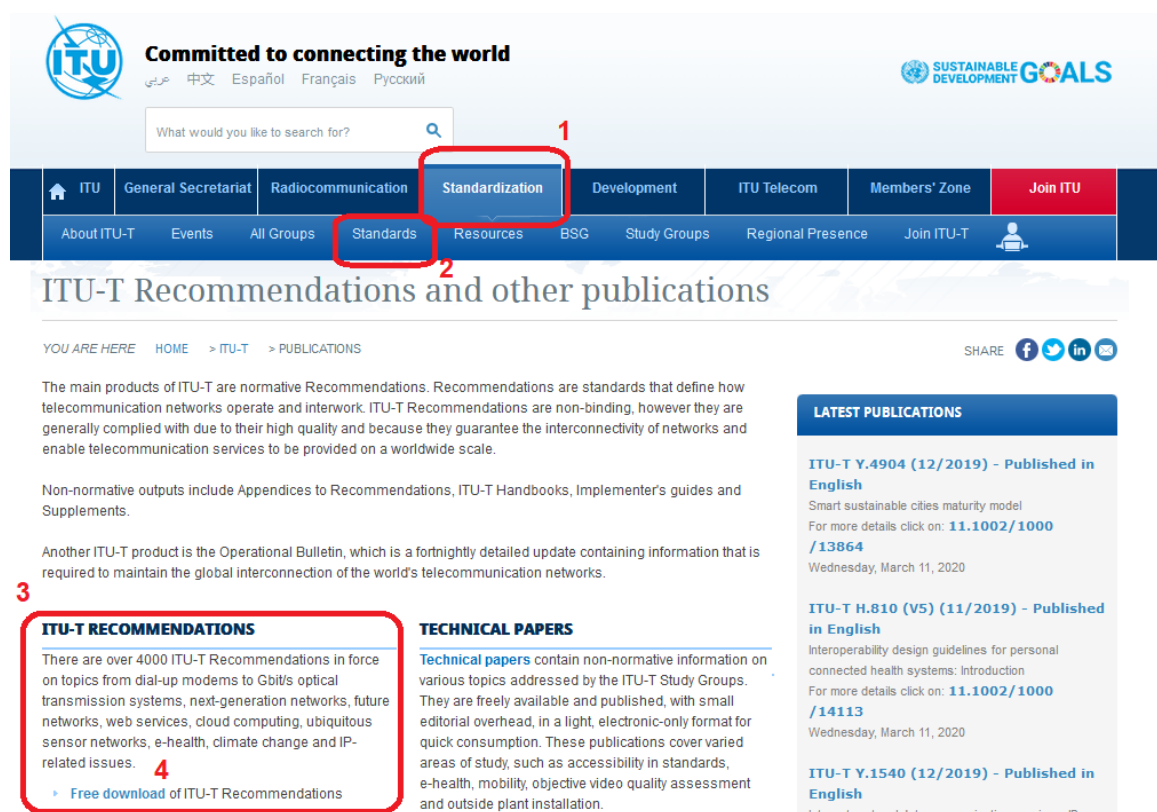
Tam hledáme doporučení podle počátečního písmene (tj. „G“) a postupně se „doklikáme“ k položce *G.993.2 (Very high speed digital subscriber line transceivers 2 (VDSL2))*. V tabulce máme různé verze, vybíráme samozřejmě co nejnovější.

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



Příklad 3.6

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



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

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).

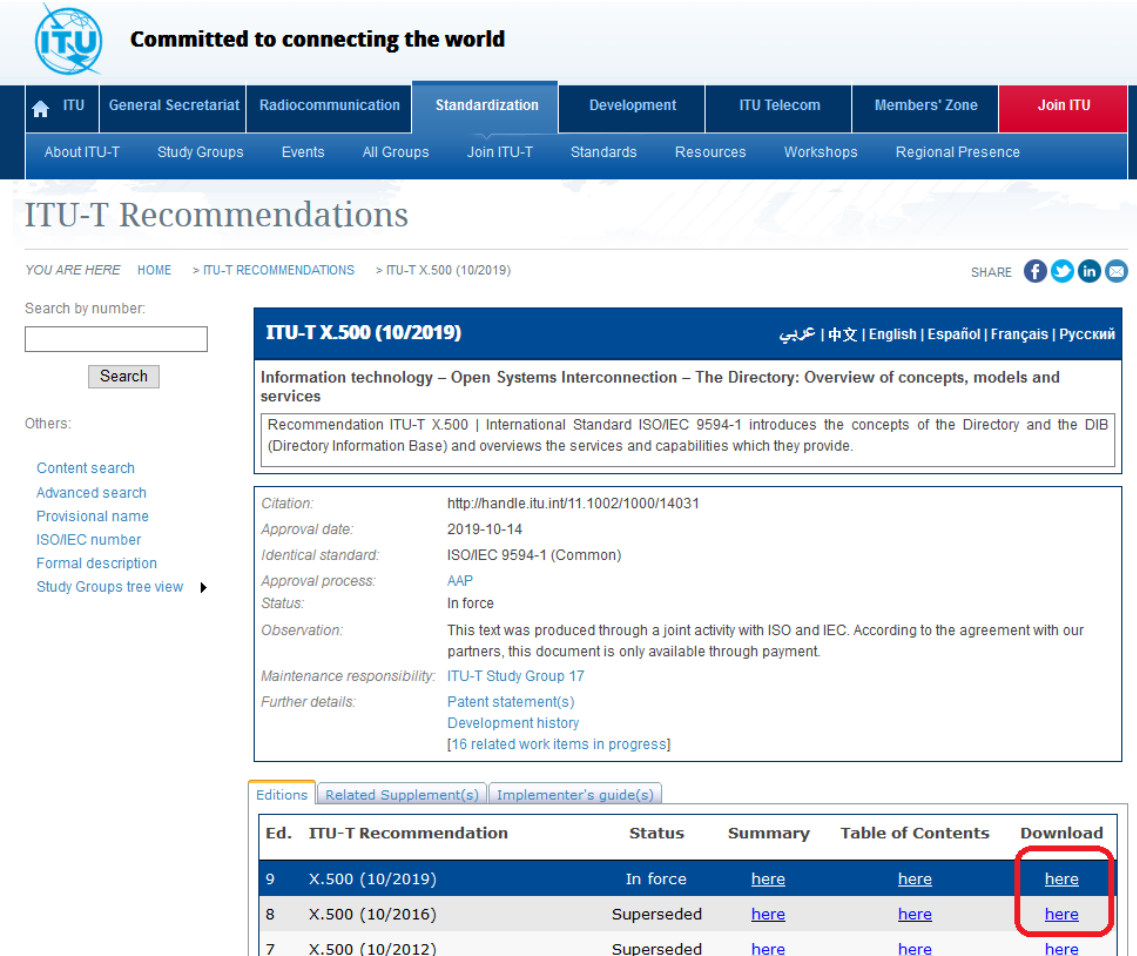
Na adrese <http://www.itu.int/itu-t/recommendations/rec.aspx?rec=X.500> najdeme základní informace o protokolu a také odkazy na PDF soubor s celým zněním. V tabulce tam najdeme různé verze. Pokud vybereme PDF soubor s verzí z roku 2019, jsme upozorněni, že přístup je pouze pro platící zájemce. Jestliže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



ITU Committed to connecting the world

Home | ITU | General Secretariat | Radiocommunication | **Standardization** | Development | ITU Telecom | Members' Zone | Join ITU

About ITU-T | Study Groups | Events | All Groups | Join ITU-T | Standards | Resources | Workshops | Regional Presence

ITU-T Recommendations

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

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)

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



Úvod do konfigurace síťových zařízení

 *Rychlý náhled:* V této kapitole se již zaměříme na aktivní síťové prvky. Naučíme se se je propojit a zprovoznit včetně základního zabezpečení. Zaměříme se na přístup do konfigurace síťových prvků přes konzolový kabel a konzolový server včetně role linek, a pro účely procvičování také použití aplikace Packet Tracer. Budeme se zabývat základní konfigurací switche a routeru: hesla, IP adresy, výpis informací z konfigurace.

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

 *Cíle studia:* Po prostudování této kapitoly se budete orientovat v základní konfiguraci switche a routeru, budete umět používat Packet Tracer a porozumíte různým módům v zařízeních Cisco.

4.1 Aktivní síťová zařízení

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.



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 jako například přepínání provozu podle tabulky MAC adres zde nehledíme.



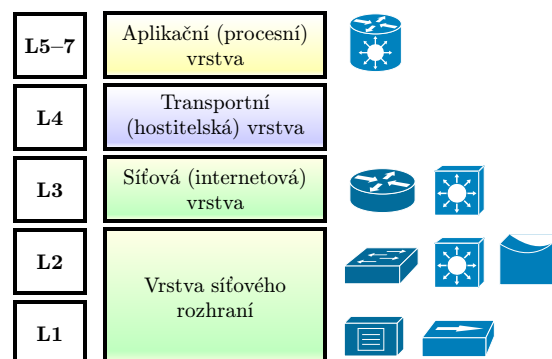
Switch s přidanou funkcionalitou vrstvy L3 (L3 switch, multilayer switch) je takový přepínač, kterému, jak název napovídá, byla přidána funkcionalita vrstvy L3 (tedy síťové). Znamená to, že se sice pořád jedná o switch, ale má vestavěný směrovací modul (protože na vrstvě L3 se směruje mezi sítěmi) a některé porty tohoto zařízení se mohou chovat jako porty routeru. Rozlišujeme zde tedy L2 porty a L3 porty. S provozem je možné zacházet na vyšší úrovni (rozbalit PDU až na úroveň vrstvy L3, pracovat s IP adresami, směrovat apod.).



Brána (gateway) slouží k propojení dvou různých typů sítí, resp. slouží jako spojovací bod či „překladač“. 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 a huby, na vrstvě L2 především switche (včetně těch s funkcionalitou vrstvy L3) a bridge, na vrstvě L3 routery a switche s funkcionalitou vrstvy L3, na aplikační vrstvě pak brány (gateway).

Ve výuce budeme používat zařízení společnosti Cisco, nicméně na podobném principu je založena konfigurace i u zařízení jiných výrobců (různé módy pro příkazy atd.).



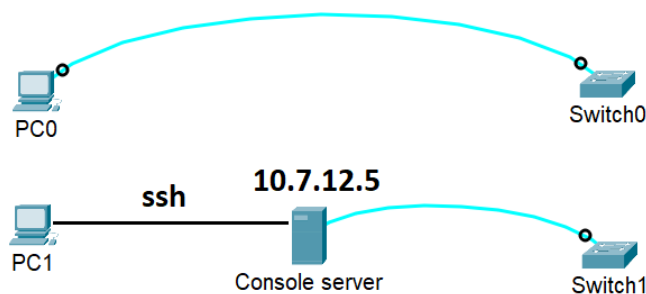
4.2 Jak se dostat ke konfiguraci zařízení

V tomto předmětu si vyzkoušíme tyto možnosti přístupu k síťovým zařízením:

- Připojení k zařízení pomocí konzolového kabelu: tuto možnost použijeme, pokud máme zařízení „v ruce“ (spíše na stole, profesionální zařízení jsou poněkud těžší než smartphone).
- Připojení k zařízení přes konzolový server: pro případ, že zařízení je umístěno v racku (rack je ta skříň, ve které se nacházejí switche, routery, servery a další podobná zařízení).
- Přístup k síťovému zařízení přes SSH: to je ovšem proveditelné až tehdy, kdy má toto zařízení přidělenou IP adresu a povolený přístup do konfigurace přes SSH, takže spíše až pro úpravy v konfiguraci. Na tuto možnost se podíváme až později, když se naučíme nakonfigurovat SSH.

- Packet Tracer: jde o program simulující síťová zařízení a jejich sítě, to můžeme použít i v případě, že zrovna žádné „fyzické“ zařízení nemáme po ruce, nebo tehdy, když chceme do sítě propojit víc zařízení než máme reálně k dispozici, nebo si něco prostě vyzkoušet.

Na obrázku 4.1 je naznačen rozdíl mezi prvními dvěma možnostmi.



Obrázek 4.1: Srovnání možností připojení k síťovému zařízení pro konfiguraci

4.2.1 Připojení k zařízení pomocí konzolového kabelu

Kromě samotného zařízení potřebujeme *konzolový kabel*, což je kabel, na jehož jedné straně je rozhraní RS-232 (také DB-9, sériové či služební rozhraní) a na druhé straně port RJ-45 (klasický port pro počítačovou síť Ethernet).



Obrázek 4.2: Vlevo: konzolový kabel RJ-45/DB-9, vpravo: redukce DB-9 na USB-A (Zdroj: https://dcloud-cms.cisco.com/help/connect_console)

Rozhraní RS-232 patří do počítače, rozhraní RJ-45 do portu na switchi či routeru, ovšem ne kteréhokoliv – musí být označeno jako „console“, tedy do konzolového portu. Obvykle bývá zvýrazněn světle modrou barvou.



Zdroj: Cisco.com

Protože dnes málokterý počítač či notebook má port RS-232 (DB-9), používáme redukci na USB, takže vlastně potřebujeme tyto části:

PC — redukce USB/DB-9 — kabel DB-9/RJ-45 — switch

Pokud neseženeme skutečný konzolový kabel DB-9/RJ-45, pak je tu ještě možnost použít redukci DB-9 na RJ-45, takže se nám posloupnost použitých prvků trochu prodlouží:

PC — redukce USB/DB-9 — kabel DB-9 (oba konce) — redukce DB-9/RJ-45 — switch

Jen je třeba si u rozhraní DB-9 pohlídat, kde jsou piny (kovové kolíky) a kde dírky, protože dvě strany s piny nepojíme. . .

 Pokud se nám podařilo zajistit „datovou cestu“ pomocí konzolového kabelu a samozřejmě i napájení (také síťové zařízení potřebuje energii), zbývá zprovoznit přístup po softwarové stránce. K tomu potřebujeme některý program pro komunikaci přes sériovou linku, například:

- Putty – program pro Windows a některé distribuce Linuxu (máme v učebně),
- minicom – jednoduchý emulátor terminálu používaný například v Linuxu (taktéž máme v učebně),
- další možnosti.

V nástroji bývá obvykle nastaveno, přes které rozhraní konkrétně se má připojit. V učebně je už vše nastaveno, ale pokud budete chtít, podíváme se, jak se to dělá.

4.2.2 Použití konzolového serveru

Předpokládejme, že máme správně nakonfigurovaný konzolový server a síťové zařízení (router, switch) je k němu připojeno konzolovým kabelem, jak je naznačeno na obrázku 4.1 dole. Pak zbývá jen navázat spojení s konzolovým serverem, což se obvykle děje přes SSH.

 Potřebujeme znát IP adresu konzolového serveru, údaje pro přihlášení a jakýkoliv program, který umí navázat SSH spojení s touto IP adresou (SSH klienta). Náš konzolový server je dostupný pouze v síti přímo v učebně, přihlašovací údaje dostanete na místě.

V učebně máme sice program Putty, který má vestavěného SSH klienta a tedy je poměrně univerzálním prostředníkem, ale jsou i další možnosti: například v každém operačním systému bývá dostupný příkaz `ssh`. Použijeme jednu z těchto možností:

```
ssh ip.adresa  
ssh uzivatel@ip.adresa
```

V prvním případě jsme dotázáni na jméno a heslo, v druhém případě jen na heslo, protože přihlašovací jméno jsme zadali před adresou.

4.2.3 Program Packet Tracer

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. Existuje pro různé platformy: Windows, Linux i MacOS.

Novější verze jsou dostupné jen těm, kdo jsou registrovaní na webu Netacad.com (to je školicí server společnosti Cisco). Registrace je zdarma, jen bohužel zabere určitý čas a pak není úplně jednoduché najít na webu program Packet Tracer, ale při troše trpělivosti se to dá. Kdyby někdo nechtěl projít registrací, je možné stáhnout verzi 6.2, která ještě nevyžaduje přístupové údaje, ale bohužel už není aktuální se všemi riziky včetně bezpečnostních.

Pokud se rozhodnete potrpět registrací, můžete jí hned využít a poohlédnout se po různých kurzech, které jsou v systému dostupné. Mnohé z nich jsou zdarma (tzv. Self-paced), a to jak z oblasti počítačových sítí, tak i například programování.



Postup (Registrace na webu Netacad)

Příslušný web má adresu <https://www.netacad.com/>. Vpravo nahoře najdete tlačítko Login, pak hledejte odkaz pro vytvoření účtu:

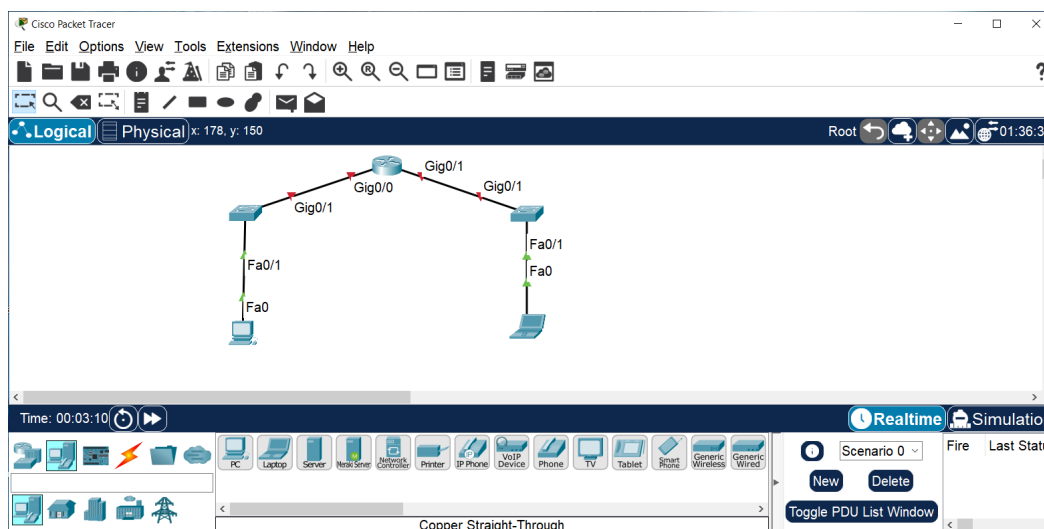
Don't have an account? [Sign up](#)

Jako světa a kyberbezpečnosti znalí studenti určitě víte, že není dobrý nápad přihlašovat se přes účet Google či podobné...

Následuje vyplnění údajů (Cisco chce především vědět, ze které jste země, kdy jste se narodili a jakou máte e-mailovou adresu) a ověřování totožnosti (je třeba potvrdit přes e-mail, že opravdu existujete, tatáž adresa se pak používá i pro další komunikaci, tedy určitě nezadávejte falešnou adresu). Volíte si také heslo, to byste si určitě měli zapamatovat.

Až budete mít registraci za sebou, Packet Tracer si můžete stáhnout na <https://www.netacad.com/resources/lab/cisco-packet-tracer-resources>

Po instalaci a spuštění se spustí průvodce, ve kterém zadáte své přístupové údaje, které jste si zvolili při registraci. Doporučuji zatrhnout, že se přihlášení má pamatovat pro příští 3 měsíce, ať tyto údaje nemusíte alespoň po tuto dobu zadávat znovu.



Obrázek 4.3: Prostředí aplikace Cisco Packet Tracer ve verzi 8.2.2

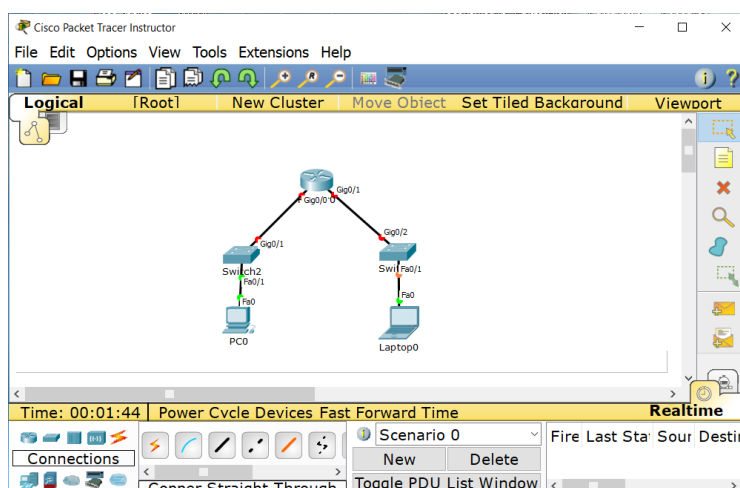


Postup (Starší verze Packet Traceru)

Pokud si chcete stáhnout Packet Tracer, ale nechcete se registrovat, starší verzi 6.2 bez nutnosti registrace najdete na <https://www.filehorse.com/download-cisco-packet-tracer-32/27899/download/>.

Jen upozorňuji, že opravdu není dobrý nápad používat starší verzi čehokoliv z bezpečnostních důvodů, a také některé funkce najdeme jen v novějších verzích, novější verze jsou také lépe vybaveny různými zařízeními (včetně Internetu věcí). Ovšem je třeba uznat, že grafické rozhraní starší verze má své kouzlo...



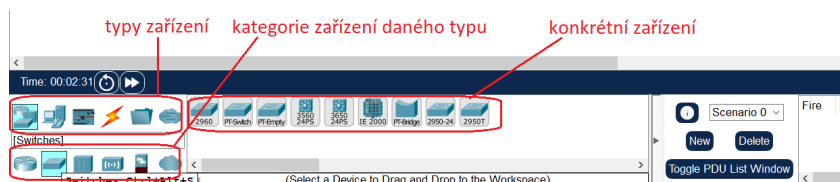


Obrázek 4.4: Prostředí aplikace Cisco Packet Tracer ve verzi 6.2

4.3 Úvod do práce v Packet Traceru

4.3.1 Pracovní plocha a vytvoření topologie sítě

Většinu okna zabírá pracovní plocha, na které budeme vytvářet topologii své sítě. Dole vlevo najdeme síťové prvky, které při tom budeme používat.



Obrázek 4.5: Typy zařízení v Packet Traceru

Na obrázku 4.5 je pohled na zásobárnu síťových prvků v novější verzi. Ve starší verzi vypadá podobně, jen tam najdeme pouze kategorie zařízení, nejsou rozčleněny podle typů, a taky je jich méně.

Nahoře vlevo je přepínač mezi logickým a fyzickým pohledem. Většinou si vystačíme s logickým pohledem, který je ostatně výchozí, ale pokud chceme mít co nejrealističtější zážitek včetně umístění zařízení do racku, můžeme zvolit fyzický pohled.

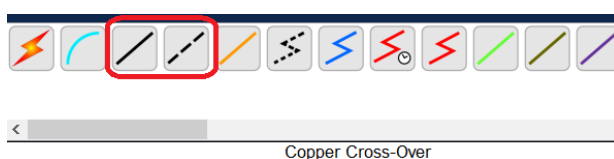


Příklad 4.1 (Orientace na pracovní ploše v Logical View)

S pracovní plochou se pracuje následovně:

- Pokud chceme na plochu umístit nový síťový prvek, vybereme si nejdřív typ a kategorii, 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 klepeme na jednotlivá zamýšlená umístění.

- Když jsme se zmýlili a chceme prvek odstranit, pak v panelu nástrojů u pravého okraje okna klepneme na , resp. v starší verzi , (čímž se přesuneme do „mazacího módu“) a pak klepneme na dotyčný prvek. Až zmizí, použijeme klávesu **[Esc]** nebo klepneme na tomtéž panelu vpravo na , resp. v starší verzi , (běžný mód) a můžeme dál pracovat.
- Jednotlivé prvky propojujeme spojem (třeba kabelem) následovně:
 - dole zvolíme typ/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.



Většinou budeme používat kabely značené plnou nebo přerušovanou černou čarou, které jsou vyznačené výše – ty první mezi zařízeními různého typu (například PC–switch, switch–router), ty druhé mezi zařízeními stejného typu z pohledu Ethernetu (PC–PC, switch–switch, router–router, PC–router).



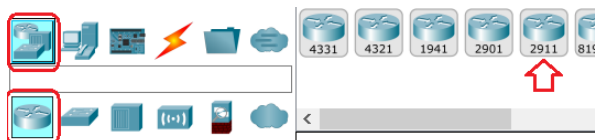
Další informace:

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 4.2 (Úprava topologie v Logical View)

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 pracovní plocha prázdná. Nejdřív umístíme router. V seznamu typů vybereme síťová zařízení (první typ), kategorii routery (takéž hned první) a v seznamu konkrétních zařízení

si zvolíme například router 2911. 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, viz obrázek vlevo. Nějaká vybereme, a tím jsme vytvořili základ naší topologie.

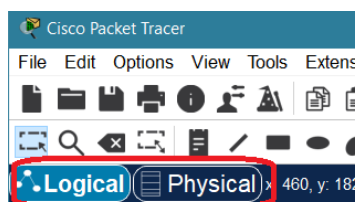
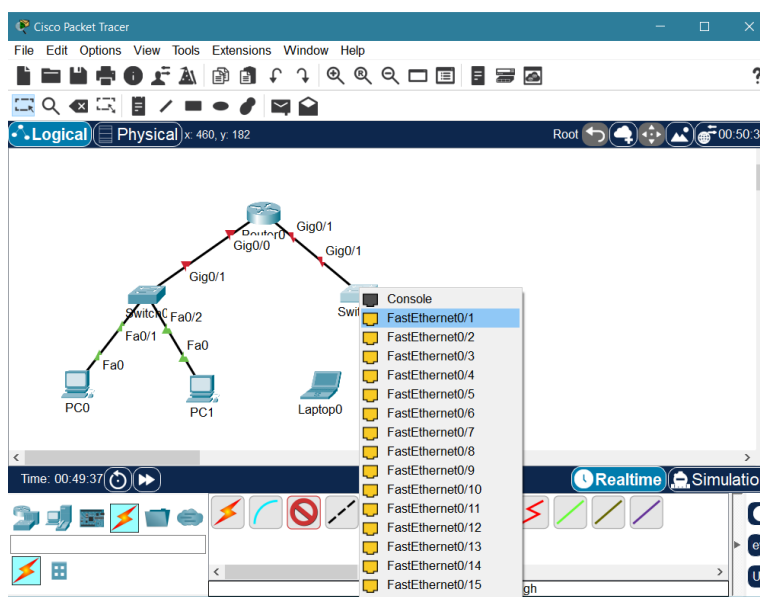
Můžeme vybrat počítače i notebooky, dle svého uvážení.

Topologii dokončíme správným propojením uzlů sítě. Dole najdeme kategorii *Connections* (ikona ⚡). 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 ⇒ 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 barevná LED indikující stav portu. Zelená znamená, že je port aktivní. Na následujícím obrázku vidíme momentální stav naší topologie ve chvíli, kdy připojujeme kabel k druhému switchi a volíme port.

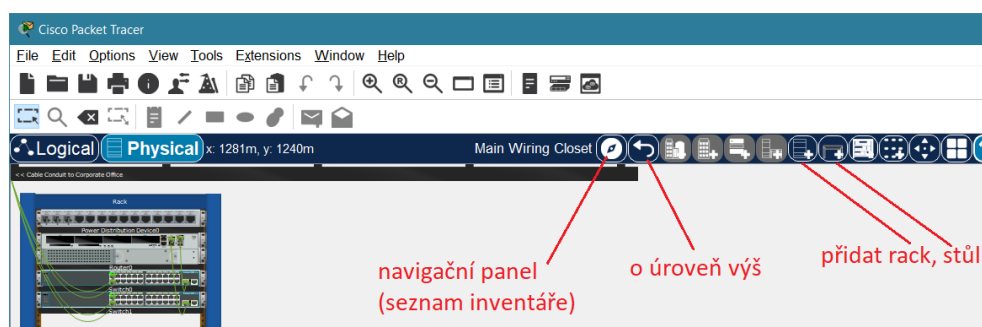


Zatím jsme pracovali v logickém pohledu (Logical View), ale existuje také fyzický pohled (Physical View). Přepínač mezi těmito dvěma pohledy najdeme vlevo nahoře, jak vidíme na obrázku vlevo. *Fyzický pohled* v terminologii Packet Traceru nám umožňuje návrh konkrétního umístění v místnostech a racku.



Příklad 4.3 (Vytvoření první topologie v Physical View)

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. Počítače a notebooky se umístily poněkud „mimo“, ale to napravíme později.

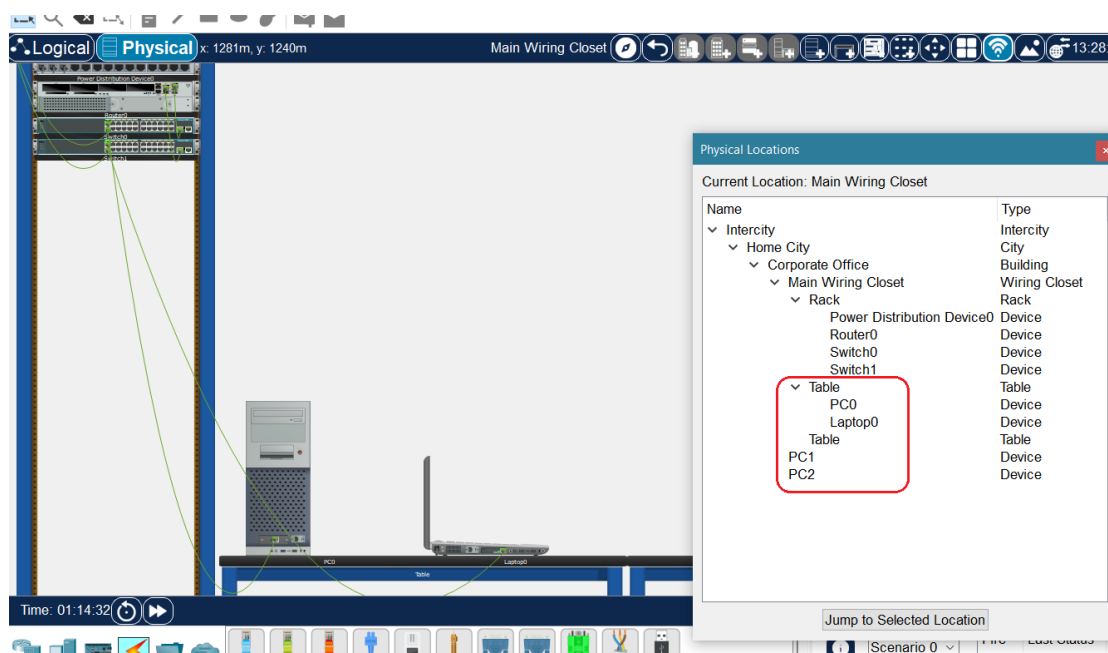


Obrázek 4.6: Ikony ve fyzickém pohledu v Packet Traceru

Na obrázku 4.6 jsou označeny některé ikony, které se nám budou hodit. Směrem hlouběji do struktury jsme se pohybovali jednoduše tak, že jsme klepli na to, do čeho jsme se chtěli přesunout. Naopak o úroveň výš se přesuneme pomocí zpětné šipky. V místnosti (v našem případě serverovně) můžeme přidávat další nábytek, například budeme chtít stůl pro počítače – k tomu máme další ikony.

Takže si přidáme dva nové stoly (ať se nám tam všechny počítače a notebooky vejdou) a teď budeme chtít přesunout tato zařízení na správné místo. Sice i pro přesuny existuje ikona, ale zařízení se mezi místnostmi jednodušeji přesouvají v navigátorovi.

Na obrázku 4.6 je vyznačena taky ikona pro zobrazení navigačního panelu. Na obrázku 4.7 je navigační panel, ve kterém už jsme přesunuli první dvě PC na stůl. Stačí v navigačním okně „vzít“ myší příslušný prvek a přesunout ho na dané umístění (vše v rámci navigátoru), v našem případě na nápis „Table“.



Obrázek 4.7: Navigační panel v Packet Traceru

Pro přesuny uvnitř místnosti už nepotřebujeme navigátoru, tam přesouváme přímo myší.

V fyzickém pohledu můžeme manipulovat také s celými místnostmi, budovami apod., což vět-

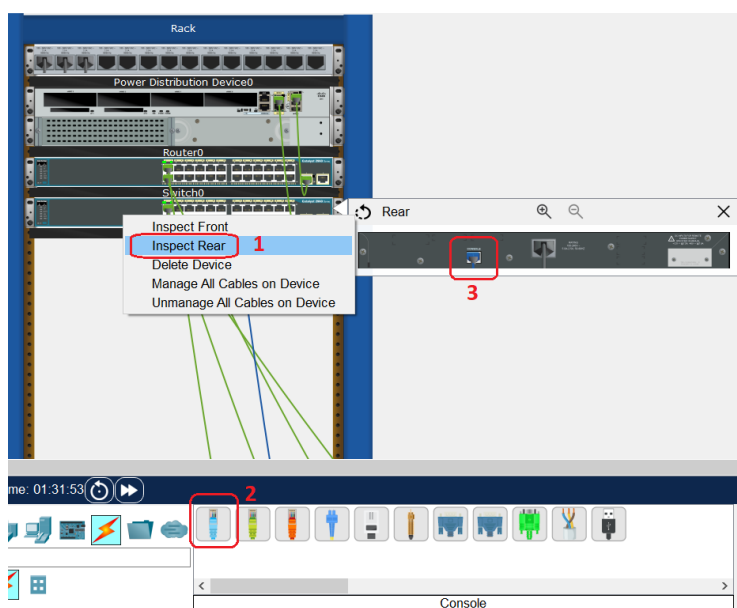
šinou nebudeme potřebovat. Můžeme také měnit rozlišení zobrazení, například pokud chceme obraz přiblížit a zvětšit detaily, držíme tlačítko **Ctrl** a kolečkem myši zvětšujeme (nebo zmenšujeme).



Příklad 4.4 (Práce v Physical View)

Také v fyzickém pohledu se dá používat panel pro přidávání nových prvků do topologie, jen například kabely na panelu vypadají poněkud realističtěji. Různá zařízení můžeme přetahovat přímo na vybrané místo, třeba do racku, také v rámci racku je můžeme přesouvat.

Kabely se připojují přímo do vybraného rozhraní, a protože příslušné porty mohou být i na té straně zařízení, která nám není zrovna viditelná, máme možnost zobrazit si jak přední, tak i zadní stranu zařízení. Stačí klepnout pravým tlačítkem myši na zařízení a vybrat „Inspect Front“ nebo „Inspect Rear“ pro zobrazení přední či zadní strany zařízení v samostatném panelu. Pak vybereme kabel a myši ho připojíme do příslušného portu.



Obrázek 4.8: Zapojení konzolového portu na zadní stranu switchu v Physical View

Na PC, notebooku a serveru máme potřebné porty na přední straně, včetně konzolového portu.



Úkol

Nainstalujte si Packet Tracer. Podle příkladů si vyzkoušejte vytvoření jednoduché topologie a prodělejte 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).



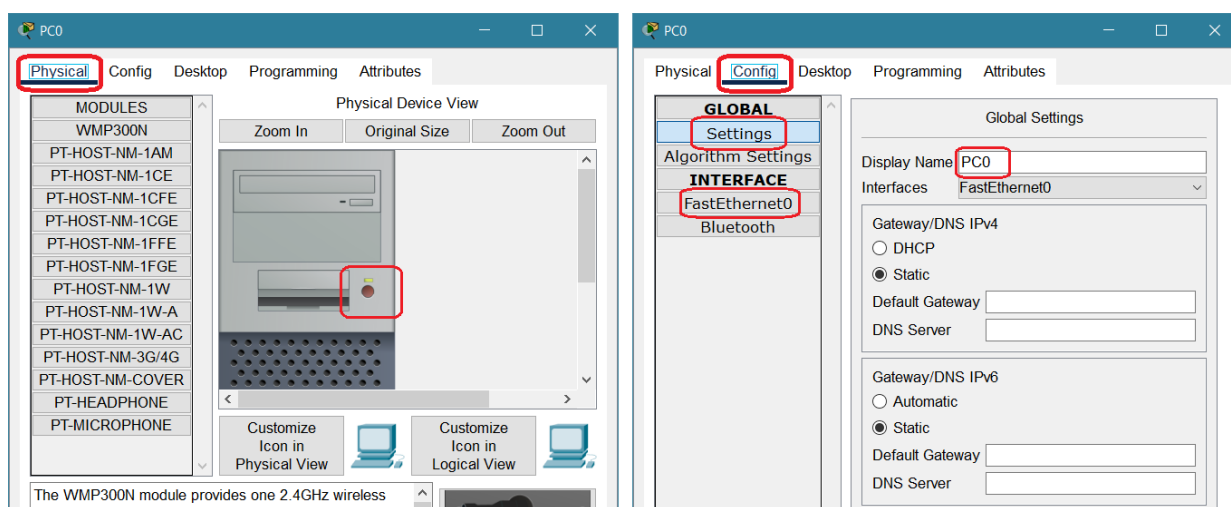
4.3.2 Konfigurace koncového zařízení

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



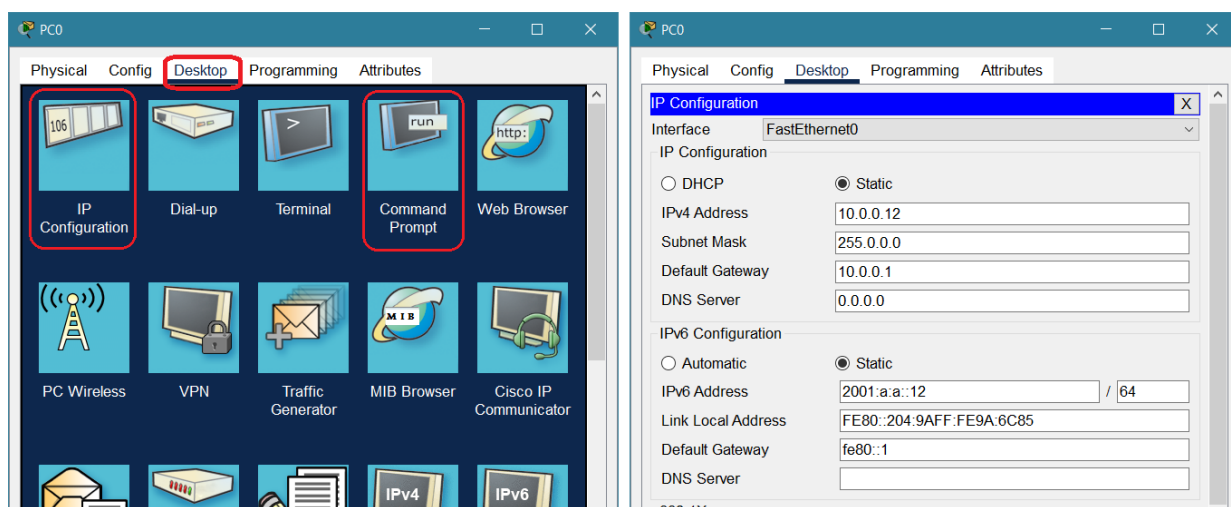
Příklad 4.5 (Konfigurace počítače)

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.



Obrázek 4.9: Zapnutí/vypnutí PC a základní konfigurace

Na obrázku 4.9 vidíme obsah prvních dvou záložek. Na té první můžeme zapnout či vypnout zařízení (tlačítko je na obrázku také vyznačeno), na druhé záložce volíme zobrazované jméno zařízení a také je zde jedna z možností, kde se dá nastavit IP adresa s maskou a adresa brány.



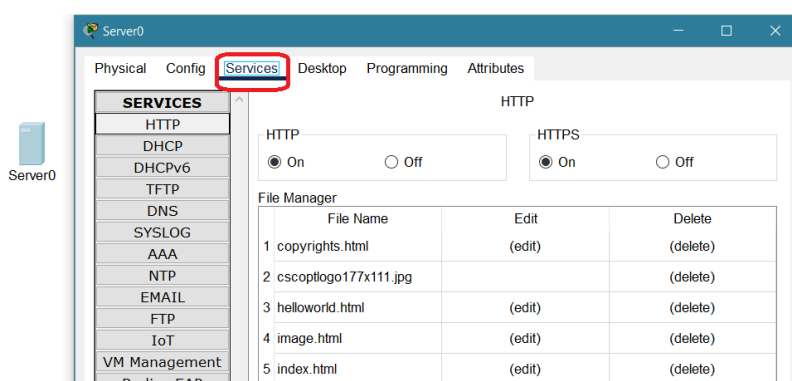
Obrázek 4.10: Desktop – „aplikace“ na PC, nastavení IP adresy v aplikaci

Na obrázku 4.10 je další záložka (Desktop), na které najdeme něco na způsob dlaždic pro různé aplikace. Uvnitř PC běží hodně osekáný systém, který funkčně připomíná Windows (i co se týče

podporovaných příkazů). Mezi vestavěnými aplikacemi najdeme například aplikaci pro konfiguraci IP adresy a brány (na rozdíl od předchozí možnosti zde máme vše v jednom okně a nemusíme přepínat mezi rozhraními a konfigurací brány), dále třeba webový prohlížeč nebo Příkazový řádek.

Zde jsme nakonfigurovali IPv4 adresu 10.0.0.12, masku 255.0.0.0, bránu 10.0.0.1, IPv6 adresu 2001:a:a::12/64, IPv6 bránu fe80::1. Na jiných PC a noteboocích bychom samozřejmě nastavili jiné IP adresy.

Základní konfiguraci máme za sebou, ale to není zdaleka všechno, co se dá ovlivňovat. Například na serveru se dají spustit či nastavit různé serverové služby – na záložce *Services*, případně můžeme jít opět přes záložku *Desktop*.



Obrázek 4.11: Služby na serveru v Packet Traceru



Úkol

Projděte si konfiguraci koncových zařízení: ať už běžného počítače, tak i serveru. Koncovému zařízení přiřadíte IP adresu, třeba tu podle příkladu. Všimněte si, jaké jsou MAC adresy zařízení.



4.3.3 Generování a monitorování provozu

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 jednotlivá zařízení a dále generovat nějaký provoz (simulovat činnost).

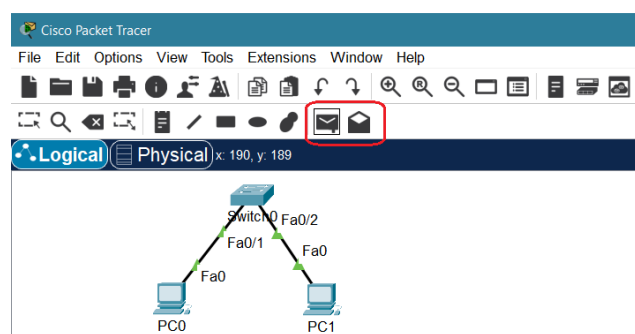


Příklad 4.6 (Generování provozu)

Podobně jako na fyzických zařízeních, i zde můžeme na koncovém zařízení použít příkaz `ping`. Nebo můžeme využít ikonu pro generování provozu:

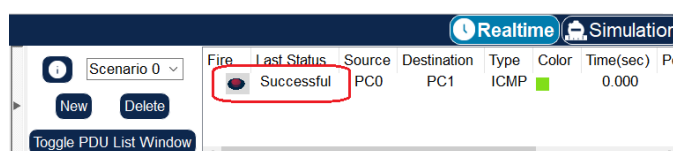
Ikona „zavřeného dopisu“ generuje jednoduchý ping, druhá ikona umožňuje vytvořit i provoz jiného typu. Nicméně nám ta první stačí.

Klepneme tedy na ikonu dopisu, pak na odesílající zařízení (u nás první PC) a pak na cílové zařízení (u nás druhé PC). Vpravo dole pak vidíme výsledek (viz obrázek 4.13). Pokud je tam napsáno „Successful“, pak ping dopadl dobře, cíl odpověděl. Kdyby náhodou ne, nemusí být na



Obrázek 4.12: Generování provozu v Packet Traceru

vině chybná konfigurace, ale tato zařízení si prostě musejí předem trochu víc popovídat a napoprvé potřebují víc času. Poklepáním na ikonu ve sloupci *Fire* spustíme opakování.

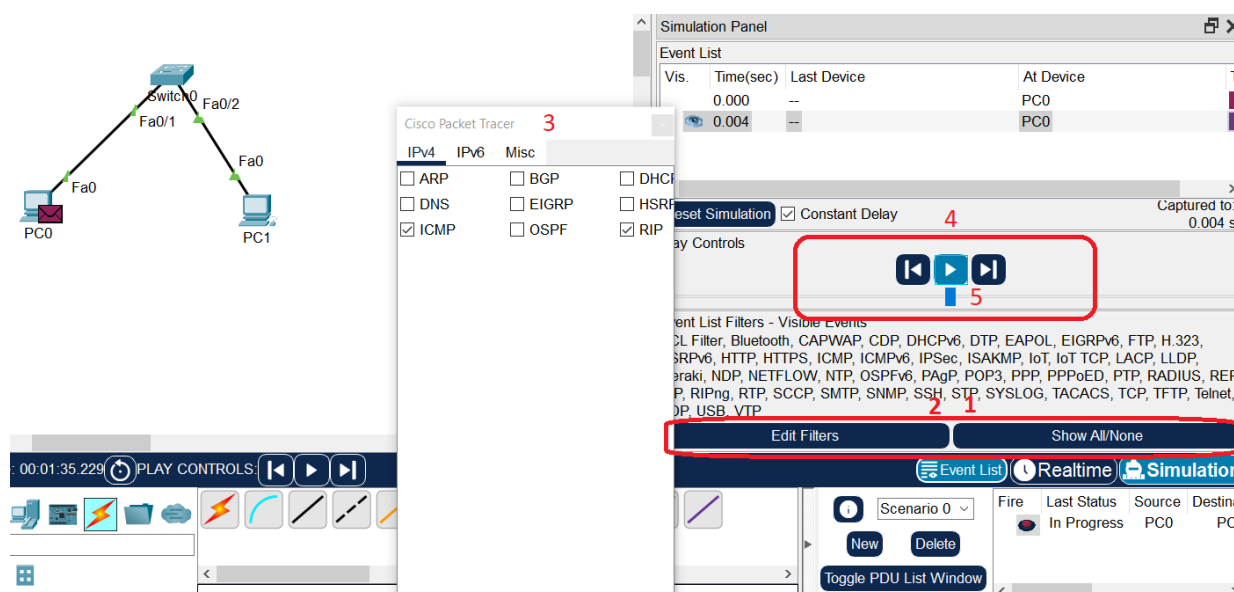


Obrázek 4.13: Výsledek generování provozu



Příklad 4.7 (Monitorování provozu na ethernetové síti)

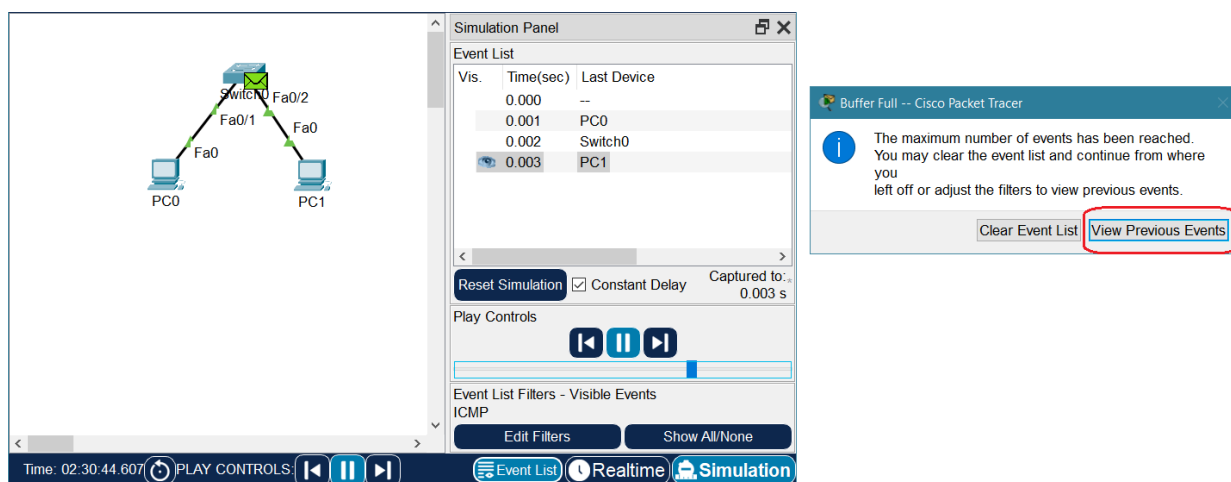
Abychom mohli podrobněji sledovat provoz na síti, musíme se přepnout do *simulačního režimu*. To provedeme v přepínači vpravo dole, viz obrázek vpravo.



Obrázek 4.14: Nastavení filtrů a spuštění simulace

Nejdřív nastavíme filtry, tedy určíme, jaké protokoly nás v simulaci zajímají. Pokud chceme sledovat pouze cestu jednoduchého pingu, vystačíme si s protokolem ICMP, ale samozřejmě mů-

žeme zvolit i jiné protokoly. Tlačítko *Show all/none* nám umožní vypnout vše, a pak přes tlačítko *Edit filters* zvolíme jen to, co chceme sledovat. Zde jsme zvolili pouze protokol ICMP.



Obrázek 4.15: Okno s informací o zaplnění bufferu při simulaci

Pak v poli *Play controls* klepneme na šipku, která spouští simulaci, posuvníkem pod těmito ovládacími tlačítky můžeme simulaci zrychlit či zpomalit. Opět můžeme využívat ikony s „dopisy“ pro generování provozu.

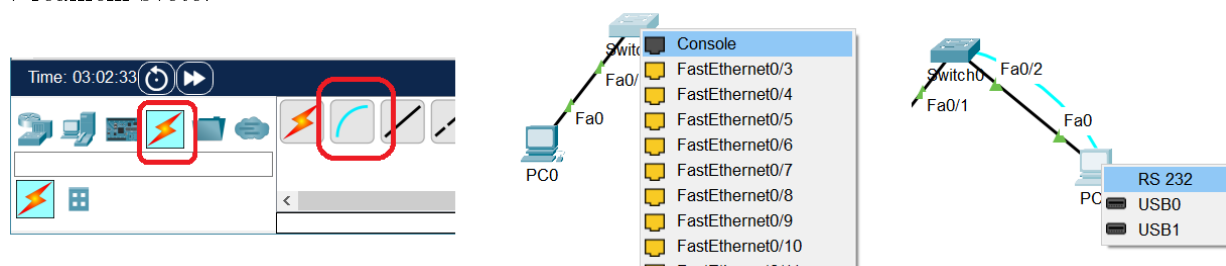
Může se stát (a taky se běžně stává), že simulace skončí předčasně, protože Packet Tracer má jen malý buffer pro zachycený provoz. Pak se objeví okno s touto informací, ve kterém stiskneme tlačítko *View Previous Events*.

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*.



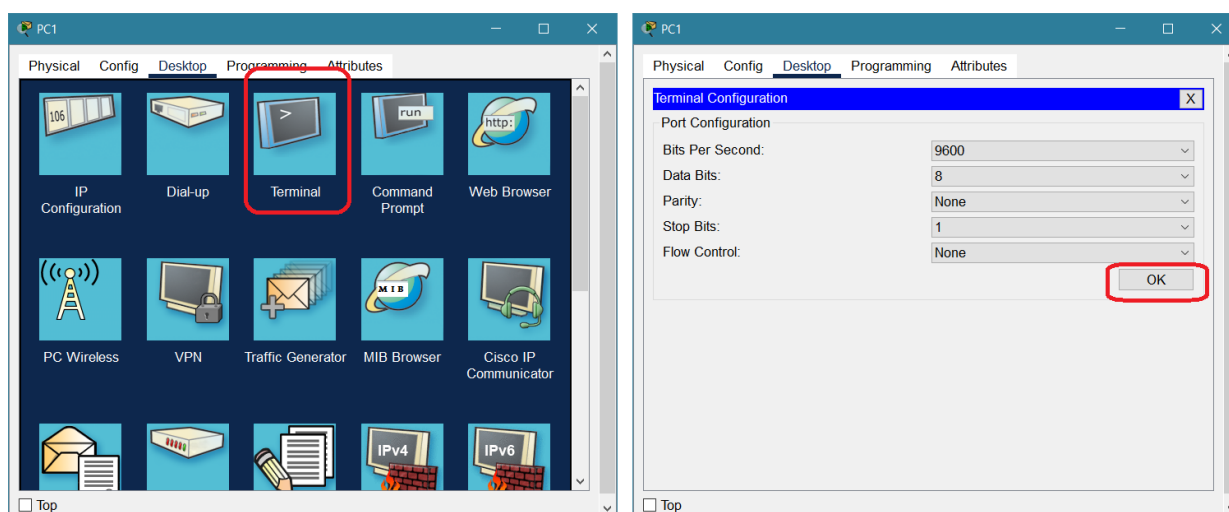
4.3.4 Jak na konfiguraci switchu a routeru

V Packet Traceru se můžeme k síťovému zařízení připojit pomocí konzolového kabelu stejně jako v reálném světě:



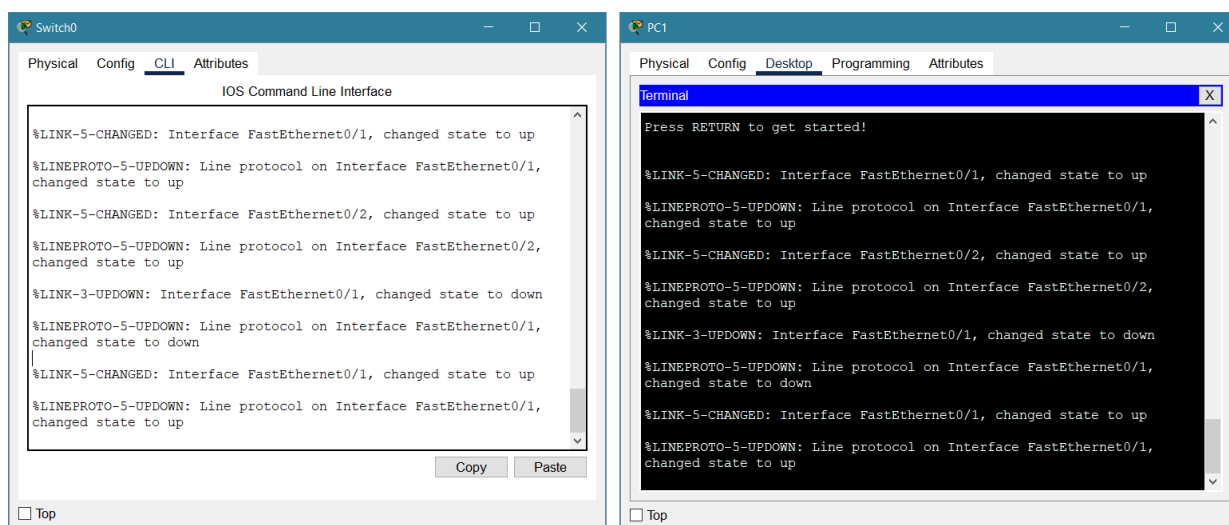
Obrázek 4.16: Propojení konzolovým kabelem v Packet Traceru

Vybereme si konzolový kabel, na straně routeru nebo switchu použijeme port označený „Console“, na straně počítače port RS-232. Pak na PC přejdeme na záložku *Desktop*, tam zvolíme *Terminal* a potvrdíme výchozí nastavení tlačítkem *OK*.



Obrázek 4.17: Přístup do terminálu/konzoly na PC

Ovšem Packet Tracer nám nabízí i jinou možnost, kterou v reálném světě nemáme: stačí klepnout na router či switch a přejít na záložku CLI. Výsledek je naprosto stejný (viz obrázek 4.18). Moc si na to nezvykejte, v reálném světě to nejde :-).



Obrázek 4.18: Okno terminálu zobrazené přes PC (vpravo) a klepnutím na switch (vlevo)

4.4 Konfigurace zařízení Cisco

Budeme předpokládat, že už jsme zvládli fyzickou stránku věci – v okně emulátoru terminálu vidíme výstup síťového zařízení a také tam můžeme zadávat vstup, tedy příkazy.

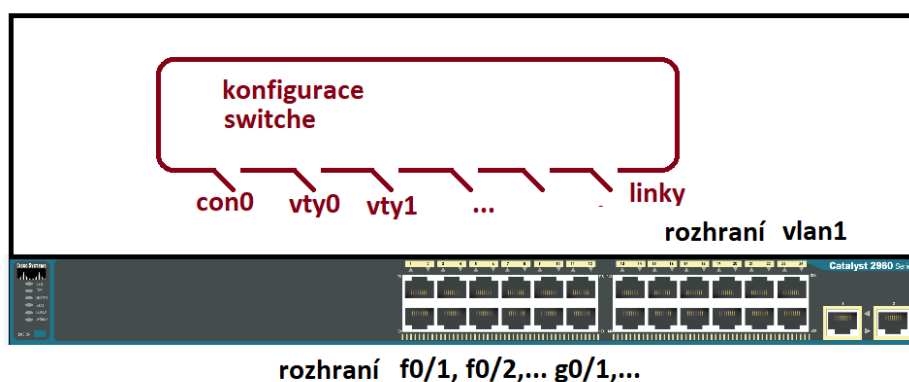
4.4.1 Linky console a VTY

Ať už jsme se k síťovému zařízení připojili jakkoliv, vždy se napojujeme na některou *linku*. Ta linka je buď konzola (console) v případě použití konzolového portu (přes konzolový kabel, včetně

možnosti využití konzolového serveru), 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 linky se propojují se síťovými porty (FastEthernetxxx, GigabitEthernetxxx apod.), podle toho, přes jaké rozhraní je síťové zařízení propojeno s okolním světem.

 Jak se tedy jednotlivé porty používají?

- Ke konzolovému portu se připojuje pouze konzolový kabel, tento port je uvnitř navázán na linku console a slouží pouze pro přístup za účelem konfigurace.
- K síťovým portům se připojuje některý typ ethernetového síťového kabelu, tyto porty slouží buď pro běžný provoz, nebo v případě použití protokolů telnet či SSH se uvnitř navážou na VTY linky a lze se takto dostat do konfigurace.



Obrázek 4.19: Přístup do konfigurace přes linky console a VTY

Na obrázku 4.19 si všimněte uvnitř switche (nebo routeru) uzavřené „místnosti“ zvané konfigurace, do které vede řada dveří a jiným způsobem se dovnitř nedá dostat. První dveře představují linku `console 0`, další je linka `vty 0`, následuje VTY 1, atd. První linka se váže na konzolový port, VTY linky se podle potřeby navazují na některé síťové rozhraní.

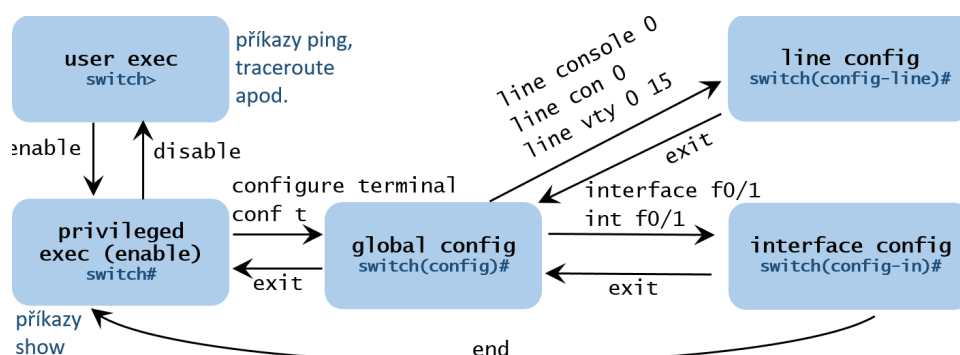
Routery většinou mají VTY linky 0–4, switche obvykle 0–15. Když konfigurujeme přístup na tyto linky, neměli bychom na žádnou zapomenout.

4.4.2 Pracovní módy

 Na zařízeních Cisco rozlišujeme několik různých pracovních módů (režimů). Na obrázku 4.20 jsou jak jednotlivé módy, tak i možnosti přechodů mezi nimi.

- *User exec* – základní režim, do kterého se dostaneme po připojení k zařízení přes některou linku (console nebo VTY) za účelem konfigurace. Prompt má formu názvu zařízení následovaného symbolem „větší“ (>). V tomto módu lze nanejvýš poslat `ping` či `traceroute`.
- *Privileged exec* (také mu říkáme *enabled*) – vyšší režim, ve kterém typicky zadáváme příkazy pro výpis (například výpis tabulky MAC adres, výpis nastavení SSH, výpis stavu rozhraní, atd.). 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. Dva nejpoužívanější typy subkonfigů jsou tyto:
 - subkonfigurační mód pro linku (buď konzolu nebo VTY) 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 můžeme rozhraní zapnout či vypnout, případně nastavujeme IP adresu.



Obrázek 4.20: Módy na zařízeních Cisco

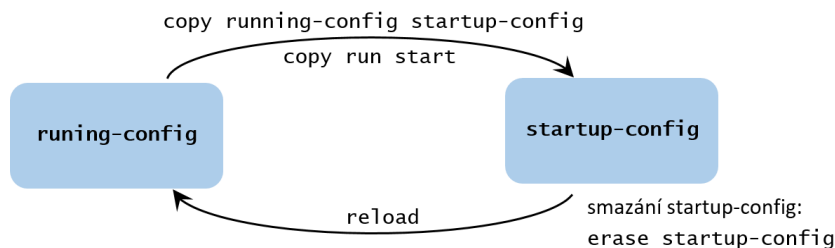
 Když provádíme změny, tyto změny se projeví pouze v běhové konfiguraci (*running-config*). To je RAM čip, v podstatě operační paměť, tedy po restartu zařízení by byly veškeré změny zahozeny. Proto se po provedení změn doporučuje uložit běhovou konfiguraci do místa, ze kterého se změny jen tak neztratí, 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 flash čip): firmware zařízení,
- *flash*: obsahuje obraz operačního systému IOS, který se používá na Cisco zařízeních (jednoduše soubor, ze kterého startuje systém) – ano, i síťová zařízení mají operační systém, ale určitě tam nenajdete Windows,
- *RAM* (běžná operační paměť): v této rychlé paměti, která se při restartu vymaže, si systém uchovává vše, k čemu potřebuje rychlý přístup, především soubor **running-config**, a také různé tabulky (tabulka MAC adres, směrovací tabulka, tabulka sousedů, atd.),
- *NVRAM* (non-volatile RAM, v reálu taktéž flash čip): v této části paměti, která si svůj obsah zachovává i po restartu, najdeme hlavně soubor **startup-config**, ve kterém máme uloženou konfiguraci určenou k tomu, aby se načetla při (re)startu zařízení.

Kdykoliv něco měníme v konfiguraci, tyto změny se projeví jen v souboru **running-config**, a obsah tohoto souboru se také používá za běhu systému (je to běhová konfigurace systému). Protože však chceme, aby se změny v konfiguraci při restartu neztratily, ukládáme je do souboru **startup-config**.

Obojí jsou soubory, a jako se soubory s nimi také můžeme zacházet. Můžeme je kopírovat, zálohovat přímo na zařízení nebo přes síť, ... Na obrázku 4.21 máme místa pro uložení konfigurace a přechody mezi nimi.



Obrázek 4.21: Uložení konfigurace na zařízeních Cisco

Na obrázku vidíme 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).

4.4.3 Základní 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 zabezpečení, alespoň hesla do enabled módu a na linky, také zajistit, aby hesla v konfiguraci nebyla snadno čitelná,
- 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.

 **Zabezpečení přístupu.** Zabezpečení nastavujeme na dvou místech: chráníme linky (konzoli a VTY linky), tedy přístup do *User exec módu*, a pak přístup do *Privileged exec módu*.

Existují tři stupně zabezpečení linek:

1. Ochrana heslem – aby se uživatel dostal přes tuto ochranu, musí zadat (sdílené) heslo. Je to nejjednodušší způsob, ale problém je, že všichni používají totéž heslo, nejsou rozlišení jednotliví uživatelé.

2. Lokální databáze uživatelů – na zařízení jsou v konfiguraci vytvořeni různí uživatelé, každý z nich má své heslo. Výhodou je, že můžeme rozlišit různé uživatele (dá se například zapnout logování, ve kterém je vidět, co kdo v konfiguraci „vyváděl“), ale pokud chceme provést změny v uživateli (některého odstranit, přidat nového, změnit heslo), musíme to provést zvlášť na jednotlivých zařízeních.
3. Autentizační server – v síti je jedno zařízení s databází uživatelů (to je autentizační server), ostatní zařízení se při žádosti o přihlášení tohoto serveru dotazují, jestli tento uživatel *může dovnitř* a případně s jakými oprávněními. Tento způsob je sice nejnáročnější (musíme mít nakonfigurovaný autentizační server), ale nejbezpečnější.

Enabled (Privileged exec) mód se typicky zabezpečuje heslem.



Příklad 4.8 (Název zařízení, zabezpečení linek a enabled módu)

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 linky. 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 linky 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**.



Příklad 4.9 (Jak se podívat do konfigurace)

Jak bylo výše řečeno, konfigurace je ve skutečnosti uložena v souborech **running-config** a **startup-config**. Jsou to soubory, tak si prostě můžeme nechat vypsat jejich obsah. Show příkazy se zadávají v Privileged exec módu:

```
S1#show running-config    takto si zobrazíme běhovou konfiguraci
S1#copy running-config startup-config    teď jsme uložili běhovou konfiguraci do té stálé
S1#show startup-config    takto si zobrazíme uloženou konfiguraci
```

Jestliže se právě nacházíme v některém konfiguračním módu, pak bychom se správně měli nejdříve přesunout do Privileged exec módu (nejrychleji to jde příkazem **end**, ale komu by se chtělo pořád přecházet mezi módy? Ve skutečnosti existuje způsob, jak zadávat **show** příkazy i v konfiguračních módech – stačí před takový příkaz přidat klíčové slovo **do**:

```
S1(config)#do show running-config    takto spustíme show příkaz patřící do enable módu i v
konfiguračním módu
```



Dalším krokem je dodatečné zabezpečení hesel. Ve výchozím nastavení je totiž řada hesel (na konzoli a VTY) uložena v souborech **running-config** a **startup-config** v textovém formátu bez šifrování a kdokoliv, kdo může vypsat konfiguraci, se k nim dostane. Když si na obrazovku vypíšeme konfiguraci, člověk stojící za naším ramenem (nebo kamera či program zaznamenávající snímky obrazovky) taktéž vidí hesla na linkách.

Řešením je spuštění služby **password-encryption**, která nahradí všechna uložená hesla jejich hashi a pak totéž provádí průběžně, kdykoliv se některé heslo mění nebo znovu nastavuje. Ve skutečnosti tato služba používá nepříliš bezpečný hashovací algoritmus, ale pořád je to lepší než nic.



Příklad 4.10 (Hashe hesel a banner)

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...“

 **Nastavení rozhraní, portů.** Abychom mohli k zařízení přistupovat vzdáleně přes Telnet nebo SSH, musí mít přiřazenou IP adresu (protože ta se zadává jako parametr dotyčného příkazu).

Příklad 4.11 (IP adresa a adresa brány, vypnutí nepoužívaných rozhraní)

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). V praxi se místo čísla 1 z bezpečnostních důvodů používá jiné číslo, ale k tomu se dostaneme později.

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`. Klíčové slovo `no` prostě znamená, že dané nastavení bude z konfigurace odstraněno. 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
```



Příklad 4.12 (Ověření konfigurace)

Teď by bylo fajn se podívat, jak jsme to všechno nakonfigurovali. K tomu slouží různé příkazy `show`. Protože `show` příkazy se mají zadávat v enabled módu, nikoliv v konfiguračních módech, tedy použijeme příkaz `end` (nebo několikrát příkaz `exit`, abychom se dostali do enabled módu. Jinak bychom museli před každým `show` příkazem napsat klíčové slovo `do`. 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 (mezer-níkem 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



 **Krácení příkazů.** 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
enable	ena
configure terminal	conf t
show ip interface brief	sh ip int br
show running-config	sh run
show startup-config	sh start
copy running-config startup-config	copy run start
ip address 10.0.0.2 255.0.0.0	ip addr 10.0.0.2 255.0.0.0
ipv6 address fe80::1 link-local	ipv6 addr fe80::1 link-lo

Tabulka 4.1: 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`.



4.4.4 Základní 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.



Příklad 4.13 (Základní nastavení routeru)



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

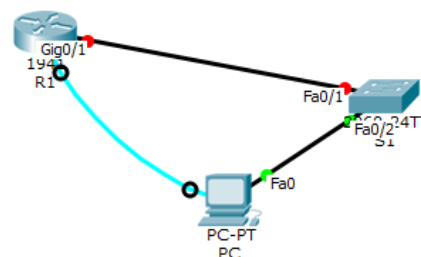
```
Router>enable
Router#configure terminal
Router(config)#hostname R1
R1(config)#enable secret heslo
R1(config)#line console 0
R1(config-line)#password heslo
R1(config-line)#login
R1(config-line)#logging synchronous
R1(config-line)#exit
R1(config)#line vty 0 4
R1(config-line)#password heslo
R1(config-line)#login
R1(config-line)#exit
R1(config)#service password-encryption
R1(config)#banner motd #Neautorizovany pristup prisne zakazan#
```

Teď 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 switchi), 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
```



Obrázek 4.22: Počítač připojený konzolovým kabelem k routeru



 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.).

Příklad 4.14 (IPv6 adresy na routeru)

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
```

Příklad 4.15 (Směrování)

Na routeru nás zajímá ještě jedna věc – směrovací tabulka. Následující příkazy nám ukazují, jak pracujeme se směrovací tabulkou (zatím jenom základ):

```
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 (zadááme 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
```

Kapitola 5

Lokální síť – Ethernet

 *Rychlý náhled:* Z přednášek bychom už měli vědět, co to jsou lokální síť, především síť Ethernet. Zde se seznámíme s nástroji, které slouží k jejímu monitorování (zejména s programem Wireshark), pak 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.

 *Klíčová slova:* Wireshark, rámec Ethernet II, LLC, SNAP, 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.

 *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.

5.1 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,

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.



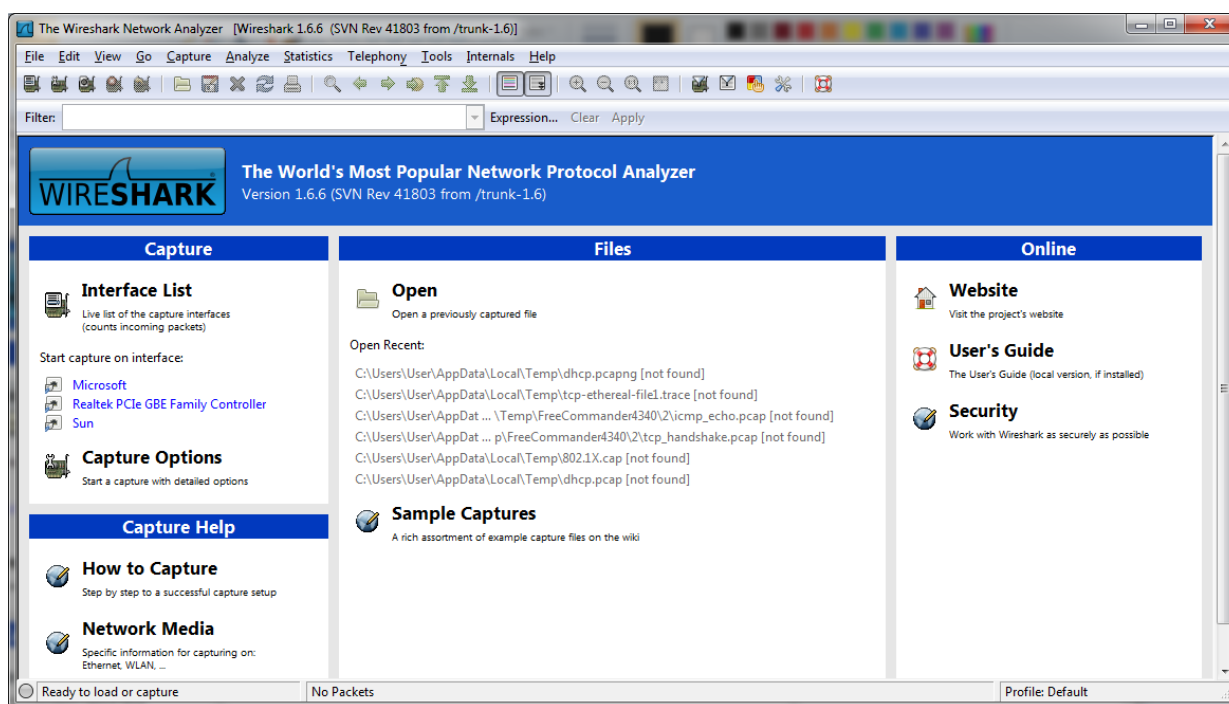
My se zde zaměříme na program *Wireshark*. Jeho výhodou je snadnost použití a přehlednost, podpora velkého množství formátů datových jednotek, a také volná dostupnost pro všechny běžné platformy (Windows, Linux, MacOS X, různé Unixy). Dá se říct, že Wireshark je jedním z nejpoužívanějších nástrojů tohoto 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 5.1: Prostředí aplikace Wireshark

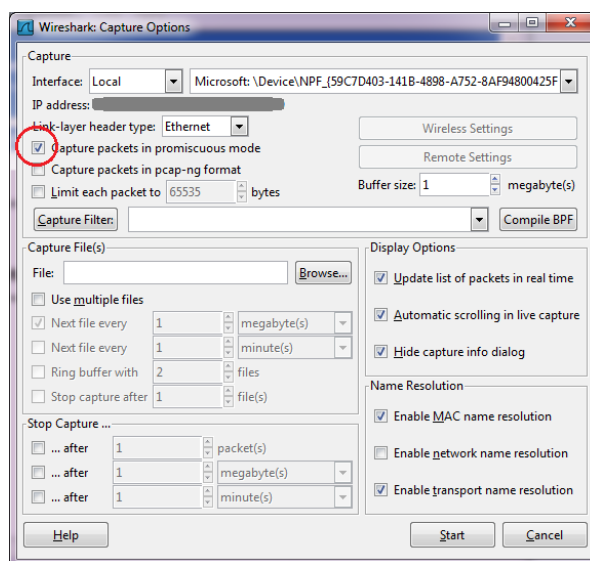
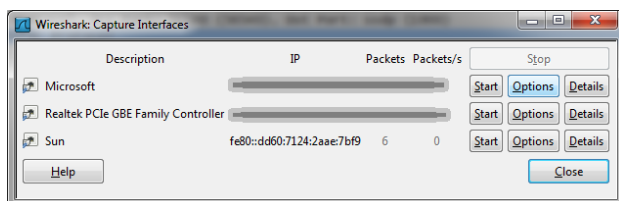


Postup (Používání Wiresharku)

Předpokládejme, že již máme Wireshark nainstalovaný a zprovozněný. Okno programu ve Windows je na obrázku 5.1.

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 5.2: 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.



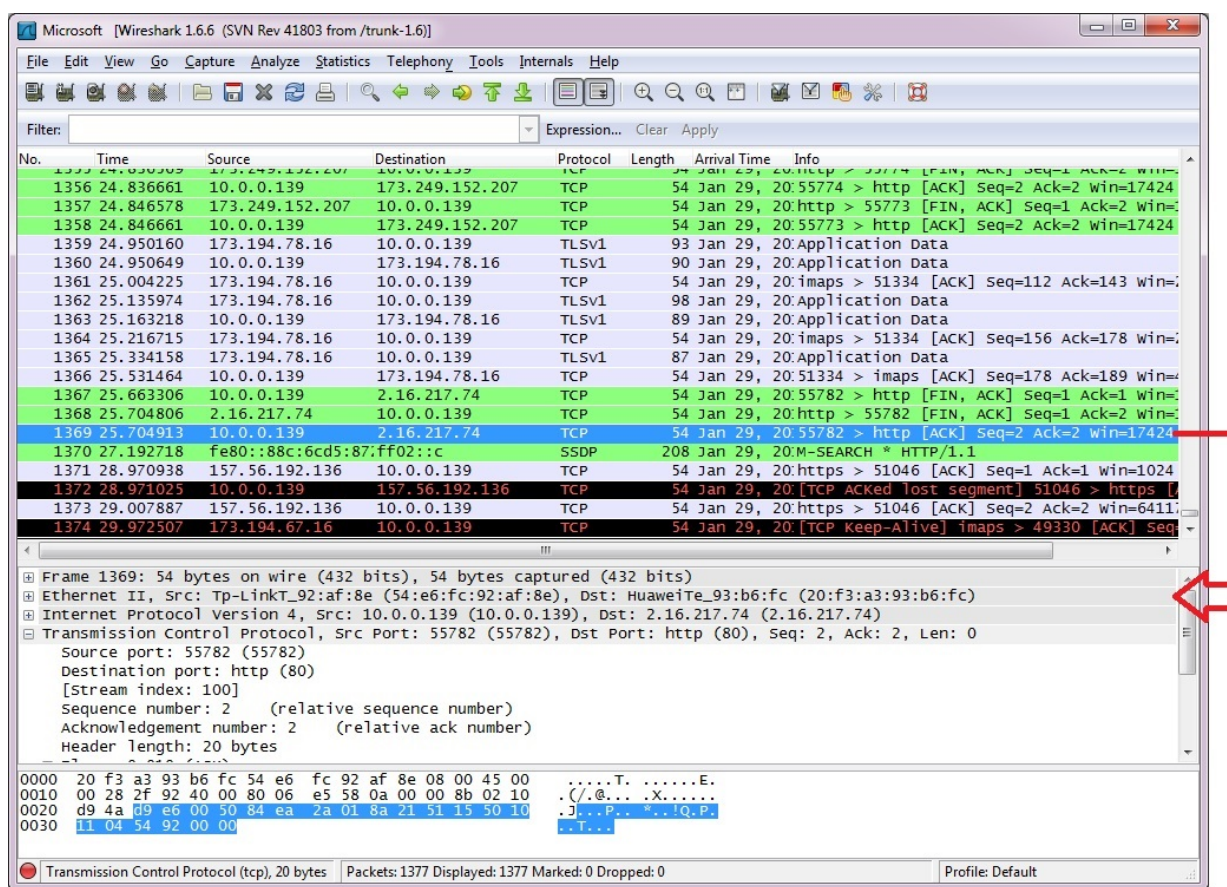
Příklad 5.1 (Zachycený rámec Ethernet II)

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 5.3 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



Obrázek 5.3: Provoz zachycený Wiresharkem

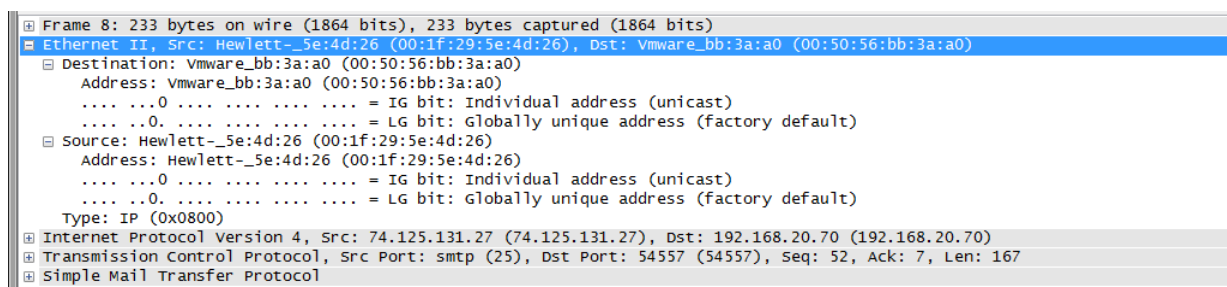
další část okna, ve které vidíme údaje o vybraném paketu. Ve spodní části okna je pak „rozbalená“ informace, která byla v paketu nalezena. Pokud je obsah šifrovaný nebo binární, nebude text srozumitelný.

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):

8 oktetů	6	6	2	46–1500	4
Preamble	DA: cílová adresa	SA: zdrojová adresa	Ether Type	SDU ze síťové vrstvy	FCS

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:



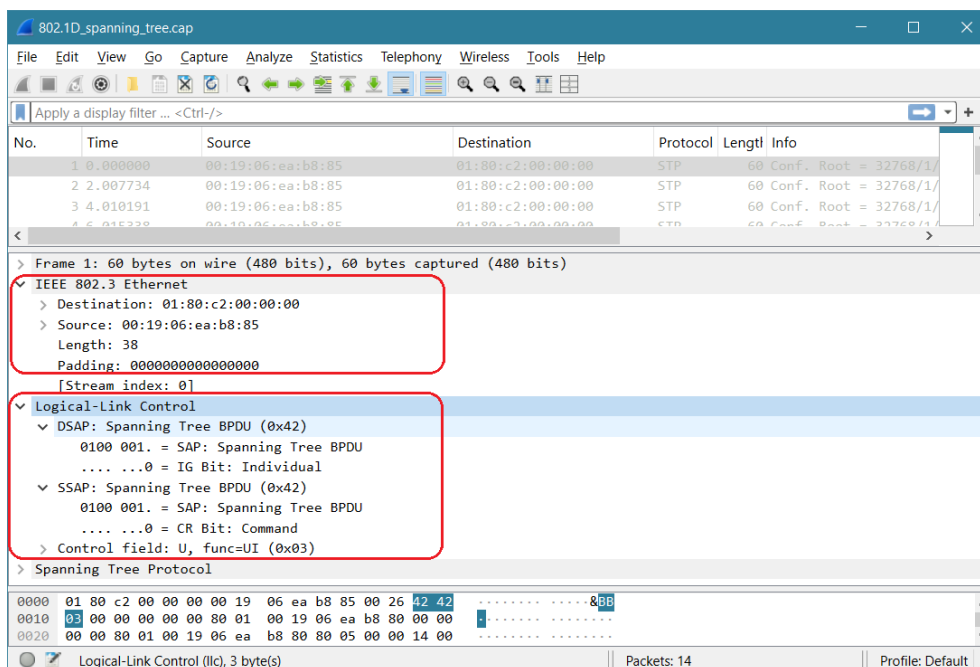
Nejdřív je uvedena adresa cíle (Destination Address) a hned za ní adresa zdroje (Source Address). Po rozbalení jejich vnitřních podstromů jsou přehledně zvýrazněny pozice speciálních bitů I/G (Individual/Group) a L/G (Local/Global). U obou adres jsou oba tyto bity nastaveny na nulu, tedy se jedná o individuální 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 0x0800 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.



Příklad 5.2 (Zachycený LLC rámec)

Prozkoumali jsme ethernetový rámec formovaný podle standardu Ethernet II, zbývají další dva typy – LLC rámec (IEEE 802.2) a SNAP rámec.



Obrázek 5.4: LLC rámec, ve kterém je zapouzdřen rámec protokolu STP

Na obrázku 5.4 je v okně Wiresharku zachycen rámec typu LLC, ve kterém je zapouzdřen provoz protokolu STP (spanning-tree). Se samotným protokolem STP se setkáme později, nyní se budeme věnovat záhlaví LLC. Jak vidíme, v LLC záhlaví jsou tři pole: DSAP, SSAP a řídicí pole. Pole DSAP a SSAP mají délku 8 bitů, z nichž 7 bitů určuje číslo zapouzdřeného protokolu,

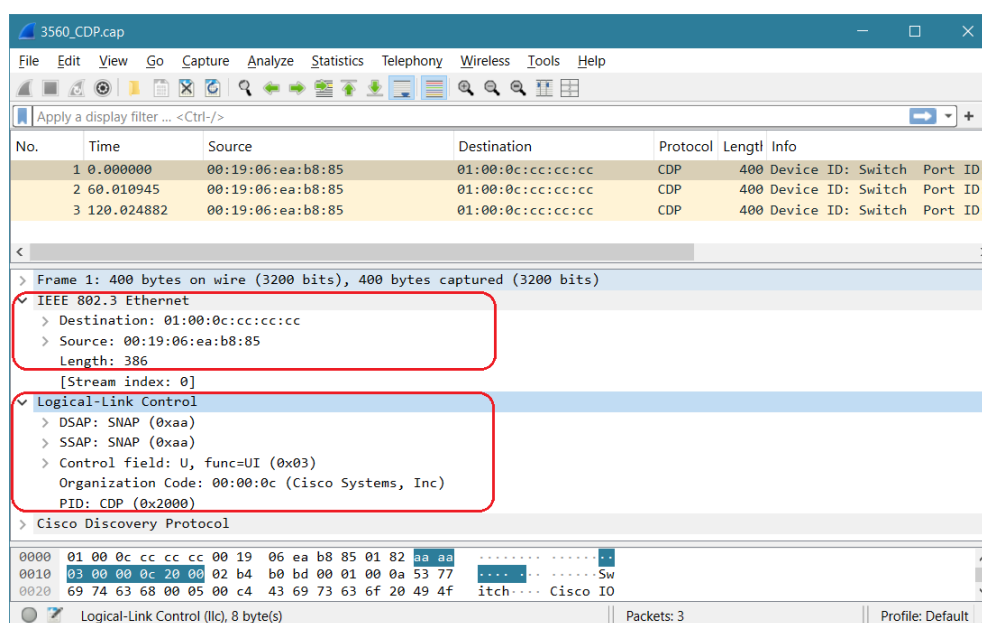
poslední má sice speciální význam, ale v Ethernetu je vždy nastaven na 0. Celých 8 bitů má hodnotu hexadecimálně 0×42 , což je číslo protokolu STP.

LLC rámec je zapouzdřen v rámci protokolu IEEE 802.3, protože LLC „umí“ jen horní podvrstvu vrstvy L2 a tedy nemá místo pro MAC adresy. V záhlaví IEEE 802.3 máme cílovou a zdrojovou MAC adresu, délku a pak „vycpávku“ (padding), protože pro velikost dat zapouzdřených v rámci existuje i spodní hranice a zrovna tento rámec by bez paddingu byl „pod míru“.



Příklad 5.3 (Zachycený SNAP rámec)

Jak už víme, SNAP rámce se většinou používají pro proprietární obsah, tedy to, co není standardizováno. Výrobci síťových zařízení mívají i své vlastní protokoly, pro které neexistuje přidělené číslo EtherType ani SAP, takže předchozí dvě možnosti pro ně nejsou použitelné.



Obrázek 5.5: SNAP rámec s daty protokolu CDP

Na obrázku 5.5 je SNAP rámec se zapouzdřenými daty proprietárního protokolu CDP od Cisca. SNAP záhlaví začíná stejně jako LLC, ale v LLC polích jsou hodnoty určující SNAP rámec. Následují dvě přidaná pole: OUI s identifikátorem výrobce (zde 00-00-0c znamenající společnost Cisco) a polem s číslem, které Cisco určilo pro protokol CDP. Celý SNAP rámec je taktéž zapouzdřen v rámci IEEE 802.3.



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:**

Wireshark User's Guide je dostupná na
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>
- <https://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 najděte *View – Coloring Rules*. Prohlédněte si momentální nastavení.



5.2 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 dotýčný adresát do té tabulky dostane?



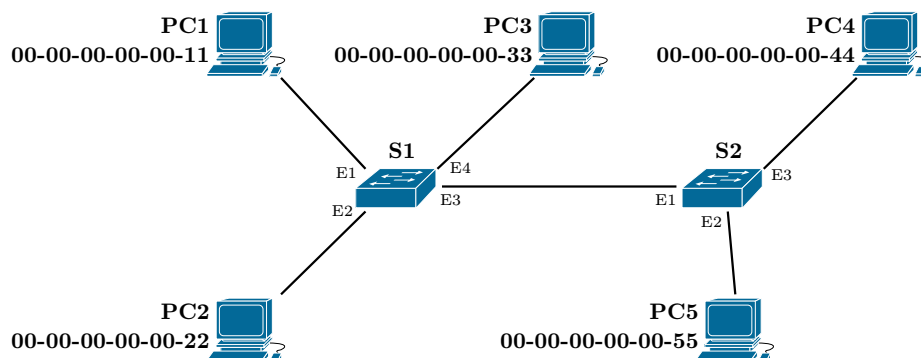
Předpokládejme, že máme nový (nebo restartovaný) switch a poprvé ho zapojíme, tabulka je prázdná. V takovém případě jsou dvě možnosti, jak tabulku naplnit. Buď jsou do ní příslušné údaje „natvrdo“ přidány pomocí ruční konfigurace (či pomocí skriptu), nebo to jednoduše necháme na dotýčném switchi, ať 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 dotýčné (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 5.4

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. Zkratka DA znamená Destination Address (cílová adresa), zkratka SA je Source Address (zdrojová adresa). Všimněte si, že flooding (tedy přeposlání na všechny porty kromě zdrojového) se ze začátku provádí poměrně často.



Následuje tento provoz, 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	...

cílovou 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.

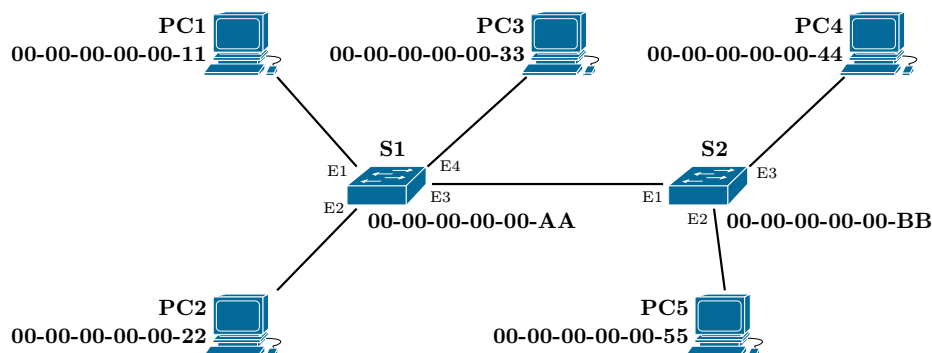
Switche mezi sebou komunikují například při použití protokolu STP, jehož účelem je zabránit logickým smyčkám v grafu switchů, aby nedocházelo k zacyklování provozu. STP je jedním z protokolů vrstvy L2, přičemž je zapouzdřován do LLC rámce (IEEE 802.2) a následně do MAC rámce IEEE 802.3.



Úkol

Zaměřte se na následující obrázek. Předpokládejte, že oba switche mají prázdné MAC tabulky a podle následujícího popisu provozu tyto tabulky postupně sestavte (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 switche může být buď koncové zařízení, nebo hub či switch, ke kterému bývá připojeno více dalších zařízení: právě jejich adresy budou k tomuto portu přiřazeny.



 Doplnňování 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.

 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:

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



5.3 Kabely

5.3.1 Metalická kabeláž – kroucená dvojlinka

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).

 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.

 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

 U kabelů tohoto typu se rozlišují různé 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.

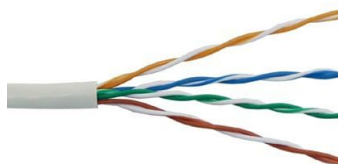
Novější standardy (vyšší kategorie) mají obvykle přísnější požadavky na kategorii použitého kabelu. V následující tabulce je přehled kategorií.

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. Šířka pásma je 100 MHz.	ano
Cat.6	Pro Gigabit Ethernet a vyšší; pro 10G omezen na cca 55 m. Šířka pásma 250 MHz.	ano
Cat.6A	Podobně jako Cat.6, ale má šířku pásma 500 MHz 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 (600 MHz).	ano
Cat.7A	Další vylepšení, ještě větší šířka pásma (1000 MHz).	ano
Cat.8.1, 8.2	Pro datová centra, na krátké vzdálenosti mezi switchem a serverem	ano

Tabulka 5.1: Kategorie kabelů typu kroucená dvojlinka

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 jsou méně běžné. Pokud někdo investuje do stíněného kabelu, obvykle raději vybere vyšší kategorii i kvůli případnému budoucímu přechodu sítě na vyšší standardy.

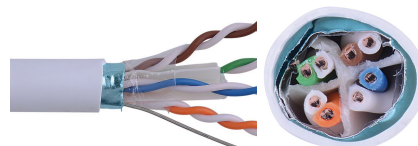
 Z výše uvedených je momentálně nejpoužívanější UTP (nestíněný) kabel kategorie 5e.



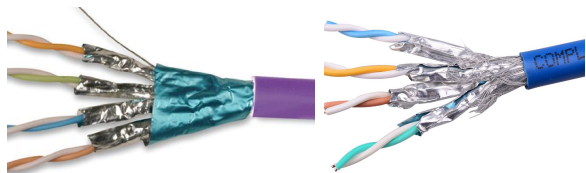
Vlevo je běžný kabel UTP Cat.5e. Vidíme 4 páry (tedy 8 vodičů), každý pár je do sebe zakroucený, díky čemuž je potlačeno vzájemné rušení. Pro odeslání signálu je určen vždy celý pár, signál je reprezentován rozdílem potenciálů obou vodičů.

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 je 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 plastový oddělovač,
- drát/licna – kabely typu drát jsou určeny na pevné instalace (do zdi), kdežto licna (lanko) jsou pružnější kabely pro připojení přímo k zařízení (na rozdíl mezi nimi se podíváme později),
- 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

- <https://www.intelek.cz>
- <https://www.softcom.cz>
- <https://www.bscom.cz/sitove-prvky-c-12376/>



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. Všímejte 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).



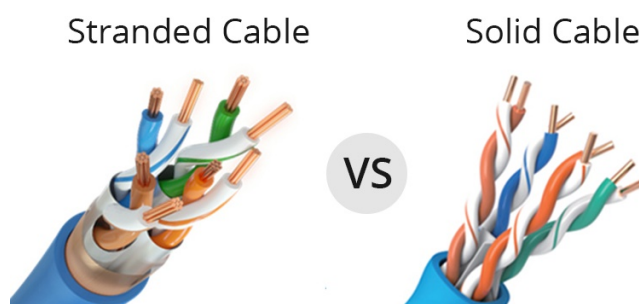
5.3.2 Vodiče v TP kabelech

Kabely typu kroucená dvojlinka (twisted pair, TP) se vyskytují ve dvou typech: typ licna a typ drát. Rozdíl mezi těmito dvěma typy ukazuje obrázek 5.6.

¹Zdroj: <https://community.fs.com/article/wiring-your-home-network-a-comprehensive-guide.html>

 *Kabel typu lanko* (licna, stranded) – vodivé jádro vodičů je ze svazku smotaných tenkých měděných drátů obalených izolací, má větší elektrický odpor. Kabel je pružnější, lépe se ohýbá.

 *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 a může být mnohem delší, protože má menší odpor.



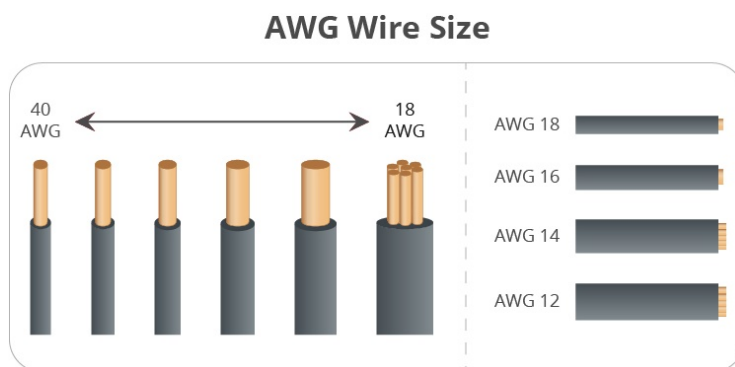
Obrázek 5.6: Rozdíl mezi kabelem typu licna a drát¹

Poznámka:

Pokud ve specifikaci či přímo na kabelu objevíte zkratku CCA (Copper Clad Aluminium), raději 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.



 U kabelů může být důležité, jak jsou vodiče v kabelu silné. Ve specifikaci kabelu můžeme vidět zkratku AWG (American Wire Gauge). Může být trochu matoucí, že čím menší číslo, tím silnější vodič: vodiče 24AWG jsou silnější než vodiče 28AWG. Srovnání několika běžných průměrů vidíme na obrázku 5.7.



Obrázek 5.7: Ukázka hodnot AWG pro TP kabely²

Číslo AWG bylo původně míněno pro vodiče typu drát. V případě vodičů typu licna se udává jako hodnota odpovídající průměru celého svazku drátků tvořících vodič, případně se můžeme setkat s posloupností čísel (AWG celého svazku, počet drátků ve svazku, AWG jednotlivých drátků ve svazku).

Příklad 5.5

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...”

²Zdroj: <https://community.fs.com/article/ethernet-cable-gauge-awg-introduction.html>

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.

Pokud se jedná o kabel typu licna, můžeme se setkat například s touto posloupností:

„...23AWG 5/30“

To znamená, že celý svazek tvořící vodič má hodnotu AWG 23, ve svazku je vždy 5 drátků, každý z těchto drátků má AWG o hodnotě 30.



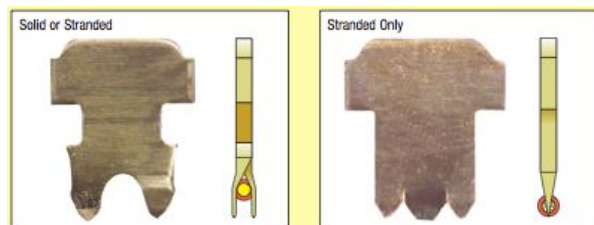
Další informace:

- <https://www.techtarget.com/searchnetworking/definition/American-Wire-Gauge>
- <https://community.fs.com/article/ethernet-cable-gauge-awg-introduction.html>
- <https://www.motorobit.com/awg-mm2-conversion-table-awg-current-table>



Úkol

Kabely se dají koupit také jako metráž, hovorově „klubo“ – bez konektorů a v délce několika set metrů. V některém internetovém obchodě (například na <https://www.intelek.cz>, <https://www.wifishop.cz/>, <https://www.tsbohemia.cz/> nebo <https://www.alza.cz/>) najdete kategorii pro takové kabely a ověřte si, kolik přibližně stojí 305 m nestíněné kroucené dvojlinky kategorie 5e jednoho i druhého typu.



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



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

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



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 zkroucený 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.



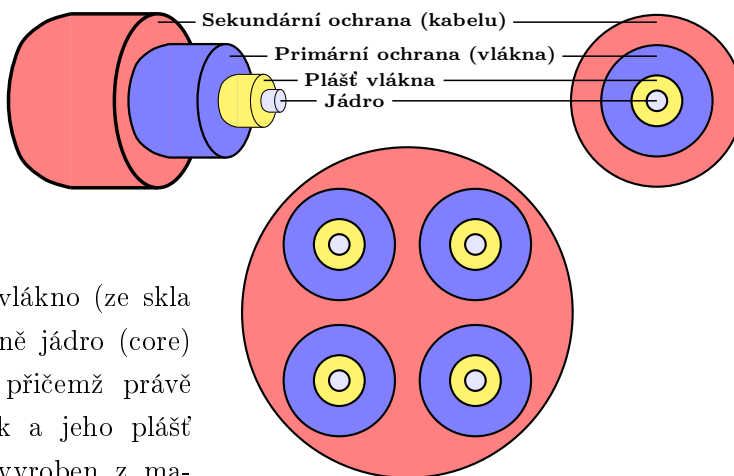
5.3.3 Optická kabeláž

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 a rychlosti).

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 doveden 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 laboratorním případě poletí fotony pořád rovně bez odrazů).

Existují dva druhy optických vláken:

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

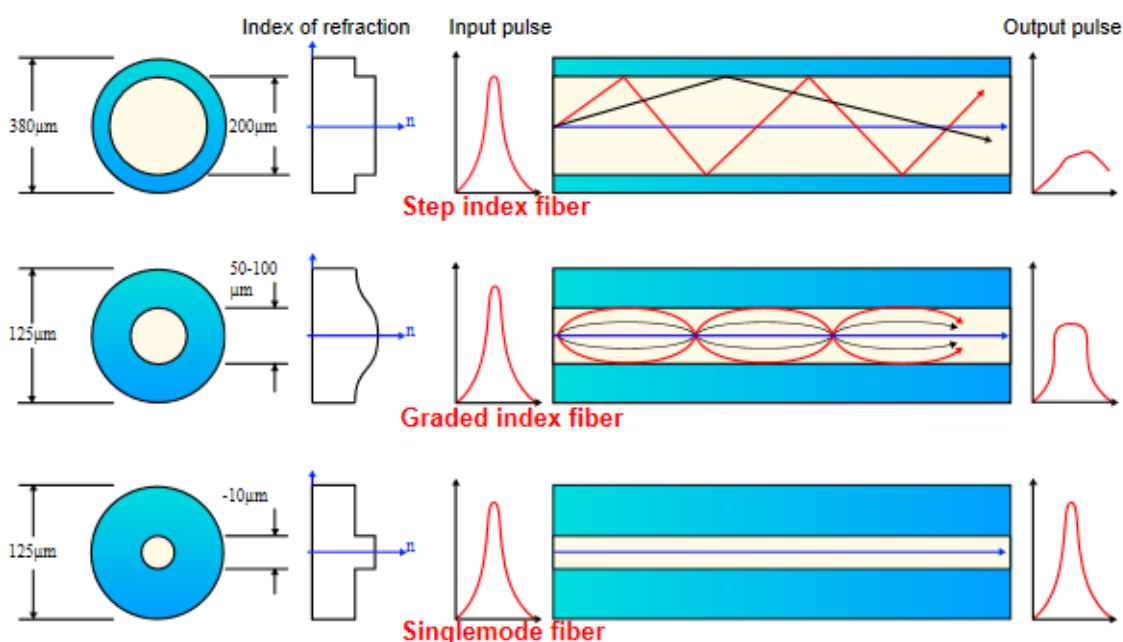
- *mnohavidové vlákno* (vícevidové, multi-mode fiber, MM) má větší průměr jádra, na kvalitu zdroje světla jsou menší požadavky, je levnější, jeho efektivní dosah je spíše menší (stovky metrů).

Liší se také rozsahy vlnových délek přenášeného signálu, jednovidová vlákna používají vyšší vlnové délky.

 V mnohavidových 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. Dnes už se s nimi moc nesetkáme.

Na obrázku 5.8 je naznačen průchod signálu v obou typech mnohavidových vláken a v jednovidovém vláknu. Neoptimálněji je signál převáděn v jednovidovém vláknu, následuje mnohavidové vlákno s gradientní změnou indexu lomu.



Obrázek 5.8: Různé druhy optických vláken³

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.



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,

³Zdroj: <https://reggle.wordpress.com/2013/02/03/osi-layer-1-part-ii-fiber/>

- 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), mnohavidové 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, mnohavidové vlákno, dosah desítky metrů až 400 m podle kvality vlákna (viz dále),
 - 10GBase-LR – laser 1310 nm, jednovidové vlákno, dosah 10 km (v praxi až 25 km),
 - 10GBase-LRM – laser 1310 nm, mnohavidové vlákno, dosah 220 m,
 - 10GBase-ER – laser 1550 nm, jednovidové vlákno, 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 mnohavidová optická vlákna používají tato označení:

- OM1 – vlákno 62,5/125, obvykle pro LED diodové zdroje signálu, max. rychlost 100 Mb/s,
- OM2 – vlákno 50/125, obvykle pro LED diodové zdroje signálu, do rychlosti 1 Gb/s (s omezením vzdálenosti na 82 m rychlost 10 Gb/s),
- OM3 – vlákno 50/125, obvykle pro laserové zdroje signálu, do rychlosti 10 Gb/s (vyšší rychlosti s podstatným omezením vzdálenosti),
- OM4 – vlákno 50/125, laserový zdroj, do rychlosti 40 nebo 100 Gb/s (do vzdálenosti 150 m),
- OM5 – vlákno opět 50/125, laserový zdroj, je určeno pro přenosy s více subkanály (WDM – obdoba frekvenčního multiplexu, ale na vlnových délkách).

Kabely kategorie OM1 a OM2 bývají oranžové (tím je myšlena vnější bužírka u patch kabelů, někdy i pigtailů – viz dále), OM3 a OM4 zase světle modré (aqua), OM5 bývá světle zelený (lime green).

Single-mode vlákna mají také své kategorie, ale méně. OS1 a OS2 jsou jednovířková vlákna 9/125, zdrojem je laser. Rozdíl mezi nimi je například v dosahu (OS2 mají při dané rychlosti o něco vyšší dosah). Single-mode kabely bývají žluté.

Pokud je v kabelu více vláken, pak jsou jednotlivá vlákna taktéž odlišena barvami, přičemž každá barva odpovídá určitému číslu. Například modrá určuje vlákno číslo 1, oranžová vlákno číslo 2, atd. Je to důležité proto, že každé z vláken musí „někam“ ústít a při jejich osazování a napojování do infrastruktury nesmí dojít k záměně.



Poznámka:

Výše uvedené informace nemusíte znát všechny zpaměti, stačí vědět, že existují různé typy optických vláken (když uvidíte například MM OM4, tak aby bylo jasné, že jde o jednu z kategorií multi-mode vláken a „spíše lepší“ a že se liší také barvami. Je dobré vědět, že žlutá indikuje single-mode vlákno.



Další informace:

Kabely vedené pod zemí (páteřní, ale i ty vedoucí na místo určení – last mile) se dnes obvykle do příslušných vodících prvků, trubiček, zafukují. K tomu se používají speciální prostředky: zafukovačky s kompresorem. Kabel je do trubičky vtlačován proudem stlačeného vzduchu. Podrobný postup najdeme například na https://www.alternetivo.cz/default.asp?inc=inc/info/b2btechn_info_zafukovani.htm



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. Všímejte si výše uvedených parametrů (OM1, ..., OS, průměr jádra/průměr pláště apod.) a také počtu vláken v kabelu. Uvědomte si, že pro každý směr potřebujeme minimálně jedno vlákno.



Další informace:

- <https://community.fs.com/blog/advantages-and-disadvantages-of-multimode-fiber.html>
- <https://www.thefoa.org/tech/ColCodes.htm>

- <https://www.root.cz/clanky/opticke-kabely-v-oceanu-tudy-doopravdy-tece-internet/>
- <http://fyzika.jreichl.com/main.article/view/557-opticka-vlakna>



5.3.4 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.

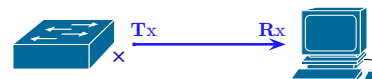


5.3.5 Křížení

Zařízení na svém síťovém rozhraní vlastně provádí dvě základní operace:

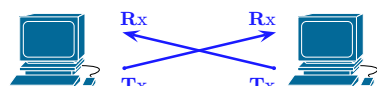
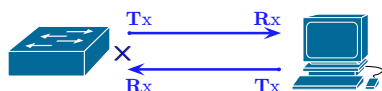
- Tx (Transmit) – odesílá,
- Rx (Receive) – přijímá (naslouchá a když zjistí, že druhá strana vysílá, vysílání přijme).

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 tímtež párem vodičů.

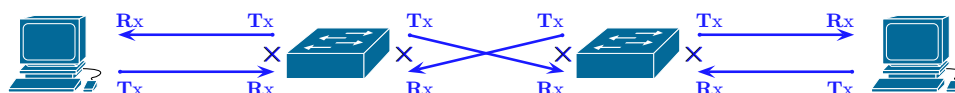


Poznámka:

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



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 5.9 vidíme případy, kdy jsou na cestě mezi-
lehlá zařízení, na portech těchto DCE je taktéž zajištěno křížení.



Obrázek 5.9: Princip křížení na kroucené dvojlince



Poznámka:

Ta křížení, která vidíme na obrázku 5.9 (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) 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 5.6

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 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 Ethernetu je router vlastně DTE, tedy koncové zařízení (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:	Switch	Hostitel	
Switch	×	—	— přímý kabel
Hostitel	—	×	× křížený kabel

Tabulka 5.2: 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 nasazené stejně (vodiče jsou na obou koncích ve stejném pořadí), kdežto křížený kabel má na jednom konci vodiče jinak seřazené než na druhém konci.



V současné době (u 100Mb Ethernetu a rychlejšího) máme situaci jednodušší – novější síťová rozhraní dokážou automaticky rozpoznat, jestli mají nebo namají zajistit vnitřní křížení, takže ve většině případů můžeme prostě použít přímý kabel. Ethernetové rozhraní tedy může být:

- MDI (Medium Dependent Interface) – bývá na koncových zařízeních (z pohledu Ethernetu), tedy DTE, nemá vnitřní křížení,
- MDIX (Medium Dependent Interface with crossover) – bývá na switchích, má vnitřní křížení.

Navíc switche obvykle na jednotlivých rozhraních podporují funkci *Auto-MDIX*. Dva switche se během propojování přímým kabelem (tedy tím, co tam teoreticky nemá co dělat) dohodnou, který z nich si křížení na rozhraní vypne, čímž se kompenzuje přítomnost křížení v kabelu. Pokud je tato funkce zapnutá, pak klidně můžeme použít přímý kabel i tam, kde bychom teoreticky měli použít křížený, ale pokud je vypnutá, musíme se držet standardu a použít „správný“ kabel.



Poznámka:

Sice se dá většinou na funkci Auto-MDIX spolehnout, ale za určitých okolností je přece jen lepší použít typ kabelu podle standardu (tedy někdy i ten křížený). Funkce Auto-MDIX může totiž být omylem nebo ve zlém úmyslu někým vypnuta, nebo může dojít k poruše rozhraní.



5.4 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.

5.4.1 Krimpování konektoru na přímý UTP kabel

 Postup nasazení konektoru na kabel se nazývá *krimpování* nebo *konektorování*. Na kabel UTP Cat.5e 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 nebo 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,
- 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.



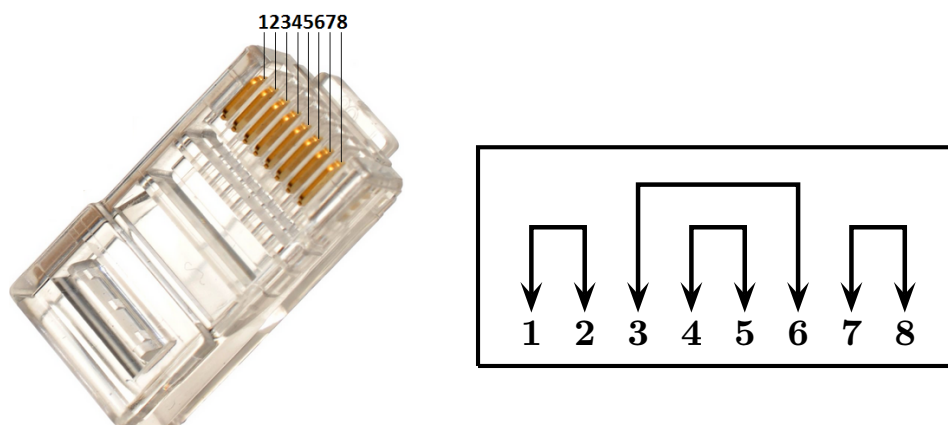
**Úkol**

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
- <https://www.conrad.cz>: Kleště – Krimpovací kleště
- <https://www.intelek.cz>: Metalická kabeláž – Nářadí a vyvazování – Kleště

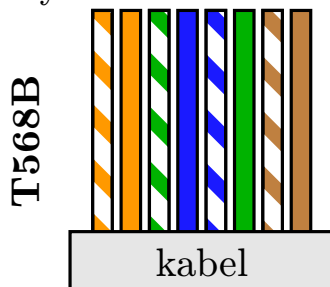


Piny jsou v konektoru v řadě, jejich čísla vidíme na obrázku 5.10 vlevo (konektor je natočen tak, že se díváme na piny). Vpravo je naznačeno, jak k sobě jednotlivé piny patří do dvojic. Ke každé dvojici pinů (podle obrázku) patří jeden pár vodičů, přičemž z těchto dvou vodičů je jeden v plné barvě, druhý v téže barvě, ale s bílým pruhem.



Obrázek 5.10: Pořadí a párování pinů v konektoru

piny: **1 2 3 4 5 6 7 8**



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 5.10 (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.

Mnemotechnická pomůcka: uprostřed teče řeka, kolem ní je zelený porost (má dost vody), dále je suchá půda (oranžová nebo hnědá).



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 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)

Teď 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 teď (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, teď ji nasuneme na konektor.

Pozor – po použití krimpovacích kleští už konektor nepůjde vysunout. Pokud dodatečně zjistíme, že je někde chyba, nezbyvá 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.



5.4.2 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 5.11: Testery pro ethernetové kabely (Zdroj: heureka.cz)

Na obrázku 5.11 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:

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



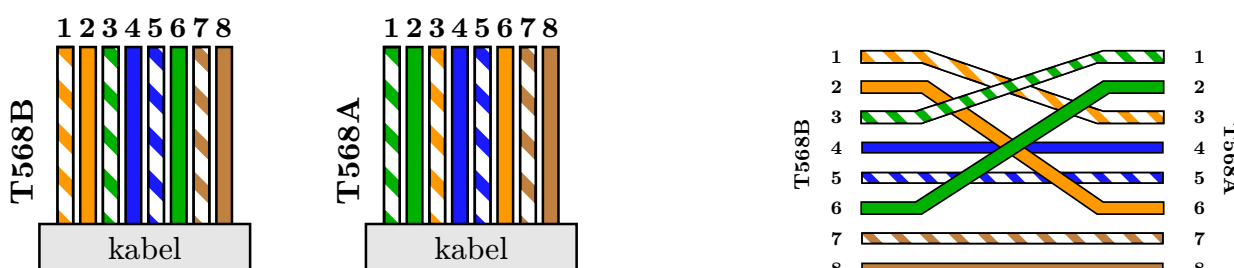
 Tester obvykle taky potřebuje napájení. Jestliže je jeden konec kabelu připojen ke spuštěnému switchi, pak může (nemusí) být napájení zajištěno switchem, ale pokud jsou oba konce kabelu v testeru, pak nutně potřebujeme baterii (typ baterie je uveden v dokumentaci). Typ a kvalita napájecího rozhraní taky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

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

5.4.3 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 5.12: Barevný kód pro křížený kabel – Fast Ethernet

Na obrázku 5.12 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.

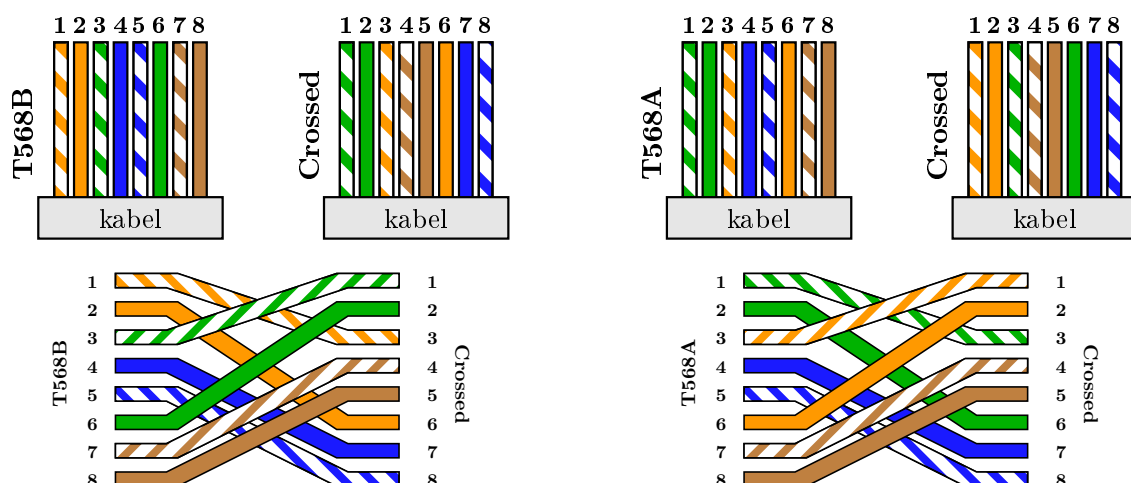
 Křížený UTP kabel pro Gigabit Ethernet už musí být krimpován podle novější verze standardu – ANSI/TIA-568-C, ve kterém je předepsáno křížení obou dvojic párů. Zde však už je porušena podmínka střídání plnobarevných a bílobarevných vodičů. Na jedné straně bude pořadí T568B nebo T568A, na druhé straně už bude pořadí vypadat zcela jinak – je třeba přehodit páry na všech pinech. Výsledek vidíme na obrázku 5.13 na straně 85.



Další informace:

Nákresy pro různé barevné kódy: <https://www.flukenetworks.com/knowledge-base/dsx-cableanalyzer-series/color-codes-dsx-cableanalyzer>





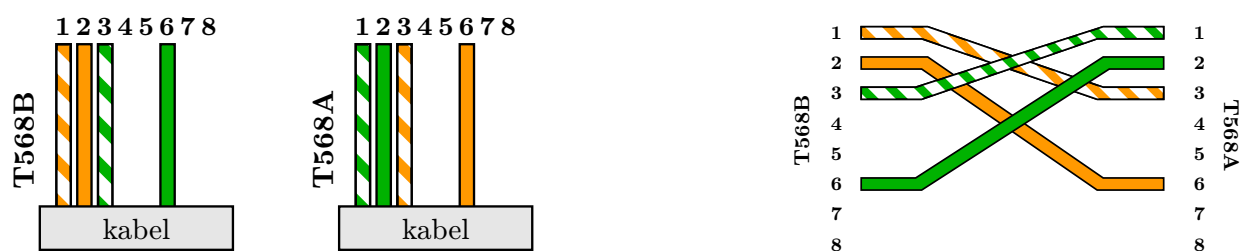
Obrázek 5.13: Barevný kód pro GbE křížený kabel – vlevo pořadí T568B, vpravo T568A

**Poznámka:**

Ve skutečnosti se naštěstí křížené kabely u rychlejších rozhraní většinou nemusejí řešit, jak jsme zjistili dříve (switche obvykle mívají funkci auto-MDIX). V některých případech přesto můžeme křížený kabel potřebovat.

**5.4.4 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 5.14: 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 5.14 zcela vlevo. U kříženého kabelu pak použijeme na jednom konci T568B a na druhém T568A, jak je naznačeno na obrázku 5.14.

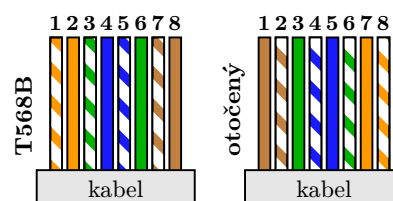
**Poznámka:**

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



5.4.5 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. S konzolovým kabelem jsme se už setkali na straně 36.

Switch či router 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 nedomluví na logické úrovni. Proto také od počítače musí směřovat komunikace podle RS-232.

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.

Někteří výrobci pro tento účel používají micro-USB port či jiné, nicméně struktura komunikace (operační kódy) bývá podle standardu pro RS-232.



Další informace:

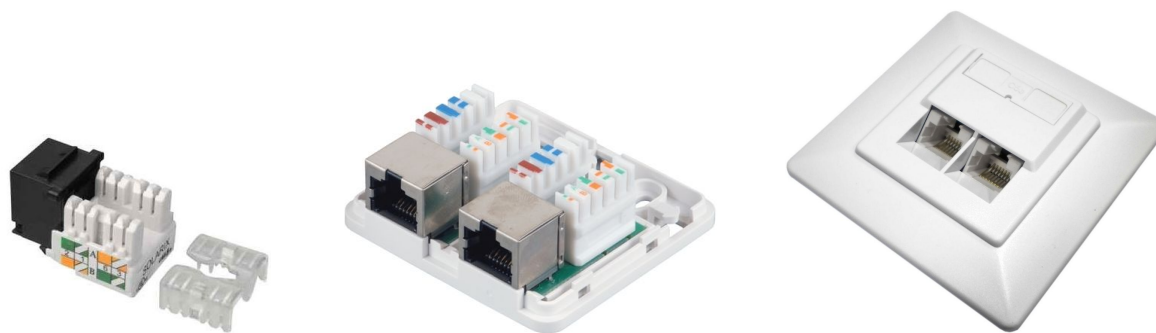
- <http://www.websitea.com/croftcom/Cisco/LaptopSerialPort.htm>
- <https://www.cable-tester.com/rj45-rs232-console-cable-pin-out/>
- <https://blog.router-switch.com/2013/05/how-to-connect-to-a-cisco-standard-console-port-rj-45/>



5.4.6 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.



Obrázek 5.15: Svorkovnice do zásuvky s jednou a dvěma zdírkami a zásuvka (Zdroj: solarix.cz)

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 (také LSA) nebo krone.



Obrázek 5.16: Patch panel, z vnitřní strany svorkovnice, z vnější UTP RJ-45 (Zdroj: bscom.cz)

Postup (Osazení UTP kabelu do zásuvky nebo patch panelu)

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

- zásuvka včetně interního konektoru, případně patch panel,
- kabel (obvykle vedený zdí),
- štípací kleště nebo jiný nástroj pro přeštípnutí nadbytečného kusu vodiče,
- zařezávací nástroj (zařezávací nůž, boxer, impact tool, punch down tool), pro svorkovnici typu 110 nebo krone (pozor – typ by měl 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 5.17: 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 5.15 vlevo).

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

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



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>
- Pořadí v kříženém kabelu pro vyšší rychlosti (Cat.6): <https://www.geeksforgEEKS.org/rj45-color-code/>, <https://cz.pinterest.com/pin/618470961310019317/>



5.4.7 Optické koncovky

Optické koncovky jsou poněkud náročnější na osazení. Je třeba si uvědomit, že optické vlákno musí naprosto přesně sednout k tomu druhému optickému vláknu, ke kterému se pomocí koncovky připojuje (tj. vlákna musí být naprosto přesně navedena proti sobě), každá nerovnost či drobný posun by měly za následek podstatný útlum.

Také je třeba udržovat vyústění vlákna v čistotě (každá nečistota by opět způsobila podstatný útlum signálu), proto pokud zrovna není koncovka zasunuta do zástrčky (v zařízení, modulu apod.), má být osazena krytkou.

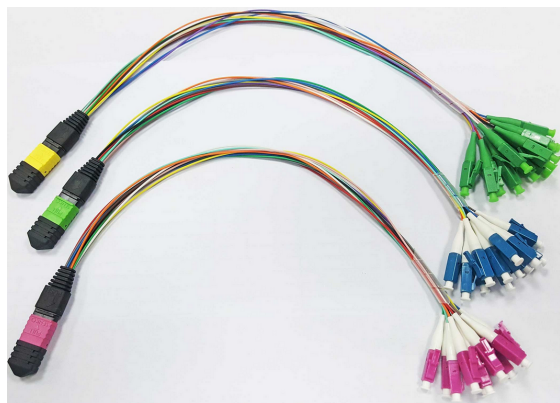
U optických kabelů se většinou setkáváme s těmito koncovkami:

- ST (Straight Tip) – spíše historie, bajonetová koncovka vzdáleně připomínající BNC konektory pro koaxiál, v současnosti se s ním prakticky nesetkáváme. Byla určena pro single-mode i multi-mode vlákna, spíše v rámci lokálních sítí.
- FC (Ferrule Connector, Fiber Channel) – dnes už také spíše historie. Vizuálně ST koncovku, ale je šroubovací, používala se většinou na single-mode vlákna.
- SC (Standard Connector, Square Connector) – použitelný pro single-mode i multi-mode kabely, ale dnes se s ním setkáme spíše na multi-mode vedeních (do 100 Gb/s), postupně je vytlačován menším konektorem LC. Nicméně dosud se hodně používá v hybridních sítích pro kabelové televize (hybridních zde znamená zároveň s koaxiálem).
- LC (Lucent Connector) – momentálně nepoužívanější, oproti SC je přibližně poloviční. Často se s ním setkáme v duplexním provedení (dvojice LC, pro každý směr jedna koncovka).
- MPO (Multi-fiber Push On) – „multikonektor“ pro více vláken (mohou být dvě, ale obvykle to bývá 12, 24, 48 nebo 72 vláken).

LC je nepoužívanějším zástupcem tzv. Small-Form Factor konektorů (SFF), je totiž menší než starší konektory „běžné“ velikosti. Jeho výhodou není jen malá velikost (obě vlákna potřebná pro duplexní komunikaci se vejdou do stejného prostoru jako RJ-45), ale také to, že se s ním zachází podobně jako s RJ-45, zacvakává se do zástrčky, včetně pojistky proti vysunutí.



Obrázek 5.18: Koncovky pro optická vlákna, zleva FC, ST, SC a LC (vždy po dvojicích)⁴



Obrázek 5.19: Koncovky typu MPO (vlevo) vyvedené do sad koncovek LC (vpravo)⁵

Podobně jako optické kabely, také optické koncovky mají své barevné kódy. Koncovky pro multi-mode kabely typu OM3 a OM4 bývají stejné barvy jako kabely, tedy světle modré (aqua), podobně OM5 kabely i koncovky bývají v jedné barvě (lime green), zatímco koncovky pro (žluté) single-mode kabely bývají tmavě modré. Multi-mode kabely OM2 mívají černé koncovky.

Ovšem pokud narazíme na svazek s větším počtem optických vláken, pak zjistíme, že ve svazku je použito velké množství barev. Není to proto, že by šlo o jiné kategorie a typy vláken, ale je třeba od sebe jednotlivá vlákna rozlišit při rozhodování, kam které vlákno patří.



Další informace:

<https://www.showmecables.com/blog/post/types-of-fiber-optic-connectors-%E2%80%93-simplex-duplex-lc-st-sc-and-more>



⁴Zdroj: <https://www.showmecables.com/>

⁵Zdroj: <https://dimiks.store/mpo-mtp-trunk-cable-patch-cord/>

5.4.8 Další ethernetové konektory a moduly

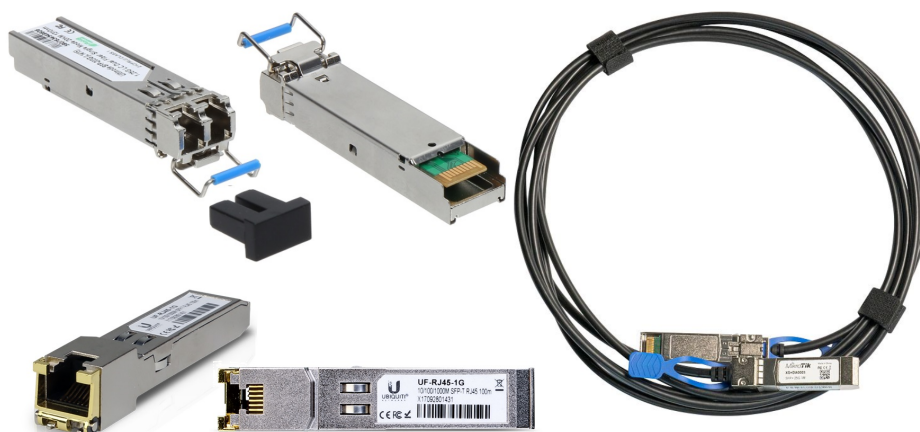
Pokud krimpujeme stíněný kabel, potřebujeme koncovky s kovovým prstencem a bočnicemi. Kov slouží k napojení na uzemnění. Na obrázku 5.20 jde o první dva (pod)obrázky zleva (první je na kabel kategorie 5e, druhý kategorie 6). V druhém případě jde o samořeznou koncovku, tedy nepotřebujeme krimpovací kleště.



Obrázek 5.20: Koncovka na stíněný kabel Cat.5e, koncovka na stíněný kabel Cat.6 samořezná, koncovka na UTP kabel Cat.6 samořezná, keystone female Cat.5e mini, koncovky Cat.8.1 (Zdroj: bscom.cz)

Na obrázku 5.20 dále vidíme samořeznou koncovku s nástavcem na kabel kategorie 6, následuje female (tj. „opačná strana“ – běžné koncovky na kabely jsou typu male) typu keystone, tedy pro napojení kabelu používáme narážecí nástroj stejně jako u zásuvky. Poslední jsou koncovky na kabel kategorie 8.1.

 U kabelů určených pro vyšší rychlosti se můžeme setkat s SFP moduly (dnes už často novějšími SFP+, QSFP+ a dalšími variantami). Tyto moduly jsou vlastně vyjímatelný transceiver (transmitter/receiver), což je část rozhraní, která je jinak uvnitř zařízení a zajišťuje funkcionalitu části vrstvy L1.



Obrázek 5.21: Vlevo nahoře optický SFP modul (singlemode s plným duplexem, LC konektor), vedle zadní strana; níže SFP modul pro metalický kabel RJ-45 na rychlost 1G/s a spodní strana, vpravo kabel s moduly kompatibilními s SFP/SFP+/SFP28. Zdroj: abctech.cz

Díky tomu, že hardwarově závislá část je „vytažená“ pryč ze zařízení, se do téže zdířky (s určitými omezeními) dá použít SFP modul pro metalický nebo optický kabel, případně i jiný. Modul může být buď přímo na kabelu, nebo může mít na vnější straně plug pro RJ-45 nebo některý optický konektor.

 Různé typy modulů (podle různých standardů) se odlišují i rychlostmi, například SFP+ umožňuje komunikaci rychlostí až 10 Gb/s, SFP28 až 25 Gb/s, QSFP+ až 40 Gb/s.

5.5 Strukturovaná kabeláž

5.5.1 Rozvaděč

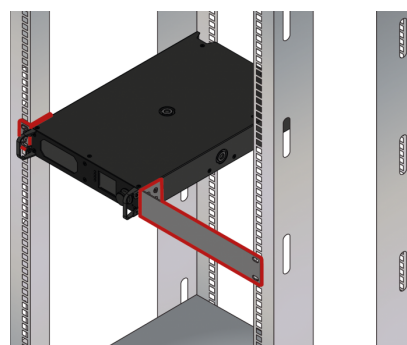
 Kam se switchem, routerem, serverem a podobnými zařízeními? Pokud se jedná o výkonná zařízení pro korporátní sféru, umísťujeme je do *rozvaděče (racku)*, což je skříň poskytující zařízením upevnění, řízené chlazení, elektroinstalaci a konektivitu. Existují různé druhy rozvaděčů – nástěnné, stojanové, 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, nevadí to, obvykle se dá vnitřní upínací mechanismus přizpůsobit (naopak to už samozřejmě vadí).

Zařízení se do racku upevňuje obvykle jen vpředu, používají se „packy“ (držáky, v angličtině „ears“) na bocích pro připevnění k čelním sloupům. Pokud do racku potřebujeme umístit i užší zařízení než jaká je šířka racku, taky to jde, jen potřebujeme držáky se správnými rozměry (viz obrázek ??).

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.

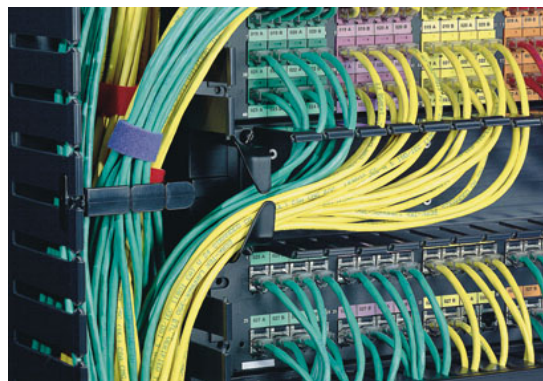
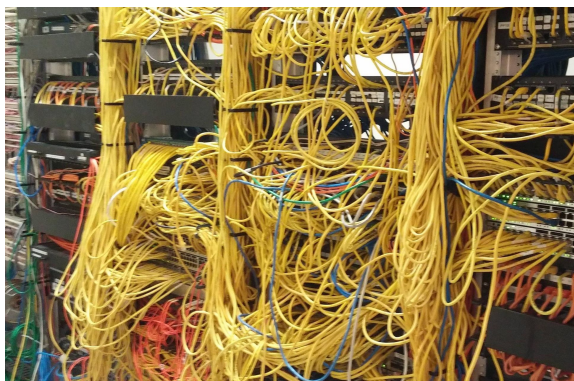
Výška rozvaděče se také udává v těchto jednotkách, například do racku o výšce 18U je možné umístit různá zařízení o výškách, jejichž součet je nejvýše 18U. Ovšem není vhodné rack příliš „přepřát“, především tehdy, když máme uvnitř výkonná zařízení vyžadující dobré chlazení.



Obrázek 5.22: Kit pro namontování užšího zařízení do racku⁶

Úkol

Podívejte se na následující dva obrázky. Na jednom z nich je něco špatně. Co to je?



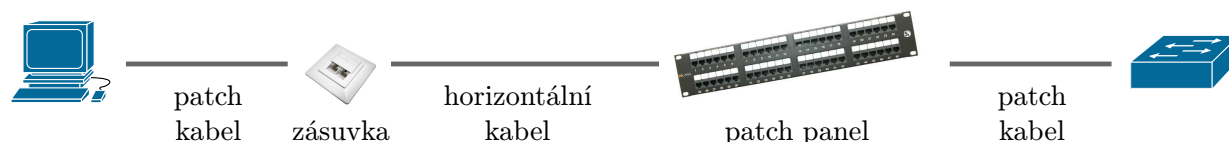
(nápověda: v barvách to není, i když barvy mohou s přehledností velmi pomoci)

⁶Zdroj: <https://lightware.com/half-rack-mounting-kit>



5.5.2 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:

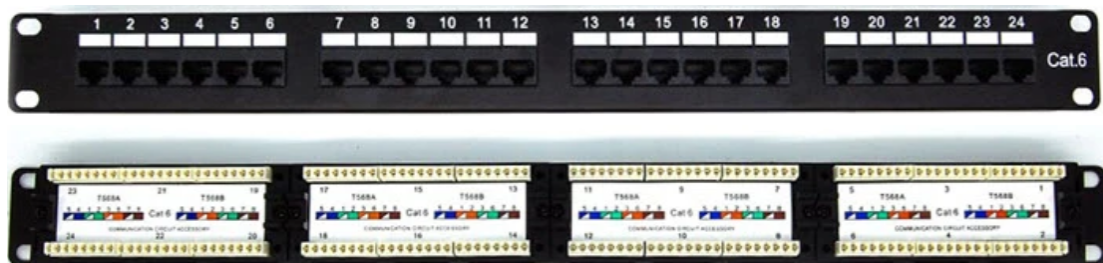


 Co se týče délky jednotlivých částí trasy, platí:

- 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.

 **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.



Obrázek 5.23: Patch panel s jednou řadou portů, čelní strana s porty a zadní strana se svorkovnicí⁷

Vlastně bychom mohli horizontální kabel připojit rovnou do switche (třebaže to standard nedoporučuje – jde obvykle o kabely s vodiči typu drát), ale použití patch panelu nám zpřehledňuje a zjednodušuje přístup k portům switche (tedy pokud je propojení uděláno správně). Patch panel může být v „přístupnější“ výšce než switch, tedy se k němu snáze dostaneme, navíc do jednoho patch panelu mohou být zapojeny kabely z více switchů (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 switche neosazujeme konektorem, ale zasuneme vodiče do zdířek na

⁷Zdroj: <https://www.cablesandkits.com/>

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



Kabeláž. O kabelech jsme se toho naučili už celkem hodně, zbývá jen rozdíl mezi patch kabelem (patch cord) a kabelem používanými do zdi pro horizontální kabeláž. Ve většině případů se pro obojí používá kroucená dvojlinka, ale:

- patch kabely bývají typu licna (lanko, stranded),
- horizontální kabely prakticky bez výjimky typu drát (solid).

Typ licna je totiž pružnější a nevadí, že nemůže být moc dlouhý, kdežto u horizontálních kabelů do zdi nepotřebujeme takovou pružnost, ovšem možnost vedení na větší vzdálenosti (do 90 m) se naopak velmi hodí.

Podle standardu by kabel typu drát dokonce neměl být zapojen přímo do zařízení (počítače, serveru apod.), měl by vést spíše mezi svorkovnicí v zásuvce na jedné straně a svorkovnicí v patch panelu na druhé straně.

5.5.3 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).

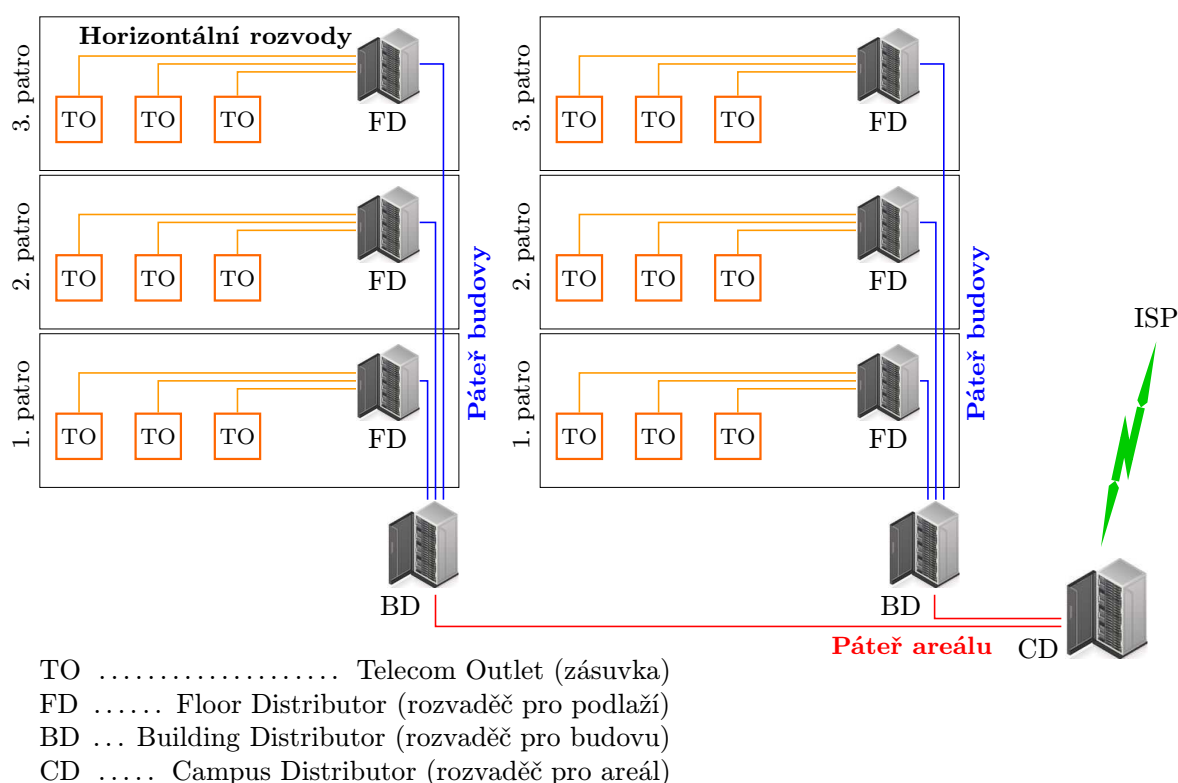
Horizontální rozvody víceméně zachovávají topologii hvězda, přičemž FD je v centru hvězdy.



Páteřní síť budovy (building backbone) propojuje všechna patra budovy, resp. všechny FD. Pokud od FD odsekne 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 zasíťovat. *Campus distributor* (CD)



Obrázek 5.24: Schéma rozvodů strukturované kabeláže

je rozvaděč zastřešující síť v celém areálu, jednotlivé BD jsou s CD propojeny *páteří síťí areálu*. Právě přes CD jde veškerá komunikace s poskytovatelem internetu (ISP) pro danou firmu.

Celá struktura sítě je naznačena na obrázku 5.24.

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.



5.5.4 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:

- Skvělý popis strukturované kabeláže podle ANSI/TIA 568-C od firmy Dintek, včetně nákresů:
https://www.dintek.com.tw/phocadownloadpap/Catalog_Files/NO14.%20ANSI%20TIA%20568C%20Standard.pdf
- <http://elektrika.cz/data/clanky/dsknn2>
- https://www.varnet.cz/soubory-ve-skladu/Karty/Spol_Zarazene/01-MANU%C3%81LY%20CS/SKS%20prirucka%20-%20man-a4.pdf



Kapitola 6

Pokračujeme v konfiguraci

 *Rychlý náhled:* V této kapitole pokračujeme v konfiguraci síťových prvků. Podíváme se zejména na konfiguraci VLAN a pak na vzdálený přístup k síťovému zařízení přes SSH.

 *Klíčová slova:* Switch, router, VLAN, Router-on-a-stick, Multilayer switch (L3 switch), Telnet, SSH, šifrování, klíč, RSA.

 *Cíle studia:* Po prostudování této kapitoly zvládnete nakonfigurovat virtuální lokální síť včetně směrování mezi různými VLAN a dále zabezpečit vzdálený přístup k síťovému zařízení pomocí SSH.

6.1 Konfigurace VLAN

Virtuální lokální sítě (VLANs), jak víme, umožňují rozdělit jednu velkou síť na několik (i hodně) menších, přičemž na rozdíl od „fyzických“ LAN nejsme omezovali tím, odkud kam vedou kabely a ke kterému rozhraní routeru jsme připojeni, popřípadě se bez routeru můžeme úplně obejít.

VLAN je mechanismus na vrstvě L2, ovšem každá L2 VLAN se typicky váže na právě jednu L3 podsít. Konfiguraci samotných VLAN budeme provádět na L2 zařízeních (tedy switchích), napojení na L3 podsítě pak na L3 zařízeních (router nebo multilayer switch). Na L3 zařízení pak může kromě samotného směrování probíhat i filtrování provozu. Můžeme například stanovit, že do určité sítě či ke konkrétnímu serveru se lze dostat jen z určité VLAN.

Pokud bychom nepotřebovali komunikaci mezi VLANami, pak by druhý krok odpadl, ovšem použitelnost takových sítí by byla omezená.

6.1.1 Oddělené VLANy na switchi

Pokud v lokální síti definujeme VLANy, znamená to, že určujeme několik logických seskupení zařízení a ke každému přiřadíme konkrétní zdroje nacházející se v síti. Obvykle platí, že každý stroj patří právě do jedné VLANy a komunikace mezi různými VLAN je omezená. Jinými slovy – rozdělení do VLAN určuje, kdo s kým má právo volně komunikovat.



Jaké příkazy budeme používat?

Na L2 switchi potřebujeme vytvořit novou VLAN – v globálním konfiguračním módu:

```
vlan číslo
name it název
```

Na rozhraní, které má pracovat v přístupovém režimu (access mode), nastavíme konkrétní VLAN, například pro rozhraní g0/1 a VLAN 10:

```
int g0/1
switchport mode access
switchport access vlan 10
```

Pokud by dané rozhraní mělo být v trunkovém módu, tedy vést k jinému switchi, nastavíme trunk:

```
int g0/1
switchport mode trunk
switchport trunk native vlan 101 ; nepovinné, výchozí native vlan je 1
switchport trunk allowed vlan 10,20,30,99 ; nepovinné, pokud neuvedeme, jsou povolené všechny
```

Jestliže se nám nedaří nastavit trunk (u prvního příkazu se objeví chybové hlášení, že nelze nastavit trunk na rozhraní, kde je encapsulation mode nastavený na auto), pak to trochu rozšíříme:

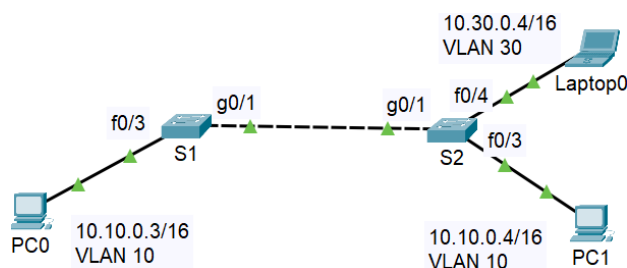
```
int g0/1
switchport trunk encap dot1q
switchport mode trunk
...
```

To se může stát na multilayer switchi nebo starším switchi, který podporuje více různých možností jak vytvořit trunk.



Příklad 6.1

Vytvoříme si síť s několika switchi a koncovými zařízeními, jak ukazuje obrázek níže. Všimněte si, jaké typy kabelů jsou kde použity: mezi switchi je křížený kabel. Adresy počítačů nastavte podle obrázku, maska je 255.255.0.0. Bránu zatím nenastavujte.



Na switchích budou porty f0/3 a f0/4 přístupové, porty g0/1 bude trunkový.

Nejdřív vytvoříme tyto VLANy (v globálním konfiguračním módu) – na obou switchích:

```
S1(config)#vlan 10 ; vytvořili jsme VLAN číslo 10
S1(config-vlan)#name IT ; pojmenovali jsme ji IT
S1(config-vlan)#vlan 20
```

```
S1(config-vlan)#name IoT
S1(config-vlan)#vlan 30
S1(config-vlan)#name others
S1(config-vlan)#vlan 40
S1(config-vlan)#name guests
S1(config-vlan)#vlan 99
S1(config-vlan)#name management
```

Příkazem `show vlan` lze zobrazit základní informace o vytvořených VLAN, případně můžeme vyzkoušet náповědu otazníkem pro zjištění dalších parametrů.

Teď vytvoříme trunk mezi switchi. Určíme nativní VLAN (zde pro jednoduchost použijeme jedničku, i když to není zrovna nejbezpečnější), a stanovíme, od kterých VLAN může jít provoz do trunku.

```
S1(config)#int g0/1
S1(config-if)#switchport mode trunk
S1(config-if)#switchport trunk native vlan 1
S1(config-if)#switchport trunk allowed vlan 10,20,30,40,99 ; nepovinné
; může být i rozsah, třeba: switchport trunk allowed vlan 10-40,99
; přidání dalších VLAN: switchport trunk allowed vlan add 50

S1(config-if)#int f0/3
S1(config-if)#switchport mode access
S1(config-if)#switchport access vlan 10
```

... ; na S2 bude na f0/3 nasazena VLAN 10, na f0/4 VLAN30

```
S1(config-if)#int vlan99 ; management vlan
S1(config-if)#ip address 10.99.0.101 255.255.0.0 ; toto provedeme na S1
S1(config-if)#ip address 10.99.0.102 255.255.0.0 ; toto provedeme na S2
S1(config-if)#no shut
S1(config-if)#exit
S1(config)#ip default-gateway 10.99.0.1
```

Nastavení můžeme zkontrolovat pomocí těchto příkazů:

```
show vlan
show vlan brief
show int trunk
show interface g0/1 status
```

Pokud by nám switch vynadal při pokusu o vytvoření trunku (což se možná stane u multilayer switchu), pak na daném rozhraní zadáme tento příkaz:

```
S1(config-if)#switchport trunk encap dot1q
S1(config-if)#do show int g0/1 switchport
```

Druhým příkazem jsme ověřili, jestli je na portu nastaveno zapouzdření do IEEE 802.1Q (tedy dot1q).

Pokud bychom chtěli některou VLAN odstranit, provedli bychom to takto (pro VLAN 30):

```
S1(config)#no vlan 30
```

Pozor, pokud tato VLAN byla nasazena na některých portech, budou tyto porty deaktivovány až do chvíle, kdy buď VLAN znovu vytvoříme, nebo přenastavíme na portu jinou VLAN.



6.2 Směrování mezi VLAN sítěmi

Existuje několik způsobů, jak zprovoznit směrování mezi různými VLAN. Učili jsme se o využití routeru se subrozhraními (router-on-a-stick) a použití multileyer switche.

6.2.1 Router-on-a-Stick

Mezi jedním switchem a routerem vytvoříme spoj, který bude z pohledu switche trunk, z pohledu routeru půjde o rozhraní rozdělené na subrozhraní (pro každé VLAN bude jedno subrozhraní). Na straně routeru použijeme tento postup (pro rozhraní g0/1, VLANy 10 a 20):

```
int g0/1.10
    encapsulation dot1q 10      ; subrozhraní pro VLAN 10
    ip address .....
int g0/1.20
    encap dot1q 20
    ip address .....
int g0/1
    no shut                    ; zapínáme celé (fyzické) rozhraní
```

Pokračujeme v předchozím příkladu.



Příklad 6.2

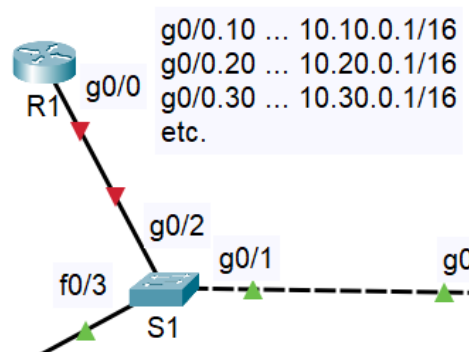
Do naší sítě připojíme router (použijte třeba model 2911). Propojíme ho se switchem S1, přičemž na straně switche použijeme port g0/2, na straně routeru g0/0.

Na straně switche vytvoříme trunk:

```
S1(config)#int g0/2
S1(config-if)#switchport mode trunk
S1(config-if)#switchport trunk native vlan 1
```

Pokračujeme na routeru, kde uvnitř rozhraní g0/0 vytvoříme jednotlivá subrozhraní, přiřadíme je ke konkrétním VLANám a přiřadíme jim IP adresu:

```
R1(config)#int g0/0.10
R1(config-if)#encapsulation dot1q 10
R1(config-if)#ip addr 10.10.0.1 255.255.0.0
R1(config-if)#int g0/0.20
R1(config-if)#encap dot1q 20
R1(config-if)#ip addr 10.20.0.1 255.255.0.0
R1(config-if)#int g0/0.30
R1(config-if)#encap dot1q 30
R1(config-if)#ip addr 10.30.0.1 255.255.0.0
R1(config-if)#int g0/0.99
R1(config-if)#encap dot1q 99
R1(config-if)#ip addr 10.99.0.1 255.255.0.0
R1(config-if)#int g0/0
R1(config-if)#no shutdown
```



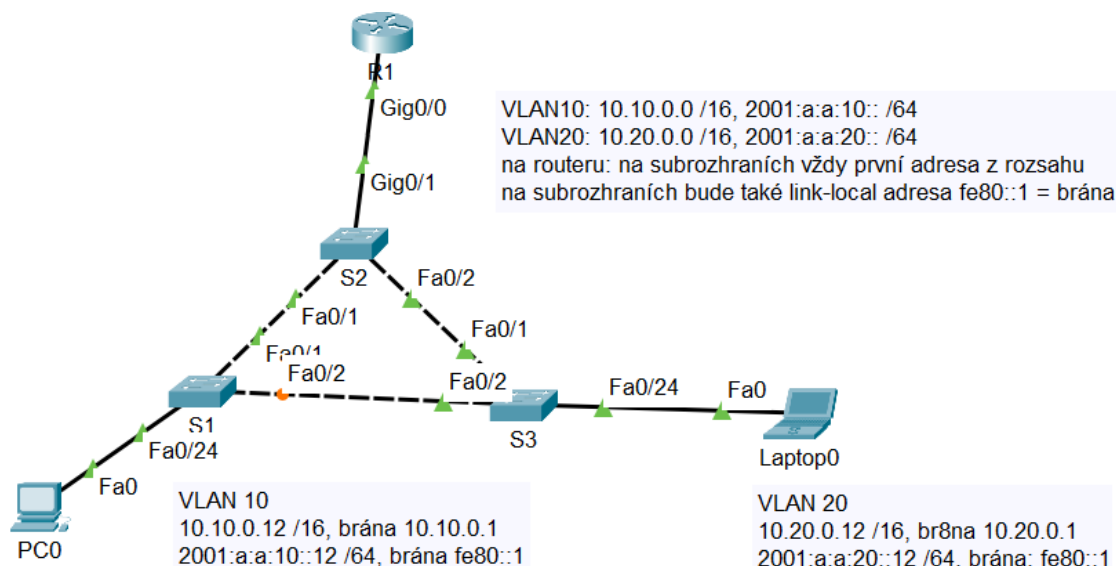
Pak by bylo dobré ještě ověřit, jestli funguje komunikace například mezi počítači s IP adresami 10.10.0.3 a 10.30.0.4, třeba pomocí pingu.





Úkol

V tomto úkolu si kromě samotného směrování mezi VLANy procvičíme také IPv6 adresy. Vytvořte topologii podle následujícího obrázku (s využitím reálných zařízení nebo v Packet Traceru, podle možnosti):



Na koncových zařízeních nastavte adresy a brány podle zadání, na switchích nastavte vše, co souvisí s VLAN (vytvoření VLAN, nasazení na portech, zprovoznění trunků). Mezi switchi budou trunky. Na routeru zprovozněte Router-on-a-Stick, na subrozhraních nasadte IPv4 i IPv6 adresy, spusťte IPv6 směrování příkazem `ipv6 unicast-routing`. Ověřte, jestli funguje ping mezi koncovými zařízeními, a to s využitím IPv4 i IPv6 adres.



6.2.2 Využití multilayer switche

Pokud chceme použít místo routeru multilayer switch, přičemž porty zůstanou L2, můžeme sloučit definování VLAN a směrování mezi nimi do jediného zařízení. Použijeme tyto příkazy:

```
int vlan10
 ip addr .....
 no shut
```

; atd. pro každou VLAN vytvoříme virtuální rozhraní a nasadíme IP adresu

`ip routing` ; zapneme IPv4 směrování, tím vytvoříme směrovací tabulku



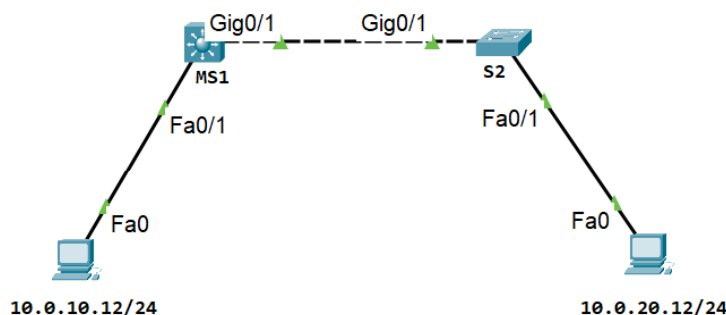
Příklad 6.3

Uvnitř multilayer switche zprovozníme „vnořený“ směrovací modul a vytvoříme virtuální rozhraní pro jednotlivé VLANy. Vytvoříme si síť podle obrázku níže. Máme jeden multilayer switch MS1 a jeden „běžný“ L2 switch S2.

Na prvním PC je IP adresa 10.0.10.12/24 a brána 10.0.10.1. Na druhém PC je IP adresa 10.0.20.12/24 a brána 10.0.20.1.

Nejdřív si nakonfigurujeme switch S2. Vytvoříme VLANy a do VLAN 20 přiřadíme port f0/1:

```
S1(config)#vlan 10
S1(config-vlan)#name hlavni
S1(config-vlan)#vlan 20
S1(config-vlan)#name hoste
S1(config-vlan)#exit
S1(config)#int f0/1
S1(config-if)#switchport mode access
S1(config-if)#switchport access vlan 20
```



Dále nakonfigurujeme trunk:

```
S1(config-if)#int g0/1
S1(config-if)#switchport mode trunk
```

Přesuneme se na multilayer switch. Vytvoříme VLANy a do první přiřadíme port:

```
MS1(config)#vlan 10
MS1(config-vlan)#name hlavni
MS1(config-vlan)#vlan 20
MS1(config-vlan)#name hoste
MS1(config-vlan)#exit
MS1(config)#int f0/1 ; linka k PC
MS1(config-if)#switchport mode access
MS1(config-if)#switchport access vlan 10
```

Nakonfigurujeme trunk. Protože jde o multilayer switch, musíme předem nastavit protokol IEEE 802.1Q.

```
MS1(config-if)#int g0/1 ; linka k S2
MS1(config-if)#switchport trunk encap dot1q
MS1(config-if)#switchport mode trunk
```

A teď přistupme ke zprovoznění L3 směrování v multilayer switchi. Pro každou VLAN vytvoříme virtuální L3 rozhraní a přiřadíme IP adresu, která zároveň bude sloužit jako brána pro podsít příslušející k dané VLAN:

```
MS1(config-if)#int vlan10
MS1(config-if)#ip addr 10.0.10.1 255.255.255.0
MS1(config-if)#int vlan20
MS1(config-if)#ip addr 10.0.20.1 255.255.255.0
```

Případně bude nutné použít příkaz `no shut` na jednotlivých rozhraních.

Zbývá zprovoznit IPv4 směrování (na routeru je defaultně spuštěno, ale na switchi ne), pak si necháme vypsát směrovací tabulku:

```
MS1(config-if)#exit
MS1(config)#ip routing

MS1(config)#do sh ip route | begin Gateway
```

```
Gateway of last resort is not set
```

```
10.0.0.0/24 is subnetted, 2 subnets
C 10.0.10.0 is directly connected, Vlan10
C 10.0.20.0 is directly connected, Vlan20
```

Jak vidíme, existují dvě (virtuálně přímo připojené) sítě a lze mezi nimi směrovat. Následující ping mezi koncovými zařízeními by měl fungovat (pokud ovšem máme na počítačích správně nastavené IP adresy a brány):

```
ping 10.0.20.12
```



 Na multilayer switchi si můžeme vyzkoušet tyto show příkazy:

```
sh ip int br    ; uvidíme obě virtuální rozhraní s nastavenými IP adresami
sh ip route     ; ve směrovací tabulce jsou obě virtuální připojené sítě
```

Jinak samozřejmě fungují jakékoliv show příkazy, které jsme si ukazovali předtím:

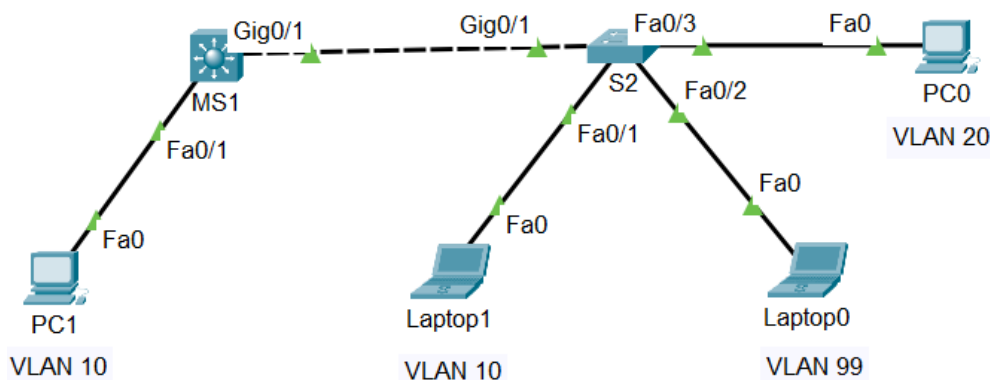
```
sh vlan br
sh int trunk
```

Případně další.



Úkol

Vytvořte topologii podle následujícího obrázku:



Koncová zařízení budou mít IPv4 adresy s prvním oktetem 10, druhý oktet bude číslo VLAN, třetí =0, čtvrtý buď číslo 11 nebo číslo 12 (volte tak, aby každé zařízení mělo jinou IP adresu, což se týká především té VLAN, kde jsou dvě koncová zařízení). Délka prefixu je 16 bitů.

VLAN 10 se jmenuje *zamestnanci*, VLAN 20 je *hoste*, VLAN 99 má jméno *management*. Mezi switchi bude trunk, ostatní porty budou přístupové s vhodnou nastavenou VLAN.

Až budete mít nakonfigurované také switche včetně směrování mezi VLAN sítěmi na multilayer switchi, vyzkoušejte komunikaci mezi zařízeními patřícími do stejné nebo různých VLAN.

Nakonfigurovanou síť si uchovejte pro procvičení následujícího tématu – přístupu na síťové zařízení přes telnet nebo SSH.



6.3 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 za-
    dáme 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)
```



Úkol

V síti nakonfigurované v předchozím úkolu zprovozníte nejdřív přístup přes Telnet na switchi S2. Nasaďte na rozhraní `vlan99` IP adresu `10.99.0.2/16`, vhodnou bránu, na VTY linkách nasaďte heslo `cisco`, přihlášení tímto heslem, povolte přihlášení přes telnet. Také nastavte heslo do enable módu, protože přes VTY linky byste se jinak do toho módu nedostali.

Vyzkoušejte, jestli se dá do konfigurace switche dostat z počítače patřícího do VLAN 99:

```
telnet 10.99.0.2
```

Vidíte, jak je to snadné? Pokud jste na reálných zařízeních a náhodou je na nich povolen telnet (možná není, proč asi?), zkuste tento provoz zachytit pomocí Wiresharku. Pokud používáte Packet Tracer, použijte simulační mód k zachycení a průzkumu provozu.

A teď náš switch zabezpečíme poněkud lépe. Nastavte minimální délku hesla na 10, vytvořte nového uživatele `admin` s patřičně dlouhým heslem (pozor na překlepy), nastavte doménu na `slu.cz`, zprovozníte SSH a na VTY linkách nastavte přihlášení proti lokální databázi a povolte přístup pouze přes SSH.

Vyzkoušejte přístup z téhož počítače – nejdřív znovu přes telnet, potom přes SSH, bude fungovat jeden z těchto příkazů, podle toho, jaký je na daném zařízení operační systém:

```
ssh -l admin 10.99.0.2
```

```
ssh admin@10.99.0.2
```

Potom vyzkoušejte přístup i z některého zařízení patřícího do jiné VLAN.



Kapitola 7

Co se děje na síťové a transportní vrstvě

 *Rychlý náhled:* V této kapitole 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.

7.1 Struktura paketu protokolu IP

Na síťové vrstvě (tedy L3) se jako nosný protokol (tedy ten, který do záhlaví ukládá adresy, a tedy se bez něho neobejdeme, musíme ho použít pro veškerý provoz z vrstvy L3 a vyšších) používá protokol IP. Jak víme, v současné době se setkáváme se dvěma verzemi, které se liší formátem adres, ale také formátem paketů.

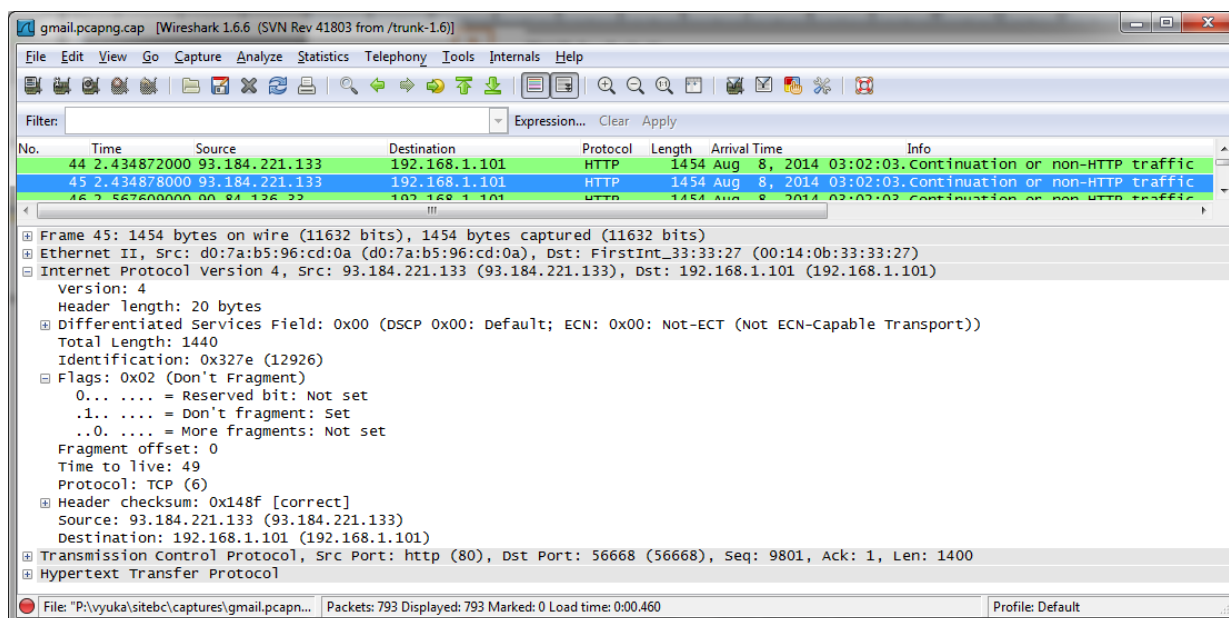
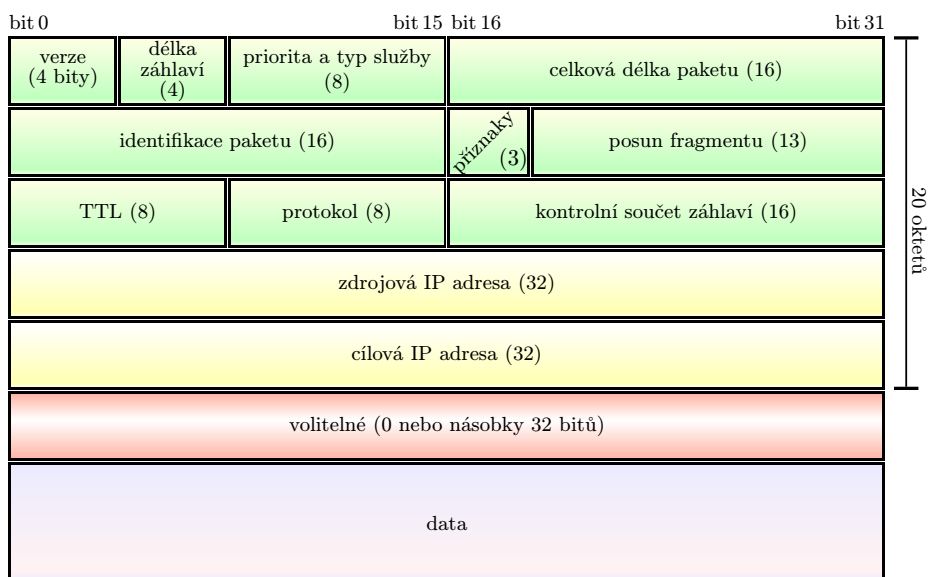
Zde pomocí Wiresharku prozkoumáme formát IPv4 a IPv6 paketů.

7.1.1 Pakety protokolu IPv4

Příklad 7.1

Program Wireshark už známe, tedy se můžeme prozkoumat některý skutečný IPv4 paket. Můžeme si buď zachytit nějaký reálný provoz na síti, nebo použít úložiště .PCAP souborů se zachyceným provozem.

Podíváme se na schéma IPv4 paketu, které známe z přednášek, a pak ho srovnáme se záhlavím skutečného IPv4 paketu. IPv4 záhlaví je mnohem delší než u rámce Ethernet II, proto je rozčleněno na několik řádků. Každý řádek je dlouhý 4 B, tedy 32 bitů, protože standard vznikl v době, kdy délka sběrnice pro komunikaci se sítí byla maximálně 32 bitů.



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

- opravdu se jedná o verzi 4 protokolu IP, protože v prvním poli je číslo 4,
- pole *Header Length* obsahuje údaj pro 20 oktetů (to odpovídá běžné velikosti IPv4 záhlaví),
- pole pro prioritu a typ služby je celé na stavěno 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 (nicméně odesílatel fragmentovat mohl, pokud od jiného protokolu, třeba TCP, dostal příliš velký balík dat k odeslání),
- *Fragment Offset* je nastaven na 0,

- hodnota TTL je 49 – pokud například byla původně 64, pak paket prošel přes pět routerů 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. Ověřte si, jestli se vám náhodou nepodařilo zachytit IP paket s volitelnými poli na konci záhlaví.

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.



7.1.2 Pakety protokolu IPv6

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

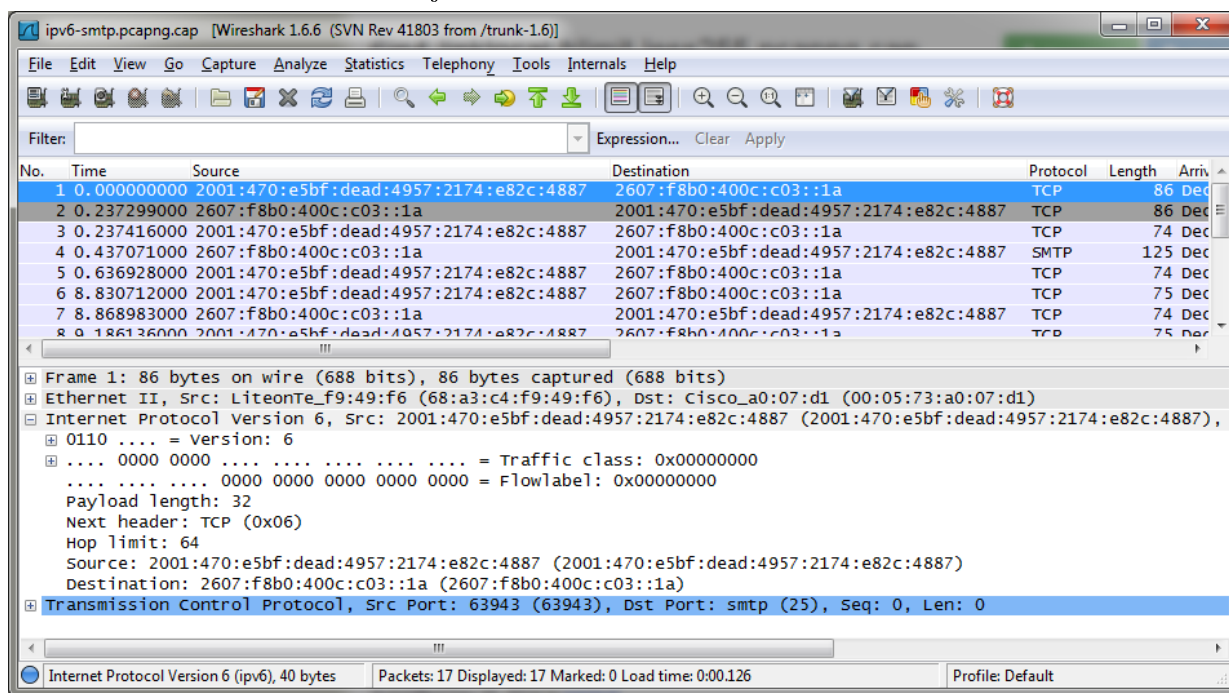
Následuje struktura IPv6 paketu (řádky jsou zarovnané na 64 bitů):

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



Příklad 7.2

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 mříčící na server.
- *Next Header* určuje, že za povinným záhlavím najdeme TCP segment (hodnota 6).
- *Hop Limit* (obdobu TTL v IPv4) je 64, což zřejmě bude výchozí hodnota pro odesílající zařízení (paket prohlížíme na tom zařízení, ze kterého je odesílán, takže ještě z této hodnoty nebylo nic odečteno).
- Následuje zdrojová a cílová adresa.



Úkol

Na https://wiki.wireshark.org/uploads/___moin_import___/attachments/SampleCaptures/DHCPv6.pcap je soubor DHCPv6.pcap. Otevřete ho a prohlédněte si jeho obsah, resp. především první paket. Všimněte si, že

- V poli *Hop Limit* je u většiny paketů hodnota 1 (a byla tam při odeslání, protože ten první je odchozí paket). To znamená, že paket půjde nejdál k nejbližšímu routeru, za něj už ne (router na něj případně reaguje, ale už dál ho nepřepoše). 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 z příslušného seznamu.



7.2 Protokol ICMP a příkazy, které ho používají

7.2.1 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 7.3 (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

Spustte 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:

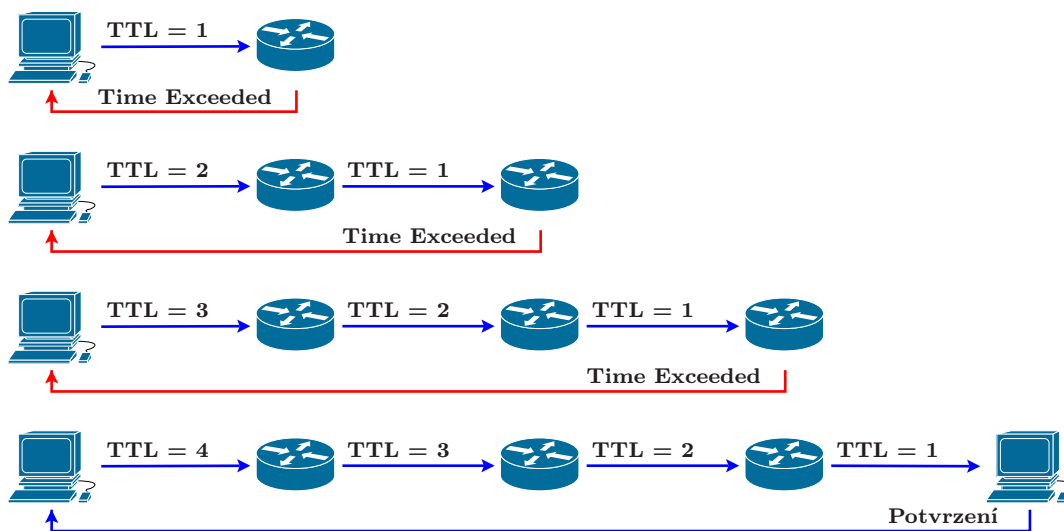
- do čeho jsou ICMP pakety zapouzdřovány,
- jaké číslo ICMP zprávy je používáno pro dotaz a odpověď,
- jaké hodnoty jsou v polích *Identifier* a *Sequence* pro dané dotazy a odpovědi,
- jaká data jsou za záhlavím zapouzdřena (podívejte se úplně dolů do podokna *Packet Bytes*, případně si zapněte jeho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



7.2.2 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 inspiroval), 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 7.4 (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:

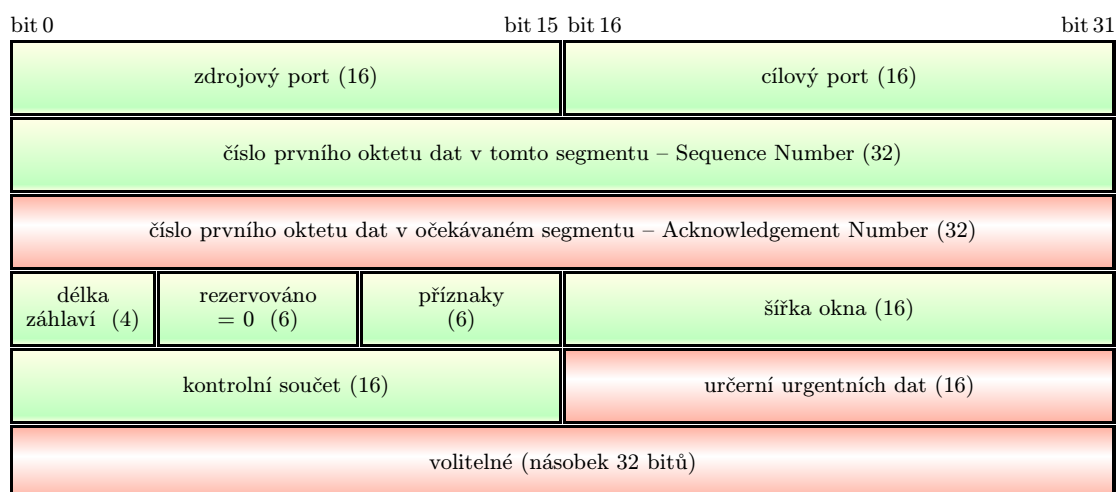
- www.slu.cz
- www.google.com
- 127.0.0.1, ::1



7.3 Protokoly na transportní vrstvě

7.3.1 Protokol TCP a průzkum parametrů 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,...

**Příklad 7.5**

Pomocí Wiresharku odchytíme provoz se serverem <http://tcpipguide.com/>. Je to jeden z mála serverů, které nešifrují, tedy se můžeme podívat „dovnitř“.

Teď potřebujeme dostat pouze záznamy týkající se daného serveru. Do řádku filtru nad seznamem zachycených paketů zadejte tento řetězec:

`ip.addr==216.92.67.219`

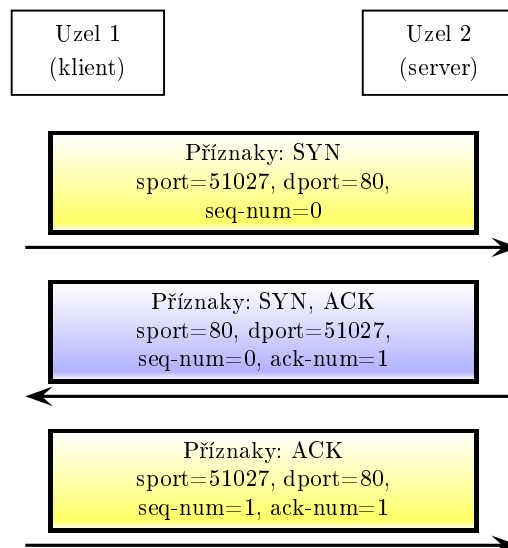
Jde o IPv4 adresu serveru <http://tcpipguide.com/>. Pak by se měly zobrazit pouze ty pakety, které mají tuto adresu jako zdrojovou nebo cílovou.

No.	Time	Source	Destination	Protocol	Length	Info
155	3.646919	10.7.12.194	216.92.67.219	TCP	66	61423 → 443 [SYN] Seq=0 Win=642...
156	3.647066	10.7.12.194	216.92.67.219	TCP	66	61424 → 443 [SYN] Seq=0 Win=642...
165	3.752418	216.92.67.219	10.7.12.194	TCP	60	443 → 61424 [ACK] Seq=1 Ac...

Obrázek vpravo naznačuje, jak vypadá TCP handshake. První TCP segment ze spojení jde od klienta k serveru a má nastaven bit SYN (a žádný jiný). Zdá se, že nejdřív se klient pokusil komunikovat přes HTTPS (číslo portu 443, jak vidíme na začátku sloupce Info), ale druhá strana zjevně neumí šifrovat (nemá certifikát). Takže později se klient pokusil spojit s portem 80 na straně serveru. Zkusíme si množství zobrazeného provozu ještě o něco zmenšit tím, že do filtru přidáme podmínku pro číslo TCP portu 80 (alternativně by se taky dalo místo `&&` napsat `AND`):

```
ip.addr==216.92.67.219 && tcp.port==80
```

Na obrázku níže jsou vyznačeny části TCP segmentu, které souvisejí s nákresem vpravo: čísla portů obou stran, číslo Sequence Number (to skutečné je uvedeno jako druhé, tedy „raw“, relativní je počítáno od 0, protože menší čísla jsou pro člověka snadněji zpracovatelná), dále Acknowledge Number (ale toto pole nemá v prvním segmentu význam, proto i raw hodnota je 0), a pak příznaky.



No.	Time	Source	Destination	Protocol	Length	Info
326	4.177529	10.7.12.194	216.92.67.219	TCP	66	54554 → 80 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SA...
332	4.282889	216.92.67.219	10.7.12.194	TCP	66	80 → 54554 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=146...
333	4.282950	10.7.12.194	216.92.67.219	TCP	54	54554 → 80 [ACK] Seq=1 Ack=1 Win=262656 Len=0
334	4.284380	10.7.12.194	216.92.67.219	HTTP	509	GET / HTTP/1.1
339	4.387827	216.92.67.219	10.7.12.194	TCP	60	[TCP Window Update] 80 → 54554 [ACK] Seq=1 Ack=1 Win=1314...

Ethernet II, Src: 70:b5:e8:aa:9e:34, Dst: 00:08:e3:ff:fc:08

Internet Protocol Version 4, Src: 10.7.12.194, Dst: 216.92.67.219

Transmission Control Protocol, Src Port: 54554, Dst Port: 80, Seq: 0, Len: 0

Source Port: 54554

Destination Port: 80

[Stream index: 14]

[Stream Packet Number: 1]

[Conversation completeness: Complete, WITH_DATA (31)]

[TCP Segment Len: 0]

Sequence Number: 0 (relative sequence number)

Sequence Number (raw): 570036866

[Next Sequence Number: 1 (relative sequence number)]

Acknowledgment Number: 0

Acknowledgment number (raw): 0

1000 ... = Header Length: 32 bytes (8)

Flags: 0x002 (SYN)

000. = Reserved: Not set

...0 = Accurate ECN: Not set

....0... = Congestion Window Reduced: Not set

....0... = ECN-Echo: Not set

....0... = Urgent: Not set

....0... = Acknowledgment: Not set

....0... = Push: Not set

....0... = Reset: Not set

....0... = Syn: Set

....0... = Fin: Not set

[TCP Flags:S.]

Projdeme si postupně první čtyři segmenty (tři pro navázání spojení a čtvrtý s tím, co klient opravdu chce poslat). Nejdřív sekvenční čísla:

1. Source Port: 54554 Destination Port: 80 [Stream index: 14] [Stream Packet Number: 1] [Conversation completeness: Complete, WITH_DATA (31)] [TCP Segment Len: 0] Sequence Number: 0 (relative sequence number) Sequence Number (raw): 570036866 [Next Sequence Number: 1 (relative sequence number)] Acknowledgment Number: 0 Acknowledgment number (raw): 0	2. Source Port: 80 Destination Port: 54554 [Stream index: 14] [Stream Packet Number: 2] [Conversation completeness: Complete, WITH_DATA (31)] [TCP Segment Len: 0] Sequence Number: 0 (relative sequence number) Sequence Number (raw): 1023220163 [Next Sequence Number: 1 (relative sequence number)] Acknowledgment Number: 1 (relative ack number) Acknowledgment number (raw): 570036867
3. Source Port: 54554 Destination Port: 80 [Stream index: 14] [Stream Packet Number: 3] [Conversation completeness: Complete, WITH_DATA (31)] [TCP Segment Len: 0] Sequence Number: 1 (relative sequence number) Sequence Number (raw): 570036867 [Next Sequence Number: 1 (relative sequence number)] Acknowledgment Number: 1 (relative ack number) Acknowledgment number (raw): 1023220164	4. Source Port: 54554 Destination Port: 80 [Stream index: 14] [Stream Packet Number: 4] [Conversation completeness: Complete, WITH_DATA (31)] [TCP Segment Len: 455] Sequence Number: 1 (relative sequence number) Sequence Number (raw): 570036867 [Next Sequence Number: 456 (relative sequence number)] Acknowledgment Number: 1 (relative ack number) Acknowledgment number (raw): 1023220164

Barevně (červená a zelená) jsou vyznačena ta čísla, která spolu souvisejí, tedy v tomtéž směru. Všimněte si, že relativní a skutečná rostou stejnou rychlostí, tedy na začátku zařízení vygenerovalo náhodné číslo, které použilo jako Sequence number v prvním posílaném segmentu, a pak přičítá buď jedničku (to platí pro úvodní a potvrzovací segmenty) nebo velikost posílaných dat. Také zde vidíme *princip pozitivního potvrzení*: Acknowledgement number je vlastně informace „všechno před tímto Bytem dorazilo v pořádku, pošli další od tohoto Byte“.

Dále se podíváme na nastavené příznaky: první dva segmenty mají nastavený bit SYN (synchronizační), ten patří do toho segmentu, který v daném směru jde jako první. V žádném dalším už tento příznak nebude nastaven, s čímž může počítat i firewall, který takto pozná, že se navazuje spojení.

1. Flags: 0x002 (SYN) 000. = Reserved: Not set ...0 = Accurate ECN: Not set ... 0... = Congestion Window Reduced: Not set0.. = ECN-Echo: Not set0. = Urgent: Not set0 = Acknowledgment: Not set 0... = Push: Not set0.. = Reset: Not set >1. = Syn: Set 0 = Fin: Not set [TCP Flags:S.]	2. Flags: 0x012 (SYN, ACK) 000. = Reserved: Not set ...0 = Accurate ECN: Not set ... 0... = Congestion Window Reduced: Not set0.. = ECN-Echo: Not set0. = Urgent: Not set1 = Acknowledgment: Set 0... = Push: Not set0.. = Reset: Not set >1. = Syn: Set 0 = Fin: Not set [TCP Flags:A..S.]
3. Flags: 0x010 (ACK) 000. = Reserved: Not set ...0 = Accurate ECN: Not set ... 0... = Congestion Window Reduced: Not set0.. = ECN-Echo: Not set0. = Urgent: Not set1 = Acknowledgment: Set 0... = Push: Not set0.. = Reset: Not set 0 = Syn: Not set 0 = Fin: Not set [TCP Flags:A....]	4. Flags: 0x018 (PSH, ACK) 000. = Reserved: Not set ...0 = Accurate ECN: Not set ... 0... = Congestion Window Reduced: Not set0.. = ECN-Echo: Not set0. = Urgent: Not set1 = Acknowledgment: Set 1... = Push: Set 0.. = Reset: Not set 0 = Syn: Not set 0 = Fin: Not set [TCP Flags:AP....]

Od druhého segmentu dále je vždy nastaven příznak ACK (acknowledgement), protože druhý a následující segmenty jsou vždy odpovědí na některý předchozí segment z druhé strany. Třetí segment má tedy pouze příznak ACK, a ve čtvrtém vidíme také příznak PSH (push). PSH je informace pro síťový zásobník odesílatele, že se tam data odesílaná v rámci tohoto spojení nemají zdržovat v bufferu, ale mají se okamžitě poslat, i kdyby mělo jít o velmi malý segment. Kdyby tento příznak nebyl nastaven, data segmentu počkají v bufferu, až se nahromadí víc dat k odeslání, což u potvrzujících, žádajících atd. segmentů nemusí být praktické.

Zbývá dojednání velikosti okna a volitelná část záhlaví:

- | | |
|--|--|
| <p>1. Window: 64240
[Calculated window size: 64240]
Checksum: 0x3327 [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0</p> <p>Options: (12 bytes), Maximum segment size, No-Operation</p> <ul style="list-style-type: none"> <div style="border: 1px solid green; padding: 2px;"> TCP Option - Maximum segment size: 1460 bytes
 Kind: Maximum Segment Size (2)
 Length: 4
 MSS Value: 1460 </div> <div style="border: 1px solid green; padding: 2px;"> TCP Option - No-Operation (NOP) </div> <div style="border: 1px solid green; padding: 2px;"> TCP Option - Window scale: 8 (multiply by 256)
 Kind: Window Scale (3)
 Length: 3
 Shift count: 8
 [Multiplier: 256] </div> <div style="border: 1px solid green; padding: 2px;"> TCP Option - No-Operation (NOP) </div> <div style="border: 1px solid green; padding: 2px;"> TCP Option - No-Operation (NOP) </div> <div style="border: 1px solid green; padding: 2px;"> TCP Option - SACK permitted
 Kind: SACK Permitted (4)
 Length: 2 </div> | <p>2. Window: 65535
[Calculated window size: 65535]
Checksum: 0xdc5a [unverified]
[Checksum Status: Unverified]
Urgent Pointer: 0</p> <p>Options: (12 bytes), Maximum segment size, No-Operation</p> <ul style="list-style-type: none"> <div style="border: 1px solid green; padding: 2px;"> TCP Option - Maximum segment size: 1460 bytes
 Kind: Maximum Segment Size (2)
 Length: 4
 MSS Value: 1460 </div> <div style="border: 1px solid green; padding: 2px;"> TCP Option - No-Operation (NOP) </div> <div style="border: 1px solid green; padding: 2px;"> TCP Option - Window scale: 6 (multiply by 64)
 Kind: Window Scale (3)
 Length: 3
 Shift count: 6
 [Multiplier: 64] </div> <div style="border: 1px solid green; padding: 2px;"> TCP Option - SACK permitted
 Kind: SACK Permitted (4)
 Length: 2 </div> <div style="border: 1px solid green; padding: 2px;"> TCP Option - End of Option List (EOL) </div> <div style="border: 1px solid green; padding: 2px;"> TCP Option - End of Option List (EOL) </div> |
| <p>3. Window: 1026
[Calculated window size: 262656]
[Window size scaling factor: 256]</p> | <p>4. Window: 1026
[Calculated window size: 262656]
[Window size scaling factor: 256]</p> |

V prvních dvou segmentech si obě strany určily počáteční velikost okna (každá jinou), a to číslo, které se vejde do vyhrazených 16 bitů (tak velké je pole v záhlaví určené pro tento účel). Ale zároveň si ve volitelných (Options) posílají info, že pro příště se má velikost okna násobit číslem 256 (první strana), resp. 64 (druhá strana), tedy v binární formě se má přidat 8 nul, resp. 6 nul.

O velikosti okna víme, že určuje maximální počet Bytů, které je možné odeslat bez nutnosti potvrzení druhou stranou (postupně v několika segmentech). Všimněte si, že třetí a čtvrtý segment mění velikost okna, ale už počítají s deklarovanou násobností. Výsledná velikost okna (Calculated window size) je tak velké číslo, že by se do 16 bitů nevešlo, proto se používá hodnota scaling factor ve volitelných polích.

Obě strany si taky navzájem poslaly informaci o dojednané maximální délce segmentu (MSS), obě strany ji nastavily na 1460, aby se po přidání TCP a IP záhlaví vše vešlo do rámce, pro jehož obsah je limit 1500 B.

Podlejší volitelné pole je SACK Permitted, čímž se obě strany dohodly, že pokud bude potřeba znovu poslat některý segment, protože se cestou ztratil, nebude nutné znovu posílat následující segmenty, pokud dorazily v pořádku.

Třetí a čtvrtý segment již žádné options neobsahují, options většinou bývají především v prvních dvou segmentech při navazování spojení, kdy je třeba dohodnout parametry.





Úkoly

1. Pokračujte v předchozím příkladu a najděte segmenty pro ukončení spojení (pokud ovšem jste příliš brzy neukončili zachytávání provozu). Která strana poslala segment s příznakem FIN jako první? Jak reagovala druhá strana?
2. Ve skutečnosti probíhá handshake i tehdy, když má být provoz šifrován. Zachyťte ve Wiresharku provoz s jakýmkoliv běžným serverem, který komunikuje přes HTTPS (například google.com), a podívejte se, jak tam probíhá handshake, přenos dat a ukončení spojení. Nezapomeňte zrušit filtr na adresu a číslo portu, případně použijte filtr (jen) pro číslo portu 443 (to je HTTPS).

Je zřejmé, že přímo k datům se ve Wiresharku nedostaneme, protože protokol TLS je šifruje, ale název domény, se kterou se komunikuje, se dá přesto najít. Propátrejte zachycené pakety a najděte místo, kde je doména serveru viditelná.



7.3.2 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 nezahlcovatel sítí, 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`.

bit 0	bit 15	bit 16	bit 31
zdrojový port (16)		cílový port (16)	
délka segmentu (16)		kontrolní součet (16)	

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

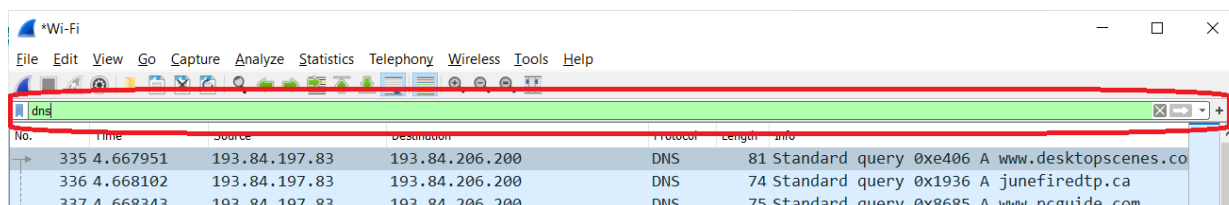


7.4 Průzkum provozu ve Wiresharku

S programem Wireshark jsme se už seznámili a tedy už víme, jak se (v základu) používá. Zde se zaměříme na pokročilejší vlastnosti programu jako jsou například dva druhy filtrů a analýza datového toku.

7.4.1 Filtry zobrazení

Filtry zobrazení (Display Filters) nám umožňují v zachyceném provozu určit, co konkrétně nás zajímá. Můžeme určovat protokoly, adresy a prakticky cokoliv dalšího v záhlavích různých PDU.



Obrázek 7.1: Zadávací řádek pro filtr zobrazení

Display filter zadáváme do řádku, který je vyznačen na obrázku 7.1.

 Nejjednodušší způsob používání je filtrování podle určitého protokolu. Například pokud chceme zobrazit pouze TCP komunikaci, zadáme do filtru `tcp` a potvrdíme klávesou `[Enter]` nebo klepnutím na šipku zcela vpravo. Pokud chceme zrušit filtr a nezadat jiný, tak výraz smažeme a opět potvrdíme (nestačí jen smazat). U některých protokolů se rozlišují verze, a v tom případě u vyšší verze musíme příslušný údaj přidat. Například:

- `icmp` znamená pouze ICMP verze 4,
- `icmpv6` znamená pouze ICMP verze 6.

Operátor	Význam
<code>==, eq</code>	rovná se
<code>!=, ne</code>	nerovná se
<code>>, gt, <, lt, <=, le, >=, ge</code>	různé relační operátory
<code>and, &&</code>	logické and
<code>or, </code>	logické or
<code>not, !</code>	negace (neplatí, že)
<code>xor, ^^</code>	logické xor (výlučné nebo)
<code>[...]</code>	pro dané pole záhlaví chceme určit, které oktety nás zajímají
<code>contains</code>	pole s řetězcem obsahuje zadaný podřetězec

Tabulka 7.1: Operátory výrazů pro filtry zobrazení

V tabulce 7.1 je přehled základních operátorů pro filtry zobrazení. Výrazy, které těmito operátory propojujeme, jsou buď jednoduše určením protokolu (jak bylo výše ukázáno), nebo nějak souvisejí s poli v záhlavích protokolových datových jednotek. Například `ip.src` znamená pole pro zdrojovou IP adresu v záhlaví IPv4, `ip.addr` je IP adresa v záhlaví IPv4 (je jedno, jestli zdrojová nebo cílová), `tcp.flags.syn` je hodnota synchronizačního bitu v záhlaví TCP apod.



Úkol

Spusťte Wireshark, odchytněte vhodný provoz (například spusťte příkaz `ping` na některou doménu, ve webovém prohlížeči si zobrazte některý web ideálně bez šifrování, třeba `http://tcpipguide.com/`, atd.).

Zastavte odchyťávání a vyzkoušejte různé filtry. Například:

- `dns` (provoz protokolu DNS)
- `tcp.port == 80` nebo `tcp.port == 443` (komunikace s webovým serverem)
- `icmp or icmpv6` (různé zprávy ICMP, vyzkoušejte také zvlášť)
- `tcp.syn == 1` (navazování spojení, TCP segmenty s nastaveným příznakem SYN)
- `eth.dst == ff-ff-ff-ff-ff-ff` (broadcast na vrstvě L2, což jsou obvykle rámce Ethernet II s broadcastovou adresou jako cílem) – MAC adresu lze ve skutečnosti zadávat i s dvojtečkami jako oddělovačem
- `dns.qry.name contains "tcpipguide"` (pole s jménem pro přeložení v DNS zprávě obsahuje zadaný řetězec)

Zjistěte svou momentální IP adresu (pokud jste ve Windows, tak třeba pomocí `ipconfig`, v Linuxu nebo v MacOS pomocí `ifconfig` nebo `ip addr`). Dále si ve Wiresharku vyfiltrujte svou vlastní komunikaci, a to jak IPv4, tak i IPv6. Například (dosadte své adresy):

```
ip.addr == 10.6.13.204 or ipv6.addr == 2001:718:2601:265:d944:f2df:8574:56e3
```

Především IPv6 adresu není od věci zkopírovat, ať ji nemusíte celou přepisovat. Upozornění: pokud jste adresu zjistili na Příkazovém řádku, pak zkopírování provedete tak, že myší řetězec „vysvítíte“ a klepnete na klávesu `Enter`, nebo použijte kontextové menu.



 Nemusíme porovnávat celá pole ze záhlaví, můžeme určovat jejich části, jednotlivé oktety. K tomu účelu se používají hranaté závorky, jejichž použití je trochu podobné polím při programování. V hranatých závorkách můžeme určit:

- `[zacatek:pocet]` od kterého oktetu v poli a kolik celkem, například: `[0:3]` znamená od začátku (oktety číslujeme od nuly), celkem 3 oktety (pokud je na začátku 0, můžeme ji vynechat)
- `[zacatek-konec]` rozsah od-do
- `[zacatek:]` od zadaného oktetu až do konce pole



Příklad 7.6

Zajímá nás komunikace vedoucí na multicastové lokální adresy. Multicastové IPv4 adresy se na vrstvě L2 mapují na MAC adresy s první polovinou ve tvaru `01-00-5e`, tedy filtr bude následující:

```
eth.dst[0:3] == 01-00-5e, nebo jednodušeji eth.dst[:3] == 01-00-5e
```



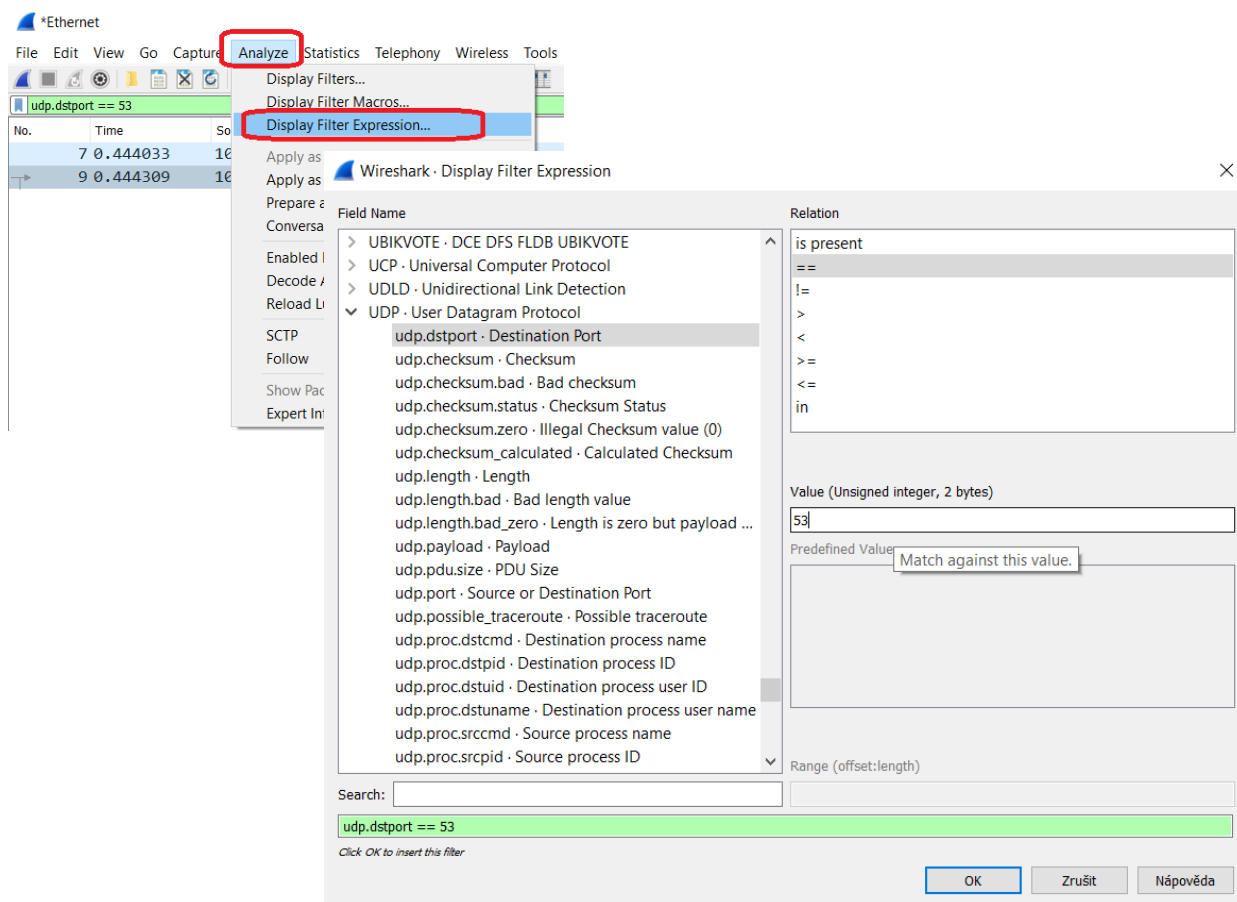
 Pokud tápeme v názvech polí ze záhlaví, může pomoci průvodce vytvářením Display Filtru, kterého najdeme v menu *Analyze – Display Filter Expressions*, viz obrázek 7.2.



Další informace:

https://www.wireshark.org/docs/wsug_html_chunked/ChWorkBuildDisplayFilterSection.html





Obrázek 7.2: Průvodce vytvářením filtrů zobrazení

7.4.2 Filtry zachytávání

Zatímco dříve popsané filtry zobrazení umožňují určovat, co z dříve zachyceného provozu nás zajímá, filtry zachytávání (capture filters) se využijí už při samotném zachytávání – určují, co konkrétně se má zachytit (ostatní provoz je ignorován).

Nejdřív je důležité vědět, kde konkrétně se tyto filtry zadávají – samozřejmě před samotným spuštěním zachytávání, viz obrázek 7.3. Na dalším obrázku vidíme postup zadání filtru v případě, že chceme změnit předchozí nastavení – přes menu *Capture*, pak to bude platit pro následující zachytávání (ale po ukončení aplikace se nastavení ztratí).

Filtry zachytávání používají trochu jinou syntaxi. Například pokud chceme zachytávat pouze komunikaci související s určitou IP adresou (což se může hodit, protože především na Wi-fi bývá pěkný chaos), napíšeme – samozřejmě s takovou IP adresou, která nás zajímá:

```
host 10.0.0.18
```

Pokud bychom chtěli zachytávat jen komunikaci související s určitým číslem portu, je to prostě klíčové slovo *port*, například pro DNS provoz (port 53):

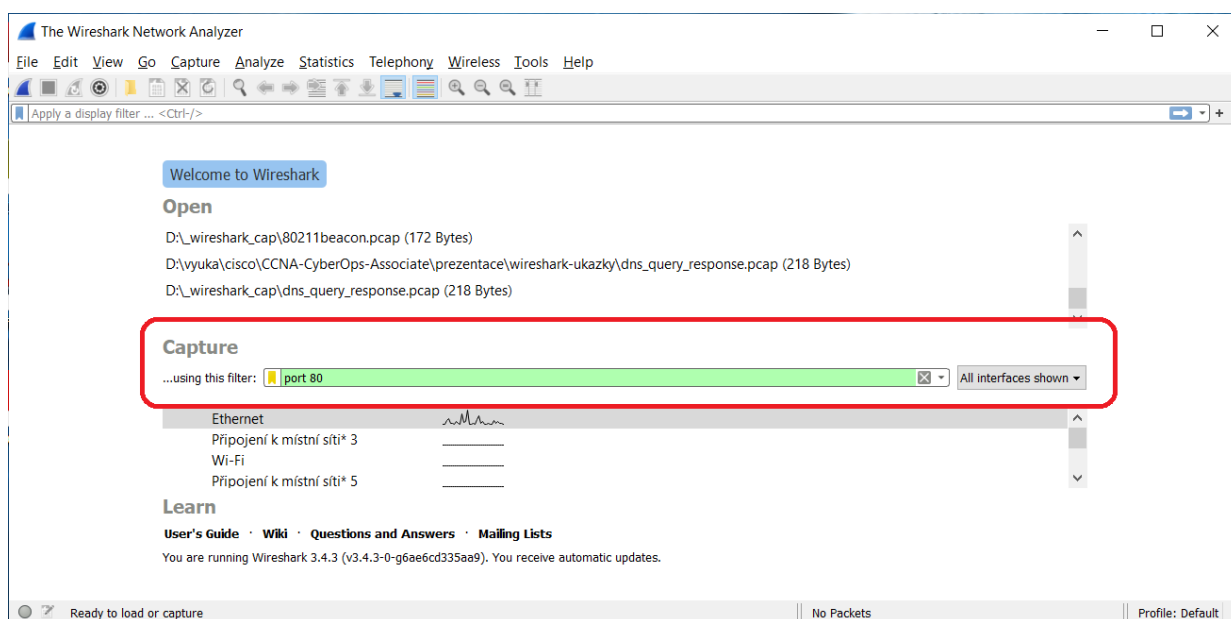
```
port 53
```

Čísla portů můžeme zadat i rozsahem, například pro provoz FTP, Telnet s SSH:

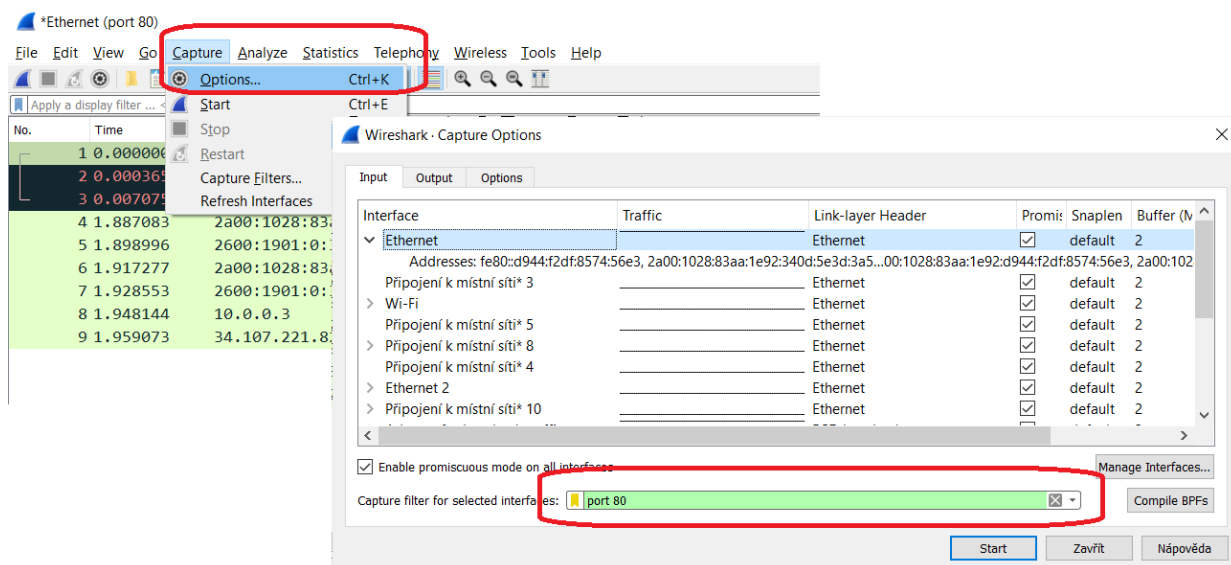
```
tcp portrange 20-23
```

Podmínky můžeme spojovat operátory *and*, *or* a *not*, například:

```
host 10.0.0.18 and not port 53
```



Obrázek 7.3: Zadání filtru zachytávání – provoz související s portem číslo 80



Obrázek 7.4: Filtr pro následné zachytávání

Můžeme zadat protokol, který nás zajímá, což je podobné jako u filtrů zobrazení, například pouze IPv4 provoz, kdy nás nezajímají ani ARP, STP a další ne-IP rámce/pakety:

```
ip
```

Pokud nás zajímá jen unicastová komunikace, můžeme napsat:

```
not broadcast and not multicast
```

Pokud nás naopak zajímá právě IPv6 multicast směřovaný všem IPv6 zařízením, což je cílová adresa `ff02::1`, bude to tento filtr:

```
dst host ff02::1
```

Pro konkrétní protokol můžeme stanovit, co ve kterém poli má být, aby byla taková PDU zachycena. K tomu používáme hranaté závorky podobně, jako jsme to viděli u filtrů zobrazení – obvykle v notaci s dvojtečkou (první číslo je určení oktetu, na kterém má posloupnost začínat,

za dvojtečkou počet oktetů). Například cílový port v UDP segmentu začíná na oktetu číslo 2 (počítáno od 0) a je dlouhý 2 B, tedy komunikace s cílovým portem 53:

```
udp[2:2]==53
```

Totéž lze použít pro jakýkoliv protokol, jenom musíme mít přehled o tom, kde které pole najdeme a jak je dlouhé. Nicméně nás může zajímat třeba jen část určitého pole, třeba u protokolu IP první byte zdrojové IP adresy:

```
ip[12:1]==10
```



Další informace:

<https://wiki.wireshark.org/CaptureFilters>



Úkoly

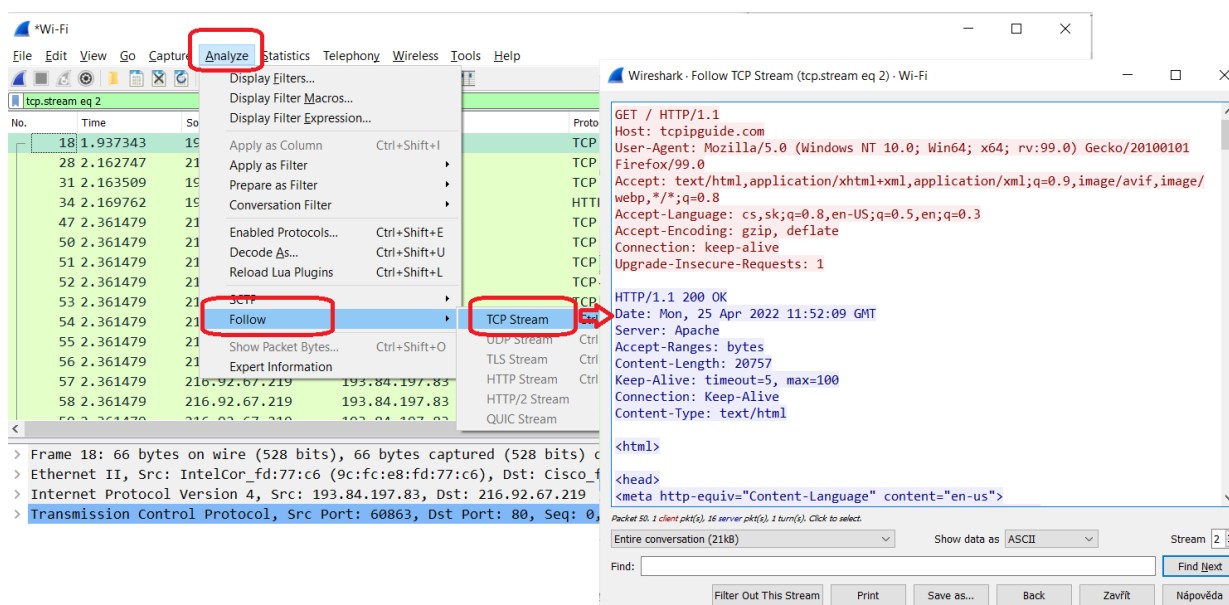
1. Vyzkoušejte alespoň některé z výše uvedených ukázek filtrů zachytávání.
2. Jak zachytíte pouze http provoz na portu 53 (DNS)?
3. Prozkoumejte zdroj nad tímto úkolem a najděte si tam některý zajímavý filtr – všimněte si, že lze používat relační operátory (třeba na čísla portů), a najdeme tady taky filtry pro rozpoznávání komunikace určitých konkrétních červů.



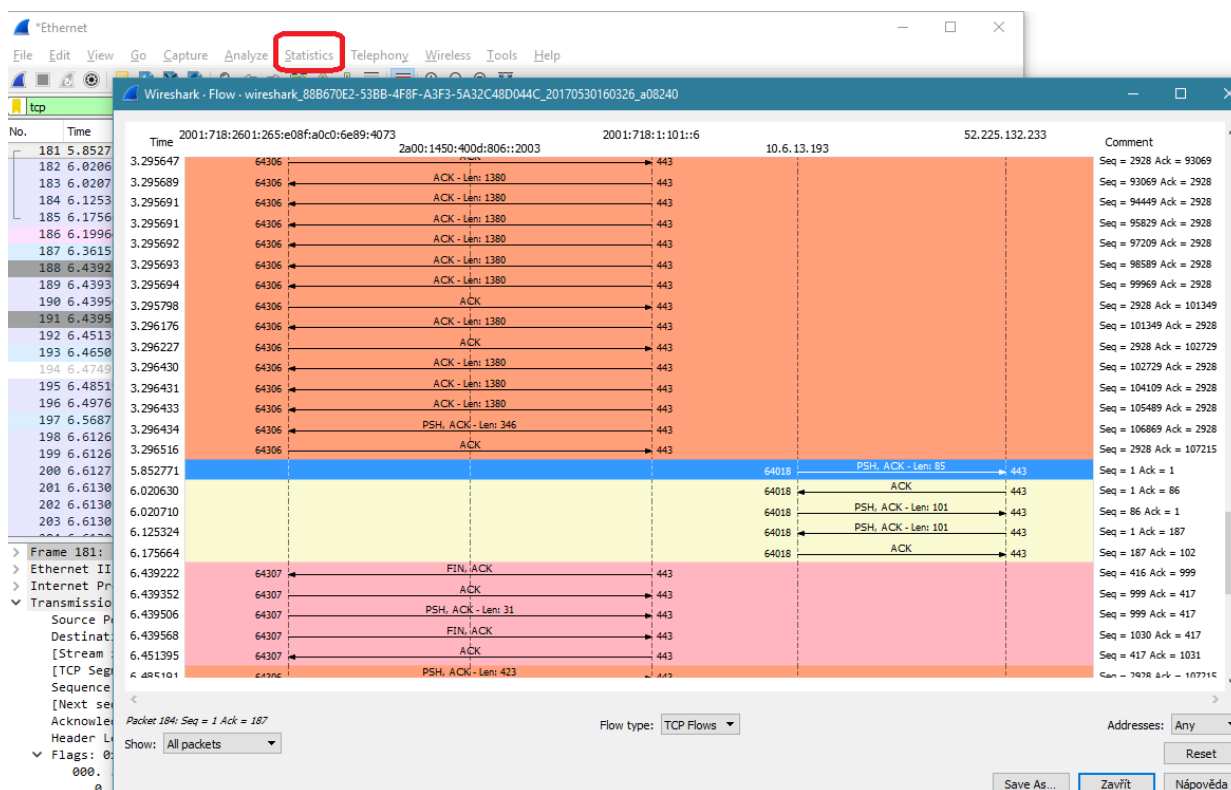
7.4.3 Analýza a statistika ve Wiresharku

 Pokud si chceme prohlédnout celou komunikaci s určitým serverem, můžeme buď postupně procházet jednotlivé rámce/pakety/segmenty, nebo si zobrazit *TCP flow*.

Na obrázku 7.5 je ukázáno, jak se k TCP streamu dostaneme. Zobrazí se okno, ve kterém jsou jednotlivé strany barevně odlišeny – klient červeně, server modře, tedy se postupně střídají červené



Obrázek 7.5: Zobrazení TCP streamu

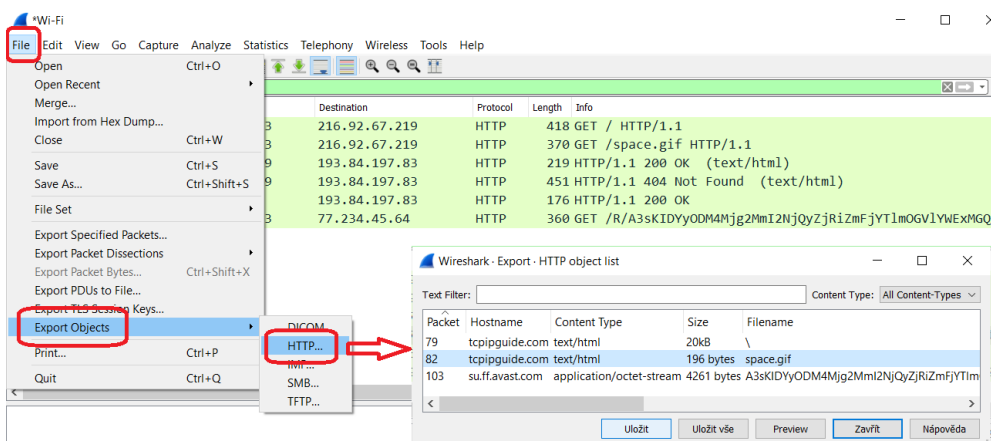


Obrázek 7.6: FlowGraph zobrazený ve Wiresharku, filtr na protokol TCP

a modré oblasti. Na obrázku vidíme, že v TCP streamu je HTTP komunikace, kde klient žádal o stránku a server mu ji poslal.

 Hodnoty Sequence Number (a také další charakteristiky síťového provozu) jsou pěkně vidět například v nástroji Flow Graph, který ve Wiresharku zobrazíme přes menu *Statistics* ⇔ *Flow Graph*, viz obrázek 7.6 (na obrázku jsou relativní čísla).

 Můžeme také exportovat objekty (například přenášené soubory). Na obrázku 7.7 je ukázáno, jak se k příslušné volbě dostat. Získáme takto všechny zachycené objekty bez ohledu na to, jaký je momentálně nastavený filtr zobrazení, samozřejmě kromě toho, co je v šifrovaném provozu. Vybraný objekt, třeba soubor, si můžeme uložit tlačítkem ve spodní části okna.



Obrázek 7.7: Export objektů z provozu



Úkoly

1. Vyzkoušejte zobrazení TCP streamu na web, který nešifruje (třeba ten, který používáme zde – <http://tcpipguide.com/>). Všimněte si, že v okamžiku, kdy v menu zvolíte zobrazení TCP streamu, se nastaví filtr zobrazení.
2. K uvedenému webu si zobrazte Flow graph (předem zrušte filtr). Podívejte se také na další volby v menu Statistics.
3. V zachyceném provozu si vyzkoušejte také exportování objektů.



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. Následuje sekce o Telnetu a SSH, ve které jde především o to jak vypadá PDU příslušného protokolu. Následuje sekce o získávání statistických informací na koncovém zařízení pomocí příkazu `netstat` a závěrem se naučíme konfigurovat DHCPv4 server.

 *Klíčová slova:* DNS, WHOIS, Telnet, SSH, netstat, DHCPv4 server.

 *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. Dále se naučíte nakonfigurovat DHCPv4 server na routeru.

8.1 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 8.1

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

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

Pokud jsme v některém UNIXovém systému, pak pro opačný překlad (známe IP adresu a chceme jmennou adresu) se to dělá stejně, jen jako parametr bude IP adresa. Ve Windows 10 a novějších je implementace příkazu `nslookup` omezená, musíme použít okliku:

```
ping -a ip.adresa.kterou.testujeme
```

V jakémkoliv operačním systému se program `nslookup` dá používat i interaktivně: napíšeme příkaz bez jakýchkoliv parametrů a dostaneme se do vnitřního prostředí, kde přímo zadáváme adresy k překladu, případně po zadání symbolu  se vypíše nápověda s informací o různých možnostech, jak překlad ovlivnit. Z interaktivního režimu se dostanete ven příkazem `exit`.



Úkoly

1. Vyzkoušejte také interaktivní režim příkazu `nslookup`, včetně výpisu nápovědy pomocí .
2. Vyzkoušejte si použití příkazu `nslookup` na některém „velkém“ serveru (Google.com, Apple.com, Microsoft.com, ...). Po zjištění IP adresy použijte reverzní překlad (v Linuxu a v MacOS také pomocí `nslookup`, ve Windows použijte výše uvedený postup s příkazem `ping`).

Zřejmě zjistíte, že se vypíše zcela jiný název domény než z jakého jste původně vycházeli, TLD doména bude pravděpodobně `.net`. Velké firmy totiž využívají služeb společností provozujících CDN sítě (Content Delivery Network), což je síť zrcadel původního serveru rozmístěných na různých místech planety. Uživatel ve skutečnosti komunikuje s tím zrcadlem, které je mu nejbližší.

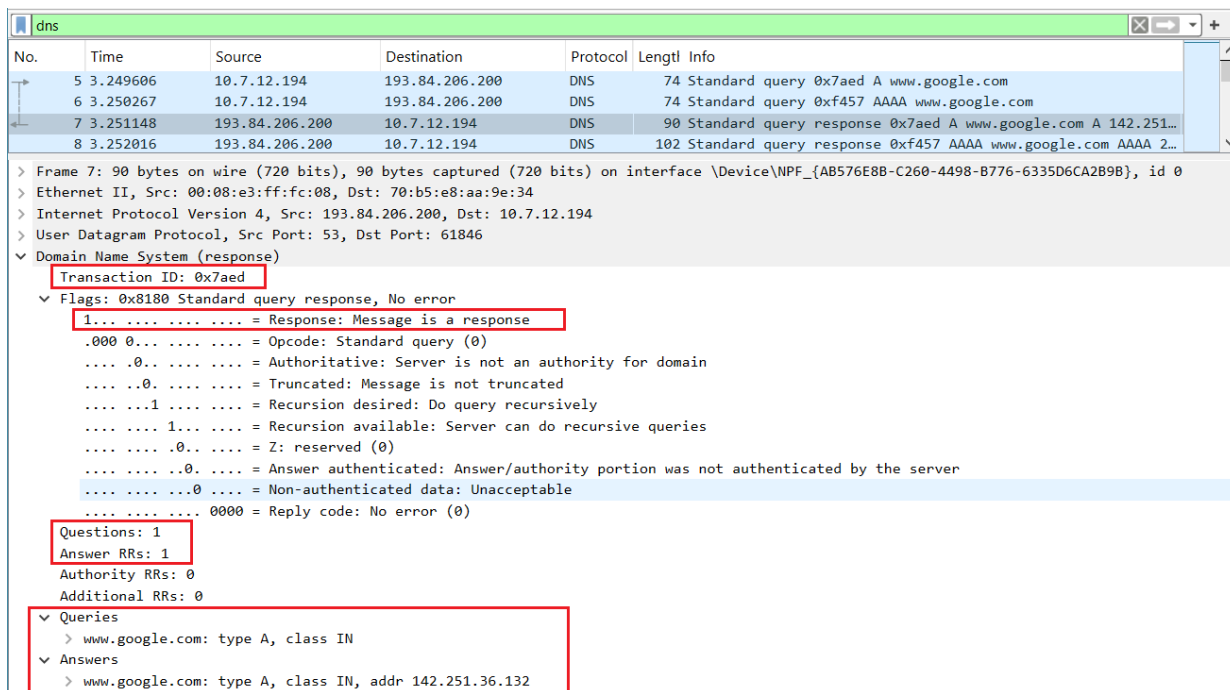


DNS pakety se většinou zapouzdřují do UDP segmentů. Podíváme se na jejich strukturu.



Příklad 8.2

Spusťte zachytávání provozu ve Wiresharku a ve webovém prohlížeči si otevřete některou webovou stránku. Pak ukončete zachytávání a vyfiltrujte si pouze DNS provoz. Prohlédněte srovnajte formát dotazu (Standard Query) a odpovědi (Standard Query Response).



The screenshot shows a Wireshark capture of a DNS Standard Query Response packet. The packet list at the top shows four packets: two queries and two responses. The selected packet is a response from 193.84.206.200 to 10.7.12.194 for the domain www.google.com. The packet details pane shows the following information:

- Transaction ID: 0x7aed
- Flags: 0x8180 Standard query response, No error
- 1... .. = Response: Message is a response
- .000 0... .. = Opcode: Standard query (0)
- ... 0... .. = Authoritative: Server is not an authority for domain
- ... 0... .. = Truncated: Message is not truncated
- ... 1... .. = Recursion desired: Do query recursively
- ... 1... .. = Recursion available: Server can do recursive queries
- ... 0... .. = Z: reserved (0)
- ... 0... .. = Answer authenticated: Answer/authority portion was not authenticated by the server
- ... 0... .. = Non-authenticated data: Unacceptable
- ... 0000 = Reply code: No error (0)
- Questions: 1
- Answer RRs: 1
- Authority RRs: 0
- Additional RRs: 0
- Queries
 - > www.google.com: type A, class IN
- Answers
 - > www.google.com: type A, class IN, addr 142.251.36.132

Na obrázku vidíme komunikaci s DNS serverem, kterého se webový prohlížeč zeptal na adresu `www.google.com`. Byly odeslány dva dotazy (A, AAAA), tedy chceme IPv4 i IPv6 adresu. Ve skutečnosti mohly být oba dotazy v jediné DNS zprávě, ale to záleží na systému...

Na obrázku je vybrána DNS odpověď, dotaz by měl podobnou strukturu, jen by pole související s odpovědí nebyla použita. DNS zpráva je zapouzdřena v UDP datagramu, který je zapouzdřen v IPv4 paketu, a to u obou typů dotazů/odpovědí. Z toho vyplývá, že klient sice zvládá obě verze IP, ale v komunikaci s místním DNS serverem byla zvolena verze 4.

Hned první pole v DNS zprávě je Transaction ID. Toto číslo slouží k párování dotazu a odpovědi. Následují příznaky, ve kterých nás zajímá především příznak určující, jestli se jedná o dotaz nebo odpověď. Za příznaky jsou čtyři pole s čísly, z nichž nás zajímají první dvě: určují počet zapouzdřených dotazů a odpovědí. Jde o DNS zprávu „Query Response“, ve které máme jeden dotaz a jednu odpověď, ty jsou vyznačeny dole.



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
```



Úkol

Spusťte si Wireshark. Zvolte si vhodnou webovou adresu v prohlížeči a 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) nebo vyčistěte DNS cache.

Prohlédněte si DNS dotaz a DNS odpověď. Všimněte si čísel portů v UDP záhlaví (pro dotaz a odpověď), a dále v DNS paketech:

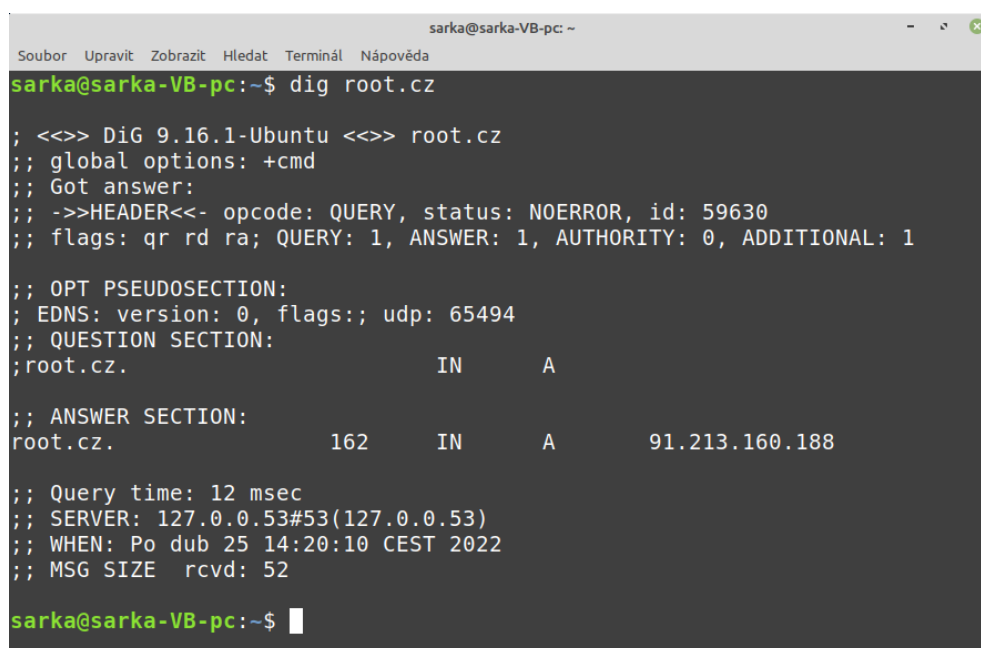
- jak je formulován dotaz (tedy na co všechno se tážeme),
- co všechno je v odpovědi,
- jak je v odpovědi nastavena hodnota TTL (tedy jak dlouho bude mít náš resolver tento záznam v cache paměti).



 V UNIXových systémech máme prográmeček s poněkud rozsáhlejšími možnostmi – `dig`. V základu vypisuje v podstatě to, co je součástí DNS zprávy, třebaže je možné si vynutit zkrácený výpis podobný tomu z `nslookup`.

Na obrázku 8.1 je běžný výstup pro případ, že jako parametr zadáme jméno pro překlad. Pokud chceme provést zpětný překlad (IP adresu na jméno), pak musíme použít přepínač `-x`, aby nástroj měl jistotu, že chceme použít záznam typu PTR (pro zpětný překlad):

```
dig -x 91.213.160.188
```



```
sarka@sarka-VB-pc: ~  
Soubor Upravit Zobrazit Hledat Terminál Nápověda  
sarka@sarka-VB-pc:~$ dig root.cz  
  
; <<>> DiG 9.16.1-Ubuntu <<>> root.cz  
;; global options: +cmd  
;; Got answer:  
;; ->HEADER<- opcode: QUERY, status: NOERROR, id: 59630  
;; flags: qr rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 0, ADDITIONAL: 1  
  
;; OPT PSEUDOSECTION:  
;; EDNS: version: 0, flags:;, udp: 65494  
;; QUESTION SECTION:  
;root.cz. IN A  
  
;; ANSWER SECTION:  
root.cz. 162 IN A 91.213.160.188  
  
;; Query time: 12 msec  
;; SERVER: 127.0.0.53#53(127.0.0.53)  
;; WHEN: Po dub 25 14:20:10 CEST 2022  
;; MSG SIZE rcvd: 52  
  
sarka@sarka-VB-pc:~$
```

Obrázek 8.1: Výstup příkazu dig

Také je možné zadat typ záznamů, které nás zajímají, třeba A, AAAA, MX, NS a další, případně si vyžádat zkrácený výpis (tj. jen výsledek překladu) parametrem `+short`. Například pokud chceme mail servery pro doménu `slu.cz` ve zkráceném formátu, napíšeme:

```
dig slu.cz MX +short
```



Úkol

Pokud máte možnost pracovat v Linuxu, MacOS nebo některém jiném UNIXovém systému, vyzkoušejte si příkaz `dig`. Srovnajte plný a zkrácený výpis.



 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/> zjistěte 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.

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



8.2 Protokoly 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 přenáší nešifrovaně jak data, tak i autentizační informace (credentials). Proto je vhodnější místo Telnetu používat protokol SSH.



Úkoly

1. Ze stránky <https://wiki.wireshark.org/SampleCaptures> si stáhněte soubor `telnet-raw.pcap` 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“.

Aby případný útočník nemusel jít „po písmenkách“, může využít zobrazení TCP streamu, s čímž jsme se setkali na konci předchozí kapitoly, str. 122.

2. Na odkazu <https://www.cloudshark.org/captures/285a99978f07> je ukázka provozu protokolu SSH. 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. Počáteční dojednávání parametrů je poměrně zdlouhavé, data se přenáší od cca 18. paketu.



8.3 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 8.3 (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 vypísáno 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 8.4 (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.



Úkol

Ve svém operačním systému použijte příkaz `netstat` s takovými přepínači, aby se vypsaly všechny informace o TCP i UDP portech včetně naslouchajícího či komunikujícího procesu. Všechny porty ve sloupci pro místní adresu jsou na vašem systému *otevřené*. Pokud se jedná o některý port z rozsahu well-known portů, který neznáte, najděte ho na webu <https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml>.



8.4 DHCP a možnosti získání IP adresy

8.4.1 Konfigurace DHCPv4



IPv4 adresu může zařízení získat některým z těchto způsobů:

- dynamicky z DHCP serveru,
- staticky, tedy ručně zadáme do konfigurace zařízení všechny potřebné údaje,
- statickou alokací, kdy se sice použije DHCP server (zařízení při připojení do sítě požádá o adresu atd.), ale v konfiguraci DHCP serveru máme přímo uvedeno, že zařízení s danou MAC adresou má vždy dostat předem určenou IP adresu.

Třetí možnost je nad rámec tohoto předmětu, druhou možnost už zvládáme, takže se podíváme na tu první.

Jako DHCP server může sloužit „opravdový“ server vyhrazený pro tento účel, nebo takto můžeme naprogramovat router (nebo L3 switch). Výhodou DHCP serveru běžícího na routeru je, že router je viditelný ve všech sítích, které jsou k němu připojeny, a tedy nemusíme řešit přeposílání žádostí o adresu mezi sítěmi.



Postup je následující:

- Promyslíme si, které adresní rozsahy budeme přidělovat dynamicky (určitě NE rozsahy určené pro servery, tiskárny, monitorovací zařízení nebo jiné, které se obvykle přidělují staticky). Typicky se jedná o podsítě, ve kterých jsou běžní klienti.
- Z každého adresního rozsahu vyhradíme část adres, která nebude podléhat přidělování. Například adresa brány, která bývá na nasazená na rozhraní routeru, musí být z přidělovaného rozsahu vyřazena, ale pro jistotu takto vyřazujeme cca 10 adres, podle situace v síti.
- Teď můžeme vytvořit tzv. *pool* adres pro přidělování a přiřadit na konkrétní rozhraní.



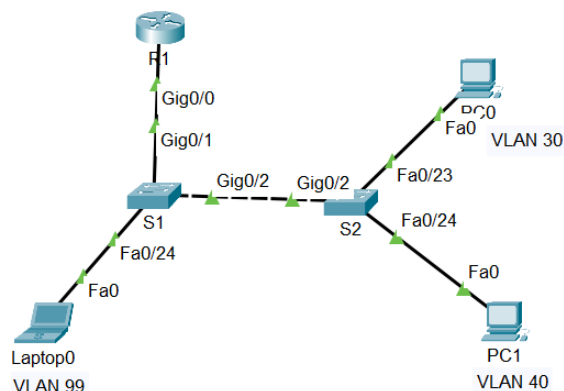
Příklad 8.5

Vytvoříme síť, ve které jsou tři VLAN sítě: VLAN 30 pro zaměstnance, VLAN 40 pro hosty a VLAN 99 pro management (správu sítě). Schéma pro IP adresy bude jednoduché:

10.číslo_VLAN.0.0/16

Tedy druhý Byte adresy určuje VLAN. Vždy první adresa z rozsahu (.1) bude určena pro bránu (tj. bude na routeru). Přes DHCP budeme řešit síť pro zaměstnance a pro hosty. V síti pro zaměstnance budeme přes DHCP přidělovat adresy od .11, prvních 10 adres vyjmeme z přidělování. V síti pro hosty bude pouze první adresa v rozsahu vyjmuta z přidělování, ostatní necháme pro DHCP.

Nejdřív nakonfigurujeme VLAN, trunky, porty na switchích. Na switchi S1 nastavíme název, pak vytvoříme jednotlivé VLAN sítě a nastavíme porty (g0/1 a g0/2 budou trunky, port f0/24 bude přístupový pro VLAN 99).



```
Switch#configure terminal
Switch(config)#hostname S1
S1(config)#vlan 30
S1(config-vlan)#name zamestnanci
S1(config-vlan)#vlan 40
S1(config-vlan)#name hoste
S1(config-vlan)#vlan 99
S1(config-vlan)#name management
S1(config-vlan)#int g0/1
S1(config-if)#switchport mode trunk
S1(config-if)#int g0/2
S1(config-if)#switchport mode trunk
S1(config-if)#int f0/24
S1(config-if)#switchport mode access
S1(config-if)#switchport access vlan 99
```

Na switchi S2 provedeme téměř totéž, jen trunkový port bude jen jeden (g0/2), port f0/23 bude přístupový pro VLAN 30, port f0/24 přístupový pro VLAN 40.

```
Switch>enable
Switch#configure terminal
Switch(config)#hostname S2
S2(config)#vlan 30
S2(config-vlan)#name zamestnanci
S2(config-vlan)#vlan 40
S2(config-vlan)#name hoste
S2(config-vlan)#vlan 99
S2(config-vlan)#name management
S2(config-vlan)#int g0/2
S2(config-if)#switchport mode trunk
S2(config-if)#int f0/23
S2(config-if)#switchport mode access
S2(config-if)#switchport access vlan 30
S2(config-if)#int f0/24
S2(config-if)#switchport mode access
S2(config-if)#switchport access vlan 40
```

Na routeru zprovozníme subrozhraní uvnitř rozhraní g0/0, na každém z nich nasadíme IP adresu, která bude brána pro příslušnou podsít/VLAN:

```
Router>enable
Router#configure terminal
Router(config)#hostname R1
R1(config)#int g0/0.30
R1(config-subif)#encapsulation dot1q 30
R1(config-subif)#ip addr 10.30.0.1 255.255.0.0
R1(config-subif)#int g0/0.40
R1(config-subif)#encapsulation dot1q 40
R1(config-subif)#ip addr 10.40.0.1 255.255.0.0
R1(config-subif)#int g0/0.99
R1(config-subif)#encapsulation dot1q 99
R1(config-subif)#ip addr 10.99.0.1 255.255.0.0
R1(config-subif)#int g0/0
R1(config-if)#no shut
```

Zůstaneme na routeru a zprovozníme DHCPv4 server. Nejdřív vyjmeme z přidělování prvních 10 adres ze sítě pro VLAN 30 a první adresu ze sítě pro VLAN 40. Pak vytvoříme oba pooly, ve kterých stanovíme adresu brány (**default-router** a adresu sítě, jejíž adresy mají být přidělovány). Opravdu je nutné NEJDŘÍV sdělit, které adresy se nemají přidělovat a až POTOM vytvořit pool, protože klienti mohou být velice rychlí s posláním žádosti :-).

```
R1(config-if)#exit
R1(config)#ip dhcp excluded-address 10.30.0.1 10.30.0.10
R1(config)#ip dhcp excluded-address 10.40.0.1
R1(config)#ip dhcp pool LAN30-Zamestnanci
R1(dhcp-config)#default-router 10.30.0.1
R1(dhcp-config)#network 10.30.0.0 255.255.0.0
R1(dhcp-config)#exit
R1(config)#ip dhcp pool LAN40-Hoste
R1(dhcp-config)#default-router 10.40.0.1
R1(dhcp-config)#network 10.40.0.0 255.255.0.0
```

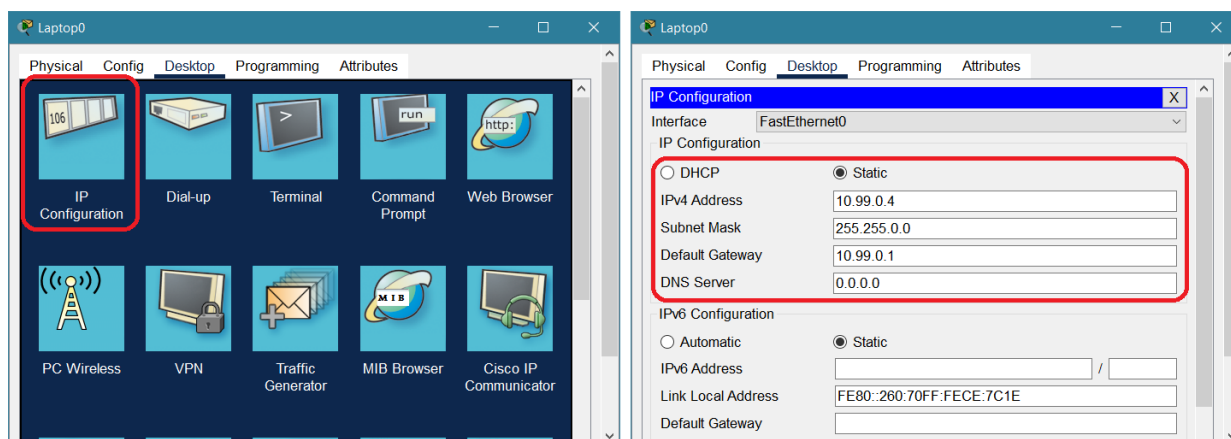
Po zadání příkazu `ip dhcp pool...` jsme se přesunuli do subrozhraní pro konfiguraci DHCP. Určení brány a adresy podsítě je povinné, ale můžeme přidat i další informace, které chceme během přidělování adresy sdělovat klientům. Možnosti zjistíme jednoduše pomocí otazníku:

`R1(dhcp-config)#?`

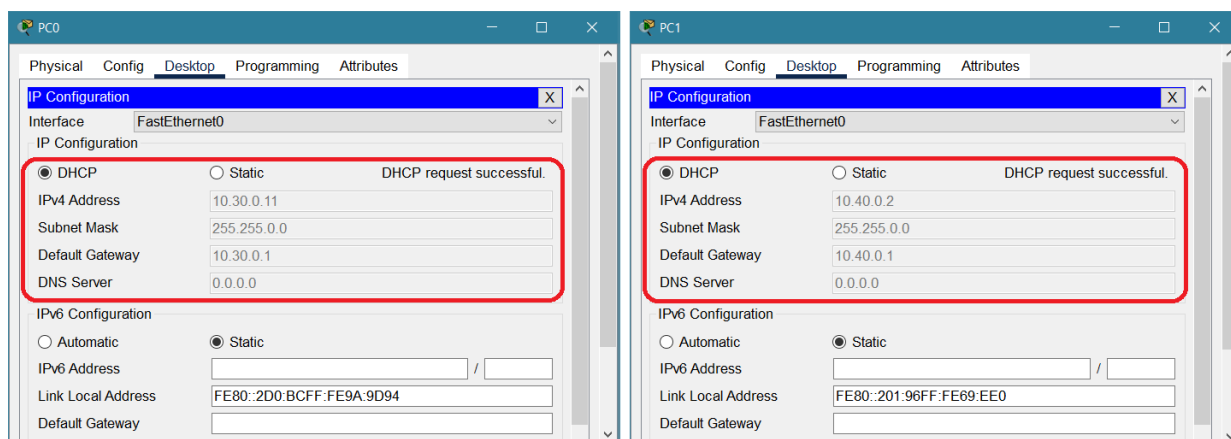
```
default-router  Default routers
dns-server      Set name server
domain-name     Domain name
exit            Exit from DHCP pool configuration mode
network         Network number and mask
no              Negate a command or set its defaults
option          Raw DHCP options
```

Volitelně lze přidat například název domény, dá se také nastavit doba pronájmu přidělené adresy, tedy lease time (adresu totiž přidělujeme jen na „dobu určitou“, pak musí klient požádat o prodloužení pronájmu). Tento výpis je z Packet Traceru, který má poněkud osekané možnosti, nastavení lease time a některé další možnosti zde tedy nevidíme.

Na notebooku, který má patřit do VLAN 99, nastavíme IP adresu, masku a bránu staticky (ručně):



Na počítačích, které mají získat toto vše od protokolu DHCP, zatrhneme volbu pro DHCP, pak by měly počítače požádat o IP adresu do sítě, což se také stalo. Jak vidíme, oba dostaly hned první adresu následující za vyjmutými adresami:



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 vypsát 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 vypsát směrovací tabulku na koncovém zařízení.

9.1 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 9.1

Také v operačním systému si můžeme vypsat 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/> najděte 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.



9.2 Práce s IPv4 adresami

9.2.1 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).



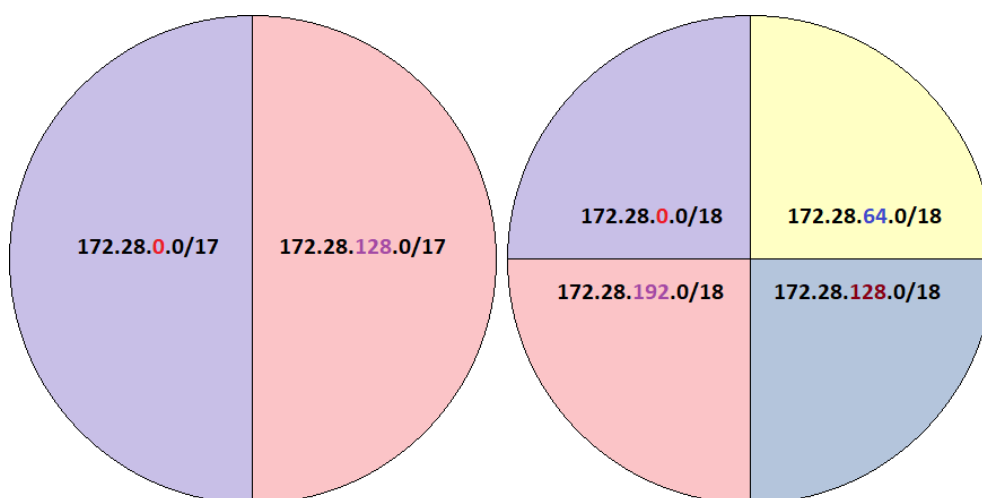
Takže adresa se při podsítování skládá ze tří částí:

- 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 9.1: Struktura IP adresy při použití podsítování



Obrázek 9.2: Rozdělení sítě 172.28.0.0/16 na dvě a čtyři podsítě

Na obrázku 9.2 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 9.2

Vezměme síť 172.28.0.0/16, o které byla zmínka dříve a platí pro ni obrázek 9.2 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 9.3

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 9.4**

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 9.1: 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 9.5**

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, podsítí 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 9.2: 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 9.3. Nenajdete tam všech 16 podsítí, jen čtyři první a dvě poslední.



Příklad 9.6

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

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 9.3: Vytvoření šestnácti podsítí pro síť 10.0.0.0/8

musí 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 9.4. 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 9.4: 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 9.5: 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 9.7

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 • 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 • 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 • 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 • 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 • 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 • 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 • 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 • první hostitel: 10.255.196.1 • poslední hostitel: 10.255.255.254 • broadcast: 10.255.255.255

Tabulka 9.6: Rozdělení sítě 10.0.0.0/8 na podsítě s využitím 10 bitů

Údaje o několika prvních a dvou posledních podsítích jsou v tabulce 9.6, 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 9.7: 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 podsítovou část vyhradíte pouze jeden bit. Pak vyzkoušejte případ, kdy pro podsítovou 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.

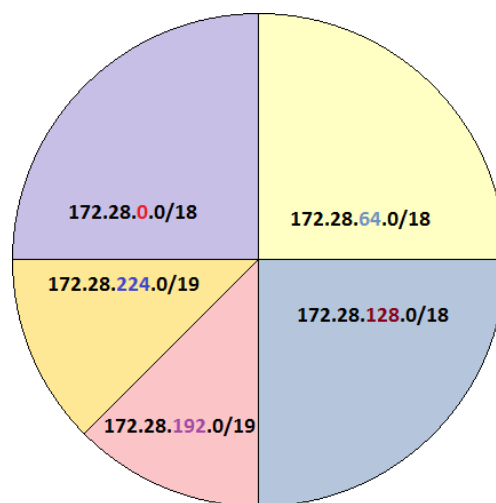


9.2.2 VLISM

Masky podsítí s proměnnou délkou – VLISM (Variable Length Subnet Mask) – jsou mechanismem rozšiřujícím možnosti podsítování. Zatímco u podsítování jsme si pevně stanovili, kolik bitů bude pro adresu podsítě, u VLISM se tento počet bitů může u různých podsítí měnit.

Při podsítování máme pro každou podsít tentýž maximální počet hostitelů, při použití VLISM 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)síť můžeme rozdělit na dvě, čtyři, osm, ... podsítí. Na obrázku 9.3 vidíme, že síť 172.28.0.0/16 byla rozdělena nejdřív 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 podsíť 172.28.128.0/18). Důležitým pravidlem je, že dělíme síť na konci rozsahu.



Obrázek 9.3: Rozdělení sítě 172.28.0.0/16 na tři větší a dvě menší podsítě



Příklad 9.8

Navážeme na příklad na straně 137. Tam jsme dělili 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 137. Všimněte si, že v tomto příkladu nám první podsít' vlastně „spolkla“ původní první a druhou podsít' z předchozího příkladu (tj. obě původní podsítě začínají 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 9.8: 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.



...	/21	/22	/23	/24	/25	/26	/27	/28	/29	/30
...	2048	1024	512	256	128	64	32	16	8	4

Tabulka 9.9: Vztah mezi délkou prefixu a počtem adres v podsíti

 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.

Z tabulky 9.9 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 9.9

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 9.10.



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 9.10: 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



9.3 Směrování

9.3.1 Směrovací tabulka

Směrovací tabulka je na každém zařízení s implementací vrstvy L3, jejím úkolem je určovat, co se má stát s odesílaným IP paketem.



Úkol

Vypište svou směrovací tabulku na PC. Jak se to dělá:

- ve Windows: `route print`
- v Linuxu a na MacOS: `route` nebo `ip route show`



Na routeru můžeme přidat do IPv4 směrovací tabulky statický záznam jedním z těchto způsobů:

```
ip route cílová_síť port
ip route cílová_síť soused
```

V prvním případě sdělujeme, že do cílové sítě vede cesta přes (náš) zadaný port, v druhém případě zadáváme adresu našeho L3 souseda, přes kterého vede do dané sítě cesta.



Příklad 9.10

Chceme do směrovací tabulky staticky přidat záznam o tom, že síť 10.20.0.0/16 je dostupná za naším portem g0/1:

```
ip route 10.20.0.0 255.255.0.0 g0/1
```

Nebo můžeme uvést, že je dostupná přes souseda s IP adresou 192.168.0.1:

```
ip route 10.20.0.0 255.255.0.0 192.168.0.1
```

Pro IPv6 je to podobné – síť 2001:a:b::/64 má být dostupná přes rozhraní g0/1, resp. přes souseda fe80::2:

```
ipv6 route 2001:a:b::/64 g0/1
```

```
ipv6 route 2001:a:b::/64 fe80::2
```



Paket se podle směrovací tabulky zpracovává tak, že se jeho cílová adresa porovnává s jednotlivými záznamy v tabulce a zjišťuje se, jestli tato adresa spadá do sítě z tohoto řádku. Porovnává se jen tolik bitů z adresy, kolik je stanoveno maskou u záznamu v tabulce. Pokud adresa spadá do několika řádků, zvolí se ten nejkonkrétnější, tedy s nejdelší maskou.



Příklad 9.11

Má být odeslán paket s cílovou adresou 10.20.45.112. Ve směrovací tabulce odpovídají tyto záznamy:

10.0.0.0	255.0.0.0	g0/1	...
10.20.0.0	255.255.0.0	g0/2	...

To není chyba, může to například znamenat, že ze sítí, jejichž první Byte je 10, je síť 10.20.0.0/16 za portem g0/2, ostatní za portem g0/1.

V tomto případě je pro odesílaný paket zvolena druhá možnost, protože záznam je konkrétnější – určuje 16 bitů, kdežto předchozí záznam pouze 8 bitů.



Default route je na routeru obdoba výchozí brány: pokud router pro odesílaný paket nenajde ve směrovací tabulce uvkonkrétní záznam, podle kterého by určil, co s paketem, pošle ho na default route. Obvykle míří k našemu ISP (poskytovateli internetu), ale samozřejmě záleží na tom, jak konkrétně jsme ke svému ISP (resp. svým ISP) připojeni.

Default route pro IPv4 a IPv6 určíme takto (v tomto případě stanovíme směr určením síťového rozhraní g0/1, ale samozřejmě to jde i s adresou souseda):

```
ip route 0.0.0.0 0.0.0.0 g0/1
```

```
ipv6 route ::/0 g0/1
```

Proč samé nuly? Protože default route se má použít pouze pro takové pakety, pro které nebyla nalezena shoda s jiným řádkem tabulky. Jak jsme si výše vysvětlili, pokud cílové adrese paketu odpovídá více než jeden záznam tabulky, volí se ten nejkonkrétnější. Default route má nulovou masku, tedy je to bezkonkurenčně nejméně konkrétní záznam, proto se zvolí jen tehdy, když není nalezena shoda s žádným jiným záznamem.

Proto tedy je maska nulová. Když je tedy maska nulová, znamená to, že z adresy se nemá porovnávat ani jediný bit. Takže není důvod, proč do samotné adresy dodávat cokoliv jiného než nuly, stejně se ta adresa při porovnávání nepoužije.

9.3.2 Směrovací protokol RIP pro IPv4

 Ukážeme si, jak zprovoznit v síti naprosto nejjednodušší směrovací protokol, což je RIP. Budeme chtít, aby směroval bez ohledu na třídy adres (tj. classless), tedy chceme verzi 2.

Na routeru (v globálním konfiguračním módu) nejdřív vytvoříme směrovací proces pro protokol RIP tímto příkazem:

```
router rip
```

Tento příkaz nás zároveň přesune do subkonfiguračního módu pro tento směrovací proces. V něm určíme, že chceme verzi 2, a taky vypneme autosumarizaci cest (kdybychom ji nechali zapnutou, tak by se ve směrovací tabulce nebrala v úvahu maska pro danou síť, tedy defacto bychom „sklouzli“ ke směrování se třídami, což nechceme). Potom routeru sdělíme, které připojené sítě má propagovat, tedy přeposílat informace o nich sousedům.

```
router rip
  version 2
  no auto-summary
  network 10.20.0.0
  network 192.168.0.0
  passive-interface f0/2
```

Poslední uvedený příkaz označil rozhraní f0/2 jako pasivní, tedy na toto rozhraní nebudou posílány RIP updaty. To se typicky dělá pro taková rozhraní, za kterými je koncové zařízení, switch, nebo třeba jde o rozhraní vedoucí k poskytovateli internetu. Koncové zařízení ani náš poskytovatel internetu nepotřebuje informace o směrovacích cestách, případně to může být i bezpečnostní riziko.

Máme k dispozici také nějaké ty show příkazy:

```
show ip route
show ip protocols
show ip rip
debug ip rip
```

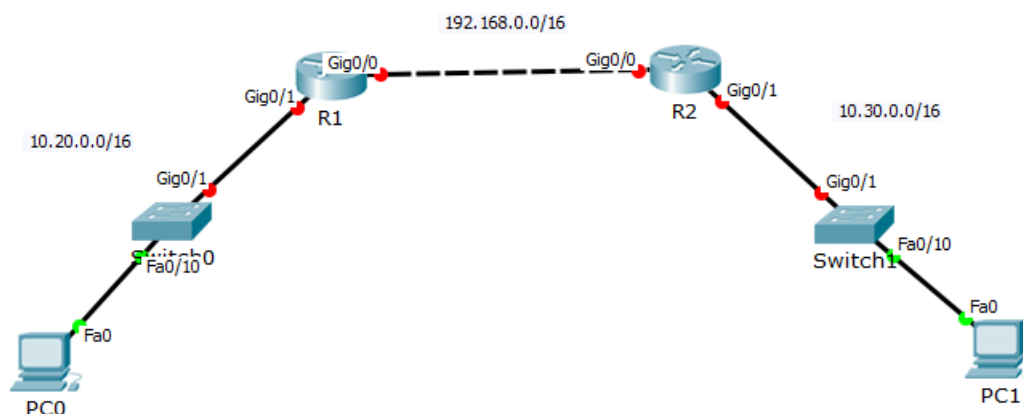
První z uvedených příkazů zobrazuje směrovací tabulku IPv4 (obdobně pro IPv6 by bylo `show ipv6 route`). Další je univerzální pro všechny směrovací protokoly, následující pouze pro RIP. Posledním uvedeným příkazem zapneme debugování protokolu RIP (tj. do konzoly se nám bude vypisovat vše, co se při směrování RIP bude dít). Vypneme příkazem `no debug ip rip`.



Úkol

Vytvořte topologii podle následujícího obrázku (přetekl na následující stranu).

Na routerech zprovozněte daná síťová rozhraní. V případě sítí začínajících „10“ nasadte na routerech adresy s jedničkou v posledním oktetu, na PC adresy s číslem 3. V případě sítě mezi R1 a R2 dejte na straně R1 adresu 192.168.0.1, na straně R2 adresu 192.168.0.2.



Na routeru R1 dále vytvořte loopback a přiřaďte mu adresu 172.16.0.1/16:

```
interface lo0
  ip addr 172.16.0.1 255.255.0.0
  no shut
```

Na routerech zprovozněte RIP podle příkladu výše, propagujte vždy síť vedoucí k sousednímu routeru a síť vedoucí k switchi. Na R2 vytvořte statickou směrovací cestu vedoucí na loopback, který jste vytvořili dříve:

```
ip route 172.16.0.0 255.255.0.0 g0/0
```

(případně změňte rozhraní na to, které jste nasměrovali k routeru R1).

Vyzkoušejte různé show příkazy na routerech. Dále pošlete ping mezi počítači a prověřte, jestli vše funguje jak má. Pingněte taky loopback – postupně z obou PC.

Následně si procvičte i další úkoly: na routeru R1 nastavte heslo do enabled módu na ciscoena, hesla na konzoli a VTY linkách na ciscoline. Zajistěte, ať jsou hesla zabezpečená (aby nebyla viditelná při výpisu konfigurace). Hesla by měla být vyžadovaná.

Na R2 zprovozněte SSH. Vytvořte uživatele admin s heslem ciscoadmin, zajistěte, aby běželo SSH verze 2, vygenerujte klíč, dále zajistěte, aby se přes VTY linky dalo použít pouze SSH, přihlašování přes lokální databázi.



9.3.3 Směrovací protokol OSPF pro IPv4 v použití single area

Protokol RIP je nejstarší používaný směrovací protokol pro vnitřní síť. To není nutně špatně, ale je použitelný jen pro velmi malé sítě routerů, délka cesty je maximálně 15. Navíc má poněkud problém se sumarizací cest. Sice od verze 2 už zvládá práci s maskou, ale jen v základní rovině, podsítě spadající do téže třídy musejí být za stejným rozhraním. To je podstatné omezení. Navíc je v síti poněkud „upovídaný“, pravidelně posílá prakticky celé směrovací tabulky.

Proto bývá obvykle lepší volbou protokol OSPF. Má kvalitnější metriku, lépe se vypořádá se sumarizací cest, je použitelné i v případě, že máme hodně L3 síťových zařízení, jen je mírně náročnější na konfiguraci.

 OSPF umožňuje rozdělit routery do několika *oblastí* (area), přičemž router má ve směrovací tabulce podrobné informace o své vlastní oblasti, kdežto o sítích v jiných oblastech má vždy v

tabulce jen jeden záznam (směr do dané oblasti), což šetří výpočetní kapacitu routeru a urychluje směrování. V menší síti si ale nebudeme komplikovat život různými oblastmi, použijeme jen jedinou oblast (area 0). Nicméně uvedena být musí.

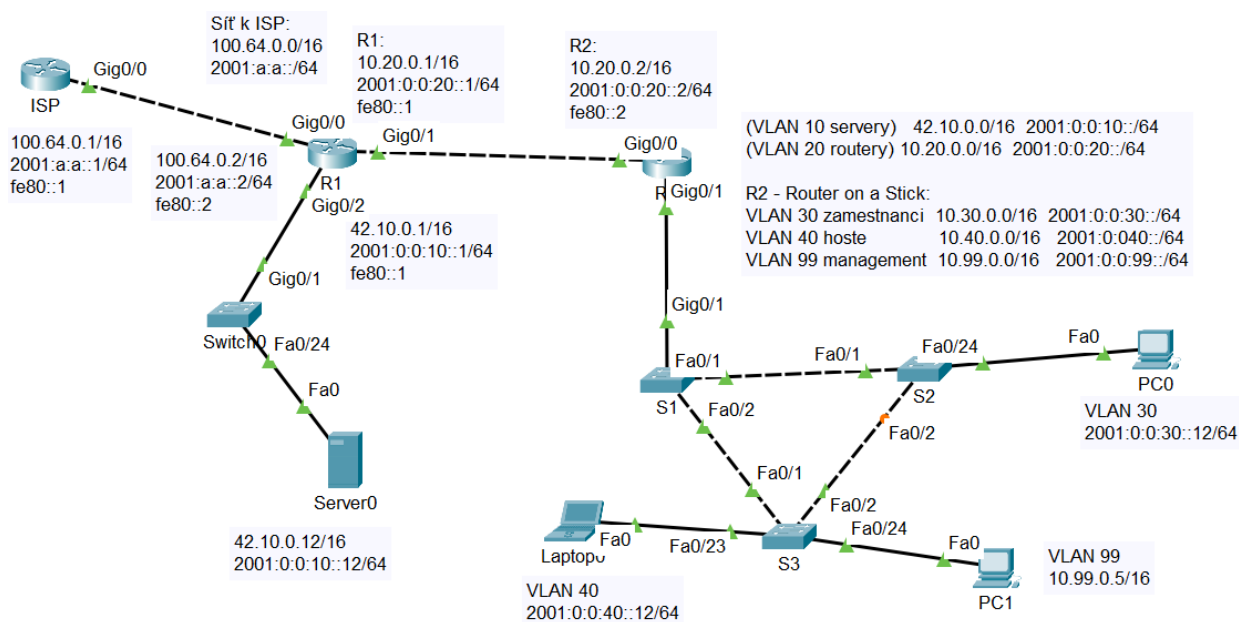
 Další změna oproti RIP je nutnost zadávat *číslo procesu*. Na jednom routeru totiž teoreticky může běžet několik procesů OSPF, každý s jinou konfigurací, i když se tato možnost moc nepoužívá. Nicméně číslo procesu je třeba určit. . .

 Každý směrovací protokol posílá L3 sousedům informace potřebné k vytvoření a aktualizaci záznamů ve směrovací tabulce. OSPF to dělá taky. Posílání těchto informací má smysl jen na ta rozhraní, za kterými je „spřátelný“ L3 soused, který tyto informace potřebuje, v našem případě soused se spuštěným protokolem OSPF. Ostatní rozhraní by měla být označena jako *pasivní*: směr k ISP, a taky směr ke switchům a koncovým zařízením.



Příklad 9.12

Vytvoříme následující síť. Je už poněkud rozsáhlá, protože se jedná o komplexní příklad, ve kterém si procvičíme většinu z toho, co jsme zatím probrali.



Potřebujeme tři routery (ISP, R1, R2, zvolte například model 2911), dále čtyři switche (např. 2960), jeden server a několik dalších koncových zařízení (počítače, notebooky). Router ISP nám simuluje poskytovatele internetu.

Levá část představuje demilitarizovanou zónu – místo pro servery, které mají být dostupné z internetu. Pravá část je čistě soukromá síť, která nemá být na internetu viditelná, nicméně zařízení z této části mohou na internet přistupovat a jsou rozdělena do několika VLAN.

Naším úkolem bude následující:

- IP adresy nastavíme podle obrázku, pokud u PC či notebooku není IPv4 adresa uvedena, bude získána přes DHCP,
- na R2 bude rozhraní g0/1 rozděleno na subrozhraní, protože zde nakonfigurujeme Router-on-a-Stick,

- pomocí statického směrování nastavíme default route mezi ISP a R1 a na ISP staticky nastavíme cestu k síti 42.10.0.0/16, obojí k rozhraní g0/0 (v reálné situaci by to na straně ISP samozřejmě bylo jinak, zde jen potřebujeme ten spoj zprovoznit),
- na R1 zprovozníme OSPF pro IPv4, rozhraní g0/0 a g0/2 označíme jako pasivní, budeme distribuovat default route (tu jsme vytvořili staticky v předchozím bodu),
- na R2 taktéž zprovozníme OSPF pro IPv4, subrozhraní pro jednotlivé VLAN sítě budou označeny jako pasivní, default route zde nebudeme distribuovat, protože její distribuci má na starosti R1,
- na R2 spustíme DHCPv4 pro podsítě odpovídající VLAN 30 a 40, u první vyjmeme prvních deset adres z přidělování, u druhé jednu,
- na switchi S1 zpřístupníme konfiguraci přes SSH, vytvoříme uživatele **admin** s heslem **admin123admin** (nojo, to není zrovna silné heslo, v praxi prosím lepší), autentizace proti lokální databázi, SSH verze 2 s délkou klíče 1024,
-

.....



Také pro OSPF máme k dispozici různé **show** příkazy, například:

```
show ip protocols
show ip ospf neighbor
show ip ospf interface
```

Pokud místo **ip** napíšeme **ipv6**, máme variantu příkazu pro IPv6 směrování. Například **show ipv6 protocols** by vypsal informace o momentálním stavu směrovacích protokolů pro IPv6.



Úkol

...



Kapitola 10

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, WifiInfoView, 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.

10.1 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 zachováváno – nejdřív příjemce, pak odesílatel.

3. Na spoji mezi dvěma AP jsou v záhlaví dokonce čtyři adresy:
 - adresa momentálního příjemce na spoji (zdrojového AP),
 - adresa momentálního odesílatele na spoji (cílového AP),
 - adresa cíle rámce (DTE),
 - adresa zdroje rámce (DTE).

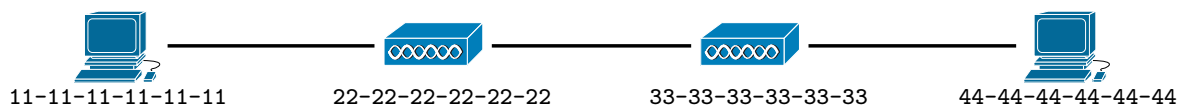
Opět se první dvě adresy vztahují k aktuálnímu spoji, ale samozřejmě musí být v záhlaví uloženo i to, kdo je zdrojem a cílem rámce. V první i druhé dvojici je zachováváno dané pořadí.

4. Na spoji mezi některým AP a cílem rámce (DTE), tedy v závěru cesty, jsou v záhlaví opět „pouze“ tři adresy:
 - adresa momentálního příjemce na spoji (cíle rámce, DTE),
 - adresa momentálního odesílatele na spoji (AP),
 - adresa zdroje rámce (DTE).



Příklad 10.1

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 spojih. *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:**

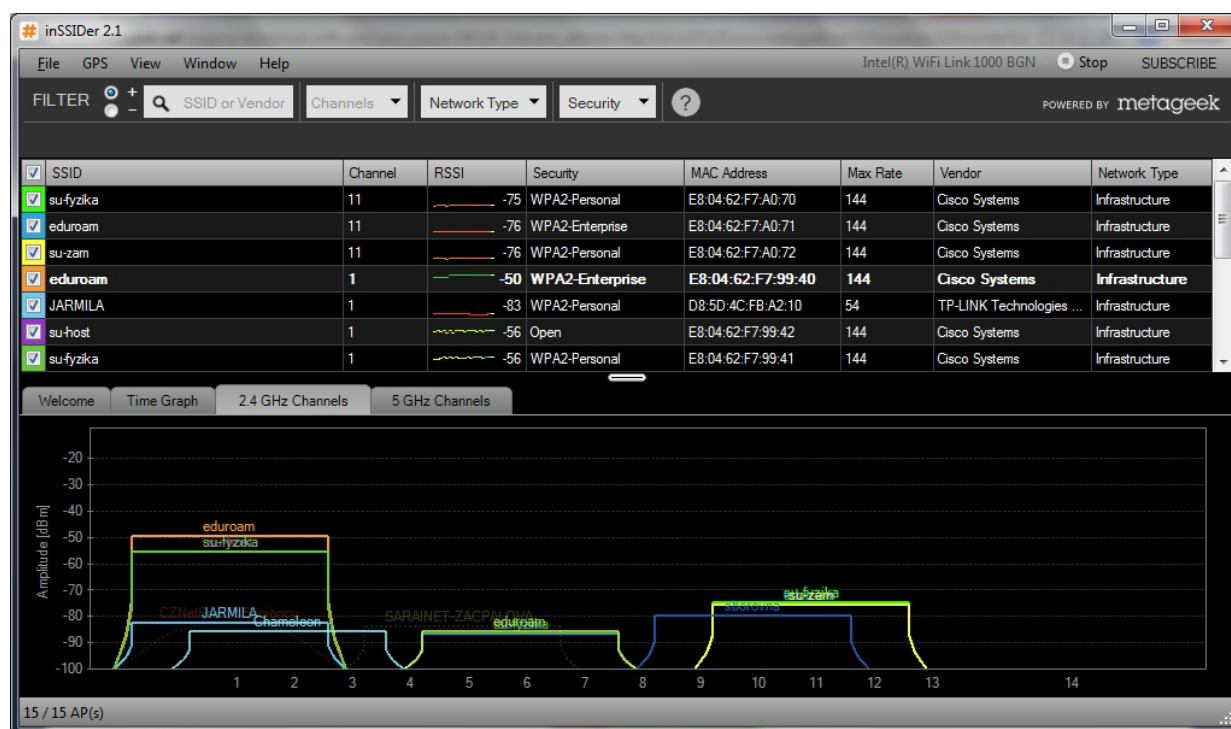
Uvědomte si, že to vše platí v případě, že komunikujete s AP pracujícím na vrstvě L2. Jestliže však jste připojeni do buňky, kde AP pracuje na vrstvě L3 (takže Wi-fi router), pak na vrstvě L2 je cílem právě ten AP, a v rámci budou opravdu jen dvě MAC adresy. Proč? Protože router (jakýkoliv) odděluje 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.



10.2 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.

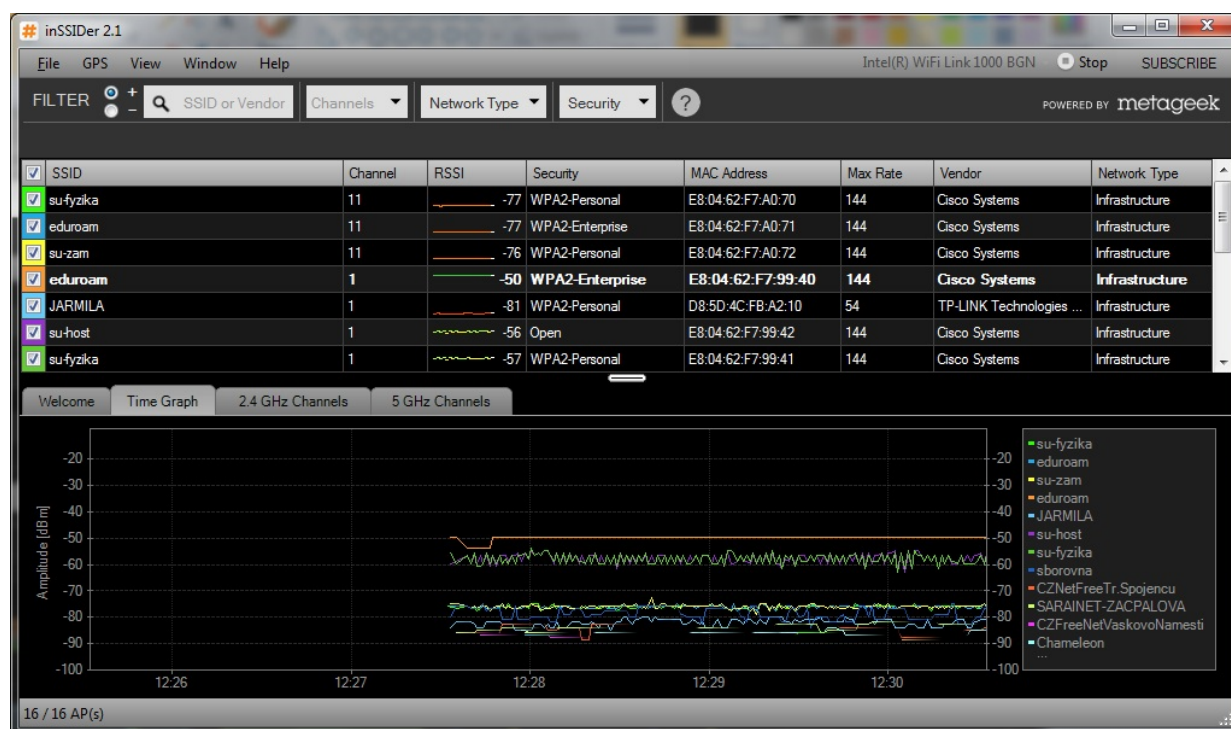


Obrázek 10.1: Program InSSIDer ve Windows

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 10.2: Program InSSIDer, záložka Time Graph

 Program *WifiInfo View* 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.

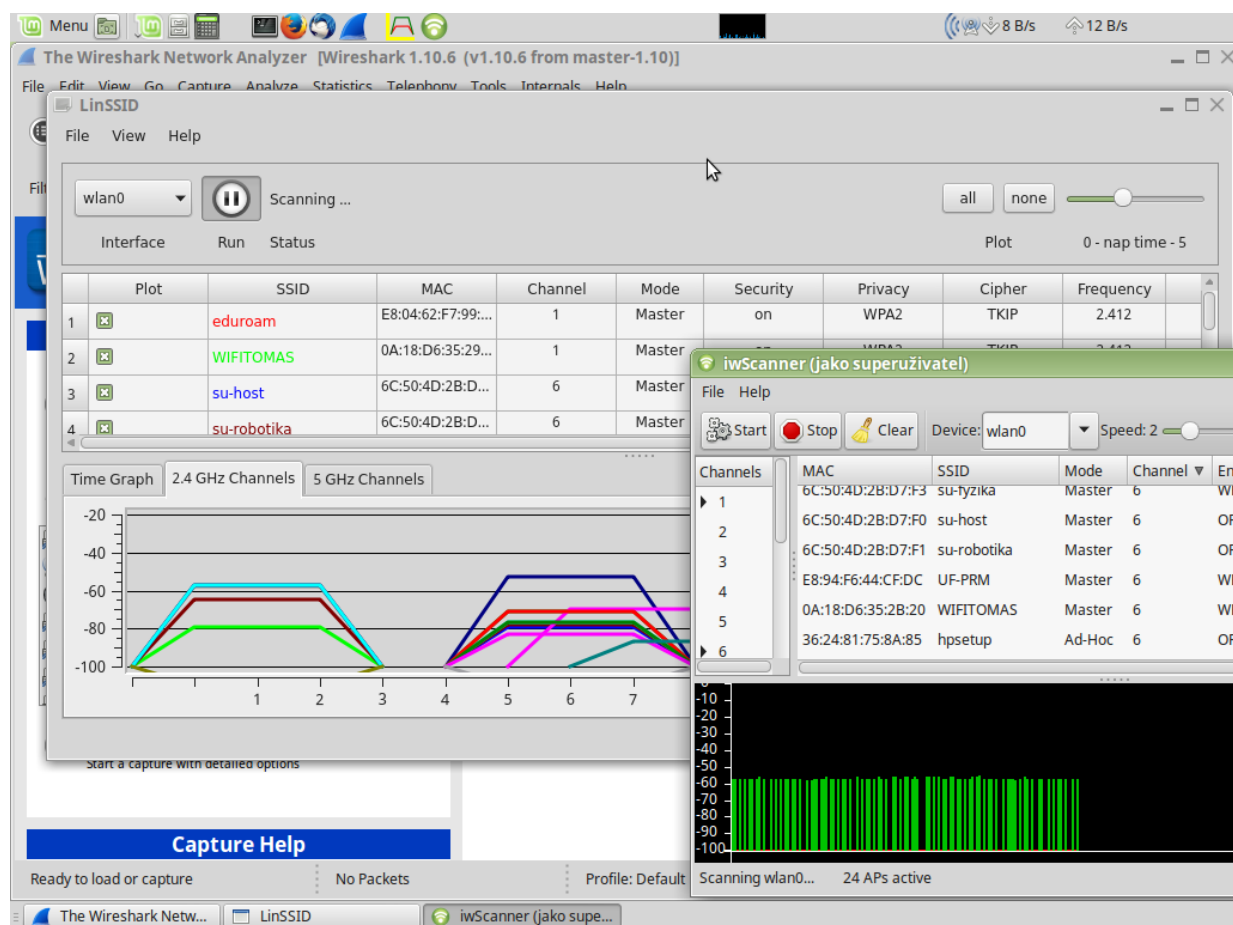


Další informace:

- <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 10.3: 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