

**5.**  
aktualizované  
vydání  
bestselleru

Libor Dostálek, Alena Kabelová

# **Velký průvodce protokoly**

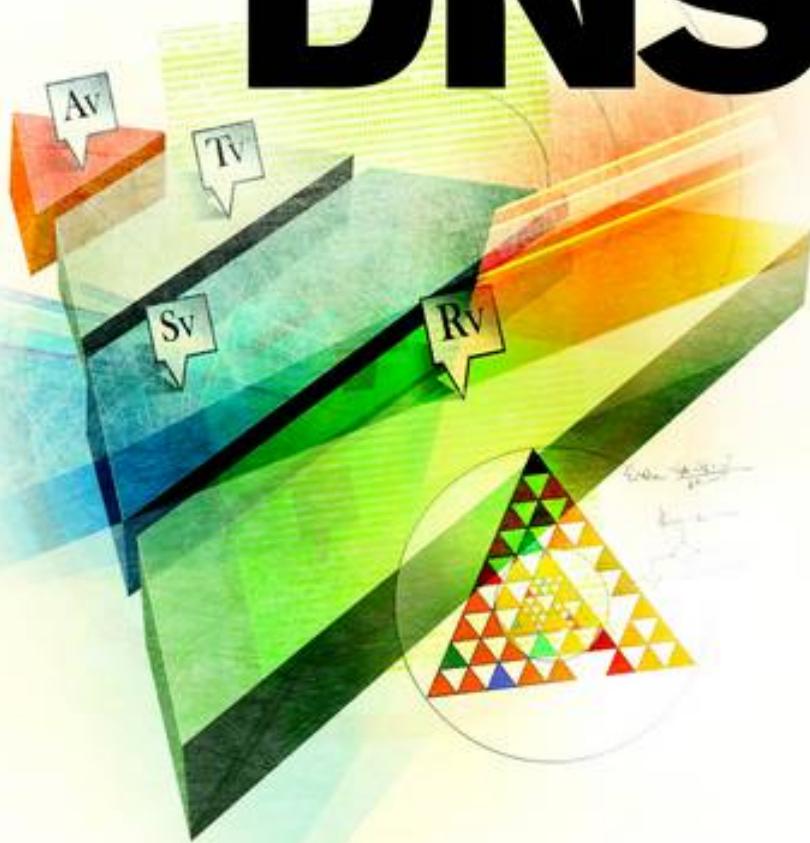
# **TCP/IP a systémem DNS**

Možnosti sledování,  
kontroly a ladění sítě

Využití proxy  
v aplikačních  
protokolech

Firewall přesně na míru  
vašeho systému

Desítky názorných  
příkladů a cvičení  
pro nastavení síťové  
komunikace



**C**PRESS

**Alena Kabelová, Libor Dostálek**

# **Velký průvodce protokoly TCP/IP a systémem DNS 5. aktualizované vydání**

---

**Computer Press  
Brno  
2012**

# **Velký průvodce protokoly TCP/IP a systémem DNS**

## **5. aktualizované vydání**

**Alena Kabelová, Libor Dostálek**

**Kapitola O sdružení CZ.NIC:** Vilém Sládek

**Odborná korektura:** Hana Straková, Jan Říha

**Ilustrace:** Martina Hegerová

**Obálka:** Martin Sodomka

**Odpovědný redaktor:** Radek Hylmar

**Technický redaktor:** Jiří Matoušek

Objednávky knih:

<http://knihy.cpress.cz>

[www.albatrosmedia.cz](http://www.albatrosmedia.cz)

[eshop@albatrosmedia.cz](mailto:eshop@albatrosmedia.cz)

bezplatná linka 800 555 513

ISBN 978-80-251-2236-5

Vydalo nakladatelství Computer Press v Brně roku 2012 ve společnosti Albatros Media a. s. se sídlem  
Na Pankráci 30, Praha 4. Číslo publikace 15 966.

© Albatros Media a. s. Všechna práva vyhrazena. Žádná část této publikace nesmí být kopírována  
a rozmnožována za účelem rozšířování v jakékoli formě či jakýmkoli způsobem bez písemného  
souhlasu vydavatele.

1. dotisk 5. aktualizovaného vydání

 **ALBATROS MEDIA** a.s.

# Obsah

## Úvod ..... 13

## Kapitola 1 Síťové protokoly ..... 15

Jaké protokoly se používají na Internetu?	17
Fyzická vrstva.....	18
Lokální síť (LAN).....	19
Linková vrstva .....	20
IP vrstva .....	21
Vrstva TCP/UDP.....	22
Aplikační vrstva.....	23
Způsoby přenosu informací .....	25
Synchronní přenos .....	25
Paketový přenos.....	26
Asynchronní přenos .....	26
Virtuální okruh.....	26
Pevná a komutované virtuální okruhy.....	28

## Kapitola 2 Nástroje pro sledování a zkoumání sítě. .... 29

Packet driver .....	30
Promiskuitní mód.....	30
Program Wireshark .....	31
Začínáme s Wiresharkem .....	32
Filtры.....	33
Coloring rules.....	36
Follow TCP stream.....	36
Statistiky.....	37
Tisk a Export.....	37
Další utility .....	39
Domácí cvičení.....	39
Program nmap .....	40
Co program nmap dělá? .....	40
První fáze: Vyhledávání připojených systémů.....	40
Druhá fáze: Scanování portů.....	41
Další zajímavé přepínače .....	42
zenmap GUI .....	42
Domácí cvičení.....	42

## Kapitola 3

### Fyzická vrstva ..... 45

Sériová rozhraní .....	46
Sériový a paralelní přenos dat.....	46
Symetrický a asymetrický signál.....	47
Synchronní nebo asynchronní přenos .....	47
Normy V.24, V.35 a X.21 .....	48
Nulový modem.....	51
Modemy .....	52
Analogová linka .....	52
Modem je „automatický“.....	53
AT-příkazy .....	53
Synchronní přenos .....	55
ISDN .....	55
Basic Rate .....	56
Základní pásmo a telefonní (hlasové) pásmo .....	57
ADSL .....	57
Přenosové rychlosti modemů.....	59
Strukturovaná kabeláž .....	60
Měděné rozvody .....	61
Optická vlákna.....	62
IEEE 802 .....	64
Ethernet, FastEthernet a Gigabitový Ethernet .....	65
Bezdrátové lokální sítě WLAN .....	69
Typické WLAN-konfigurace .....	71
Antény .....	71
Sledování WiFi .....	72

## Kapitola 4

### Linková vrstva ..... 75

SLIP .....	75
HDLC.....	76
Křídlová značka (Flag).....	77
Adresní pole.....	78
Datové pole a typ přenášeného protokolu .....	78
Řídicí pole.....	78
Kontrolní součet (Frame Check Sequence – FCS).....	81
Základní vlastnosti protokolu HDLC .....	81
Cisco HDLC (cHDLC).....	81
PPP.....	82
Vytáčení telefonní linky .....	84
Protokol LCP.....	85
Autentizace.....	90
Protokol řízení zpětného volání.....	95
Další protokoly .....	96
Protokol IPCP .....	101
Frame Relay .....	103

Rámec protokolu Frame relay.....	106
IP přes Frame Relay.....	109
LMI.....	110
Závěr k protokolu Frame Relay.....	111
Ethernet.....	111
Bezdrátové lokální síť WLAN – IEEE 802.11 .....	114
Obecný formát rámce .....	116
Ovládací rámce.....	117
Řídicí rámce .....	118
Datové rámce .....	122
WEP .....	123
IEEE 802.1X.....	124
IEEE 802.11i.....	124
LLC (IEEE 802.2).....	124
Domácí cvičení.....	126

## Kapitola 5

### **IPv4..... 127**

IP datagram .....	131
Protokol ICMP .....	135
Echo.....	137
Nedoručitelný IP datagram .....	138
Sniž rychlosť odesílaní .....	138
Změň směrování (Redirect) .....	138
Žádost o směrování.....	139
Čas vypršel (time exceeded) .....	139
Žádost o masku .....	141
Časová synchronizace .....	141
Fragmentace .....	142
Volitelné položky IP záhlaví .....	145
Záznamenávej směrovače .....	147
Záznamenávej čas.....	149
Explicitní směrování .....	150
Upozornění pro směrovač (IP Router Alert Option).....	152
Protokoly ARP a RARP .....	153
Filtrace ARP .....	156
Proxy ARP .....	156
RARP .....	157
IGMP .....	158
Oběžníky a linkový protokol .....	161
QoS .....	162
ECN .....	164
Domácí cvičení.....	165

**Kapitola 6****IPv4 – adresa ..... 167**

Síť – historická epocha I .....	168
Speciální IP adresy .....	169
Síťová maska .....	170
Síť – historická epocha II .....	171
Subsítě a supersítě .....	172
Supersítě a autonomní systémy .....	176
IP adresy v intranetu .....	181
Nečíslované sítě .....	181
Dynamicky přidělované adresy .....	182
Adresní plán .....	183
Více než 254 rozhraní na LAN .....	184
Domácí cvičení .....	184

**Kapitola 7****Směrování ..... 185**

Předávání (forwarding) a filtrace (filtering) .....	187
Směrování (routing) .....	187
Zpracování .....	189
Manipulace se směrovacími tabulkami .....	190
Výpis obsahu směrovací tabulky .....	190
Výpis obsahu směrovací tabulky v UNIXu/Linuxu .....	190
Výpis v operačních systémech Microsoft .....	191
Výpis obsahu směrovací tabulky na směrovačích CISCO .....	192
Naplnění směrovací tabulky a rušení položek .....	193
Směrovací protokoly .....	194
Princip Routing Vector Protocols (RVP) .....	195
RIP, RIP2 a RIPng .....	198
Princip Link State Protocols (LSP) .....	198
OSPF .....	203
Redistribuce .....	204
Domácí cvičení .....	205

**Kapitola 8****IPv6 ..... 207**

Další hlavičky v IP datagramu .....	210
Informace pro směrovače .....	210
Směrovací informace .....	212
Záhlaví fragmentu .....	213
Autentizační hlavička (protokol AH) .....	215
Bezpečnostní hlavička (protokol ESP) .....	215
ICMPv6 .....	216
Překlad IP adres na linkové adresy .....	218
Zjištění adresy směrovače na LAN .....	221

Změn směrování .....	223
IPv6 – adresa .....	224
Zápis adresy .....	225
Oběžníky (Multicasts) .....	226
Jednoznačné adresy .....	226
Identifikátor (index) síťového rozhraní .....	228
Domácí cvičení .....	228

## Kapitola 9

### **Protokol TCP (Transmission Control Protocol)..... 229**

TCP segment .....	231
Volitelné položky TCP záhlaví .....	234
Příklad výpisu TCP segmentu .....	☒
Navázání a ukončení spojení protokolem TCP .....	236
Navazování spojení .....	236
Ukončování spojení .....	240
Odmítnutí spojení .....	242
Zjištění stavu spojení .....	242
Technika zpozdění odpovědi .....	243
Technika okna .....	245
Zahlcení sítě .....	247
Pomalý start .....	247
Vyhýbání se zahlcení .....	248
Ztráta segmentu .....	248
Volba zvětšení okna .....	249
Domácí cvičení .....	250

## Kapitola 10

### **Protokol UDP (User Datagram Protocol) ..... 251**

Fragmentace .....	252
Příklad UDP datagramu .....	253
Oběžníky .....	253
Domácí cvičení .....	253

## Kapitola 11

### **DNS..... 255**

Domény a subdomény .....	256
Syntaxe jména .....	257
Reverzní domény .....	258
Doména 0.0.127.in-addr.arpa .....	259
Zóna .....	259
Speciální zóny .....	259
Rezervované domény a pseudodomény .....	260

---

Dotazy (překlady) .....	.260
Round Robin .....	.263
Konfigurace resolveru .....	.264
Konfigurace resolveru v Unixu .....	.264
Konfigurace resolveru ve Windows .....	.265
Jmenné servery .....	.267
Předávání dotazů DNS .....	.270
Věty RR .....	.271
Databáze DNS .....	.273
SOA .....	.274
Záznamy typu A .....	.276
CNAME .....	.276
HINFO a TXT .....	.277
NS .....	.277
MX .....	.278
PTR .....	.279
Věta typu SRV .....	.280
\$ORIGIN .....	.282
\$INCLUDE .....	.283
Rozšíření DNS pro IP verze 6 .....	.283
Záznam typu AAAA .....	.283
Záznam typu A6 .....	.283
Reverzní domény .....	.285
Záznam typu DNAME .....	.285
Nástroje pro sledování DNS .....	.286
Program nslookup .....	.286
Domácí cvičení: .....	.288
Ladicí režim .....	.289
Ladicí úroveň debug .....	.289
Ladicí úroveň d2 .....	.290
Změna implicitního jmenného serveru .....	.291
Dig .....	.291
Domácí cvičení .....	.292
<b>Kapitola 12</b>	
<b>Protokol DNS .....</b>	<b>293</b>
DNS QUERY .....	.293
Formát paketu DNS query .....	.294
Záhlaví paketu DNS query .....	.294
Sekce dotaz (Question section) .....	.296
Sekce odpověď, autoritativní servery a doplňující informace .....	.298
Komprese .....	.298
Domácí cvičení .....	.299
Inverzní dotaz .....	.299
DNS UPDATE .....	.299
Sekce záhlaví .....	.300
Sekce zóny .....	.301
Sekce předpokladů .....	.301
Sekce update .....	.302

Sekce doplňujících informací .....	303
Soubor žurnál .....	303
DNS Notify .....	303
Zpráva Notify .....	304
Inkrementální zone transfer .....	305
Formát dotazu .....	306
Formát odpovědi .....	306
Strategie čištění (purging) .....	306
Negativní caching (DNS NCACHE) .....	307
Jaké negativní odpovědi ukládat do paměti? .....	307
Jak dlouho udržovat negativní odpovědi v paměti? .....	308
Pole MINIMUM ve větě SOA .....	308
Pravidla ukládání negativních odpovědí .....	309
Domácí cvičení .....	309

## Kapitola 13

### Mezinárodní a národní organizace Internetu ..... **311**

ICANN .....	312
RIR .....	312
Národní organizace .....	313
O sdružení CZ.NIC .....	313
Protokol whois .....	314
TLD .....	315
Adresní prostor IPv4 .....	324
Adresní prostor IPv6 .....	328
Domácí cvičení .....	330

## Kapitola 14

### Telnet ..... **311**

Charakteristika protokolu .....	331
Bezpečnost .....	332
Protokol Virtuální terminál (NVT) .....	332
Příkazy protokolu Telnet .....	334
Příklad komunikace klienta z Windows .....	340
Příklad komunikace klienta z Unixu .....	342
Domácí cvičení .....	344

## Kapitola 15

### FTP ..... **345**

Charakteristika .....	345
Architektura .....	345
Aktivní režim komunikace protokolu FTP .....	348
Pasivní režim komunikace protokolu FTP .....	350

---

Příkazy FTP .....	.353
Proxy .....	.355
Návratové kódy .....	.357
Abnormální ukončení příkazu .....	.357
Anonymní FTP.....	.358
Domácí cvičení.....	.359

## Kapitola 16

### **HTTP..... 361**

Klient-server.....	.361
Domácí cvičení.....	.364
Proxy.....	.364
Brána.....	.367
Tunel.....	.368
Více mezilehlých uzlů.....	.369
URI .....	.370
Schéma http .....	.370
Schéma ftp.....	.371
Schéma mailto .....	.372
Schéma nntp: .....	.372
Schéma Telnet.....	.372
Schéma file.....	.372
Schéma pop.....	.372
Relativní URI.....	.372
HTTP dotaz.....	.373
Metoda GET.....	.374
Metoda POST.....	.377
Metoda HEAD .....	.378
Metoda TRACE.....	.378
Metoda OPTIONS.....	.379
HTTP odpověď .....	.380
Přehled výsledkových kódů.....	.381
Ostatní hlavičky .....	.381
Hlavíčky Accept .....	.381
Autorizace klienta .....	.382
Proxy autentizace.....	.383
Hlavíčky Content .....	.384
Přesměrování a dočasná nedostupnost .....	.384
Hlavíčka Upgrade.....	.385
Cache .....	.385
Informace o softwaru .....	.387
Cookie .....	.387
Hlavíčka Set-Cookie2 .....	.389
Hlavíčka Cookie .....	.389
Domácí cvičení.....	.389

**Kapitola 17****Elektronická pošta ..... 391**

Elektronická pošta a DNS .....	397
Format poštovní zprávy.....	397
Přehled základních hlaviček z RFC-822 .....	398
SMTP .....	400
ESMTP.....	403
VERB.....	403
BITMIME.....	404
SIZE .....	404
ETRN .....	405
Potvrzení o doručení zprávy .....	405
DSN (Delivery Status Notifications). ....	406
POP3 .....	408
IMAP4 .....	411
Neautentizovaný stav.....	413
Autentizovaný stav .....	414
Otevřená schránka.....	418
Domácí cvičení.....	423

**Kapitola 18****Filtrace, proxy a NAT ..... ☒ ..... 425**

Filtrace .....	425
Filtrace na úrovni protokolu IP .....	427
Filtrace na úrovni TCP.....	432
Reflexivní filtry.....	436
Filtrace protokolů UDP, ICMP a případně dalších protokolů.....	440
Zakázané adresy .....	440
Aplikační protokoly a filtrace.....	440
Závěr k filtraci.....	445
Proxy .....	445
Klasická proxy .....	447
Generická proxy .....	448
Transparentní proxy .....	449
Autentizace.....	451
SOCKS.....	451
Skryté sítě .....	453
NAT .....	455
Jednoduchý NAT .....	455
Rozšířený NAT .....	457
Dvojitý NAT .....	458
Rozložení výkonu.....	459
ALG .....	459
Domácí cvičení.....	460

**Kapitola 19****Firewall ..... 461**

Architektura firewallů .....	462
TIS Firewall Toolkit .....	463
SEAL .....	463
Jednopočítáčové firewalls s dvěma síťovými rozhraními .....	465
Firewalling .....	465
Personální firewall .....	466
Demilitarizované zóny (DMZ) .....	466
Firewall on Firewall .....	467
Extranet .....	467
Viruswall a antispamový filter .....	469
DNS .....	470
DNS v uzavřených podnikových sítích .....	470
DNS a firewall .....	472
Společné DNS pro Internet i intranet .....	473
Na firewallu je jen jmenný server pro Internet, a nikoliv pro intranet.....	476
Duální DNS .....	477
Ostatní aplikační protokoly a firewall .....	479
HTTP a FTP .....	479
SSL/TLS .....	479
Telnet či SSH .....	479
Elektronická pošta .....	479
NTP (Network Time Protocol) .....	480

**Rejstřík ..... 483**

# Úvod

Vážený čtenáři,

dostává se vám do rukou poslední, zcela přepracované vydání Velkého průvodce TCP/IP. Jedná se o originální českou příručku a učebnici TCP/IP, která nejenom vychází v 5. českém vydání, ale mezi-tím vyšla anglicky, rusky a druhý díl dokonce polsky. Díky aktualizacím v jednotlivých vydáních došlo k takovým změnám, že z prvního vydání toho zbylo opravdu málo (třeba příklad na str. 399).

## Velký průvodce TCP/IP má nyní 19 kapitol:

1. **Sítové protokoly** – v této kapitole jsem s gustom vypustil model ISO OSI a jeho sedm vrstev na důkaz definitivního vítězství rodiny TCP/IP nad protokoly ISO OSI. Vím, že spíše došlo ke zkřížení obou rodin protokolů, ale pro mne, který zažil v první polovině 90. let minule-ho století monopol sítí ISO OSI (X.25) stvrzený tehdy ještě československým státem, který zakazoval veřejné využívání TCP/IP, to navždy bude vítězství.
2. **Nástroje pro sledování a zkoumání sítě** – tato kapitola byla přepracována a nyní je posta-vena na bázi programu Wireshark a byla dopracována část zabývající se sledováním sítí programem nmap.
3. **Fyzická vrstva** – z nostalgických důvodů jsem ponechal sériová rozhraní, modemy, struk-turovanou kabeláž a dokonce i ISDN, protože mi stále připadá, že to patří k všeobecnému vzdělání. Dopracováno bylo ADSL a bezdrátové sítě.
4. **Linková vrstva** – byla vypuštěna obtížná část o van Jacobsonově kompresi IP a TCP záhlaví. Ponechány byly protokoly HDLC, PPP, Ethernet a dokonce i Frame Relay. Dopracovány byly části o WiFi a vrstvě LLC.
5. **IPv4** – všimněte si, že tato kapitola už má ve svém názvu v4. Protokol IPv4 je dnes sice na svém vrcholu, ale z vrcholu jdou cesty jen dolů.
6. **IPv4 adresa** – tato kapitola popisuje tvar IP adres. Kapitola zůstává bez zásadních změn.
7. **Směrování** – tato kapitola se asi nejvíce liší od předchozích vydání. Popisuje detailně neje-nom Bellman-Fordův algoritmus, ale i Dijkstrův algoritmus.
8. **IPv6** – kapitola o IP protokolu verze 6 již uvádí příklady z Microsoft Windows Vista, protože od verze Vista je IPv6 standardní součástí Windows.
9. **Protokol TCP** – zde byly zásadně přepracovány obrázky, které byly údajně málo demonstra-tivní. Doufám, že tentokrát bude laskavý čtenář spokojen.
10. **Protokol UDP** – tento protokol je tak jednoduchý, že toho k němu už mnoho nelze dodat.
11. **DNS** – tato kapitola objasňuje principy a základní termíny DNS.
12. **Protokol DNS** – obsahuje vlastní popis protokolu DNS.
13. **Mezinárodní a národní organizace internetu** – zde čtenář najde informace, jak jsou celo-světově rozděleny internetové IP adresy a domény a kdo je přidělováním těchto světových zdrojů pověřen.
14. **Telnet** – popisuje stejnojmenný aplikační protokol, který je jedním z nejstarších protokolů Internetu.

15. **FTP** – popisuje protokol FTP.
16. **HTTP** – popisuje protokol HTTP.
17. **Elektronická pošta** – tato kapitola objasňuje principy elektronické pošty a následně popisuje protokoly POP3, IMAP4 a SMTP.
18. **Filtrace, proxy a NAT** – popisuje některé bezpečnostní mechanismy TCP/IP.
19. **Firewall** – popisuje využití uvedených bezpečnostních mechanismů TCP/IP pro ochranu vnitřních sítí.

Jak jste si asi všimli, byla přepracována i skladba kapitol. Přibyly kapitoly o filtraci, proxy a firewallu, které původně byly součástí až 2. dílu.

Druhý díl se tak zabývá výhradně PKI a tak mohl vyjít pod názvem „Velký průvodce PKI a technologií elektronického podpisu“.

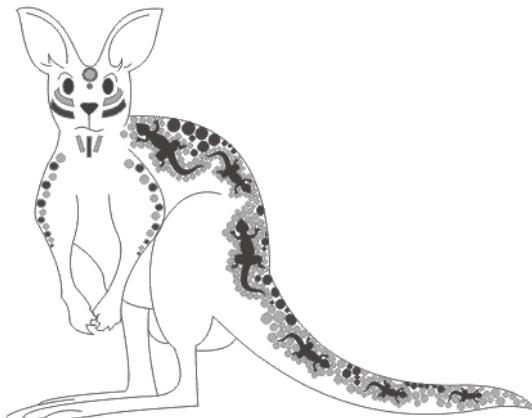
Zatímco první vydání sloužila zejména jako příručky TCP/IP, tak v poslední době se Velký průvodce TCP/IP hojně využívá jako učebnice na vysokých, a dokonce i na středních školách. Proto téměř za každou kapitolou přibily náměty na domácí cvičení.

Českým čtenářům bych chtěl poděkovat za vesměs kladné ohlasy na tuto publikaci. Nemohu ale nevzpomenout na extrémně negativní reakci jednoho australského čtenáře. Velice jej totiž rozlítalo, že to je už myslím 12. učebnice TCP/IP, kterou si koupil, a zase to TCP/IP nepochopil. Pokaždé, když si na tuto zlostnou reakci vzpomenu, tak mi to zlepší náladu. Dokonce mě to povzbudilo, abych připravil toto 5. české vydání. A aby se ten Australan už tolik nezlobil, tak mu posílám klokana – třeba aspoň pozná, co to je za zvíře, i když neumí česky.

Závěrem bych chtěl poděkovat Haně Strakové a Janu Říhovi za velice zodpovědnou revizi a podnětné připomínky. Dále děkuji Vilému Sládkovi, který doplnil informace o sdružení CZ.NIC.

From: Neposílejte mi spamy <[libor.dostalek@siemens.com](mailto:libor.dostalek@siemens.com)>

Date: Mon, 11 Aug 2008 17:17:48 +0200



## Kapitola 1

# Síťové protokoly

Podobně jako diplomati při svých jednáních používají diplomatický protokol, tak počítače v počítačových sítích používají pro vzájemnou komunikaci síťové protokoly. Síťových protokolů existuje celá řada. V Internetu se používají síťové protokoly TCP/IP.

Síťový protokol je norma napsaná na papíře (dnes spíše pořízená textovým editorem na počítači). V Internetu se používají normy nazývané **Request For Comments** (zkratka RFC), které se číslují průběžně od jedničky. V současné době je jich již více a pět tisíc. Mnohé však postupem času zastaraly, takže z první tisícovky jich je aktuálních už jen několik. Důležité je, že tyto normy jsou volně ke stažení na Internetu (viz [www.rfc-editor.org](http://www.rfc-editor.org)).



**Poznámka:** V první tisícovce RFC jsou i superstar roku 1981: RFC-791 a RFC-793, které specifikují protokoly IP a TCP.

Vedle RFC se budeme ještě setkávat zejména s normami vydanými organizacemi:

- ◆ ISO (*International Organization for Standardization*). Tyto normy lze získat např. v informačním středisku Českého normalizačního institutu (Praha 1, Biskupský dvůr 5).
- ◆ ITU (*International Telecommunication Union*) se sídlem v Ženevě, která je jednou z nejstarších celosvětových organizací – byla založena roku 1865 (jen Červený kříž byl založen dříve – roku 1863). Tyto normy lze studovat v knihovně ústavu TESTCOM (Technický a zkušební ústav telekommunikací a pošt, Hvoždanská ulice, Praha 4).

Na fyzické a linkové vrstvě se pak dále setkáme s americkými normami:

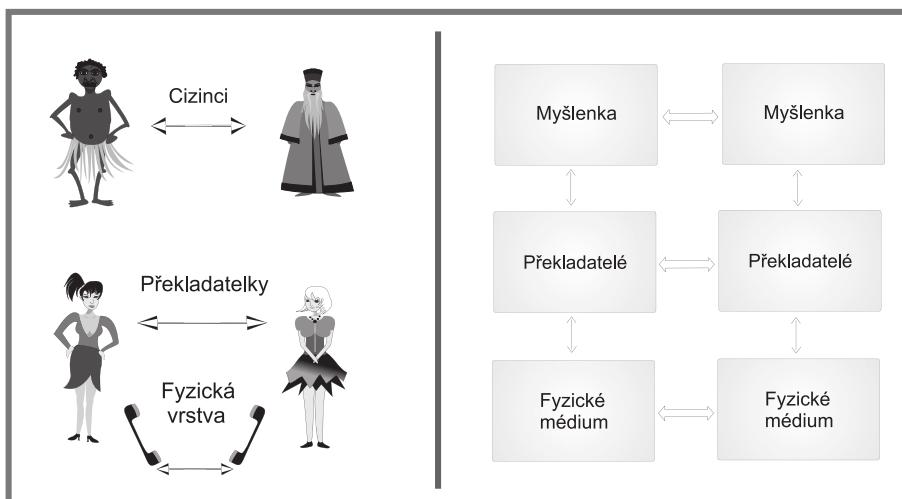
- ◆ IEEE – profesní organizace zabývající se pokročilými technologiemi. Setkáme se s jejími standardy řady IEEE 802. Tyto normy jsou tč. dostupné i na Internetu. V názvech norem řady IEEE 802 následuje za řetězcem „IEEE 802“ tečka a číslo vlastní normy (např. IEEE 802.11 standard pro WiFi). Důležité je, že mnohé tyto normy mají i řadu dodatků, které se označují zpravidla písmenem na konci (např. IEEE 802.11g – dodatek standardu IEEE 802.11 pro rychlosť do 54 Mb/s).
- ◆ TIA (*Telecommunications Industry Association*) – asociace amerických společností zabývajících se telekomunikacemi. Setkáme se s její normou TIA/EIA-568-B.
- ◆ EIA (*Electronic Industries Aliance*) – asociace amerických výrobců elektroniky. Setkáme se s jejími normami RS-232 a TIA/EIA-568-B.

Celá problematika síťové komunikace je poměrně komplikovaná, a tak je potřeba ji rozdělit na různé vrstvy. Většina publikací o síťových protokolech uvádí přirovnání ke komunikaci dvou cizinců (či filozofů, šamanů apod.). Každý sedí ve své vesnici a umí jen svůj jazyk. Aby si vyměnili své myšlenky, musí si každý obstarat překladatelku do společného jazyka – např. do češtiny. Viz obr. 1.1.

Vzájemně si oba cizinci předávají své myšlenky, tj. komunikují mezi sebou. Jenže mezi sebou komunikují jen pomyslně (virtuálně). Ve skutečnosti oba předávají své informace překladatelkám. Avšak překladatelkám se nepodařilo přemluvit cizince, aby se sešli na jednom místě, a tak překladatelky musí vzájemně komunikovat pomocí telefonu – jejich hlas musí být tedy přenášen po telefonních linkách.

Rozeznáváme virtuální komunikaci ve vodorovném směru (filozofickou, společným jazykem a mezi překladatelkami) a skutečnou komunikaci ve svislém směru, tj. komunikaci cizinec – překladatelka a komunikaci překladatelka – telefon. V našem příkladě tedy rozlišujeme celkem tři vrstvy komunikace:

- ◆ Komunikace mezi cizinci.
- ◆ Komunikace mezi překladatelkami.
- ◆ Fyzický přenos informací po médiu (např. telefonní vedení, zvukové vlny atp.).



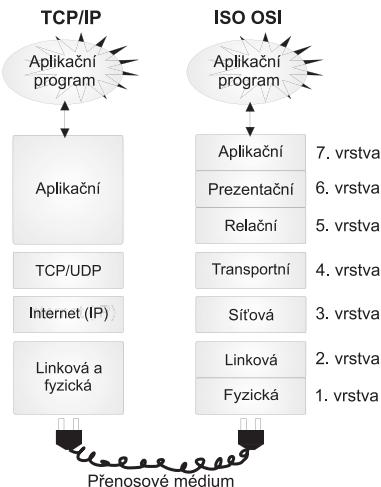
**Obrázek 1.1:** Třívrstvá komunikační architektura

Komunikace cizinec – cizinec a komunikace překladatelka – překladatelka je pouze pomyslná (virtuální). Ve skutečnosti (reálně) komunikuje cizinec vždy jen s překladatelkou.

V počítačových sítích používáme vrstev více. Počet vrstev závisí na tom, jakou soustavu síťových protokolů použijeme. Nejčastěji se budeme setkávat se soustavou protokolů, která vznikla pro přenos dat na Internetu, též nazývanou **rodinou protokolů TCP/IP**, která je čtyřvrstvá.

Kromě protokolů TCP/IP se ještě občas potkáme se sedmivrstvou soustavou **ISO OSI**, kterou standardizovalo ISO. Soustavu ISO OSI její tvůrci dokonale propracovali a jeden čas se dokonce zdálo, že odsune do pozadí i rodinu TCP/IP.

Soustavy síťových protokolů TCP/IP a ISO OSI se od sebe liší – jsou vzájemně neporovnatelné. Z obrázku 1.2 je však patrné, že na síťové a linkové vrstvě jsou si velice blízké. Rodina síťových protokolů TCP/IP stejně nerěší (až na výjimky, jako jsou např. protokoly SLIP a PPP) linkovou a fyzickou vrstvu, proto se i na Internetu setkáváme s linkovými a fyzickými protokoly z modelu ISO OSI.

**Obrázek 1.2:** Porovnání rodiny protokolů TCP/IP a ISO OSI

Jinou skupinou protokolů jsou standardy pro telekomunikační přenosy vydané ITU. Ty se tvoří už přes 100 let. Před 100 lety se jistě nikomu ani nesnilo o tom, že by se skrze **telefoni okruhy** mělo přenášet něco jiného než hlas. Cílem telekomunikací bylo sestavovat telefonní (hlasové) okruhy mezi koncovými účastníky. Vývoj ale šel nezadržitelně dál. Nejprve se původní telefonní okruhy začaly využívat pro přenos dat. Avšak dnes se již naopak vytváří datové okruhy, které, pokud mají sloužit pro přenos hlasu, vyžadují, aby se hlas nejprve digitalizoval a pak přenášel jako data. To je případ i dnešních mobilních telefonů standardu GSM.



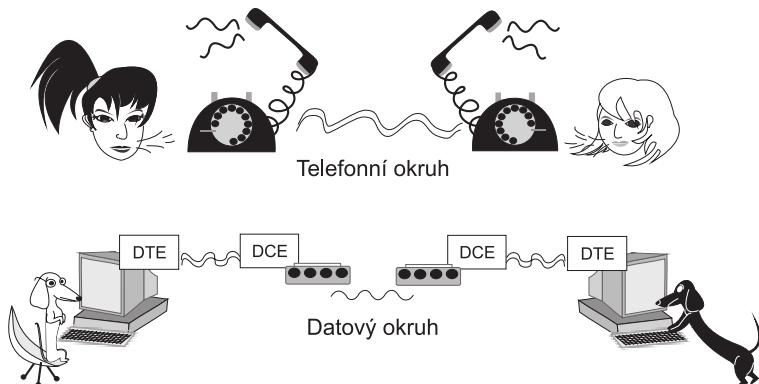
**Poznámka:** A jestlipak víte, že standard GSM je standardem naším – evropským? Konkrétně standardem ETSI - European Telecommunications Standards Institute.

Na hlasových okruzích nemáme v běžném hovoru s terminologií žádné potíže. Vždyť přece „hlasový okruh končí v telefonním aparátu“ a koncový účastník pak hovoří/naslouchá do telefonního aparátu (obr. 1.3).

Na datových okruzích (obr. 1.3) si ale budeme muset zvyknout na odbornou terminologii, která je trochu zvláštní. V případě datového okruhu si místo telefonního aparátu představíme tzv. DCE (*Data Circuit Equipment*). Takovým zařízením je např. modem. Jenže to nestačí, data přece potřebujeme dovést z DCE až do počítače – přesněji do DTE (*Data Terminal Equipment*).

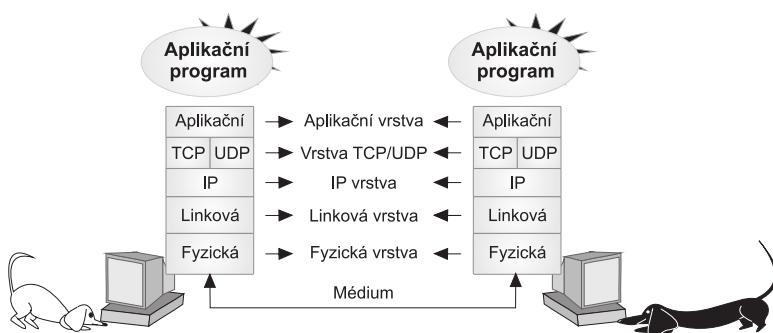
## Jaké protokoly se používají na Internetu?

Na Internetu se používá mix rodiny protokolů TCP/IP, protokolů ITU i ISO. Přitom protokoly ITU a ISO se vyskytují zejména na spodních dvou vrstvách, fyzické a linkové. Avšak protokoly ITU nám už vystrkují růžky i na aplikační vrstvě (např. protokol H.323 pro audiovizuální komunikaci přes Internet).



**Obrázek 1.3:** Telefonní a datový okruh

Dnes už ani příliš nefilozofujeme, který protokol má původ ve standardech ITU, který v ISO a který v rodině TCP/IP. Komunikaci mezi dvěma počítači prostě schematicky znázorňujeme tak, jak je tomu na obr. 1.4.



**Obrázek 1.4:** Komunikace mezi dvěma počítači

## Fyzická vrstva

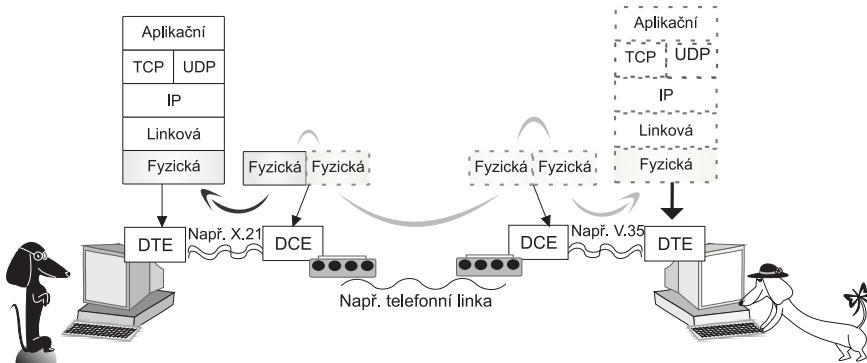
Fyzická vrstva se zabývá elektrickými, elektromagnetickými či optickými signály používanými při komunikaci mezi bezprostředními komunikačními sousedy. Pro nás, koncové uživatele, je ale asi nedůležitější, že fyzická vrstva specifikuje i tvary konektorů a vůbec nejrůznějších propojovacích kabelů.

Protokoly fyzické vrstvy specifikují:

- ◆ Elektrické signály (např. +1V).
- ◆ Tvary konektorů (např. V.35).
- ◆ Typ média: kroucená dvojlinka, koaxiální kabel, optické vlákno atd.
- ◆ Přenosovou rychlosť (např. 128kB/s).
- ◆ Modulaci (např. FM, PM apod.).

- ◆ Kódování (např. RZ, NRZ apod.).
- ◆ Synchronizaci: synchronní či asynchronní komunikace, zdroj hodin atd.

Pokud použijeme terminologii ITU, pak fyzická vrstva je zodpovědná za aktivaci, komunikaci a deaktivaci fyzického okruhu mezi DTE a DCE. Komunikace mezi DCE a DCE už pak zajišťují jiné protokoly fyzické vrstvy. Pod DTE si opět představíme počítač. Pod DCE si představíme zase modem.

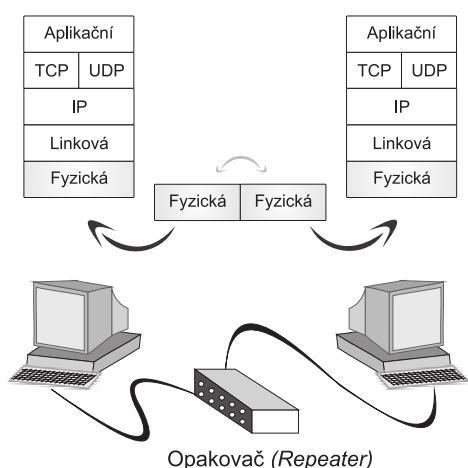


**Obrázek 1.5:** Komunikace mezi DTE a DCE

Všimněte si, že na obrázku 1.5 je jiný protokol fyzické vrstvy mezi DTE a DCE na levém počítači a jiný mezi DTE a DCE na pravém počítači.

### Lokální síť (LAN)

Lokální síť (LAN) jsou počítačové sítě, kde spolu komunikuje několik stanic zpravidla na sdíleném médiu. V rámci jedné LAN se používá stejný linkový protokol (např. Ethernet). V případě lokálních sítí o DTE a DCE nehovoříme. Tam fyzická vrstva bývá realizována společně s částí linkové vrstvy



**Obrázek 1.6:** Opakovač (někdy nazývaný též rozbočovač)

přímo v hardwaru počítače nebo jiného zařízení. Tento hardware pak nazýváme **síťovým rozhraním (Network Interface)**. Opět i zde nás na fyzické vrstvě zajímají zejména konektory a propojovací kabely – ostatní ponecháme technikům, jejichž profesí je právě fyzická vrstva.

Geografický rozsah lokální sítě je omezen. Pro zvětšení tohoto rozsahu lze na fyzické vrstvě využívat tzv. **opakovač (repeater)**, označovaný též jako **rozbočovač**. Opakovače (obrázek 1.6) naslouchají na svých síťových rozhraních a automaticky veškerou komunikaci opakují na ostatní svá síťová rozhraní.

## Linková vrstva

Linková vrstva (též nazývaná spojová vrstva) zajišťuje v případě sériových linek výměnu dat mezi sousedními počítači a v případě lokálních sítí výměnu dat v rámci lokální sítě.

Od linkové vrstvy výše se data přenáší v blocích, které obecně nazýváme **datovými pakety**. Datovému paketu na linkové vrstvě se říká **linkový rámec**, na IP vrstvě **IP datagram** a na TCP/UDP, vrstvě zase máme **TCP segmenty** a **UDP datagramy**.

Vraťme se ale zpět na linkovou vrstvu, kde základní přenosovou jednotkou je linkový rámec (viz obr. 1.7). Linkový rámec se skládá ze **záhlaví (Header)**, přenášených dat (**Payload**) a často i **zápatí (Trailer)**.

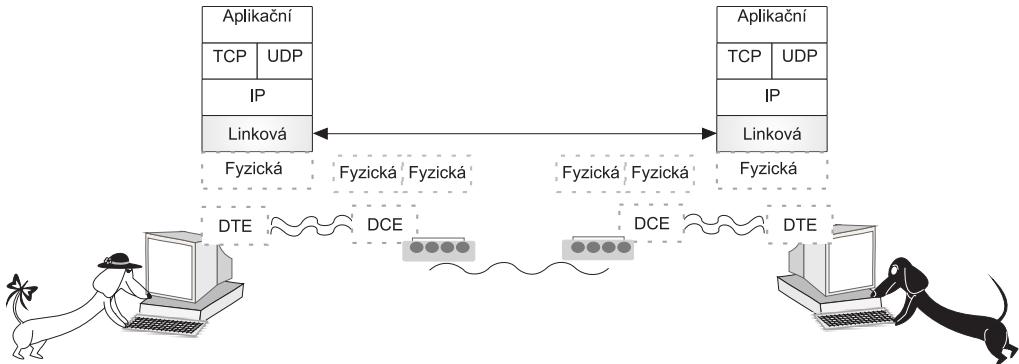


**Obrázek 1.7:** Linkový (spojový) rámec

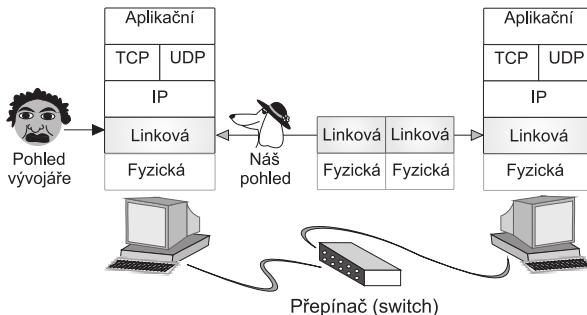
Linkový rámec zpravidla nese v záhlaví linkovou adresu příjemce, linkovou adresu odesilatele a další řídicí informace. V zápatí pak obvykle nese kontrolní součet (CRC) z přenášených dat. Pomocí kontrolního součtu lze zjistit, zdali nedošlo při přenosu k porušení dat. V přenášených datech je pak zpravidla nesen IP datagram nebo ARP-paket.

Z obr. 1.5 je vidět, že na fyzické vrstvě mohou být pro každý konec datového okruhu použity jiné protokoly. V našem případě jeden konec používá protokol X.21 a druhý konec používá protokol V.35. Tento fakt neplatí jen pro sériové linky, ale i pro lokální sítě. U lokálních sítí se ale spíše setkáváme s komplikovanějším případem, kdy mezi oba konce spojení je vložen **přepínač (Switch)**, který umí i konvertovat linkové rámce jednoho linkového protokolu na rámce jiného linkového protokolu (např. Fast Ethernet na FDDI), což má pochopitelně za následek i použití různých protokolů na fyzické vrstvě. V poslední době přepínače spíše přepínají mezi Ethernetem, Fast Ethernetem a Gigabitovým Ethernetem, které mají z našeho pohledu stejný tvar linkového rámce. Avšak z pohledu fyzické vrstvy jsou mezi nimi zásadní rozdíly (viz kap. 3).

Na obr. 1.8 je znázorněno, že linkové rozhraní se tváří, že nevidí fyzickou vrstvu, ale spolehá na ni, že funguje. To platí zejména pro specifikaci linkových protokolů. Avšak na konkrétním počítači linková vrstva data předává fyzické vrstvě (vertikální směr přenosu). Jestliže fyzická vrstva z nějakého důvodu není připravena data převzít/předat, tak to pochopitelně linkové vrstvě signalizuje. Ale tato signalizace je věcí konkrétní implementace, nikoliv linkového protokolu jako takového. Linkový protokol ji neřeší – ten specifikuje komunikaci jakoby uprostřed sítě a je povznesen nad nějaké problémy fyzické vrstvy. Implementátor, tj. vývojář, který programuje knihovny realizující linkovou vrstvu, se zase musí zabývat i ošetřením všech stavů při předávání/přebírání dat ve vertikálním směru.



**Obrázek 1.8:** Komunikace na linkové vrstvě „nevidí“ fyzickou vrstvu



**Obrázek 1.9:** Linkový protokol (jako všechny ostatní síťové protokoly) specifikuje síťovou komunikaci jakoby uprostřed sítě

Linkovým rozhraním pro počítač může být např. síťová karta pro Ethernet či sériový port. Zejména v lokálních sítích má každé linkové rozhraní linkovou (hardwareovou) adresu. Ta je jednoznačná v rámci lokální sítě.

## IP vrstva

IP vrstva zabezpečuje přenos dat mezi vzdálenými počítači v Internetu (obecně v rozsáhlé síti – WAN). Základní jednotkou přenosu je **IP datagram**, který se balí (zapouzdřuje) do linkového rámce linkové vrstvy. IP datagram se skládá z IP záhlaví a pole Data (viz obr. 1.10).

Z obr. 1.10 je patrné, že IP záhlaví společně s daty IP datagramu tvoří datové pole (*Payload*) linkového rámce.

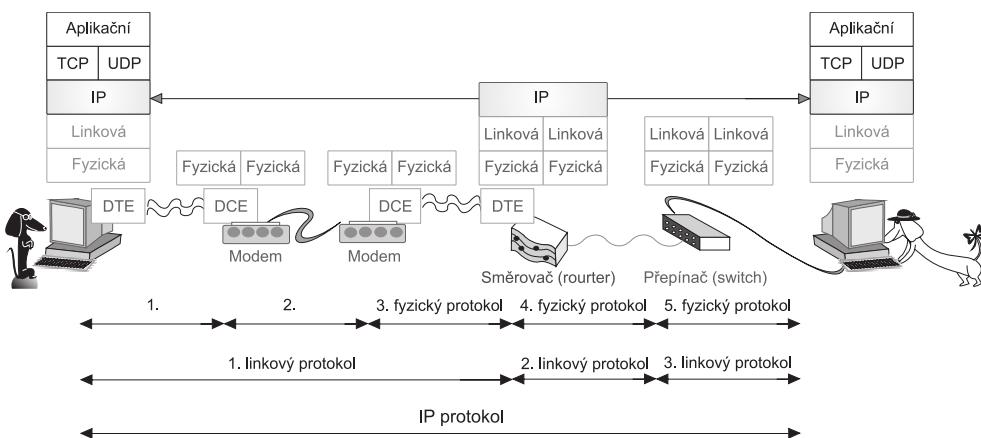
V Internetu mezi počítači může ležet jeden nebo více **směrovačů (routers)**. Z praktického pohledu říkáme, že komunikuje:

- ◆ V rámci LAN (lokálně), neleží-li mezi zdrojovým a cílovým počítačem žádný směrovač.
- ◆ V rámci WAN, jestliže komunikace je zprostředkována skrze alespoň jeden směrovač.



**Obrázek 1.10:** IP datagram se i se svým záhlavím vkládá (zapouzdruje) do linkového rámce

Mezi sousedními směrovači je na vrstvě IP vždy přímé spojení. Směrovač vybalí síťový paket z datového rámce (jednoho linkového protokolu) a před odesláním do jiné linky jej opět zabalí do jiného datového rámce (obecně jiného linkového protokolu). IP vrstva nevidí zařízení pracující na fyzické a linkové vrstvě (modemy, opakovače/rozbočovače, přepínače apod.).



**Obrázek 1.11:** Komunikace na IP vrstvě

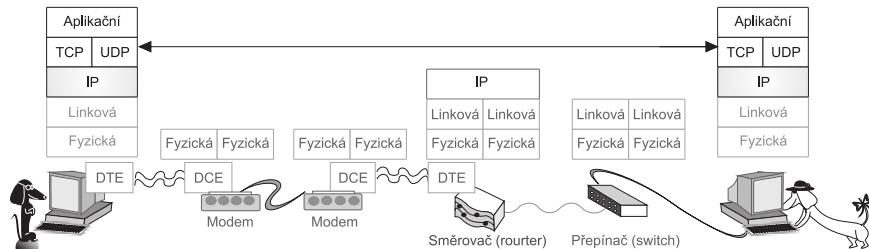
IP vrstvu nezajímá, jaké jednotlivé protokoly linkové (a natož fyzické) vrstvy byly na cestě mezi oběma konci spojení.

Síťovým rozhraním může být, opět jako na linkové vrstvě, např. karta pro Ethernet či sériový port. Síťové rozhraní má na vrstvě IP jednoznačnou IP adresu v rámci Internetu/intranetu (tj. v rámci rozsáhlé sítě – WAN). Každý IP datagram přitom ve svém záhlaví nese IP adresu příjemce, což je úplná směrovací informace pro dopravu IP datagramu až k adresátovi. Takže síť může přenášet každý IP datagram samostatně. IP datagramy tak mohou k adresátovi dorazit v jiném pořadí, než byly odesány.

## Vrstva TCP/UDP

IP vrstva zabezpečí spojení mezi vzdálenými počítači, takže vrstvě TCP/UDP se jeví jakoby žádné modemy, opakovače/rozbočovače či směrovače na cestě nebyly. Tato vrstva žádná taková zařízení síťové infrastruktury nevidí. Vrstva TCP/UDP se zcela spoléhá na služby nižších vrstev. Je jí zcela jedno, jestli komunikující počítače stojí vedle sebe nebo každý na jiném kontinentu.

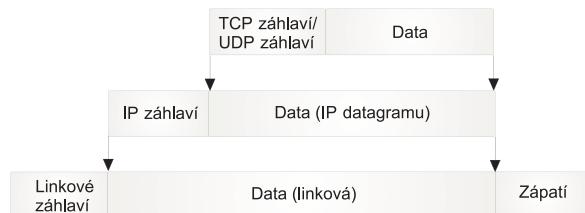
Vrstva TCP/UDP předpokládá, že spojení mezi počítači je zajištěno, proto se bez zbytečných starostí může věnovat předávání dat mezi aplikacemi na vzdálených počítačích.



**Obrázek 1.12:** Komunikace na vrstvě TCP/UDP nevidí síťovou infrastrukturu

I mezi dvěma týmiž počítači může komunikovat několik aplikací současně. Např. jedna pro přenos elektronické pošty a druhá pro přenos webových stránek. Z hlediska IP vrstvy je přitom protější počítač adresován zpravidla stejnou IP adresou. Jak se má tedy protější rozhodnout, které aplikaci má která data předávat? A to je právě základním úkolem vrstvy TCP/UDP. Pro adresaci aplikací zavádí tato vrstva tzv. porty. Konkrétní datový tok na sousední počítač je pak určen nejenom IP adresou, ale i číslem portu (a navíc ještě tím, jedná-li se o protokol TCP nebo UDP).

Základní přenosovou jednotkou na této vrstvě je buď **TCP segment** (v případě protokolu TCP), nebo **UDP datagram** (v případě protokolu UDP). Jak TCP segment, tak i UDP datagram se zapouzdřují do IP datagramu – viz obr. 1.13.



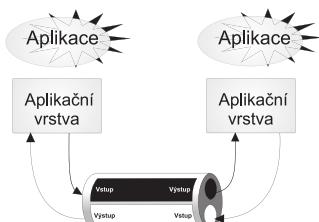
**Obrázek 1.13:** Zapouzdřování TCP/UDP do IP datagramu

K dispozici na této vrstvě máme dva protokoly:

- ◆ Protokol TCP, který je spojovanou službou, tj. příjemce potvrzuje přijímaná data. V případě ztráty dat (ztráty TCP segmentu nebo jeho části) si příjemce vyžádá zopakování přenosu.
- ◆ Protokol UDP přenáší data pomocí datagramů (obdoba telegramu), tj. odesilatel odešle datagram a už ho nezajímá, zdali byl doručen.

## Aplikační vrstva

Aplikační protokoly se spolehají na protokoly TCP nebo UDP, které vytvářejí datové toky mezi počítači. Aplikační vrstva pouze požádá nižší vrstvu (tj. protokol TCP nebo protokol UDP) o vytvoření datového toku, do kterého pak vkládá aplikační data. Datové toky jsou obecně oboustranné (obr. 1.4).



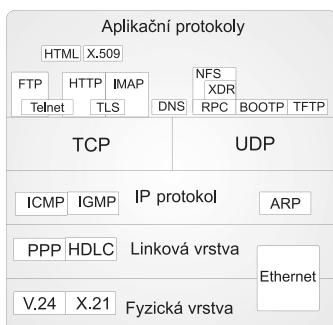
**Obrázek 1.14:** Applikační vrstva využívá služeb protokolu TCP

Applikační vrstva mj.:

- ◆ Předepisuje dialog komunikace mezi aplikacemi. V tomto dialogu je např. jedna strana klientem a druhá serverem. Např. protokol HTTP popisuje, jaký má být dialog mezi webovým klientem a webovým serverem.
- ◆ Předepisuje formát dat předávaných mezi aplikacemi. Např. jazyk HTML specifikuje, jak mají být formátována webová data či kódování BER specifikuje formát dat pro bezpečnou elektronickou poštu, adresářové služby atd.
- ◆ Protokol TLS je zase příkladem protokolu, který zabezpečuje spojení mezi aplikacemi proti útočníkům.

Applikačních protokolů je velké množství. Z praktického hlediska je lze rozdělit na:

- ◆ Uživatelské protokoly, které využívají koncov uživatelské (např. pro vyhledávání informací v Internetu). Příkladem takových protokolů jsou protokoly: HTTP, SMTP, Telnet, FTP, IMAP, POP3 atd.
- ◆ Služební protokoly, tj. protokoly, se kterými se běžní uživatelé Internetu nesetkají. Tyto protokoly slouží pro správnou funkci Internetu. Jedná se např. o směrovací protokoly, které používají směrovače mezi sebou, aby si správně nastavily směrovací tabulky. Dalšími příklady jsou protokoly SNMP a RADIUS, který slouží k managementu sítí.



**Obrázek 1.15:** Některé protokoly používané na Internetu

## Způsoby přenosu informací

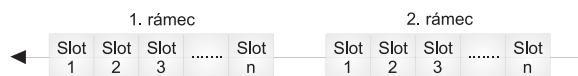
Síťových protokolů je velké množství, dokonce na jedné vrstvě máme často k dispozici několik protokolů. Zejména u protokolů nižších vrstev rozlišujeme, jaký typ přenosu protokol zabezpečuje, zda zabezpečuje službu spojovanou nebo nespojovanou, jestli protokol používá virtuální okruhy atd.

Rozeznáváme přenos synchronní, paketový a asynchronní.

### Synchronní přenos

Synchronní přenos je vyžadován např. pro zvuk a video, tj. v případě, kdy je třeba po dobu přenosu zajistit požadovanou stejnoměrnou šíři pásma. Stane-li se, že odesílatel nevyužije zajištěné pásmo, pak zůstává nevyužito.

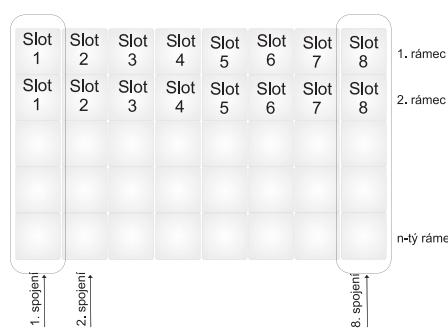
Synchronní přenos používá rámce konstantní délky, které jsou přenášeny sítí konstantní rychlostí.



**Obrázek 1.16:** Rozdelení rámců na sloty u synchronního přenosu

Garance šíře přenosového pásmo se u synchronního přenosu zajistí rozdelením přenášených rámců na sloty. Pro dané spojení se pak v každém přenášeném rámci vyhradí jeden či více slotů, viz obr. 1.16. Představme-li si, že v každém rámci je např. slot číslo 1 vyhrazen pro naše spojení, pak, jelikož rámcem stejnoměrně plynou sítí za sebou, má naše aplikace garantovanu šíři pásma, která je dána tím, kolik slotů číslo jedna přenese síť za sekundu.

Podstatu pochopíme, když si několik rámců nakreslíme pod sebe do tzv. **superrámců**, viz obr. 1.17. Sloty pod sebou patří těmuž spojení.



**Obrázek 1.17:** Superrámc

Se synchronním přenosem se setkáváme např. u připojení podnikové telefonní ústředny do telefonní sítě. Ta bývá připojena např. linkou E1, která obsahuje 32 slotů, každý o šířce pásmá 64 kb/s. Slot lze využít pro telefonní hovor. Současně je tak teoreticky garantováno 32 hovorů (některé sloty se však používají jako služební).

Internet nepoužívá synchronní přenos, tj. obecně negarantuje šíři přenášeného pásmo. Kvalitní přenos zvuku či videa se v Internetu zpravidla dociluje předimenzováním přenosových linek. V poslední době vzrůstají požadavky na přenos zvuku či videa i přes Internet, proto se stále častěji setkáváme se systémy, které zajišťují šíři pásmo i v Internetu – tzv. **Quality of Services - QoS (QoS)**. Aby však bylo dosaženo kýženého efektu, musejí tyto služby podporovat veškerá zařízení od odesilatele k příjemci. Takže se dnes spíše setkáváme jen s oblastmi Internetu, které garantují šíři přenosového pásmo, např. v rámci konkrétního poskytovatele Internetu.

## Paketový přenos

Paketový přenos používaný v Internetu je výhodný zejména pro přenos dat. Pakety nesou data obecně různé délky.



**Obrázek 1.18:** Paketový přenos dat

Paket nese data vždy jedné aplikace (jednoho spojení). Jelikož jsou pakety různé délky, nelze garantovat šíři pásmo. Výhodou je efektivní využití pásmo, protože v případě, že aplikace nepotřebuje přenášet data, mohou pásmo využít jiné aplikace.

## Asynchronní přenos

Asynchronní přenos kombinuje paketový přenos se synchronním přenosem.



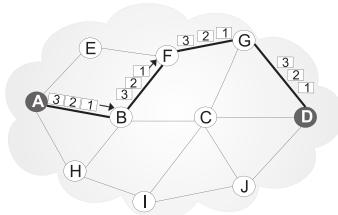
**Obrázek 1.19:** Asynchronní přenos dat

Podobně jako u paketového přenosu jsou u asynchronního přenosu data přenášena v paketech, které jsou však malé a stejně velké; nazývají se **buňky**. Obdobně jako u paketového přenosu se v jedné buňce přenáší data jedné aplikace (jednoho spojení). Avšak buňky mají stejnou délku, takže garantuji-li se, že každá x-tá buňka bude k dispozici konkrétní aplikaci (konkrétnímu spojení), pak se tím garantuje i šířka pásmo. Navíc, pokud aplikace buňku neodešle – nevadí, může být odeslána buňka jiné aplikace. Asynchronní přenos používá protokol ATM, jehož sláva pohasíná.

## Virtuální okruh

Některé síťové protokoly vytváří v síti **virtuální okruh** (*Virtual Circuit*), který se vytváří vždy před zahájením komunikace. Klasickým virtuálním okruhem je telefonní okruh. Před telefonováním přece vždy nejprve vytočíte číslo, což způsobí vytvoření telefonního okruhu.

Virtuální okruh je vedený sítí a všechny pakety spojení pak prochází tímto okruhem. V případě, že se okruh někde přeruší, přeruší se spojení a poté je třeba vytvořit nový okruh, aby se opět mohla přenášet data.

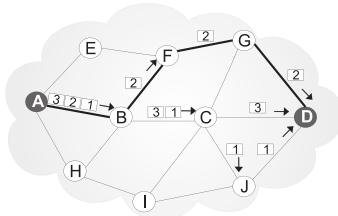
**Obrázek 1.20:** Virtuální okruh

Na obr. 1.20 je vytvořen virtuální okruh mezi uzly A a D přes uzly B, F a G. Všechny pakety musí procházet tímto okruhem.

Na virtuálním okruhu je možné buď přenášet datagramy, kdy okruh negarantuje doručení datagramu příjemci (tj. v případě zahlcení sítě může datagram i zahodit); takovýmto protokolem je např. protokol Frame Relay. Nebo naopak virtuální okruh může navázat spojení a doručení dat garantovat, tj. přenášená data čísluje a příjemce potvrzuje příjem dat. Pokud by se nějaká data ztratila, je dozádáno jejich opakování. Tento mechanismus používal např. protokol X.25.

Výhodou virtuálního okruhu je, že je nejprve sestaven (pomocí signalizace) a teprve do sestaveného okruhu se vkládají data. Každý paket obecně nemusí ve svém záhlaví nést globálně jednoznačnou adresu příjemce, ale pouze identifikaci okruhu.

V Internetu se na úrovni vrstvy IP mechanismus virtuálních okruhů nepoužívá, protože zničení uzlu ve virtuálním okruhu znamená přerušení spojení, což nevyhovovalo tvůrcům rodiny protokolů TCP/IP, která byla prvotně vytvořena pro Ministerstvo obrany USA. Každý IP datagram nese IP -adresu příjemce (tj. úplnou směrovací informaci), a je proto dopravován samostatně. Zničení uzlu sítě může zničit pouze IP datagram právě procházející zničeným uzlem v okamžiku zničení uzlu. Další IP datagramy jsou směrovány přes jiné uzly.

**Obrázek 1.21:** IP protokol nepoužívá virtuální okruhy

Z obr. 1.21 je vidět, že IP datagramy 1, 2 a 3 odešly z uzlu A společně na uzel B, ale z tohoto uzlu jsou již datagramy 1 a 3 směrovány jinou cestou než datagram 2. Do cílového uzlu D pak každý z našich datagramů dorazí jinou cestou. Obecně IP datagramy mohou dorazit i v jiném pořadí, než byly odesány, tj. např. v pořadí

2, 1 a 3.

Nad nespojovaným IP protokolem se v Internetu používá protokol TCP (protokol vyšší vrstvy), který spojení naváže a jenž garantuje doručování dat. Pokud byla data ztracena kvůli zničení některého

uzlu sítě a v síti existuje ještě jiná cesta, pak opakování dat již automaticky proběhne po této záložní cestě. Pokud bychom použili telefonní terminologii, pak protokol TCP vytváří virtuální okruh. Naopak protokol UDP nikoliv.



**Poznámka:** Teď už je jasné, proč jsou dva samostatné protokoly IP a TCP.

## Pevné a komutované virtuální okruhy

Virtuální okruhy rozeznáváme:

- ◆ Pevné (*Permanent Virtual Circuit – PVC*), tj. virtuální okruhy pevně sestavené administrátorem sítě. PVC používá např. protokol Frame Relay.
- ◆ Komutované (*Switched Virtual Circuit – SVC*), tj. virtuální okruhy dynamicky vznikající podle okamžité potřeby. SVC se vytváří pomocí tzv. signalačních protokolů, což jsou protokoly, pomocí kterých spolu mohou komunikovat uživatel a samotná síť. Síť signalizuje uživateli různé mimořádné stavy, pomocí nichž lze spravovat i monitorovat síť, ale právě i vytvářet okruhy. Na SVC se tak komunikace skládá ze dvou kroků: z vytvoření virtuálního okruhu a z jeho vlastního využití ke komunikaci. S SVC se setkáme např. u protokolu ISDN. V přeneseném smyslu opět můžeme říci, že SVC vytváří protokol TCP.

PVC je obdobou pevných linek a SVC je obdobou komutovaných (vytáčených) linek v telefonní síti.



**Poznámka:** Protokoly využívající virtuální okruhy se anglicky nazývají Connection Oriented Network Services (CONS) a protokoly dopravující své pakety bez sestavení virtuálních okruhů se anglicky nazývají Connection-Less Network Services (CLNS), tj. spojované a nespojované síťové služby.

## Kapitola 2

# Nástroje pro sledování a zkoumání sítě

Mohlo by se zdát, že využívání nástrojů (programů) pro sledování a zkoumání sítě je výsadou hackerů a správců sítě. Avšak nejenom jich. Jsou totiž skvělou demonstrační pomůckou i pro studium síťových protokolů. Pokud si na PC nainstalujete program Wireshark, kterým si budete pakety popisované dále v této publikaci odchytávat na síti, uvidíte, že studium TCP/IP bude podstatně snazším.

První skupinou nástrojů popisovaných v této kapitole jsou nástroje pro sledování (monitorování) sítě (*Packet sniffers*). Tyto nástroje slouží ke sledování dat procházejících sítí. Sledováním se rozumí zachytávání linkových rámců v síti a jejich ukládání do paměti. Následně pak prohlížení obsahu jednotlivých zachycených rámců. Tyto nástroje jsou také nepostradatelným pomocníkem pro programátory, kteří vyvíjejí síťové aplikace. Např. napíšete-li aplikaci klient-server, spusťte ji „a nic“ – klient se serverem ani nenavází spojení. V takovém okamžiku nevíte, zdali je na vině klient, nebo server. Pokud budete odchytávat datové rámce, pak např. zjistíte, že klient odesal datový rámec, ale server na něj vůbec nezareagoval, takže jste zjistili, že závada bude pravděpodobně na straně serveru. Nebo jste si všimli, že data odeslaná klientem jsou jiná, než jste předpokládali...

V celé této publikaci budeme využívat zejména program Wireshark, který je pokračováním oblíbeného programu Ethereal. Ve starších verzích Windows byl k dispozici skvělý program „Sledování sítě“ (*Network Monitor*), který jsme ale v MS Vista již neobjevili. Obdobných programů je celá řada, ale za ty se většinou musí platit. V operačních systémech UNIX je běžně k dispozici program tcpdump. Program `tcpdump` nemá na rozdíl od zmíněných programů grafické uživatelské rozhraní, takže je spíše určen pro použití ve skriptech. To nám však nebrání si v programu Wireshark prohlédnout pakety nasbírané programem `tcpdump`.



**Poznámka:** Nástroje pro sledování sítě jsou pouze spodním patrem pro odhalování bezpečnostních incidentů na síti, tj. pouze sbírají data, ve kterých mohou další nástroje vyhledávat a signalizovat bezpečnostní incidenty. Těmito nadstavbovými nástroji se však v této publikaci nezabýváme.

Existují i hardwarové nástroje pro sledování sítě. Jaké jsou jejich výhody? Tyto nástroje jsou nepostradatelné zejména pro technický personál. Softwarové nástroje zobrazí pouze rámce, které jsou v pořádku. Může se např. stát, že stanice na síti má poškozenou síťovou kartu a generuje poškozené rámce. Takovou stanici lze těžko objevit pomocí softwarových nástrojů. A navíc na nejrůznějších jednoúčelových síťových boxech (např. přepínačích) vlastní nástroj pro sledování sítě nespustíte. My použijeme pouze ten nejjednodušší hardwarový nástroj AirPcap – a to pro sledování bezdrátových sítí (kap. 3 a 4).

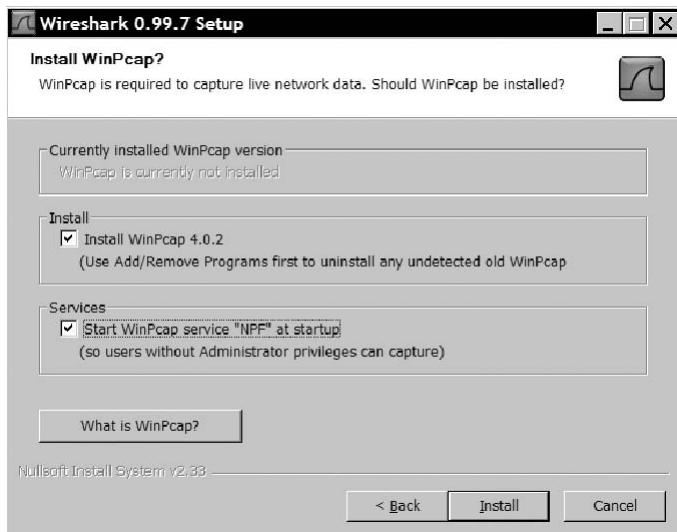
Druhou skupinou nástrojů jsou nástroje pro zkoumání ostatních systémů připojených na síť. Zatímco nástroje pro sledování sítě bezbranně sledují a shromažďují data procházející sítí, nástroje pro zkoumání sítě už tak bezbranné nejsou. Při zkoumání sítě se totiž již generují pakety, kterými se síť zkoumá, takže můžeme odhalovat slabiny v počítačích připojených na síť. Jako příklad programu pro zkoumání sítě si uvedeme nástroj `nmap`.

Asi nejožehavějším problémem nasazení nástrojů pro sledování a zkoumání sítě je otázka bezpečnosti. Argumentem proti používání těchto nástrojů bývá skutečnost, že pomocí nich lze velice snadno zjistit stálá hesla uživatelů nebo obecně slabá místa.

My naopak považujeme demonstraci odchycení hesel či zjištění slabých míst za užitečnou, protože firma po takové demonstraci (nikoliv až po prvních bezpečnostních incidentech) přejde na jinou autentizaci než pomocí jména uživatele a stálého nezabezpečeného hesla či obecně přijme nápravná opatření pro eliminaci slabých míst.

## Packet driver

Abychom mohli sledovat přicházející (resp. odcházející) pakety, musíme mezi síťové rozhraní a zbytek operačního systému vložit nějakou komponentu, která bude schopna procházející pakety sledovat a případně je předávat nějakému programu, který je bude protokolovat či nějak zobrazovat. Taková komponenta se často označuje jako *packet driver* nebo *packet filter*. Součástí instalace programu Wireshark pro Windows je již i instalace *packet driveru WinPcap* (obr. 2.1).



Obrázek 2.1: Instalace WinPcap ve Windows

## Promiskuitní mód

Sítové karty systémů připojených na LAN naslouchají provozu na LAN, tj. čtou jednotlivé procházející linkové rámce, které zpravidla začínají linkovou adresou příjemce. Pakliže stanice zjistí, že při-

jímaný rámec jí není adresován, zpravidla se jím již dále nezabývá. Prakticky to znamená, že *packet driver* může obdržet pouze rámce, které jsou adresované stanici, na které *packet driver* běží (plus rámce, které stanice sama odesílá do sítě). Pokud nám to nestačí a chceme sledovat veškerý provoz na našem segmentu LAN, pak musíme kartu přepnout do tzv. promiskuitního módu. V promiskuitním módu pak karta čte všechny rámce procházející sítí a my můžeme sledovat kompletní provoz na našem segmentu LAN.

**Upozornění:** Nezapomínejte, že stanici jsou adresovány i oběžníky!

Pokud nemáme kartu v promiskuitním módu, jsme schopni sledovat rámce odesílané naší stanicí a rámce adresované naší stanici. To jsou jednak rámce mající některou z adres naší stanice, ale i všechny všeobecné oběžníky. Dále uvidíme adresné oběžníky, které naše stanice přijímá. To je však složitější problém, kterému je věnována kapitola „Oběžníky a linkový protokol“. Všeobecným oběžníkem rozumíme rámec určený všem stanicím na LAN, adresným oběžníkem pak rámec určený více stanicím na LAN (nikoliv všem).

Jiným problémem je sledování provozu na tzv. přepínaném Ethernetu, tj. na LAN, kdy každá stanice leží na samostatném segmentu LAN. Obecně totiž platí, že pokud chceme sledovat komunikaci na jiných segmentech LAN, než na kterém je připojen náš počítač, musíme si uvědomit, že námi hledané rámce nejsou standardně opakovány na segment, na kterém je náš počítač. To lze na mnohých přepínačích obejít za využití ARP-proxy (viz str. 156-157). Hacker zase využije techniku nazývanou „Otrava ARP“ (*ARP poisoning*). Tato technika je detailně popsána v publikaci „Hacking bez tajemství“ (vydána též v nakladatelství Computer Press).

Pro běžnou práci správce sítě jsou ale takové techniky krkolemné. Správce potřebuje přijít k počítači a odchytit jeho provoz. V takovém případě je naprosto nepostradatelným pomocníkem malý opakovač (*repeater*) s alespoň třemi porty. Sledovaný počítač odpojíme ze zásuvky pro LAN a připojíme do tohoto opakovače. Další port opakovače propojíme se zásuvkou pro LAN a na třetí port opakovače si připojíme svůj notebook s programem Wireshark. Pokud opakovač nesezenete už ani v bazaru, pak mnohé přepínače (*switch*) lze přepnout do opakovacího módu. Některé přepínače mají dokonce tzv. servisní rozhraní, které je mj. určeno k monitorování veškerého provozu.

Trochu jiná situace je u bezdrátových sítí. Zde pro odchytávání provozu sousedů je dobré se vybavit specializovaným hardwarem jakým je např. AirPcap.

## Program Wireshark

Wireshark je nástroj pro sledování sítě (*sniffer*) a síťový analyzátor, tj. nástroj, který nejenom zachycuje pakety v síti, ale navíc umožňuje jejich analýzu, která zahrnuje jak pitvání datových paketů do nejmenších detailů, tak vytváření nejrůznějších statistik a grafů. Dokáže však pitvat i síťové pakety zachycené jinými nástroji jako jsou: tcpdump, LANalyzer, HP-UX nettl trace, Microsoft Network Monitor, Sun Snoop a mnoho dalších.

Silnou stránkou Wiresharku je podpora více než 1000 síťových protokolů, jejichž pakety dokáže právě pitvat do nejjemnějších detailů. Za zmínu rovněž stojí dokumentace, která je obsáhlá a volně dostupná na Internetu.

## Začínáme s Wiresharkem

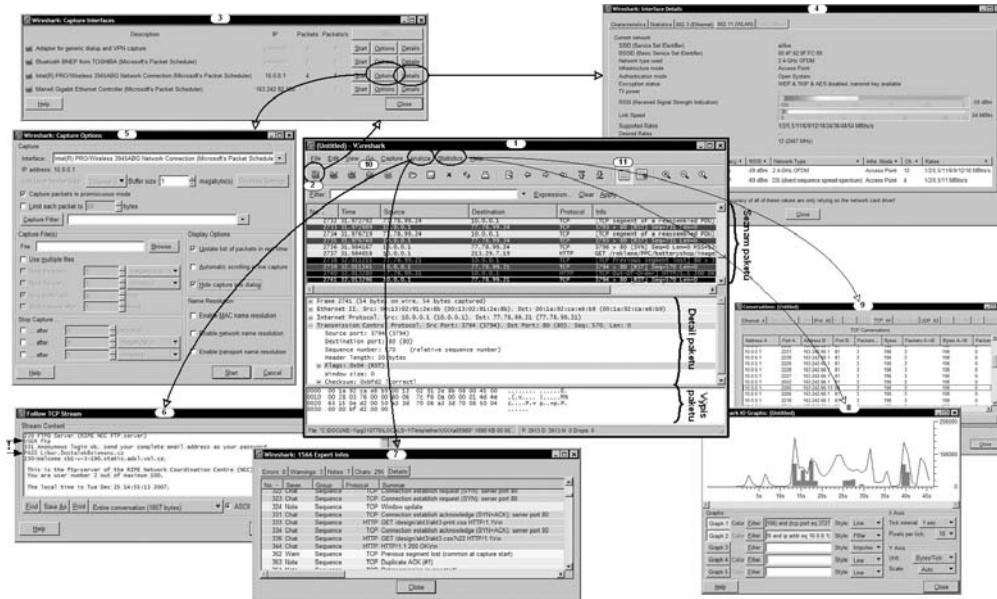
Předpokládáme, že na počítači máme předem spuštěn packet driver (tj. ve Windows WinPcap). Nyní spustíme program Wireshark (obr. 2.2). Hlavní okno (1) se na počátku objeví bez oblastí Seznam paketů, Detail paketu a Výpis paketu.

Sběr paketů začneme pomocí menu Capture → Interfaces nebo pomocí ikony „List the available capture interfaces“ (2). Tato volba nám zobrazí okno Capture Options (3) se seznamem zjištěných síťových rozhraní. Pokud by toto okno neobsahovalo žádné síťové rozhraní, pak nejspíše nemáme správně nainstalován packet driver.

V okně (3) vidíme u každého síťového rozhraní tři tlačítka „Start“, „Options“ a „Details“:

- ◆ Tlačítko „Details“ zobrazí maximální množství informací o síťovém rozhraní (4). Tyto informace program „Wireshark“ nijak neověruje – přebírá je od ovladače síťového rozhraní.
- ◆ Tlačítko „Options“ zobrazí okno Capture Options (5), kde
  - Volbou „Interface“ je možné zvolit jiné síťové rozhraní.
  - Volba „Capture packets in promiscuous mode“ přepne síťové rozhraní do promiskuitního módu.
  - Volbou „Limit each packet to“ se z každého sebraného paketu zaznamená pouze x prvních zvolených bajtů.
  - Volbou „Capture Filter“ omezujeme filtrem (podmínkou) množství nasbíraných paketů. Při sběru paketů je zásadním problémem to, že se nasbírá velkém množstvím paketů, ve kterém je pak obtížné něco najít. Prvním řešením tohoto problému je sbírat jen některé pakety. Touto volbou je možné zadat filtr specifikující, které pakety se mají sbírat. Pokud se nic nezadá, pak se sbírají všechny pakety. My preferujeme, alespoň na počátku, sběr všech paketů a následně použití filtrů až při zobrazování paketů. Zákon schválnosti totiž hovoří neúprosnou řečí: „Capture Filter“ vám odfiltruje právě ten paket, který hledáte.
  - Oblast „Capture File(s)“ umožňuje specifikovat soubory, do kterých se budou zapisovat sebrané pakety. V případě, že se žádný soubor nespecifikuje, pak se pakety zaznamenávají do dočasného souboru.
  - Oblast „Stop Capture“ umožňuje specifikovat událost, po které se sběr paketů automaticky zastaví. Takovou událostí může být stanovený počet paketů, dosažení velikosti souboru se zaznamenanými pakety či vypršení času zaznamenávání paketů.
  - Oblast „Display Options“ obsahuje:
    - „Update list of packets in real time“ – pokud tuto volbu nezvolíme, tak se pakety sbírají do souboru a my je při sběru nevidíme – vidíme pouze statistiku sběru. Pakety se zobrazí až po zastavení sběru. Pokud chceme průběžně vidět sebrané pakety, pak tuto volbu musíme zatrhnout.
    - „Automatic scrolling in live capture“ – automaticky posouvá zobrazené pakety při jejich sběru.
    - „Capture dialog box“ – potlačí okno statistiky sběru paketů.
  - Oblast „Name resolution“ specifikuje vrstvy, jejichž adresy se mají překládat na jména.
- ◆ Tlačítko „Start“ odstartuje sběr paketů.

Sběr paketů ukončíme volbou Capture → Stop nebo pomocí ikony (10).



Obrázek 2.2: Program Wireshark

Nyní můžeme v klidu pitvat nasílané pakety. V hlavním okně (1) máme tři oblasti:

- ◆ Seznam paketů (*Packet List*) – obsahující seznam nasílaných paketů. V tomto seznamu si zvolíme jeden paket, který se nám zobrazí ve zbylých dvou oblastech.
- ◆ Detail paketu (*Packet Details*) – zobrazující detailní (rozpitovaný) výpis konkrétního paketu zvoleného v oblasti *Packet List*.
- ◆ Výpis paketu (*Packet Bytes*) – zobrazující konkrétní hodnoty jednotlivých bajtů paketu. Pokud v oblasti *Packet Details* zvolíme konkrétní položku pitvaného paketu, jsou její bajty v oblasti *Packet Bytes* zvýrazněny.

Základní práce uživatele s Wiresharkem se pak skládá z pohybu po seznamu zobrazených paketů a z jejich prohlížení.

## Filtry

Největším problémem při analýze nasílaných dat je nalezení konkrétního paketu, který nás zajímá. To je problém zejména v případě, že nasílaných paketů je velké množství. Cílem filtrů je odfiltrovat pro nás nezajímavé pakety.

Wireshark používá dva typy filtrů:

- ◆ Capture Filters (*Capture → Capture Filters*), které filtroují pakety již při jejich sběru. Zejména na sítích s velkým provozem je tato volba výhodná, abychom nemuseli zpracovávat velké soubory.

- ◆ *Display Filters* (Analyze → Display Filters), které filtruji pakety až při jejich zobrazování v oblasti *Packet List*.

Bohužel každý z těchto typů filtrů má jinou syntaxi. Zatímco v případě *Display Filters* máme k tvorbě filtrů uživatelsky přívětivý nástroj (tlačítko „Expression...“), tak v případě *Capture Filters* musíme takový výraz napsat „rukou“. A navíc tyto filtry se píší podle pravidel filtrů pro program *tcpdump*.

## Display Filters

Než se pustíme do tvorby filtrů, musíme se seznámit s operacemi, pomocí kterých se filtry tvoří:

1. Operace porovnávání se zapisují buď anglickými zkratkami, nebo operandy používanými v jazyce C:
  - ◆ eq (nebo ==) pro rovnost
  - ◆ ne (nebo !=) pro nerovnost
  - ◆ gt (nebo >) pro větší než
  - ◆ lt (nebo <) pro menší než
  - ◆ ge (nebo >=) pro větší nebo rovno
  - ◆ le (nebo <=) pro menší nebo rovno
2. Logické výrazy pro vytváření složitějších filtrů obsahujících více operací:
  - ◆ and (nebo &&) pro logický AND
  - ◆ or (nebo ||) pro logický OR
  - ◆ xor (nebo ^^) pro logický XOR
  - ◆ not (nebo !) pro negaci

Nyní již můžeme přistoupit k tvorbě filtru. Abychom vytvořili nějaký byť jednoduchý výraz, potřebujeme znát ještě operandy. Jedním operandem bude vždy název nějakého pole v paketu a druhým operandem pak jeho hodnota.

Např. filtr tvořený výrazem:

```
ip.version == 4
```

nám ze všech datových paketů vybere všechny, které nesou IP datagram verze 4.

Pokud se ptáte, kde jsme vzali ten název pole „*ip.version*“, pak si otevřete okno Analyze → *Display Filters*. Stiskem tlačítka „New“ se vytvoří nový filtr a nyní již stiskněte „Expression...“ a můžete se nechat inspirovat při tvorbě filtrů.

Zajímavým typem filtru je jednoduchý filtr, který specifikuje, že se mají zobrazit/sbírat pakety určitého protokolu. Takový filtr se skládá jen ze jména tohoto protokolu. Např. filtr:

```
pop
```

zobrazí všechny pakety protokolu POP. Jenže je třeba upozornit na jednu vlastnost Wiresharku týkající se speciálně tohoto typu filtrů. Pokud chceme sledovat všechny pakety TCP spojení protokolu POP3, pak tímto filtrem budou vybrány jen ty „výživné“. Tento filtr např. nevybere pakety TCP spojení, kterými protějšek potvrzuje příjem dat apod. Pokud např. sledujeme statistiky, pak spíše než filtr „*pop*“ je lépe využít filtr:

```
tcp.port == 110
```

který zachytí veškeré pakety TCP spojení navázaného pro protokol POP3, protože POP protokol používá port 110/tcp.

Nyní již máme dva výrazy, takže je můžeme spojit např. logickým AND:

```
ip.version == 4 && pop
```

nám zobrazí pakety protokolu IP verze 4, které nesou aplikační protokol POP.

Uvedeme si ještě složitější příklad:

```
(ip.addr eq 195.47.37.196 and ip.addr eq 172.24.122.109) and (tcp.port eq 110  
and tcp.port eq 3053)
```

Tento filtr filtrouje všechny pakety jednoho TCP spojení mezi počítači 195.47.37.196 a 172.24.122.109, přičemž jeden počítač používá TCP port 110 (tj. POP3) a druhý používá klientský port 3053:

Z uvedeného příkladu je vidět, že do filtrů pro Wireshark se nepíše, která adresa je zdrojová a která cílová, čímž se zápis filtru zjednoduší. To je rozdíl např. od zápisu filtrů pro boxy CISCO či program `tcpdump`. Nutno však dodat, že i ve Wiresharku lze specifikovat zdrojovou či cílovou adresu (`ip.src` či `ip.dst`).

Pokud chceme např. zobrazit jen pakety protokolu POP3, které nesou heslo, pak doporučují filtr:

```
frame[54:4] == 50:41:53:53
```

Paket protokolu POP3 nesoucí heslo totiž od 55 bajtu obsahuje čtyři znaky dlouhý řetězec PASS (což je hexadecimálně 50 41 53 53). Jenže samotné heslo je vcelku na nic – ještě potřebujeme jméno uživatele a to se vyskytuje v paketech, které mají ve stejném místě řetězec USER (hexadecimálně 55 53 45 52). Takže výsledný filtr bude:

```
frame[54:4] == 55:53:45:52 or frame[54:4] == 50:41:53:53
```

## Capture filter

Capture filter je skvělý tehdy, když sledujeme síť s velkým provozem. Pak nechceme shromažďovat ohromné soubory dat, ale chceme vybrat je tu třešinku, která nás zajímá. Jak jsme již uvedli, Capture filter má odlišnou syntaxi než Display filter. Má totiž stejnou syntaxi jakou používá program `tcpdump`!

Pro ilustraci si popíšeme Capture filter, který je obdobou filtru z předchozího odstavce, tj. filtr, kterým budeme sledovat, jestli v naší síti někdo nezabezpečeně nepřistupuje na server POP3 pomocí jména a hesla. Posunutí tentokrát ale nebudeme počítat od počátku linkového rámce, ale od počátku TCP segmentu.

Úvaha je jednoduchá. POP3-protokol je aplikačním protokolem. To znamená, že jeho pakety jsou zabalenы v segmentech TCP. TCP záhlaví má standardně 20B. Takže:

- ◆ POP3 paket standardně začíná za 20. bajtem TCP záhlaví.
- ◆ POP3-paket nesoucí jméno uživatele začíná 4 bajty dlouhým řetězcem „USER“, tj. hexadecimálně 0x55534552.
- ◆ POP3-paket nesoucí heslo začíná 4 bajty dlouhým řetězcem „PASS“, tj. hexadecimálně 0x50415353.

Pokud chceme programem Wireshark sbírat jen aplikační pakety začínající buď řetězcem „USER“, nebo začínající řetězcem „PASS“, pak použijeme Capture filter:

```
tcp[20:4] = 0x55534552 or tcp[20:4] = 0x50415353
```

Tím dostaneme seznam uživatelských jmen a hesel přihlašovaných uživatelů k serveru POP3.

### Příklad

Jelikož jsme uvedli, že Capture filter využívá stejnou syntaxi jako program `tcpdump`, tak si můžeme stejný příklad uvést i pro příznivce Linuxu. Představme si, že se stejným filtrem spustí správce program POP3 `tcpdump` na serveru POP3:

```
# tcpdump -w soubor tcp[20:4] = 0x55534552 or tcp[20:4] = 0x50415353
```

Pak v souboru `soubor` bude mít všechna uživatelská jména a hesla přihlašovaných uživatelů k serveru POP3. A pokud nemá na serveru Wireshark, tak si soubor `soubor` pošle na PC, na kterém ho jistě má. Nyní si uvědomíme, jak mocným nástrojem je protokol SSL/TLS, kterým rovněž můžeme zabezpečnit komunikaci i se serverem POP3.



**Poznámka:** Nyní si uvědomíme, jak mocným nástrojem je protokol SSL/TLS, kterým můžeme zabezpečnit komunikaci i se serverem POP3.

Detailní popis Capture filter získáte příkazem `man tcpdump`.

### Coloring rules

Asi si řeknete, že vybarvování oken ve Windows není to podstatné při analýze nasbíraných paketů. Jenže kupodivu tomu tak není. Pokud si jednotlivé typy paketů v seznamu barevně odlišíme, pak se v tomto seznamu budeme rozhodně lépe orientovat. Zapnutí „Coloring Rules“ je možné pomocí ikony (11) na obr. 2.2.

Barevné rozlišení se provádí volbou menu `View → Coloring rules`. Zde můžeme vytvářet, upravovat i rušit pravidla pro vybarvování. Základem každého pravidla je opět filtr, kterým definujeme, které pakety patří do konkrétního vybarvovacího pravidla. Pak již jen nastavíme barvu textu a barvu pozadí pro konkrétní pravidlo a můžeme jej aplikovat.

### Follow TCP stream

Tato volba (v menu `Analyze`) je podle našeho názoru tím nejlepším, co Wireshark poskytuje. Pokud si totiž zvolíme jeden paket z nějakého TCP spojení, pak nám Wireshark automaticky vytvoří filtr zobrazující všechny pakety tohoto spojení. Avšak navíc ještě spustí okno, ve kterém je dialog celého spojení vidět v plné kráse v textovém tvaru.

Skvěle se to demonstruje např. na protokolu FTP – okno (6) na obr. 2.2. Nejprve si nachytáme paketu např. FTP spojení na server `ftp.ripe.net`. Zvolíme jeden paket z řídicího kanálu spojení (protokol ftp používá dva kanály: řídicí pro přenos povelů a datový pro přenos dat). Řídicí kanál používá port serveru 21. Nyní již můžeme zvolit `Analyze → Follow TCP Stream` a dostaneme okno (6).

Všimněte si, že z tohoto výpisu je opět velice pěkně vidět přihlašovací jméno (ftp) a heslo (e-mailová adresa).

Takový výpis s heslem je na první pohled efektivní. Ale neméně zajímavý je tento výpis i v případě binárních dat. Ten oceníme např. při analýze protokolu S/MIME. Analýzu protokolu S/MIME nelze totiž dělat nad jednotlivými síťovými pakety, ale nad celým TCP tokem (*stream-em*). Celou zprávu S/MIME si pak v hexadecimálním tvaru zkopiujeme do jiného programu k další analýze (např. do programu WinVi, openssl či dumpasn1). Ale o tom se více dozvíte ve Velkém průvodci PKI a technologií elektronického podpisu vydaném rovněž vydavatelstvím Computer Press.

## Statistiky

Menu Statistics nám umožňuje získat mnohé statistiky. Získání jednotlivých statistik je intuitivně velmi snadné. Efektní je volba IO Graphs (8), pomocí níž je možné získat velice užitečné grafy. Implicitně vidíme krivku veškerého síťového provozu.

Jak je vidět z (8), je možné specifikovat až 5 grafů. Specifikace grafu se opět provádí pomocí filtru (jak jinak!) a pak se ještě volí styl grafu. Na obr. 2.2 vidíme příklad takového grafu v okně (8). Pomocí dvou filtrů jsme specifikovali dva toky a každý pak vyjádřili jiným grafem v okně (8).

Kromě volby „IO Graphs“ jsou neméně zajímavé statistiky vztahující se k jednotlivým protokolům. Statistiky vztahující se k protokolu HTTP nám tak sdělí, kolik bylo jakých odpovědí webového serveru (kolik bylo informativních odpovědí serveru, kolik bylo úspěšných odpovědí serveru, kolik bylo přesměrování atd.). Pokud používáte protokol RTP (Real Time Protocol), pak jsou statistiky Wiresharku rovněž velice užitečné. Zajímavé jsou také statistiky protokolu TCP, na jejichž grafech je pěkně vidět RTP či throughput protokolu TCP. Statistika „Conversation“ nám sumárně ukáže rozpad komunikace mezi jednotlivými uzly sítě atd.

## Tisk a Export

Ne vždy se nám hodí si analyzované pakety prohlížet na obrazovce. Někdy je potřeba mít obsah paketů i vytisknout. Wireshark má pro tyto účely uzpůsoben tisk. A pokud výstup potřebujeme zkopirovat do jiného programu, pak použijeme export do souboru. Je možné zvolit, jestli ve vypisovaných paketech mají být úplně všechny položky rozvinuty (Packet details: All collapsed) anebo mají být rozvinuty jen ty, které jsou rozvinuty na obrazovce (Packet details: As displayed).

Následující příklad obsahuje výpis (Packet details: As displayed):

No.	Time	Source	Destination	Protocol	Info
70	2.683246	160.217.208.35	10.0.0.1	TCP	23 > 3096 [ACK]
Ethernet II,					
Destination: 00:13:02:91:2e:8b (00:13:02:91:2e:8b)					
Address: 00:13:02:91:2e:8b (00:13:02:91:2e:8b)					
.... .0 .... .... .... = IG bit: Individual address (unicast)					
.... ..0. .... .... .... = LG bit: Globally unique address (factory default)					
Source: 00:1a:92:ca:e6:b9 (00:1a:92:ca:e6:b9)					
Address: 00:1a:92:ca:e6:b9 (00:1a:92:ca:e6:b9)					
.... .0 .... .... .... = IG bit: Individual address (unicast)					

```

.... .0. .... .... .... = LG bit: Globally unique address (factory default)
Type: IP (0x0800)
Trailer: 000000000000
Internet Protocol, Src: 160.217.208.35 (160.217.208.35), Dst: 10.0.0.1 (10.0.0.1)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x20 (DSCP 0x08: Class Selector 1; ECN: 0x00)
    0010 00.. = Differentiated Services Codepoint: Class Selector 1 (0x08)
    .... ..0. = ECN-Capable Transport (ECT): 0
    .... ..0 = ECN-CE: 0
Total Length: 40
Identification: 0x33b2 (13234)
Flags: 0x04 (Don't Fragment)
    0... = Reserved bit: Not set
    .1.. = Don't fragment: Set
    ..0. = More fragments: Not set
Fragment offset: 0
Time to live: 55
Protocol: TCP (0x06)
Header checksum: 0x9500 [correct]
    [Good: True]
    [Bad : False]
Source: 160.217.208.35 (160.217.208.35)
Destination: 10.0.0.1 (10.0.0.1)
Transmission Control Protocol, Src Port: 23 (23), Dst Port: 3096 (3096), Seq: 22, Ack: 29
Source port: 23 (23)
Destination port: 3096 (3096)
Sequence number: 22      (relative sequence number)
Acknowledgement number: 29      (relative ack number)
Header length: 20 bytes
Flags: 0x10 (ACK)
    0... .... = Congestion Window Reduced (CWR): Not set
    .0... .... = ECN-Echo: Not set
    ..0. .... = Urgent: Not set
    ...1 .... = Acknowledgment: Set
    .... 0... = Push: Not set
    .... .0.. = Reset: Not set
    .... ..0. = Syn: Not set
    .... ...0 = Fin: Not set
Window size: 5888 (scaled)
Checksum: 0x4436 [correct]
    [Good Checksum: True]
    [Bad Checksum: False]
[SEQ/ACK analysis]
    [This is an ACK to the segment in frame: 69]
    [The RTT to ACK the segment was: 0.034854000 seconds]

0000 00 13 02 91 2e 8b 00 1a 92 ca e6 b9 08 00 45 20  ....E
0010 00 28 33 b2 40 00 37 06 95 00 a0 d9 d0 23 0a 00  .(3.@.7.....#..

```

Pro případ, že bychom chtěli obsah paketů zpracovávat moderními nástroji pro práci s XML, lze exportovat zachycené pakety ve formátu XML, a to buď v detailu (XML packet detail), nebo jako přehled (XML packet summary).

## Další utility

Wireshark zpravidla spouštíme bez parametrů a vše zadáváme z menu. Pro ty, kteří nejsou tak líní, je možné specifikovat téměř vše na příkazovém řádku, když se spouští Wireshark. Kromě programu Wireshark se nám zpravidla nainstalují i další utility:

- ◆ TShark – je terminálová obdoba programu Wireshark, která je velice podobná programu tcpdump na UNIXu. Příkazem „TShark -D“ si nejprve zjistíme názvy síťových rozhraní a pak např. příkazem „TShark -i rozhraní“ získáme výpis paketů, který je velice podobný výpisu programu tcpdump.
- ◆ Dumpcap – rádková utilita na sbírání paketů (rádkový sniffer). Např. příkaz „Dumpcap -i interface -w soubor“ nám nasbírá na uvedeném rozhraní pakety do souboru „soubor“.
- ◆ Editcap – utilita umožňující upravovat a konvertovat soubor s nasbíranými pakety.
- ◆ Mergecap – sloučuje dva nebo více souborů s nasbíranými pakety.

## Domácí cvičení

Nyní jste již připraveni pro první experimenty s programem Wireshark. Jako první cvičení doporučujeme odchycení vlastního hesla (v žádném případě nedoporučujeme odchytávání cizího hesla – to stejně nejde, pokud nemáte síťovou kartu přepnutou do promiskuitního módu nebo jste na přepínaném Ethernetu). Nejprve přemýšlejte, nemáte-li někde stálé heslo. Může se např. jednat o vybírání poštovní schránky protokolem POP3 nebo o komunikaci s webovým serverem či serverem FTP. Pokud je ovšem heslo posíláno přes kanál, který je zabezpečen např. protokolem SSL, pak nebude zachyceno v čitelné podobě!

Pokud vás nenapadne žádný server FTP, pak vyzkoušejte anonymní server *ftp.ripe.net*. V případě anonymních serverů FTP se uvádí jméno uživatele „anonymus“ nebo „ftp“ a jako heslo se používá vaše e-mailová adresa.

V případě, že budete odchytávat heslo v protokolu HTTP, máte dvě možnosti:

- ◆ Jméno a stálé heslo se zadává ve webovém formuláři a zpracovává přímo aplikací – je to opravdu snadné.
- ◆ Využívá se standardní autentizace „Basic“ protokolu HTTP (nesmí se jednat o protokol HTTPS – tam heslo odchytit nelze, protože spojení je šifrováno). Při autentizaci „Basic“ je jméno a heslo součástí každého dotazu (pokud se nejedná o přístup na anonymní server). Avšak protože hlavičky protokolu HTTP nesmí obsahovat znaky, které nejsou ASCII, jsou jméno uživatele a heslo odděleny dvojtečkou a celé je to kódováno Base64. Je třeba provést dekódování buď ručně, nebo nějakým programem. Hlavička, která nese tyto informace, je:

Authorization: Base64(uživatel:heslo)

nebo v případě autentizace vůči firewallu (proxy) se jedná o hlavičku:

Proxy-Authorization: Base64(uživatel:heslo)

`Base64()` zde vyjadřuje, že argument je Base64 kódován do sedmibitového tvaru.

Pokud nemáme trpělivost si dekódování Base64 provést ručně na papíře, pak je možné použít např. program `openssl` s parametry „`enc -a -d`“.

## Program nmap

*Tato kapitola je určena pro pokročilejší uživatele.*

Druhým nástrojem, bez kterého se při studiu sítí těžko obejdeme, je program `nmap`. Zatímco Wireshark bezbranně monitoroval síť, Pak programem `nmap` můžeme síť zkoumat aktivně. Program `nmap` je součástí standardních distribucí Linuxu. Na ostatní systémy jej můžeme stáhnout z Internetu.

Obecná syntaxe příkazu `nmap` je:

```
nmap [přepínače] {sledované systémy/sítě}
```

### Co program nmap dělá?

Program `nmap` zkoumá zadané počítače postupně ve dvou fázích:

- ◆ Nejprve zjistí, jsou-li počítače vůbec dostupné. Tato fáze se nazývá vyhledávání připojených systémů do sítě (zobecnění příkazu `ping`). Celou tuto fázi lze přeskočit zadáním přepínače `-P0`.
- ◆ Pro každý vyhledaný počítač zjišťuje, jaké protokoly jsou na něm dostupné, jaké porty TCP/UDP jsou na něm otevřené, běží-li na systému FTP proxy, jaké verze OS na něm běží apod. Tato fáze se označuje jako „scanning“. Celou tuto fázi lze potlačit volbou přepínače `-sP`.

### První fáze: Vyhledávání připojených systémů

Vyhledávat systémy připojené k síti je mj. možné na základě následujících protokolů:

- a. ICMP echo (přepínač `-PE`) – lidově „**ICMP ping**“. V tomto případě se opravdu jedná o obdobu příkazu `ping` (viz kap. 5). Tento přepínač je implicitní volbou pro zkoumání počítačů mimo LAN.
- b. ICMP timestamp (přepínač `-PP`). Viz kap. 5.
- c. ICMP netmask (přepínač `-PM`). Viz kap. 5.
- d. TCP SYN (přepínač `-PS`) – lidově „**SYN-ping**“. Na zadané počítače program `nmap` odešle segmenty TCP s nastaveným příznakem SYN (navazování spojení) a sleduje odpovědi (viz kap. 9).
- e. TCP ACK (přepínač `-PA`) – lidově „**ACK-ping**“. Na zadané počítače odešle segmenty TCP s nastaveným příznakem SYN (navazování spojení) a sleduje odpovědi (viz kap. 9).
- f. UDP (přepínač `-PU`) – lidově „**UDP ping**“. Na zadané počítače odešle datagramy UDP a sleduje odpovědi (viz kap. 10).
- g. ARP ping (přepínač `-PR`) – lidově „**ARP-ping**“. Zadaných počítačů na LAN se ARP protokolem dotáže na jejich linkové (MAC) adresy (viz kap. 5). Tento přepínač je implicitní volbou na LAN. Jedná se o velice efektivní cestu, jak zjistit, jaké systémy jsou k LAN připojeny. Jedinou obranou je vypnutí protokolu ARP, což je ale velice nepraktické.
- h. Volbou `-P0` je možné fázi vyhledávání dostupných systémů vynechat. Tj. předpokládáme, že všechny zadané systémy jsou dostupné, tj. všechny zadané systému budou i scanovány.

- i. Volbou `-sP` je možné potlačit další fáze prohledávání.
- j. Nezadáním žádné z uvedených voleb program `nmap` testuje dostupnost zadaných systémů v závislosti, jsou-li na LAN (ARP-ping), nebo ne (ICMP ping).

### Příklad 1

Vyhledání všech počítačů na síti 10.0.0.0/24 pomocí ARP-ping (s potlačením 2. fáze „scanning“ parametrem `-sP`):

```
C:\Program Files\Nmap> nmap -PR -sP 10.0.0.0/24
```

```
Starting Nmap 4.53 ( http://insecure.org ) at 2008-02-17 15:50
Host 10.0.0.1 appears to be up.
Host 10.0.0.2 appears to be up.
MAC Address: 00:19:D2:29:51:53 (Intel)
Host 10.0.0.138 appears to be up.
MAC Address: 00:1A:92:CA:E6:B9 (Asustek Computer)
Nmap done: 256 IP addresses (3 hosts up) scanned in 3.328 seconds
```

Výsledkem příkladu 1 je, že jsme zjistili, že na lokální síti 10.0.0.0/24 máme 3 počítače. Pod jejich IP adresami jsou pak uvedeny jejich linkové (MAC) adresy. Pouze u počítače, ze kterého by příkaz prováděn, linková adresa není vypsána.

### Příklad 2

Vyhledání všech počítačů na síti 160.217.208.0/26 pomocí TCP SYN-pingu (opět s potlačením 2. fáze „scanning“ parametrem `-sP`):

```
C:\Program Files\Nmap>nmap -PS -sP 160.217.208.0/26
```

```
Starting Nmap 4.53 ( http://insecure.org ) at 2008-02-17 16:04
Host rum.bf.jcu.cz (160.217.208.32) appears to be up.
Host botanika.bf.jcu.cz (160.217.208.33) appears to be up.
Host tomcat1.prf.jcu.cz (160.217.208.34) appears to be up.
Host tomcat.bf.jcu.cz (160.217.208.35) appears to be up.
Nmap done: 64 IP addresses (4 hosts up) scanned in 8.906 seconds
```

## Druhá fáze: Scanování portů

Na vzdálených systémech je možné scanovat porty mj. následujícím způsobem:

- a. Odesílat na jednotlivé porty vzdáleného systému segmenty TCP s nastaveným příznakem SYN – lidově „**SYN-scan**“ (přepínač `-sS`), a vyhodnocovat odpovědi vzdáleného systému.
- b. Odesílat segmenty TCP s nastaveným příznakem ACK – lidově „**ACK-scan**“ (přepínač `-sA`).
- c. Segmenty TCP se všemi nastavenými bity (rozsvícenými jako vánoční stromeček) – lidově „**Xmas-scan**“ (přepínač `-sX`).
- d. Odesílat segmenty TCP s nenastaveným žádným příznakem (přepínač `-sN`).
- e. Pokoušet se postupně navazovat spojení TCP (přepínač `-sT`).
- f. Odesílat segmenty UDP – lidově „**UDP ping**“ (přepínač `-sT`).

### Příklad 3

Scan TCP portů pomocí techniky „SYN-scan“ počítače [www.playboy.com](http://www.playboy.com). Jelikož tento počítač neodpovídá na „ICMP ping“ a my víme, že je zapnut, potlačujeme fázi vyhledávání připojeného systému přepínačem -P0:

```
# nmap -P0 -sS www.playboy.com

Starting Nmap 4.11 ( http://www.insecure.org/nmap/ ) at 2008-02-17 21:02 CET
Interesting ports on www.playboy.com (216.163.137.3):
Not shown: 1673 closed ports
PORT      STATE      SERVICE
80/tcp    open       http
135/tcp   filtered  msrpc
136/tcp   filtered  profile
137/tcp   filtered  netbios-ns
138/tcp   filtered  netbios-dgm
139/tcp   filtered  netbios-ssn
445/tcp   filtered  microsoft-ds

Nmap finished: 1 IP address (1 host up) scanned in 17.536 seconds
```

### Další zajímavé přepínače

- ◆ Přepínačem -0 zjišťujeme informace o operačním systému sledovaného počítače.
- ◆ Přepínač -s0 je tzv. „IP scan“, který nezjišťuje porty TCP/UDP, ale porty protokolu, které sledovaný systém podporuje (TCP, ICMP, IGMP atd.).
- ◆ Přepínačem -6 otevíráme sledování v protokolu IPv6.
- ◆ Přepínačem -v zapínáme podrobný výpis.
- ◆ Přepínačem -d zapínáme podrobnější „ladicí“ výpis.
- ◆ Přepínač -n potlačuje překlad IP adres na jména DNS.
- ◆ Přepínačem -p se specifikují TCP porty (-p T:porty) nebo UDP porty (-p U:porty), které se mají sledovat.
- ◆ Atd.

### zenmap GUI

zenmap GUI je grafická nadstavba na program `nmap`. Umožňuje snadněji zadávat příkazy a přehledně sledovat výsledky (obr. 2.3).

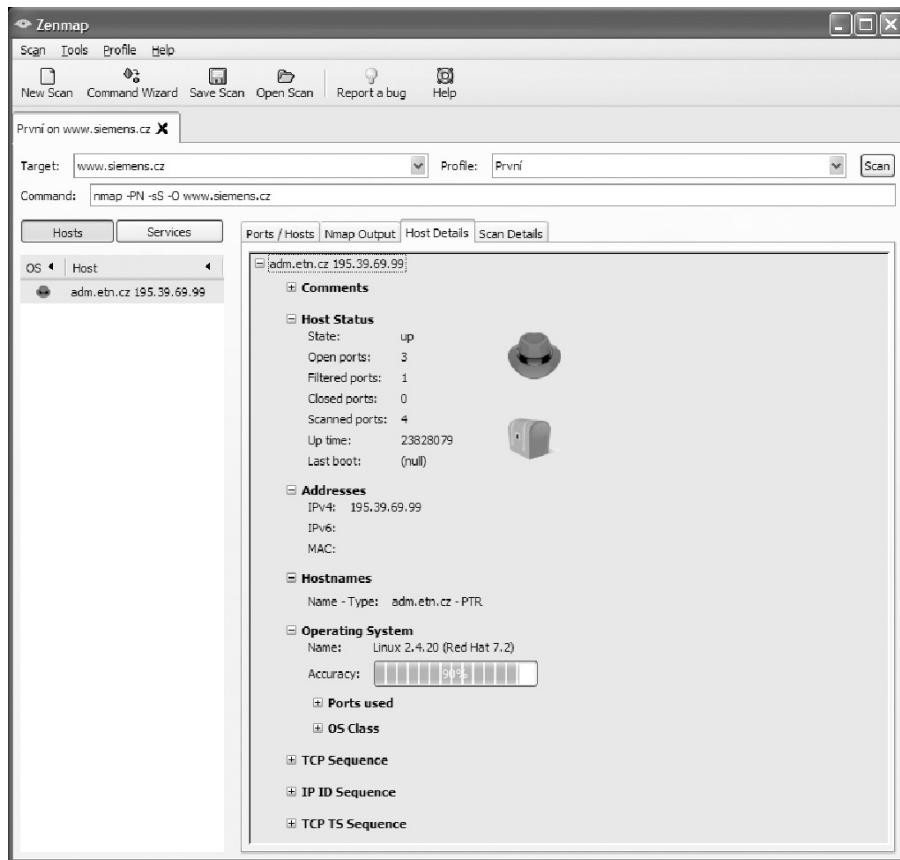
Avšak doporučujeme si nejprve procvičit příkaz `nmap` samotný, teprve pak se bude příjemně pracovat i s zenmap GUI.

### Domácí cvičení

Na LAN zjistěte:

- ◆ Na vaši LAN pomocí ARP-pingu zjistěte připojené počítače.

- ◆ Pokuste se zjistit, jaký běží na připojených počítačích operační systém (volba -O).
- ◆ Zjistěte, jaké porty jsou otevřené na připojených počítačích vaší LAN.



Obrázek 2.3: zenmap GUI



## Kapitola 3

# Fyzická vrstva

Pro studium ostatních kapitol této publikace není bezprostředně nutná znalost této kapitoly, kterou ocení zejména správci sítí.

Pro drtivou většinu uživatelů jsou protokoly fyzické vrstvy „naprosto odtažité“ protokoly, které popisují signály na konektorech (uživateli říkají zástrčkách) na zadní straně počítače, na které je připojena šňůra propojující počítač s počítačovou sítí<sup>1</sup>. Uživatelé starosti o fyzickou vrstvu přenechávají technikům, o nichž se domnívají, že to jsou lidé, „kteří natahují dráty a případně na nich něco měří nějakým voltmetrem nebo čím“. Situace je dnes úplně jiná: technik je spíše správcem programového vybavení, které ovládá všechny „tajuplné krabičky v zamčených místnostech“. Tato představa se v podstatě netýká pouze fyzické vrstvy, ale i linkové vrstvy. Uživatelé se většinou zabývají až IP protokolem. V IP protokolu už totiž „vidí“ nebo „nevidí“ servery či sousedy, kdežto fyzická a linková vrstva zajišťuje pouze komunikaci s nějakou krabičkou na půl cesty k serveru, o které běžný uživatel ani neví, že tam je.

V zásadě rozlišujeme několik typů počítačových sítí: rozlehlé sítě (WAN), lokální sítě (LAN), personální sítě (PAN) apod. Z hlediska fyzické vrstvy jsou v podstatě protokoly pro WAN jednou skupinou protokolů a protokoly pro LAN druhou skupinou.

### **WAN (Wide Area Network)**

Rozlehlé sítě pokrývají velkou škálu situací: od připojení domácího PC k Internetu pomocí sériové asynchronní linky (viz 3.1) rychlostmi uváděnými v kb/s až po mezikontinentální linky realizované podmořskými kably či družicovými spoji o rychlostech uváděných v mnoha Gb/s.

### **LAN (Local Area Network)**

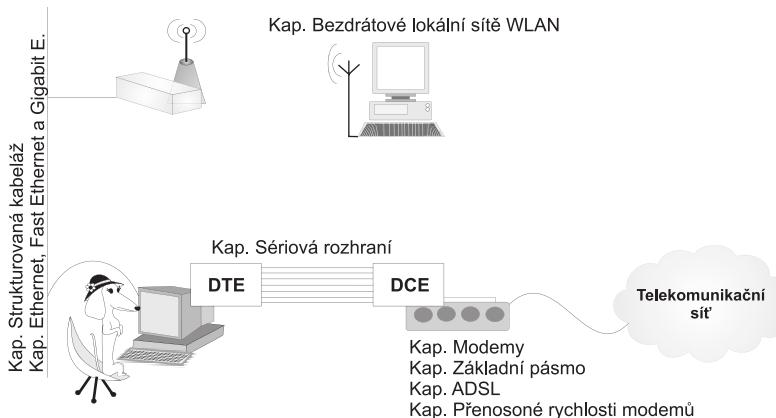
Na LAN spolu komunikuje několik stanic zpravidla na sdíleném médiu. V rámci jedné LAN se používá stejný linkový protokol (např. Ethernet). Dnes se však často pod pojmem LAN míní tzv. rozšířené LAN, které jsou tvořeny jednotlivými LAN. Rozšířené LAN vzniknou spojením jednotlivých LAN pomocí přepínačů (*switch*). Přepínače mají často rozhraní pro různé typy linkových protokolů a umí konvertovat rámce jednoho linkového protokolu na rámce jiného linkového protokolu. Stále častějším je případ, kdy jednotlivou LAN tvoří pouze dva prvky. Přitom jedním z nich je právě přepínač, který tvoří jádro rozšířené LAN.

Z hlediska fyzické vrstvy nás však budou zajímat pouze jednotlivé LAN, protože na rozšířené LAN se fyzická vrstva dívá jako na soustavu jednotlivých LAN. Na LAN je běžné použití všeobecných oběžníků, tj. rámců určených všem stanicím na LAN.

Přenosové rychlosti na dnešních LAN se pohybují od 10 Mb/s až po 10 Gb/s – a vývoj jde pochopitelně dále.

### PAN (Personal Area Network)

PAN je určena pro výměnu dat mezi zařízeními v bezprostřední blízkosti jedné osoby, typicky do vzdálenosti 10 m. Nejčastějšími protokoly PAN jsou Bluetooth (IEEE 802.15) a též WiFi (IEEE 802.11).



**Obrázek 3.1:** V této kapitole se budeme zabývat: sériovým rozhraním, modemy, LAN a WLAN

## Sériová rozhraní

Dříve mělo každé PC dva sériové porty: COM1 a COM2. Dnes, zejména u notebooků, je port COM již vzácnost. Zejména sítovým administrátoremům však doporučujeme si kupovat notebook se sériovým portem, protože se stále občas hodí. A to nejenom pro připojení modemů nebo jako konzola sítových prvků, ale třeba i čteček čipových karet.



**Poznámka:** Sériová rozhraní slouží zpravidla pro komunikaci počítače/směrovače s modelem – viz obr. 1.5.

Sériové výstupy PC používají signály specifikované normou ITU V.24 (známou též pod označením EIA RS-232). Jedná se o rozhraní pro sériový asynchronní arytmický přenos dat. V praxi se běžně používá do 128 kb/s (obvyklá horní hranice pro COM-porty PC).

### Sériový a paralelní přenos dat

Sériový přenos znamená, že pro přenos informace od odesilatele k příjemci je jen jedna dvojice vodičů (resp. u asymetrických rozhraní jeden vodič a společná zem), takže jednotlivé bity každého znaku se přenáší postupně za sebou – tj. sériově.

Paralelní přenos k přenosu používá osm vodičů (nebo násobek osmi), tzn. že všechny bity přenášeného znaku se mohou přenést najednou – tj. paralelně. Paralelní přenos se používá zejména uvnitř počítače na jeho sběrnicích, ale třeba i pro komunikaci s paralelní tiskárnou.

## Symetrický a asymmetrický signál

U sériových rozhraní bývají minimálně dva signály: příjem dat a vysílání dat. Pokud je každý signál realizován dvěma vodiči (4 dráty), pak hovoříme o symetrickém nebo diferenciálním signálu. Symetrické signály pro přenos dat používá např. rozhraní X.21.

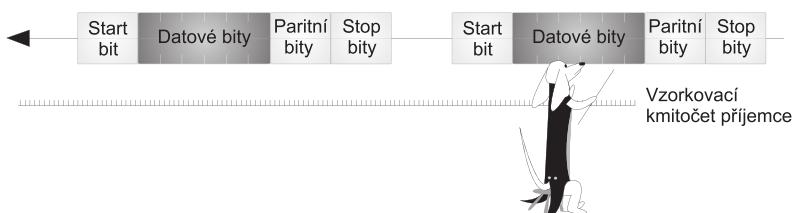
Pokud je každý signál realizován jedním vodičem proti společné zemi, pak hovoříme o asymmetrickém signálu. Asymetrické signály používá např. rozhraní V.24.

## Synchronní nebo asynchronní přenos

Chcete-li se s někým (např. telefonem) o něčem domluvit, musíte mluvit tak rychle, aby on byl schozen vám rozumět. Např. budete-li mluvit desetkrát rychleji, pak vám stěží porozumí. Ten, kdo poslouchá, se musí synchronizovat s tím, kdo mluví.

Z hlediska synchronizace rozeznáváme přenos:

- ◆ Synchronní, kdy se informace přenášejí po jednotlivých bitech. Okamžiky přechodu od přenosu jednoho přenášeného bitu k přenosu dalšího bitu jsou vždy stejně vzdáleny.
- ◆ Asynchronní, kdy okamžiky přechodu od přenosu jednoho bitu k přenosu dalšího bitu nejsou stejně vzdáleny. Zvláštním případem asynchronního přenosu dat je tzv. arytmický přenos. U arytmického přenosu je přenos znaků asynchronní, ale jednotlivé bity v přenášeném znaku se přenáší synchronně. Pokud se hovoří o asynchronním přenosu, pak se má většinou na mysli asynchronní arytmický přenos.



**Obrázek 3.2:** Asynchronní arytmický přenos znaků

Při asynchronním arytmickém přenosu je odesílaný znak obalen obálkou tvořenou startovacím bitem, paritními bity a stop-bity (viz obr. 3.2).

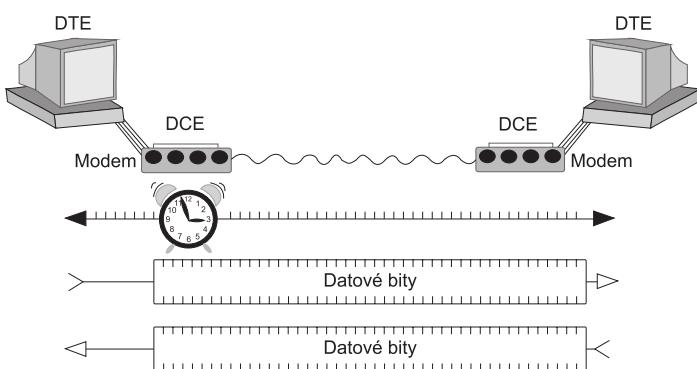
Přijímač generuje tzv. vzorkovací kmitočet, tj. kmitočet o řád vyšší frekvence, než je maximální možná frekvence přenosu jednoho bitu. Přijímač touto frekvencí měří vzorky přijímaného signálu. Pokud vzorek odpovídá s jistou pravděpodobností startovacímu bitu, předpokládá, že narazil na začátek přenášeného znaku. Pokračuje ve vzorkování až do stop-bitů. Mezi start bitem a stop-bity považuje všechno za bity přenášeného znaku. Navíc tam může být ještě paritní bit zabezpečující jednoduchý kontrolní součet přenášeného znaku.

Asynchronní přenos má výhodu v tom, že se přijímač může se značnou tolerancí přizpůsobit frekvenci vysílače. Avšak obálka přenášených dat se zpravidla skládá z jednoho start-bitu, jednoho paritního bitu a jednoho, jeden a půl či dvou stop-bitů. Což znamená, že obálka svou režii může způsobit i 50% nárůst požadavků na přenos (přenášený znak má nejčastěji 5 až 8 bitů). Pokud vás překvapil

termín jeden a půl bit, tak to je slang. Bitem se zde totiž nerozumí jednotka informace, ale doba, po kterou se přenáší jeden bit.

U synchronního přenosu se režie snižuje. V minulosti se používal blokové synchronní přenos dat (BSC), kdy se data přenášela v blocích. Na začátku bloku byl jeden nebo více synchronizačních znaků, které byly obdobou start-bitu. Přijímač se synchronizoval na těchto synchronizačních znacích (tj. přijímač si přizpůsobil své hodiny). Blok se pak přenáší synchronně.

Dnes je však běžnější zcela jiný princip. Kromě přenášených dat se vedením přenáší ještě synchronizační signál (hodiny). Na obr. 3.3 se na komunikaci podílí čtyři zařízení (dva modemy a dva počítače).



**Obrázek 3.3:** Synchronní přenos

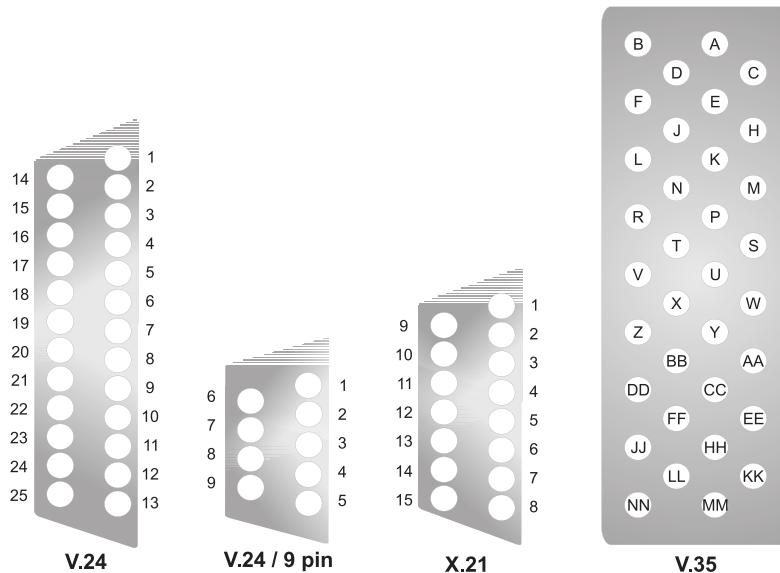
Podobně jako v orchestru může být jen jeden dirigent, tak zdrojem hodin může být jen jedno z těchto čtyř zařízení. Zpravidla to bývá jeden z modemů (*originator*). Ostatní zařízení přizpůsobí takt svých obvodů tomuto dirigentovi (synchronizují se). Jelikož všechna čtyři zařízení jsou synchronizována, mohou mezi sebou přímo komunikovat (bez vzorkování).

Pokud by byl zdrojem hodin jeden z počítačů, nastavíme v komunikaci mezi modemy jako *originator* modem na straně počítače generujícího hodiny. Tento modem však nastavíme do režimu, že používá externí hodiny (z počítače).

### Normy V.24, V.35 a X.21

Na fyzické úrovni se pro sériová rozhraní často používají normy V.35, X.21 a u PC v minulosti velice oblíbená norma V.24. Pochopitelně existují i další normy. Nejčastěji se s těmito normami uživatel setká na rozhraní modemu a počítače (resp. směrovače).

Rozhraní V.24 je u PC oblíbené proto, že mnoho PC je vybaveno porty COM1 a COM2, konstruovanými podle normy V.24. Rozhraní V.24 se zpravidla nedoporučuje pro přenosové rychlosti nad 128 kb/s, proto pro vyšší rychlosti je nutné použít rozhraní V.35 nebo X.21. Každá z těchto tří uvedených norm používá jiný typ konektoru, takže propojovací kabely lze těžko zaměnit.



**Obrázek 3.4:** Konektory používané pro rozhraní V.24, X.21 a V.35

U rozhraní V.35 a X.21 jsou pro přenos dat určeny vždy páry vodičů a hodnota signálu je určována mezi vodiči v daném páru (symetricky). Pouze pro signalizaci řízení toku dat se používají signály se společnou zemí (tj. asymetricky).

Pomocí rozhraní V.24, V.35 či X.21 je sice možné mezi sebou propojit dva počítače, ale pouze na vzdálenost několika metrů. Na větší vzdálenosti je třeba použít modemy. Přímé propojení dvou počítačů (propojení DTE-DTE) se provádí tzv. nulovým modelem – viz odstavec „Nulový modem“.

V tabulce 3.1 uvádíme význam jednotlivých signálů rozhraní V.24, X.21 a V.35. Pro jednoduchost jsme upravili terminologii na situaci komunikace počítače s modelem, tj. popisujeme tzv. moderový kabel propojující počítač s modelem.

První otázkou je, proč je nutné, aby v každém konektoru bylo zapojeno tolik vývodů – tolik signálů. Nikdo asi nebude zpochybňovat vývody pro přenos dat. V těchto normách máme zpravidla několik skupin vývodů:

- ◆ **Zem.** Přitom „země“ máme dvojí. Jedenak „Signálová zem“, kterou je společný vodič pro asymetrické signály. Pro jednotlivé signály pak „stačí“ jeden vývod. Asymetrický signál se pak měří vůči tomuto vývodu „Signálová zem“. Naopak vývod „Ochranná zem“ slouží pro propojení koster (tj. elektricky vodivých součástí) propojovaných zařízení.
- ◆ **Řízení,** které slouží k řízení toku přenášených dat. Řízení toku přenášených dat je nutné proto, že přenosová rychlosť mezi počítačem (DTE) a modelem (DCE) a přenosová rychlosť od modelu dále jsou obecně různé. Případný přebytek neodvysílaných dat na jednu nebo druhou stranu se řeší ukládáním těchto dat do vyrovnávací paměti. Když se však začne paměť povážlivě zaplňovat, je třeba druhé straně signalizovat, aby s dalšími daty posečkala. Dalšími řídicími signály jsou např. signály pro sdělování druhé straně, že zařízení je vůbec zapnuto, a tudíž má

smysl se nějakou komunikací zabývat. Signál „Zvonek“ signalizuje příchozí hovor (na vytáčecích linkách). A signál „Nosná“ zase signalizuje, že protější modem komunikuje nebo se alespoň o komunikaci snaží.

- ◆ **Hodiny**, které slouží pro přenos synchronizačního signálu (hodin) v případě synchronního přenosu.
- ◆ **Test** sloužící pro testování okruhu.

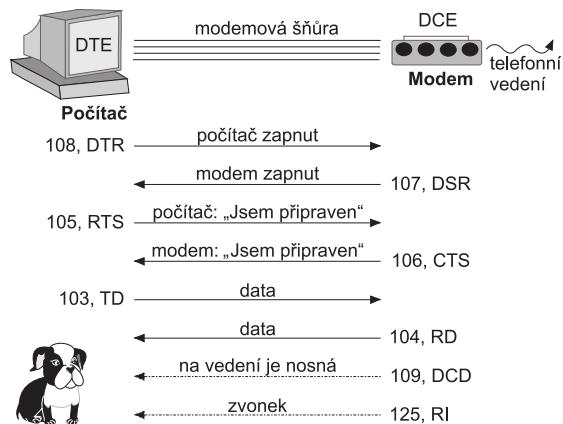
**Tabulka 3.1:** Význam jednotlivých signálů. Sloupec *od* vyjadřuje, kdo je zdrojem signálu: buď počítač (p), nebo modem (m). Sloupec *Typ* vyjadřuje, zda se jedná o symetrický signál (mezi A a B), nebo asymetrický (mezi A a společnou signálovou zemí).

	<b>Popis signálů</b>	<b>od</b>	<b>Označení signálů</b>		<b>V.24</b>		<b>X.21</b>		<b>V.35</b>		
			EIA	ITU	<b>25 Pin</b>	<b>9 Pin</b>	<b>A</b>	<b>B</b>	<b>15 Pin</b>	<b>Typ</b>	<b>34 Pin</b>
<b>Zem</b>	Ochranná zem Signálová zem		FG	101	1		1				A B
			SG	102	7	5	8				
<b>Data</b>	Vysílání Příjem	p m	TD	103	2	3	2	9	sym.	P R	S T
			RD	104	3	2	4	11	sym.		
<b>Řízení</b>	Výzva k vysílání „Počítač: jsem připraven komunikovat“ Pohotovost k vysílání „Modem: jsem připraven“ Modem zapnut Počítač zapnut Nosná Zvonek	p m m p m m	RTS	105	4	7	3	10	asym.	C	
			CTS	106	5	8			asym.	D	
			DSR	107	6	6			asym.	E	
			DTR	108/2	20	4			asym.	H	
			DCD	109	8	1	5	12	asym.	F	
			RI	125	22	9			asym.	J	
<b>Hodiny</b>	Generované počítačem Vysílané (z modemu) Přijímané (z modemu)	p m m	TTC	113	24				sym.	U Y	W AA
			TC	114	15				sym.	V	X
			RC	115	17		6	13	sym.		
<b>Test</b>	Vzdálená digitální smyčka (V.54/2) Lokální analogová smyčka (V.54/3) Testovací mód	p p m	RLB	140	21						
			LLB	141	18						
			TM	142	25						

Dialog řízení toku přenášených dat mezi počítačem (DTE) a modemem (DCE) je schematicky vyjádřen na obr. 3.5. Signály DTR a DSR signalizují svému protějšku, že zařízení je zapnuto. V praxi se tyto signály někdy nepoužívají (vývody se nezapojují nebo naopak přímo v konektoru jsou vývody DTR a DSR propojeny). V případě, že signály DTR a DSR nejsou zapojeny, je třeba, aby oba konec spojení byly na tuto situaci připraveny (konfigurovány), aby nekonečně nečekaly na signál druhé strany.

Význam signálů RTS a CTS spočívá v řízení toku dat. V případě, že modem má svou vyrovnávací paměť plnou, shodí signál CTS, čímž signalizuje svému protějšku, aby pozastavil odesílání dat. Po uvolnění vyrovnávací paměti modem opět obnoví signál CTS a počítač jej může opět zásobovat daty. V opačném směru, pokud počítač momentálně není schopen přijímaná data zpracovávat, shodí signál RTS.

Je možná i eventualita, že se signály RTS a CTS nepoužijí vůbec (např. nejsou zapojeny příslušné vývody ve šňůře). Pak je třeba konfigurovat oba konce na tuto situaci. V tomto případě je možné k řízení toku dat využít datové signály (vždy v opačném směru). Je-li např. třeba pozastavit příjem, vloží se do vysílání znak XOFF. Obnovení se provede znakem XON. Tento protokol XON/XOFF je také třeba na obou koncích spojení shodně konfigurovat a navíc ho lze využít pouze u asynchronních spojů, kdy se přenáší znaky.



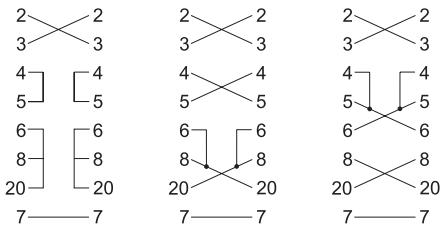
**Obrázek 3.5:** Schematické znázornění dialogu mezi počítačem a modemem

Signály data (TD i RD) mohou na počátku komunikace přenášet pouze data mezi počítačem a modelem – např. AT-příkazy pro vytáčení. Teprve později, po navázání spojení mezi modemy, mohou být signály TD a RD využívány i pro přenos dat mezi počítači.

Řízení toku dat pomocí signálů RTS a CTS je efektivní zejména u rozhraní V.24. Rozhraní V.35 a X.21 jsou spíše určena pro vyšší rychlosti a můžeme jim být připojeni např. k poskytovateli sítě FrameRelay. V takovém případě fyzicky jedním rozhraním (*interface*) prochází několik logických rozhraní (*subinterface*), v případě FrameRelay každé logické rozhraní odpovídá jednomu virtuálnímu okruhu. V případě zahlcení jednoho virtuálního okruhu nelze pozastavit tok dat pomocí signálů RTS či CTS, protože takové pozastavení by znamenalo pozastavení všech logických rozhraní (*subinterface*) bez ohledu na to, že zahlceno může být jen jedno.

## Nulový modem

Pokud byste chtěli pomocí rozhraní V.24, X.21 či V.35 přímo propojit dva počítače stojící vedle sebe (tj. propojit DTE s DTE), musí být propojovací kabel speciálně zapojený. Problém je v tom, že signály, které jedna strana na příslušném vývodu vysílá, musí přijímací strana přijímat na vývodu určeném pro příjem („vysílání musí být překříženo s příjemem“). Takováto propojovací šňůra se nazývá nulový modem (obr. 3.6).



**Obrázek 3.6:** Některé varianty zapojení nulového modemu pro rozhraní V.24/25

V případě synchronního přenosu musí jedna strana poskytovat hodiny. Neumí-li ani jedna ze stran generovat hodiny, pak se nulový modem nemůže skládat pouze ze šňůry, ale musí mezi oběma počítači být ještě krabička, která vytváří synchronizační takt (hodiny).

## Modemy

Pro komunikaci do Internetu přes telefonního operátora máme principiálně tři možnosti:

- ◆ Namísto hlasu přenášet data, tj. digitální signál modulovat do hlasového pásmo. Pro přenos dat pak můžeme využít stávajících komunikačních sítí (včetně GSM). Tuto možnost využívají modemy. Tato možnost nám nabízí velice omezenou přenosovou rychlosť. U sítí GSM kolem 10 kb/s, u analogových linek do 56 kb/s a u ISDN 128 kb/s na každý kanál B.
- ◆ Ponechat přenos hlasu na telefonním okruhu beze změn a využít přenosové pásmo, které se pro přenos hlasu nevyužívá. Tuto možnost využívá ADSL nebo GPRS.
- ◆ Pronajmout si připojení výhradně k přenosu dat. Přitom se nemusí jednat jen o klasickou místní telefonní smyčku, ale můžeme si pronajmout i např. optická vlákna. Na těchto pronajatých linkách lze pak dosahovat nejvyšších rychlostí. Tuto možnost využívají zejména firmy pro své připojení k Internetu nebo i samotní poskytovatelé Internetu.

Pokud chceme využít k připojení do Internetu místní telefonní smyčku, pak máme v zásadě dvě možnosti:

- ◆ analogová linka,
- ◆ linka ISDN („Digitální linka“).

Obě dvě tyto možnosti lze kombinovat s ADSL.

## Analogová linka

Analogová linka je klasickým telefonním připojením koncového uživatele, které se používá od telefonní nepaměti. Je realizována tzv. místní telefonní smyčkou, tj. telefonní kroucenou dvojlinkou od místní telefonní ústředny ke koncovému uživateli. Hlasová frekvence se v analogovém telefonu převádí na stejnou frekvenci elektrického napětí. Přitom telefonní přenos garantuje jen přenos omezeného pásmá (viz ADSL). Odtud pramení termín telefonní kvalita přenosu.

Analogová linka je u koncového účastníka dnes již zpravidla zakončována konektorem RJ-11 (nebo širším konektorem, do kterého lze konektor RJ-11 vložit). Hovoříme o rozhraní a/b.

V minulosti se též pronajímalý pevné analogové linky. Tato praxe byla nahrazena pevnými digitálními okruhy.



**Tip:** Překvapením je, že stávající místní telefonní smyčky je možné využít jak pro analogové linky, tak pro digitální linky (ISDN) a obojí je ještě možné kombinovat s ADSL.

Doposud jsme předpokládali, že analogová linka je využívána pro telefonování, tj. pro hlasovou komunikaci. Chceme-li použít telefonní vedení pro počítačovou komunikaci, musí se datové informace na telefonní vedení modulovat a na druhé straně demodulovat. Komunikace je ale obousměrná, takže na obou koncích je potřeba **modulátor/demodulátor**, tj. **modem**.

Modem je zařízení, které se bude připojovat k počítači či směrovači modemovým kabelem (tj. v případě PC rozhraním V.24 na COM-port PC). Na druhý vývod modemu se připojuje telefonní linka.

Modem může být do PC i vestavěn nebo realizován na kartě PCMCIA vkládané do notebooku. V takovém případě není třeba modemový kabel. Jiným typem propojení počítače a modemu je skrze sběrnici USB. I zde pak dochází k emulaci klasických COM-portů PC.

## Modem je automatický

Na komutované (vytáčené) lince je problém: kdo vytvoří telefonní číslo. Kdysi se i na komutovaných linkách používaly „neautomatické“ modemy. Znamenalo to, že uživatel musel nejprve pomocí telefonního aparátu vytvořit číslo a pak ručně přepnout modem do režimu přenosu dat (přepínač VOICE/DATA).

Tzv. „automatické“ modemy umí po zapnutí přijímat příkazy od počítače, kterými se mj. vytvoří číslo. Po navázání spojení se takové modemy samy vzájemně dohodnou na nejvyšší možné přenosové rychlosti a automaticky se přepnou do datového režimu.



**Poznámka:** Znalost modemu (včetně AT-příkazů pro řízení modemu) uplatníme nejenom na telefonních linkách, ale i v případě GSM-sítí. Nejenom mobilní telefony, ale i nejrůznější 4G-karty totiž emulují klasický modem.

## Příkazy AT

Pro komunikaci, kterou počítač ovládá modem, se dnes používají tzv. **příkazy AT**, zavedené před lety firmou Hayes. Příkazy AT jsou určeny pro ovládání modemu na asynchronních rozhraních. Každý příkaz AT je tvořen znaky kódovanými v ASCII. Často se příkazy AT používají i pro nastavení modemu k synchronnímu přenosu v případě, že je použito rozhraní V.24. Postupuje se tak, že se pomocí tlačítka na modemu přepne modem na tovární nastavení. V továrně přitom bývá nastaven asynchronní přenos. Pak se modem připojí na port COM, který je asynchronní. Z PC se např. pomocí aplikace Hyperterminal vyšlou do modemu příkazy AT, kterými se modem nastaví. Posledním příkazem pak je přepnutí modemu na synchronní režim. Modem jakoby zmrzne (protože již chce komunikovat synchronně), ale PC komunikuje asynchronně. Nastavený modem se pak přenese např. ke směrovači, kde bude sloužit pro synchronní přenos.

Pomocí příkazů AT lze z počítače ovládat modem. Příkazy AT jsou jednoduché povely. Např. příkaz ATH znamená, že počítač do modemu odešle (resp. odešle na COM-port) znakový řetězec ATH. Modem pak řetězec ATH interpretuje jako příkaz.

Počítač nejprve komunikuje pomocí AT-příkazů s místním modelem, po navázání spojení mezi modemy místní modem signalizuje AT-příkazem CONNECT navázání spojení a přepne se do datového režimu. Od tohoto okamžiku mohou počítače přímo komunikovat mezi sebou, tj. z hlediska počítačů nastane situace, jako by na lince žádné modemy nebyly (jako by počítače byly propojeny nulovým modelem). Pokud chce počítač přepnout modem opět do příkazového režimu, aby mohl odesílat příkazy AT, odešle jako data řetězec „+++“.

*Pokud pracujete např. na Windows XP, spusťte si aplikaci Hyperterminal. Vytvořte připojení libovolného jména na libovolné telefonní číslo. V menu Soubor -> Vlastnosti (File -> Properties) zvolte připojený zapnutý modem a zmáčkněte tlačítko Konfigurovat. Ve volbě Upřesnit zvolte „Okno terminálu zobrazit před vytvořením čísla“. Volby potvrďte a nakonec zvolte Vytočit. Objeví se okno „Obrazovka terminálu před vytvořením“. V tomto okně pak můžete procvičovat příkazy AT popisované v následujících odstavcích (např. zadejte AT a stiskněte Enter).*



**Poznámka:** Pokud pracujete ve Windows Vista, pak máte smůlu, protože jejich součástí už program HyperTerminal není. Stačí si ale nainstalovat např. program putty, který se často používá jako klient protokolu SSH. Ten ve verzi 0.60 má i volbu serial, kterou můžeme specifikovat příslušný COM-port a zadávat AT-příkazy.

PC nejprve pošle do modemu dotaz „AT“ („modeme, jsi schopen pracovat?“). Je-li modem připraven, pak odpoví „OK“ (vyzkoušeli jste si napsat do okna „Obrazovka terminálu před vytvořením“ znaky AT?).

Nyní může PC poslat modemu příkaz k vytvoření čísla ATDtč (např. ATDP1234560), kde t je typ vytáčení (P=pulsně, T=tónově) a č je telefonní číslo protějšího účastníka. Protější modem zvedne a oba modemy se dohodnou na nejvyšší možné přenosové rychlosti. Modem na straně PC vrátí PC příkaz CONNECT, kde jako parametr může udat dohodnutou rychlosť mezi modemy. Poté se oba modemy přepnou do datového režimu, tj. oba počítače mohou začít mezi sebou přenášet data jako by tam žádné modemy nebyly. Celý mechanismus je znázorněn v tabulce 3.2, kde je abstrahováno od chybových stavů i od příkazů AT pro nastavení modemu, který spojení očekává.

**Tabulka 3.2:** Dialog sestavení datového okruhu mezi dvěma počítači

	Místní počítač	Místní modem (vytáčející)	Telefonní okruh	Vzdálený modem (očekávající)	Vzdálený počítač
<b>Signalizace</b>	105, RTS → ← 106, CTS		nečinný		← 105, RTS → 106, CTS
<b>AT-příkazy</b>	AT → ← OK ATDtč →				
<b>vytáčí</b>			vytváření okruhu (dohoda na nejvyšší možné rychlosti)	<b>zvedá</b>	
Signalizace	← 109, DCD		vytvorený	109, DCD →	
AT-příkazy	← CONNECT				
Přenos dat			↔		

## Synchronní přenos

Již jsme se zmínili o tom, že přenos může být **synchronní** nebo **asynchronní**. Synchronní modemy slouží pro synchronní přenos, asynchronní pro asynchronní přenos. Dnešní modemy umí zpravidla oba druhy přenosu.

Poznamenejme, že PC podporuje standardně pouze asynchronní přenos, proto modem, který je nastaven jako synchronní, je před použitím na PC třeba nastavit do asynchronního režimu – jinak se jeví jako pokažený. Jiná situace je u modemů vkládaných do počítače jako karty PCMCIA nebo přídavné karty. Ty nevyužívají standardní porty COM, proto se v takovémto případě může teoreticky použít i synchronní přenos.

Při konfiguraci synchronního modemu nesmíme zapomenout právě jeden modem nastavit jako zdroj hodin (*Originator*). V případě, že zdrojem hodin je počítač, nastavíme jako *originate* modem na straně počítače generujícího hodiny. Navíc musíme tomuto modemu nakonfigurovat, že využívá externí hodiny.

Dnešní modemy pro rychlosti pod 64 kb/s umí většinou jak asynchronní, tak synchronní režim. Modemy pro rychlosť 64 kb/s a výše bývají zpravidla synchronní.

Existují i modemy podporující tzv. autosynchronní režim, tj. s počítačem komunikují asynchronně, data ukládají do paměti a poté je synchronně modulují do telefonního vedení. S těmito modemy se v Internetu setkáváme jen zřídka, jejich hlavním těžištěm jsou veřejné datové sítě na bázi protokolu X.25, resp. X.32.

## ISDN

Doposud jsme popisovali analogové okruhy. Život však jde dále a analogové rozvody jsou nahrazovány digitálními. Nejprve se tak dělo uvnitř telekomunikačních operátorů. Dnes však i uživatelé mohou používat digitální okruhy – ISDN.

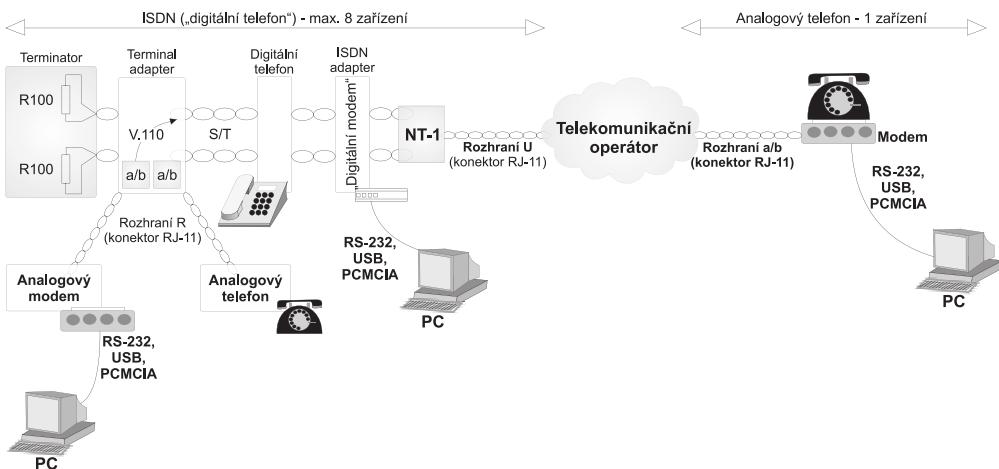
V Evropě je zpravidla nabízeno buď připojení euroISDN2, nebo připojení euroISDN30. To jsou spíše obchodní označení, v literatuře se spíše setkáme s anglickými názvy:

- ◆ BRI (*Basic Rate*) pro euroISDN2, což je typ připojení, kdy ve fyzicky jednom vedení (jedné kroucené dvojlince místní telefonní smyčky) jsou dva datové kanály (označované jako **kanály B**), každý o kapacitě 64 kb/s, a jeden signalační kanál D o kapacitě 16 kb/s. BRI se zpravidla používá pro připojení koncových uživatelů.
- ◆ PRI (*Primary Rate*) pro euroISDN30, což je typ připojení, kdy ve fyzicky jednom vedení je třicet datových kanálů B, každý o kapacitě 64 kb/s, a jeden signalační **kanál D** o kapacitě 64 kb/s. PRI se např. používá pro připojení podnikové telefonní ústředny do telekomunikační sítě.

Kanál D v ISDN slouží k signalačaci, tj. např. k sestavení virtuálního okruhu („vytočení čísla“) či v případě, kdy jsou oba kanály B obsazeny telefonními hovory, může být kanálem D signalačován další přicházející hovor. Současný hovor můžeme pozdržet a hovořit na nově příchozím hovoru, je možné signalačovat číslo volajícího účastníka atd.

## Basic Rate

BRI využívá stávající telefonní rozvody kroucenou dvoulinkou, tj. většinou lze využít pro rozvod euroISDN2 i stávající metalické rozvody pro analogové telefony. Kroucená dvoulinka přicházející od telefonního operátora vytváří tzv. **rozhraní U** (viz obr. 3.7).



**Obrázek 3.7:** ISDN (Basic Rate) versus analogové připojení místního účastníka telefonní sítě

Rozhraní U je rozhraním mezi telekomunikačním operátorem a zařízením (krabičkou) NT-1, které zpravidla dodává i instaluje telekomunikační operátor. Ze zařízení NT-1 vychází dvě dvoulinky **rozhraní S/T**. Rozhraní S/T má charakter sběrnice, na kterou se připojují jednotlivá digitální zařízení. Pokud např. kupujeme směrovač pro připojení na ISDN, pak jej kupujeme s rozhraním S/T – nikoliv s rozhraním U, protože rozhraní U je u nás v režii telekomunikačního operátora.

Jak je znázorněno na obr. 3.7, jednotlivá zařízení se na rozhraní S/T připojují jako na sběrnici. Jelikož *Basic rate* má k dispozici dva datové kanály B, mohou v jednom okamžiku komunikovat současně dvě zařízení (např. digitální telefon a digitální modem nebo dva digitální telefony atd.).

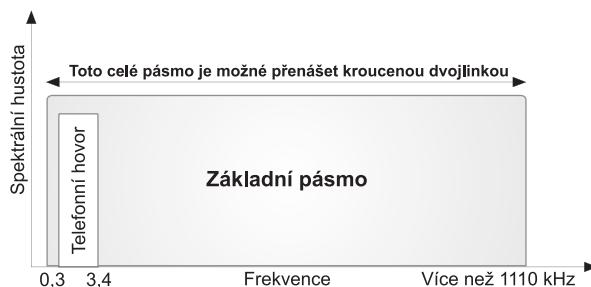
I když máme k dispozici pouze dva datové kanály B, na rozhraní S/T můžeme umístit více než dvě digitální zařízení (v jednom okamžiku však mohou komunikovat nejvýše dvě), např. pět digitálních telefonů. Každý z nich může mít své samostatné číslo. Z hlediska uživatele se to jeví, jako by měl pět „telefonních linek“, ale současně může používat pouze dvě.

Na rozhraní S/T může být také připojen tzv. terminál adaptér, který umožnuje připojení klasických analogových modemů, faxů a telefonů. Tato komunikace se pak zabalí do protokolu V.110 a přenese k protějšímu analogovému zařízení.

Ještě se musíme zmínit o termínu „digitální modem“. Mnohým je toto označení trnem v oku, protože modem je zařízení, které moduluje/demoduluje digitální signál na analogový. Digitální modem přitom nic takového nedělá, pouze konvertuje jeden typ digitálního rozhraní (např. V.24 v případě PC) na rozhraní S/T (konektor RJ45). Proto je pro takovéto zařízení vhodnějším označením adaptér ISDN než „digitální modem“.

## Základní pásmo a telefonní (hlasové) pásmo

Kroucená telefonní dvojlinka, která se používá mezi koncovým účastníkem telefonní sítě a nejbližší telefonní ústřednou (tzv. telefonní smyčka), umožňuje přenos dat v poměrně širokém pásmu 0,3 kHz až několik MHz v závislosti na kvalitě vedení. Pro přenos zvuku v telefonní kvalitě však stačí přenášet pásmo 0,3 až 3,4 kHz (viz obr. 3.8.), a tak zbytek pásma zůstává nevyužit



**Obrázek 3.8:** Pásmo pro telefonní hovor je jen zlomkem základního přenosového pásma telefonní kroucené dvojlinky

Pokud tedy využíváme modem na vytáčené lince, musí se veškerá datová komunikace vejít do pásm pro telefonní hovor. Proto se nesmíme divit, že přes vytáčené linky se dosahuje maximální přenosové rychlosti do 56 kB/s.

## ADSL

ADSL (*Asymmetric Digital Subscriber Line*) přišlo s tím, že po místní telefonní smyčce poběží současně stávající telefonní komunikace (analogová nebo ISDN) a datová komunikace. Myšlenka spočívá v tom, že obě komunikace poběží po téže kroucené dvojlince, ale na obou stranách bude komunikace oddělena:

- ◆ na telefonní hovor (analogový nebo ISDN) v pásmu do 4 kHz,
- ◆ na přenos dat ve zbytku základního pásma.

Výsledkem je, že stávající kroucenou dvojlinku je možné využít nejenom pro telefonní hovor, ale pro přenos dat. A to i současně, protože každá komunikace využívá jinou část přenosového pásma (obr. 3.9).

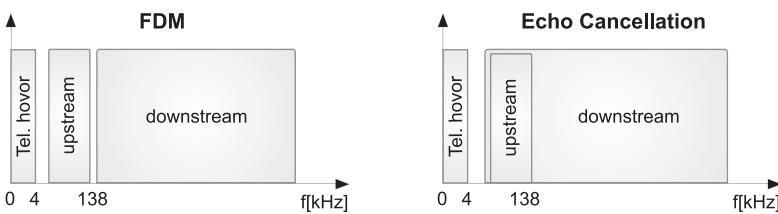
Ve zkratce ADSL je důležité slovo „*Asymmetric*“, které vyjadřuje, že pro každý směr přenosu se využívá jiná přenosová rychlosť:

- ◆ Ve směru od koncového uživatele k telefonní ústředně („*upstream*“) se používá vždy menší přenosová rychlosť.
- ◆ V opačném směru („*downstream*“) je pak rychlosť vyšší.

Takové řešení je výhodné pro domácí uživatele Internetu, kteří více dat z Internetu stahují, než na něj posílají. Pro připojení firem to však může být problematické, např. firemní webový server za takovou linkou je problém.

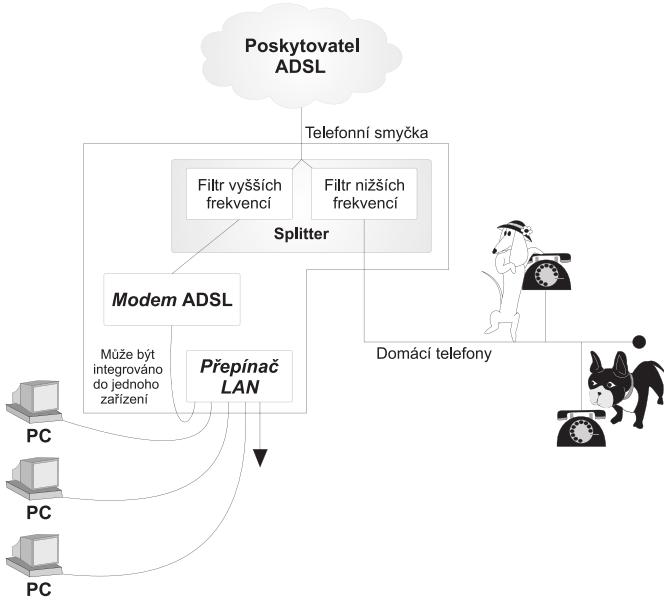
Na obr. 3.9 jsou znázorněny dvě metody rozdělení přenosového pásma:

- ◆ **FDM (Frequency Division Multiplexing):** Tato metoda odděluje pásmo pro upstream a downstream, což je technicky jednodušší řešení.
- ◆ **EC (Echo cancellation):** ADSL základní pásmo vždy rozděluje na jednotlivé přenosové kanály. Kanály na nízkých frekvencích nevyužívá – ty jsou určeny pro telefonní hovor. Na frekvencích pro ADSL pak modemy ADSL na počátku komunikace otestují linku a pro další komunikaci využijí jen ty kanály, na kterých jsou schopny kvalitně komunikovat. Spodní kanály pak využijí pro upstream a horní pro downstream. Nevadí tedy ani fakt, kdyby linka neumožňovala přenos v pásmu do 138 kHz, což by v případě FDM přenosu zásadně vadilo.



Obrázek 3.9: ADSL používá buď metodu FDM, nebo metodu EC

Už jsme se zmínili o tom, že v případě ADSL se na obou koncích telefonní smyčky musí oddělit telefonní komunikace od datové. Nás bude nejvíce zajímat, jak se to dělá na straně koncového zákazníka. Zpravidla se do stávající telefonní zásuvky zapojí malá krabička – **splitter**. Ta rozbočí nižší frekvence do domácího telefonního rozvodu a vysílá do modemu ADSL.



Obrázek 3.10: Splitter

Technicky může být Splitter, ADSL-modem a přepínač LAN integrován do jednoho domácího zařízení (obr. 3.10).

Na straně telekomunikačního operátora je pak ADSL přivedeno do zařízení DSLAM (*Digital Subscriber Line Access Multiplexer*). Maximální přenosová rychlosť přitom prudce klesá se vzdáleností koncového uživatele od DSLAM. Maximální vzdálenost koncového uživatele by tak neměla přesáhnout cca 7 km.

DSLAM-data balí data koncového uživatele zpravidla do protokolu ATM, kterým se data přenesou na přístupový server poskytovatele Internetu. Přístupový server pak IP datagramy vybalí z buněk/paketů ATM a vypustí do Internetu. Toto řešení umožňuje, aby v jedné telekomunikační síti mělo více poskytovatelů Internetu své přístupové servery. V poslední době se i zde začíná upouštět od protokolu ATM a využívá se protokol IP – hovoříme pak od IP DSLAM.

## Přenosové rychlosti modemů

Modem posílá/přijímá data na dvě strany: jednak směrem k počítači/směrovači a jednak směrem do telefonního vedení. Pro komunikaci mezi počítačem/směrovačem (DTE) a modemem (DCE) zpravidla nastavujeme nejvyšší možnou přenosovou rychlosť na použitých sériových rozhraních (ta např. v případě klasických COM-portů PC bývá 128 kb/s – často však jen 115 kb/s). Přenosová rychlosť na telefonním vedení pak závisí zejména na použitém standardu ITU:

	<b>Protokol</b>	<b>Standard ITU</b>	<b>Přenosová kapacita</b>		
			<b>Symetrická</b>	<b>Asymetrická</b>	
<b>Přenos v hlasovém pásmu (např. vytáčené linky)</b>	V.32	V.32	9,6 kb/s		
	V.32bis	V.32bis	14,4 kb/s		
	V.34	V.34	28,8 kb/s		
	V.34	V.34	33,6 kb/s		
	V.90	V.90		Až 33,6 kb/s	Až 56 kb/s
<b>ISDN</b>	Kanál B	I.430 (BRI)	128 kb/s		
		I.431 (PRI)	128 kb/s		
<b>xDSL</b>	HDSL ( <i>High data rate Digital Subscriber Line</i> )	G.991.1	1,544 Mb/s (T1) nebo 2 Mb/s (E1)		
	ADSL ( <i>Asymmetric Digital Subscriber Line</i> )	G.992.1		Až 1,3 Mb/s	Až 12 Mb/s
		G.992.2 (G.Lite)		Až 0,5 Mb/s	Až 4 Mb/s
	ADSL2	G.992.3		Až 3,5 Mb/s	Až 12 Mb/s
	ADSL2+	G.992.5		Až 1 Mb/s	Až 24 Mb/s

Modemy pro přenos v hlasovém pásmu se často připojují přímo k PC (např. na jeho sběrnici USB nebo port COM), kdežto modemy ADSL se s PC zpravidla propojují přes LAN, tj. některým z protokolů Ethernet.

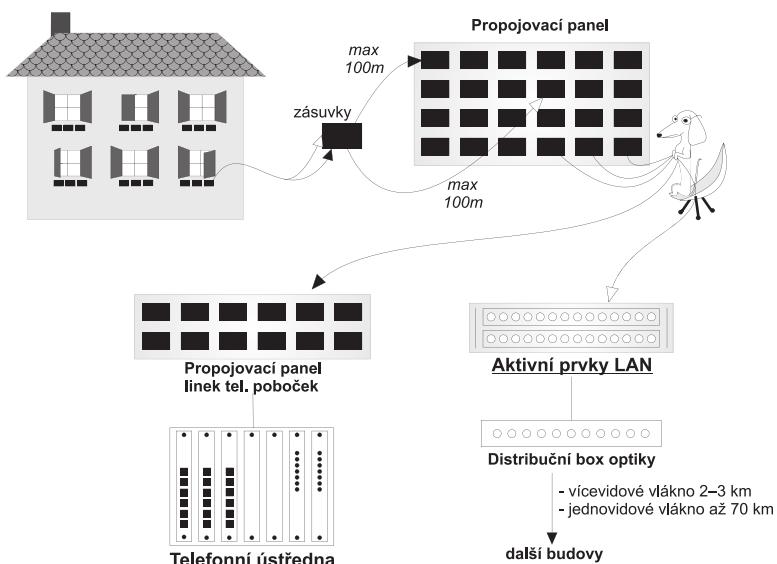


**Poznámka:** Všimněte si, že termín „modem ADSL“ jednou vyjadřuje jen modem ADSL a jindy celé integrované zařízení obsahující modem ADSL i směrovač.

## Strukturovaná kabeláž

Strukturovanou kabeláží se rozumí komplexní řešení nízkonapěťových rozvodů v budově. Zahrnuje zejména telefonní rozvody a rozvody pro LAN. Od společných rozvodů s požární a bezpečnostní signálizací se postupně upustilo, protože na takové rozvody jsou kladený odlišné bezpečnostní nároky.

V jednotlivých místnostech budovy jsou umístěny telefonní zásuvky, zásuvky LAN a jiné vývody. Z těchto zásuvek vedou rozvody na propojovací panel budovy. V případě optických rozvodů jsou optická vlákna vyvedena na distribuční box optiky.



Obrázek 3.11: Rozvody v budově

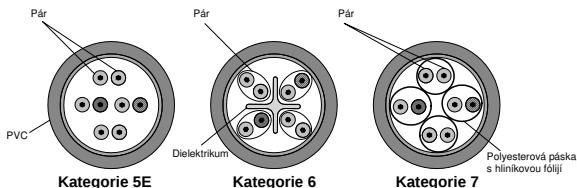
Propojovací panel a distribuční box optiky bývají uzavřeny v jedné skříni (*RackMount*) spolu s aktivními prvky LAN, či dokonce i s telefonní ústřednou. Propojení mezi propojovacím panelem a aktivními prvky se provádí propojovacími kably (*Patch Cord*).

Rozvod od zásuvek na propojovací panel je poměrně drahou záležitostí, protože se mnohdy jedná i o stavební úpravy. Snahou je proto rozvod provést maximálně kvalitně, aby se rozvody nemusely často předělávat. Základní filozofie nových síťových protokolů je pak v maximální míře využít stávající kabeláže. Proto také kvalitním rozvodům původně vytvořeným pro Ethernet nedělal problémy přechod na Fast Ethernet.

Pro rozvody strukturované kabeláže máme dva standardy:

- ◆ TIA/EIA-568-B, který je pokračovatelem standardu TIA/EIA-568-A. Tento standard zavádí tzv. kategorie rozvodů. Jednotlivé kategorie se liší zejména maximální frekvencí přenosu, pro kterou jsou určeny.
- ◆ ISO/IEC 11801, specifikující jak měděné rozvody, tak i optické rozvody. Pro měděné rozvody pomocí páru kroucené dvojlinky zavádí třídy A až F v závislosti na maximální přenosové frekvenci.

Frekvence	Využití	TIA/EIA-568-B	ISO/IEC 11801
Do 100 kHz			Třída A
Místní telefonní smyčka	Dnes se i pro telefonní rozvody v budovách využívají páry v kabelech pro rozvody LAN (tj. Cat 5e – 7)	Kategorie 1	
Do 1 MHz			Třída B
4 Mb/s	Sítě Token Ring (dnes nepoužíváno)	Kategorie 2	
Do 16 MHz	Historicky – používalo se pro 10BASE-T	Kategorie 3	Třída C
Do 100 MHz	10BASE-T, 100BASE-T i 1000BASE-T	Kategorie 5 (historická) a Kategorie 5e (aktuální)	Třída D
Do 250 MHz	10BASE-T, 100BASE-T, 1000BASE-T i 10GBASE-T	Kategorie 6	Třída E
Do 600 MHz	10BASE-T, 100BASE-T, 1000BASE-T, 10GBASE-T a v budoucnu 100GBASE-T	Někdy se hovorově označuje jako „Kategorie 7“	Třída F



Obrázek 3.12: Příklad řezu propojovacími kably

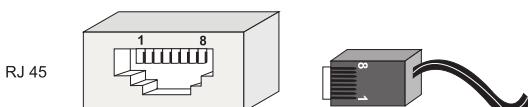
## Měděné rozvody

Měděné rozvody se provádí svazky kroucených dvojlinek. Klasickým kabelem je kabel obsahující 4 páry dvojlinek. Je pak na výrobcích, aby vyrobili kably příslušné kategorie. Na obr. 3.12 jsou znázorněny příklady kabelů jednotlivých kategorií.

Měděné rozvody se ukončují zpravidla konektorem **RJ-45** (RJ = registered jack), který je znázorněn na obr. 3.13. Konektor RJ-45 („kostka cukru“) obsahuje 8 vývodů pro 4 páry. Přitom se jedná vlastně o stavebnici. Na obr. 3.13 je uvedeno zapojení 8P8C (8 Position 8 Contact). Tak lze do téže zásuvky vložit např. i konektor RJ-11 (ten je 2P2C) a tak připojuje pár číslo 1 (obr. 3.14), který např. není využíván 10BASE-T ani 100BASE-T.



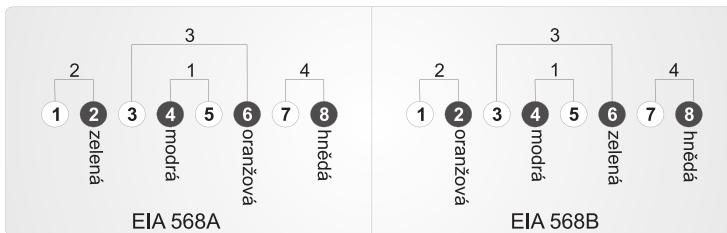
**Tip:** Lze tak do jedné zásuvky přivést jak Ethernet/Fast Ethernet, tak analogový telefon.



Obrázek 3.13: Konektor RJ-45

Norma **TIA/EIA-568-B** definuje zakončení kabelu buď T568A, nebo T568B (nezaměňovat s novým a starým standardem TIA/EIA-568-A a TIA/EIA-568-B!!). Základním zapojením je T568A, ale v praxi se dává z historických důvodů přednost T568B (viz obr. 3.14). Obě zapojení umožňují např. pár číslo 1 použít pro telefon (analogový) a páry 2 a 3 např. pro Ethernet (pár 4 zůstává v tomto případě volný). Pro Ethernet a Fast Ethernet existuje i jiná možnost využití volných páru. A to ta, že po dvou volných párech pustíme připojení dalšího počítače. Je to sice atypické zapojení, ale když potřebujeme

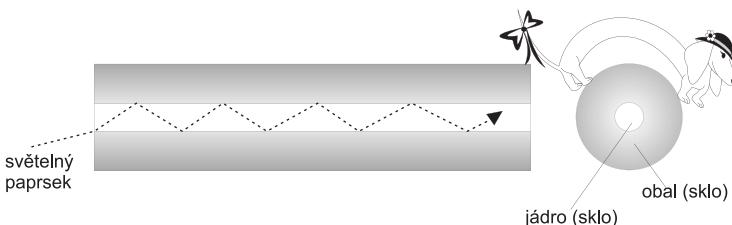
do místnosti přidat počítač a nemáme pro něj volnou zásuvku, tak jej rozhodně oceníme. Dokonce se prodávají „rozdvojký“, které nasadíme na oba konce rozvodu.



**Obrázek 3.14:** Zapojení jednotlivých párů v konektorech dle T568A a T568B

## Optická vlákna

Optická vlákna jsou tvořena dvěma vrstvami skla. Jeden typ skla je použit pro jádro vlákna a jiný typ skla pro obal vlákna. V jádře vlákna je veden optický paprsek, který se postupně odráží od rozhraní mezi dvěma druhy skla (viz obr. 3.15).



**Obrázek 3.15:** Optické vlákno

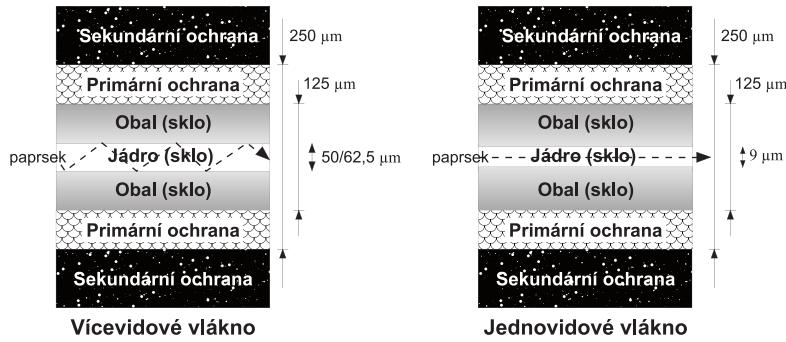
Sklo má nízký optický odpor pouze pro tři vlnové délky světla: 850 nm, 1 300 nm a 1 500 nm, proto se vždy k buzení optického signálu používá jedna z těchto vlnových délek.

Optické vlákno je vždy simplexní spoj, tj. na jedné straně je vysílač a na druhé straně přijímač. Pro duplexní spoje (což potřebujeme téměř vždy) je nutná dvojice vláken – pro každý směr jedno vlákno.

I když vlákno má zpravidla průměr 0,125 mm, jádro vlákna máme dvojího průměru:

- ◆ průměru 50 µm (resp. 62,5 µm) – pak hovoříme o vícevidovém vlákně (*multi mod*). Vícevidová vlákna se budí pomocí LED, avšak v poslední době se např. u gigabitového Ethernetu již setkáváme také s buzením laserem.
- ◆ průměru 9 µm – pak hovoříme o jednovidovém vlákně (*single mod*). Jednovidová vlákna mají již tak úzké jádro, že paprsek se šíří jádrem vlákna rovnoběžně, tj. neodráží se od rozhraní mezi oběma druhy skel. Jednovidová vlákna se zásadně budí laserem a jsou určena pro spoje na velké vzdálenosti.

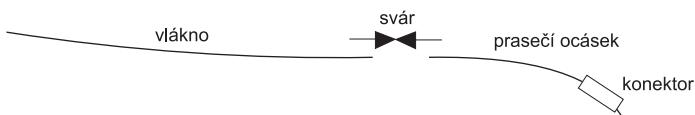
Na obr. 3.16 je znázorněna ochrana optických vláken. Optická vlákna jsou nejprve obalena tzv. primární ochranou, která zajišťuje pružnost vlákna. Bez primární ochrany je vlákno velice křehké. Sekundární ochrana pak zvyšuje ochranu vlákna. S odstraněnou sekundární ochranou se již setkáváme u optických propojovacích kabelů.



Obrázek 3.16: Vícevidové a jednovidové vlákno

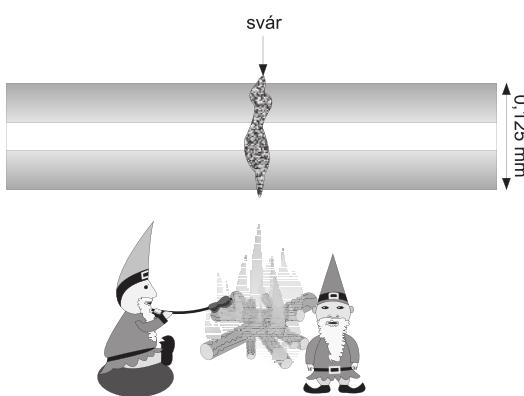
S optickými kably, které mají odstraněnu sekundární ochranu, se v běžných firemních podmínkách obtížně pracuje. V této sféře jsou populární optická vlákna s tzv. těsnou sekundární ochranou (průměr 900 µm = 0,9 mm), která integruje primární i sekundární ochranu. Takové kably jsou o něco dražší (proto se nehodí na propojování velkých vzdáleností), ale na druhou stranu je možné na tyto kably svépomocí nasazovat optické konektory.

Pokud se použijí kably s primární ochranou, musí se používat továrně připravené optické konektory nasazené na kus optického vlákna, tzv. prasečí ocásy (*pig tail*). Prasečí ocásek (obr. 3.17) se pak navařuje na vlákno.



Obrázek 3.17: Prasečí ocásek

Svár dvou optických kabelů (tj. i navaření prasečího ocásku) je „věda“. Stačí si představit, že se jedná o navaření vlákna skládajícího se ze dvou skel (obalu a jádra) – při prostém navaření by se sklovina obou vláken slila a vznikla by tak neprostupná překážka – viz obr. 3.18.



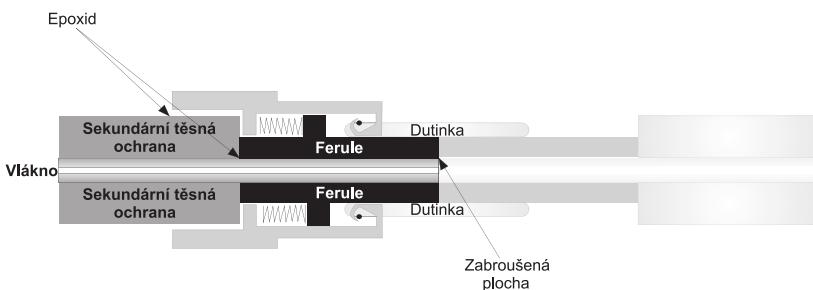
Obrázek 3.18: Chybný svár optického vlákna způsobí světelnému paprsku nepropustnou překážku

Vlákna je třeba svařit tak, aby překážka nevznikla. Kvalitní svár lze provést pouze pomocí nákladného zařízení.

V mnoha případech je tedy efektivnější vyhnout se svařování vláken a použít vlákna s těsnou sekundární ochranou. Na konce vláken si pak můžeme nasadit optické konektory, pomocí kterých vlákna propojíme.

Optická vlákna včetně primární a sekundární ochrany (resp. včetně těsné ochrany) jsou zpravidla uložena v optických kabelech. V optickém kabelu jsou svazky vláken opředeny kevlarem a vše je zapouzdřeno do vnějších obalů kabelu. Vnější obaly jsou pak provedeny podle toho, zdali je kabel určen pro uložení v budově, mezi budovami či na mořském dně.

Problém je i se zlomením optického vlákna. Pokud by se vlákno zlomilo v ruce, dojde k roztríštění jeho konce, proto se lámání provádí speciálním nástrojem, který roztríštění konců omezuje. Přesto se zlomené konce musí ještě zabroušit a zkontolovat mikroskopem, zdali byly všechny praskliny odbrášeny.



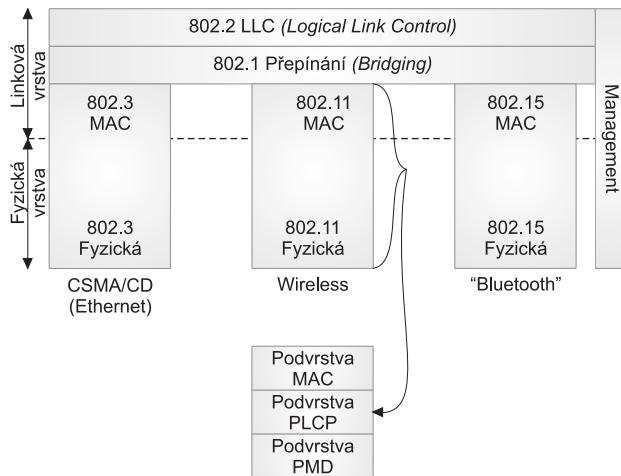
**Obrázek 3.19:** Optický konektor

Pomocí optického konektoru se propojují dvě optická vlákna. Cílem optického konektoru je přitlačit dvě zabroušená optická vlákna jádry proti sobě, aby světelný paprsek mohl přejít z jednoho vlákna do druhého.

Na obr. 3.19 je znázorněn princip propojení optického vlákna se sekundární těsnou ochranou s optickým konektorem. Z konce vlákna byla odstraněna ochrana a na místo ochrany byl na konec vlákna nasazen keramický kroužek – ferule. Konec vlákna s ferulí byl zabroušen a zabroušená plocha zkontolována pod mikroskopem. Dvě takto zakončená vlákna se vkládají proti sobě do dutinky, aby se zabránilo posunutí vláken. V dutince přiléhá jádro jednoho vlákna na jádro druhého vlákna, a paprsek tak může projít z jednoho vlákna do druhého.

## IEEE 802

IEEE 802 je rodina standardů pro lokální síť (LAN), metropolitní síť (MAN) i osobní síť (PAN). Tyto standardy pokrývají nejenom fyzickou, ale i linkovou vrstvu, přičemž linkovou vrstvu si rodina standardů IEEE 802 ještě dělí na několik podvrstev. Důležité je, že horní patra těchto linkových protokolů pod označením IEEE 802.1 a IEEE 802.2 jsou společná pro všechny protokoly pod nimi.

**Obrázek 3.20:** Rodina protokolů IEEE 802

V rodině protokolů IEEE 802.2 máme skupiny protokolů jako např.:

- ◆ Skupina IEEE 802.3, která specifikuje Ethernet. Tyto protokoly se oficiálně nazývají CSMA/CD podle mechanismu přístupu k přenosovému médiu.
- ◆ Skupina IEEE 802.11, která specifikuje bezdrátové lokální sítě (WLAN) v nelicencovaných pásmech.
- ◆ Skupina 802.15 pro osobní sítě (PAN).

Každá ze skupin protokolů IEEE 802 má více či méně společnou vrstvu MAC (*Medium Access Control*), která definuje linkový rámec. A tak např. pro všechny Ethernety standardu IEEE 802.3 máme stejný linkový rámec, který si popíšeme ve 4. kapitole.

Na fyzické vrstvě jsou již rozdíly mezi jednotlivými protokoly též skupiny. Jako příklad vezmeme skupinu 802.11 (*Wireless*). Linkový rámec 802.11 je obalen fyzickým záhlavím vrstvy PLCP (*Physical Layer Convergence Procedure*). Bez nadsázky můžeme říci, že MAC rámec 802.11 je zabalen do paketu podvrstvy PLCP, která obsahuje: synchronizační preambuli, přenosovou rychlosť, délku přenášených dat a kontrolní součet záhlaví PLCP podvrstvy. Podvrstva PMD (*Physical Medium Dependent*) pak zajistí kódování jednotlivých bitů a jejich odeslání do éteru. Výsledkem je, že jednotlivé protokoly jako 802.11b či 802.11g mají každý své odlišné podvrstvy PMD, kdežto podvrstva MAC je již pro všechny protokoly rodiny 802.11 stejná.

## Ethernet, FastEthernet a Gigabitový Ethernet

Protokol Ethernet byl vyvinut firmami DEC, Intel a Xerox pro přenosovou rychlosť 10 Mb/s. Pod označením Ethernet II. IEEE později standardizovalo Ethernet pod označením IEEE 802.3. Pozor! Linkové rámce (tj. rámce podvrstvy MAC) obou protokolů se liší!

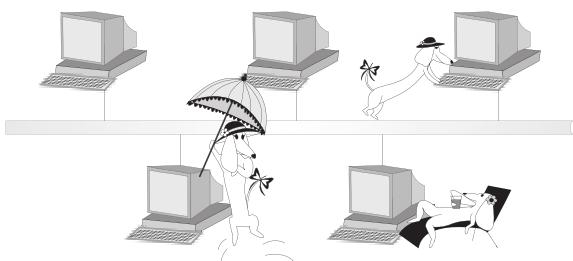
IEEE tyto protokoly oficiálně nenazývá Ethernet, ale CSMA/CD-podle mechanismu přístupu k přenosovému médiu. Avšak v záleptí uvidíte, že ironií osudu nejrychlejší protokoly Ethernet už od samotného mechanismu CSMA/CD upouštějí (ale i přesto se tak oficiálně nazývají).



**Poznámka:** Všechny stanice Internetu, které chtějí využívat protokol Ethernet, musí podporovat rámce tvaru Ethernet II. Pokud se na tom předem dohodnou, pak mohou používat i rámce IEEE 802.3.

Jako přenosové médium se původně pro Ethernet využíval koaxiální kabel. I když tato technologie patří minulosti, zanechala nám dva zajímavé termíny:

1. Vyjádření lokální sítě jako sběrnice, na kterou jsou navěšeny jednotlivé počítače. Takové znázornění LAN je mimořádné názorné. Tou sběrnicí byl právě již zmíněný koaxiální kabel.
2. Protokol CSMA/CD, s kterým mají dnešní správci potíže, zejména když si jej zapomenou vypnout na přepínači. V Gigabitovém Ethernetu je CSMA/CD teoreticky možné používat, ale 10G Ethernet jej již zcela opustil.



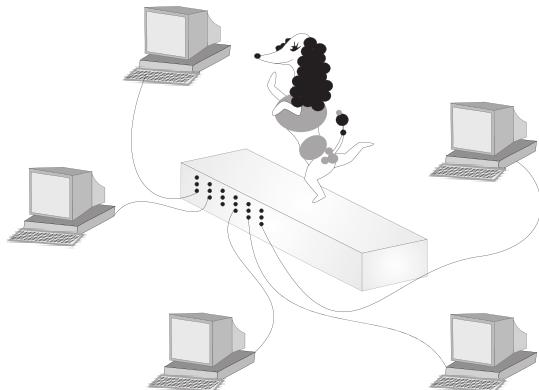
**Obrázek 3.21:** Segment Ethernetu tvořený koaxiálním kabelem

Koaxiální kabel byl společným médiem pro přenos informací. Pro výměnu rámců mezi stanicemi na společném médiu se používal protokol CSMA/CD. V tomto protokolu jsou si všechny stanice na společném médiu rovny. Potřebuje-li nějaká stanice vysílat, poslechně si, zdali jiná stanice právě nevysílá. V případě, že médium není používáno (jiná stanice nevysílá), může stanice začít vysílat. Tento postup se nazývá *Carrier Sense Multiple Access – CSMA* (*Carrier Sense* – česky „Naslouchání nosné“ – tím se míní, že vysílač naslouchá nosné vlně před pokusem vysílat).

Jenže naslouchat před pokusem vysílat může napadnou více stanic současně. Takže kromě toho, že stanice začne vysílat data, ještě připoslouchává, jestli nezačala vysílat současně ještě jiná stanice. V případě, že současně začala vysílat jiná stanice, dochází ke kolizi. Aby byla kolize detekovatelná i ostatními, nemohou při kolizi obě stanice okamžitě přestat vysílat, proto ještě nějakou dobu vysílají bezvýznamné znaky a pak se na náhodně zvolený časový interval odmlčí. A po tomto intervalu to zkusí znova. Tento postup se nazývá *Collision Detection – CD*. A už máme celý název CSMA/CD.

Čím je na společném médiu větší provoz, tím je větší pravděpodobnost vzniku kolizí. Optimální využití sítě je tak asi do 20 %. U varianty Ethernetu s teoretickou přenosovou rychlostí 10 Mb/s kalkulujeme tedy propustnost sítě asi na 2 Mb/s (tj. 256 KB/s). To je trochu zklamání. A pokud si říkáte, že společné přenosové médium je minulostí, pak vezte, že se s ním setkáme znova hned v další

kapitole o WLAN, protože v případě bezdrátového přenosu je radiové pásmo (resp. kanál radiového pásmá) přece také společným přenosovým médiem.



**Obrázek 3.22:** Opakovač (rozbočovač), most nebo přepínač

Vraťme se k Ethernetu. Koaxiální kabel měl jen omezenou délku nejvýš na několik set metrů. Pokud bylo nutné dělat rozsáhlejší rozvody, pak se položilo více segmentů s tím, že každý segment měl svůj koaxiální kabel. Tyto segmenty se pak propojily tzv. opakovačem (*Repeater*). Ve slovu opakovač cítíme propojení jen dvou segmentů, a tak se dnes místo slova opakovač používá slovo rozbočovač, vyjadřující fakt, že tento box může vzájemně propojit i více segmentů.

Později se příšlo na to, že není nutné opakovat síťový provoz na segmenty, na kterých neleží adresáti tohoto provozu. A na světě byl box, který se jmenoval Most (*Bridge*). Most má předávací tabulku. V tabulce je seznam všech linkových adresátů na jeho jednotlivých síťových rozhraních, tj. most má v předávací tabulce u každé linkové adresy poznamenáno, za kterým síťovým rozhraním mostu se nachází. Objeví-li se datový rámec na nějakém síťovém rozhraní mostu, podívá se most do datového rámce na adresu adresáta a z předávací tabulky zjistí, za jakým rozhraním se adresát nachází. Rámec pak zopakuje pouze do rozhraní, za kterým je adresát. V případě, že se adresát nachází za stejným rozhraním, neopakuje jej vůbec. Oběžníky se pochopitelně opakují do všech rozhraní.

Postupně začaly vznikat další linkové protokoly a každý segment mohl používat jiný linkový protokol. Bylo třeba vyvinout boxy, které by uměly konvertovat jeden linkový protokol na druhý. A na světě byl přepínač (*Switch*).

Mezitím se upustilo od koaxiálního kabelu a přešlo se na páry kroucené dvojlinky (na menší vzdálenosti) a na páry optických vláken na větší vzdálenosti. Největší revolucí pak byla skutečnost, že segment lokální sítě je, v případě přepínačů, možné změnit jen na komunikaci mezi stanicí a přepínačem, tj. na segmentu jsou jen dvě stanice. V čem je tak velká změna? Mezi koaxiálním kabelem a kroucenou dvojlinkou (resp. párem optických vláken) je jeden naprostě zásadní rozdíl. Tento rozdíl spočívá v tom, že se již nejedná o sdílené médium v tom původním smyslu. Na rozdíl od koaxiálního kabelu se totiž v případě kroucené dvojlinky používá jeden páru pro vysílání a jiný páru pro příjem (obr. 3.23) (resp. v případě optického vlákna se jedno vlákno používá pro vysílání a druhé pro příjem).

Takže pokud jsou na segmentu pouze dvě stanice, pak se jednoduše vysílání jedné stanice přivede na příjem druhé stanice. Každá stanice může tedy vysílat tak, jak je jí libo, a nikdy nemůže dojít ke kolizi. To znamená, že:

- ◆ je možné vypnout CSMA/CD, aby při poslouchávání zbytečně nezatěžovalo.
- ◆ je možné plně využít přenosovou rychlosť téměř až k její teoretické hranici, a to v každém směru nezávisle! (Teoretické hranice nelze dosáhnout, protože rámce začínají synchronizační preambuli a navíc mezi rámci musí být alespoň minimální mezera.)

Síťová rozhraní na takovémto bezkolizním segmentu přepínáme do tzv. plně duplexního provozu (*Full Duplex*), kdy je zcela odděleno vysílání od příjmu. Odměnou je pak stabilní vysoká přenosová rychlosť.

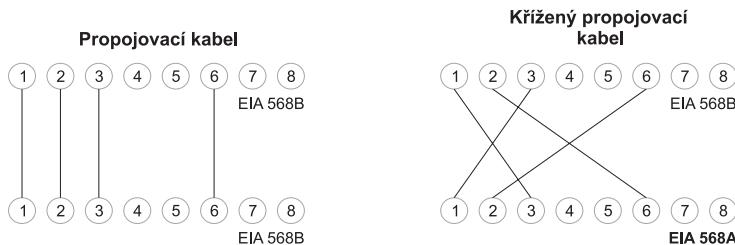
**Přehled jednotlivých protokolů Ethernet**

Ethernet		Max. přenosová rychlosť	Kabel	Max. délka segmentu	Zakončení
10Base5	Původní Ethernet	10 Mb/s	Koaxiální – nepřerušovaný	500 m (největší vzdálenost mezi stanicemi na koaxiálním kabelu)	Na kabel „napíchnutý“ transceiver s rozhraním AUI
10Base2	Tenký koaxiální kabel	10 Mb/s	Koaxiální kabel přerušený u každé stanice vložením tzv. BNC T-konektoru	185 m (největší vzdálenost mezi stanicemi na koaxiálním kabelu)	BNC konektor
10Base-T	T = <i>Twisted pair</i>	10 Mb/s	Kategorie 5 a výše – dva páry kroucené dvojlinky (zbylé dva páry zůstávají neobsazeny)	100 m	RJ-45
10Base-F	F = <i>Fiber</i>	10 Mb/s	Pár optických vláken		Optický konektor
100Base-TX	FastEthernet	10 Mb/s	Kategorie 5 a výše – dva páry kroucené dvojlinky (zbylé dva páry zůstávají neobsazeny)	100 m	RJ-45
100Base-FX		100 Mb/s	Pár optických vláken		Optický konektor
1000Base-T	Gigabitový ethernet	1 Gb/s	Kategorie min. 5e (všech 8 vodičů)	100 m	RJ-45
1000Base-SX		1 Gb/s	Pár vícevidových optických vláken	500 m	Optický konektor
1000Base-LX		1 Gb/s	Pár jednovidových optických vláken	10 km	Optický konektor
10GBase-T	10G Ethernet	10 Gb/s	Kategorie min. 6 (všech 8 vodičů)	55 m (cat 6) 100 m (cat 7)	RJ-45
10GBase-SR	„short range“	10 Gb/s	Pár vícevidových optických vláken	300 m	Optický konektor
10GBase-LR	„long range“	10 Gb/s	Pár jednovidových optických vláken	10 km (30 km)	Optický konektor
10GBase-ER	„extended range“	10 Gb/s	Pár jednovidových optických vláken	40 km	Optický konektor

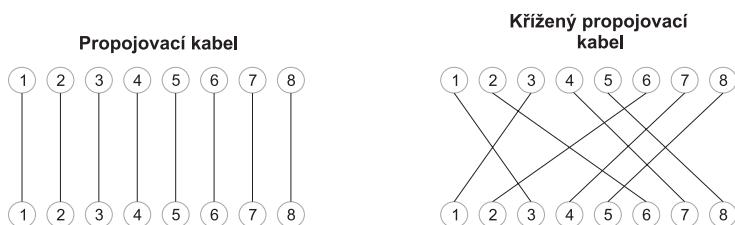
Zapojení vývodů pro Ethernet a Fast Ethernet v konektoru RJ-45 je na obr. 3.23. Při propojování je důležitá ještě jedna věc. Počítač se jeví jako DTE a aktivní prvek (rozbočovač, most, přepínač atp.) se jeví jako DCE. Takže pokud spojujeme počítač s počítačem (tj. DTE s DTE) nebo aktivní prvek s aktivním prvkem (tj. DCE s DCE), tak, podobně jako v případě nulového modemu, musíme vysílání jedné strany převést na příjem na druhé straně, tj. k propojení musíme použít tzv. křížený propojovací kabel. Modernější síťová rozhraní si umí detekovat, jestli je druhá strana DTE nebo DCE, a samy se přepnou, takže pak ani není třeba používat křížený propojovací kabel (obr. 3.24). U Gigabitového Ethernetu pak musí být propojeno všech 8 vývodů konektoru RJ-45 (obr. 3.25).



**Obrázek 3.23:** Zapojení vývodů pro 10BASE-T, resp. 100BASE-TX



**Obrázek 3.24:** Propojovací kably pro 10Base-T a 100BaseTX



**Obrázek 3.25:** Propojovací kably pro 1000Base-T

## Bezdrátové lokální sítě WLAN

Bezdrátové lokální sítě (*Wireless Local Area Network*) WLAN jsou v současné době na velkém vzestupu. Označují se též Wi-Fi, což je obchodní značka (trademark), kterou zavedlo konsorciump výrobců „Wi-Fi Alliance“, aby označovala skutečnost, že jím testované výrobky splňují standardy skupiny IEEE 802.11.

Bezdrátové sítě mají několik výhod:

- ◆ Mobilita – uživatel není vázán na zásuvku a kabel.
- ◆ Rychlosť a jednoduchost uvedení do provozu.

- ◆ Nižší celkové náklady na vybudování sítě (není nutné budovat nákladné rozvody strukturované kabeláže).
- ◆ Rozšiřitelnost – vhodnou volbou a polarizací antén lze snadno a rychle nejenom zvyšovat kapacitu přístupových bodů, ale také vykryt území podle potřeby.
- ◆ Roaming – je důležitou funkcí WLAN. Pokud je nastaven, mohou se mobilní stanice volně pohybovat v oblasti, která je pokryta signálem více přístupových bodů. Podmínkou je, aby tyto přístupové body byly vzájemně propojeny, např. páteřní sítí (označovanou jako Distribuční Systém – DS). Mobilní stanice se přihlašují k připojněmu bodu s nejlepším poměrem signálu k šumu. Když poklesne tento poměr pod stanovenou mez nebo se signál připojného bodu zcela ztratí, přepne se mobilní stanice do promiskuitního módu a vyhledá jiný připojny bod s nejlepším poměrem signál/šum. Roaming nijak nenaruší komunikaci aplikací v síti.

V běžné praxi se s WLAN setkáváme:

#### **Vnitřní prostory (indoor):**

- ◆ V provozech, kde jsou mobilní zařízení, která nemohou být vázány na zásuvku a kabel, jako jsou např. nemocnice či výrobní haly.
- ◆ V kombinaci se strukturovanou kabeláží, kdy stolní počítače jsou zapojeny klasicky pomocí rozvodů strukturované kabeláže a přenosné počítače jsou připojeny bezdrátově.
- ◆ V dočasně budovaných sítích, jako jsou např. výstaviště, studentské koleje apod.

#### **Vnější prostory (outdoor):**

- ◆ Lokální poskytovatelé Internetu (v rámci vesnice či bloku domů většího sídla).

WLAN používají jako přenosové médium rádiové vysílání o kmitočtu 2,4 GHz nebo 5 GHz. Avšak WLAN na 5GHz není u nás povoleno. Na provoz WLAN není třeba licence Českého telekomunikačního úřadu – ČTÚ. Nikdo tak ani nekoordinuje přidělování licencí, a tak se může stát, že budeťe rušeni i od jiných sítí WLAN (např. od poskytovatelů Internetu). Při nevhodném uspořádání své sítě se dokonce můžeme úspěšně rušit i sami. Dalším zdrojem rušení mohou být zařízení využívající stejného pásmá, jako jsou např. mikrovlnné trouby, bezdrátové telefony apod.

WLAN specifikuje norma **IEEE 802.11**. WLAN používají protokol přístupu k médiu (*Media Access Protocol - MAC*) nazývaný *Carrier Sense Multiple Access/Collision Avoidance – CSMA/CA*. Je odvozen z protokolu CSMA/CD (*Collision Detection*), který známe z Ethernetu. Na rozdíl od Ethernetu však u bezdrátového vysílače není jednoduché detekovat kolize, proto se k jejich detekci využívá systém potvrzování.

Byla a je vytvářena celá řada rozšíření normy IEEE 802.11, která vylepšují funkci WLAN. Patří sem zejména 802.11b a 802.11g.

<b>Protokol</b>	<b>Frekvence</b>	<b>Propustnost</b>	<b>Max. přenosová rychlosť</b>
Původní 802.11	2,4 GHz	0,9 Mb/s	2 Mb/s
802.11a	5 GHz (u nás není povoleno)	23 Mb/s	54 Mb/s
802.11b	2,4 GHz	4,3 Mb/s	11 Mb/s
802.11g	2,4 GHz	19 Mb/s	54 Mb/s

V České republice můžeme v pásmu 2,4 GHz využívat 14 kanálů (kanály 0 až 13).

## Typické WLAN-konfigurace

### Sítě Ad-hoc

Bezdrátové stanice mezi sebou komunikují přímo. Není potřeba žádný přístupový bod ani žádná konfigurace. Tato metoda není vhodná pro více než 8–10 počítačů. Tato metoda je náročnější na konfiguraci, proto není příliš oblíbená.

### Infrastrukturní síť

Základem tohoto typu sítě je přístupový bod (*Access point* - AP). Přístupový bod je nepohyblivý a tvoří zároveň základnovou rádiovou stanici a datový most. Přístupový bod je zpravidla připojen dále do LAN.

Přístupový bod provede tzv. asociaci se stanicí. Stanice se může pohybovat v dosahu přístupového bodu. Pokud spolu komunikují dvě stanice v rámci společného přístupového bodu, pak komunikace běží dvakrát: jednou od odesilatele na přístupový bod a podruhé z přístupového bodu k adresátovi. Přesto je tento typ sítě oblíbený, mj. i proto, že je nenáročný na konfiguraci jednotlivých stanic.

Přístupový bod má také možnost nastavení určitých bezpečnostních prvků, jako je šifrování, filtrace linkových či IP adres atd. Počet koncových stanic připojených k jednomu přístupovému bodu se udává jako 15–38.

### Více přístupových bodů (roaming)

Tato síťová topologie se též označuje jako oblast rozšířených služeb. V případě infrastrukturních sítí má stanice problém, pakliže se pohybuje mezi více přístupovými body. Vždy totiž musí dojít k nové asociaci. Roaming umožňuje vytvořit virtuální oblast (ESS), která je vykryta více přístupovými body. Stanice se pak může pohybovat jakoby volně v rámci celé této virtuální oblasti a jednotlivé přístupové body si ji přebírají jeden od druhého.

Tato topologie je omezena tím, že všechny přístupové body musí být v téže LAN (resp. VLAN).

### Páteřní spoje Point-to-Point

Propojení dvou sítí pomocí přístupových bodů v konfiguraci Point-to-Point. Toto spojení je typické pro venkovní (*outdoor*) řešení za použití přídavných směrových antén, které pak umožňují spojení i na mnoho kilometrů.

### Antény

WLAN se nepoužívají pouze jako varianta k „drátovému“ ethernetu, ale stále častěji i v otevřeném terénu. V otevřeném terénu se navíc používají přídavné antény a také nesmíme zapomenout na bleskojistky sloužící jako ochrana proti přírodním elektrostatickým výbojům. Kombinací výkonu vysílače a zisku antény lze komunikovat na značné vzdálenosti. Je však nutné dát pozor, aby nebyla překročena norma vyzářeného výkonu stanovená Českým telekomunikačním úřadem. Zařízení pracují zpravidla poloduplexně, ale je možné z nich udělat i plně duplexní spoj tak, že použijeme souběžně

dva páry antén, jednu pro vysílání a druhou pro příjem. Vysoce ziskové antény na příjmu nám navíc pomohou i s problémy ohledně vyzářeného výkonu.

Antény se používají s horizontální, vertikální nebo kruhovou polarizací. Vertikální či horizontální polarizaci nastavíme natočením antény, na kruhovou polarizaci musíme mít jiný typ antény, tzv. šroubovici. Pro spoje Point-to-Point používáme zpravidla paraboly různého průměru, pro Point-to-Multipoint se používají na přístupových bodech antény Yagi, svislé vše směrové antény nebo sekundové antény.

Při budování vnějších spojů je vždy nutné proměření a respektování fyzikálních zásad radiových spojů, jako je např. Fresnelova zóna, určující maximální výšku překážky mezi anténami.

## Sledování WiFi

Sledování v bezdrátových sítích není triviální. Pokud totiž použijete standardní WiFi-kartu, pak rychle ve Wiresharku zjistíte, že:

1. karta nepracuje v promiskuitním módu. Promiskuitní mód totiž výrobci novějších karet nepodporují, ale v bazarech lze koupit starší karty, které promiskuitní mód umožňují.
2. nachytané rámce se budou jevit jako rámce protokolu Ethernet.

Řešením je obstarat si speciální kartu, jejíž nabídku naleznete např. na webových stránkách Wiresharku (AirPcapp). Pokud si obstaráte tento specializovaný hardware, pak hned je najednou život veselý.

V následujícím výpisu programu Wireshark nevidíme přímo pole podvrstvy PLCP (viz odstavec 3.7), ale informace o jednotlivých rámcích, které jsou získány z ovladače síťové karty. Tyto informace tvoří tzv. záhlaví Radiotap (<http://www.radiotap.org/>). V tomto pseudozáhlaví pak např. máme:

- ◆ Verzi, která je tč. jen 0.
- ◆ Header length – délka pseudozáhlaví.
- ◆ Present flags – příznaky jednotlivých dále následujících polí pseudozáhlaví (1 – příslušné pole se v pseudo-záhlaví vyskytuje, 0 – nevyskytuje se).
- ◆ Pole Flags – vlastnosti odchyceného rámce.
- ◆ Pole Data Rate – přenosová rychlosť.
- ◆ Pole Channel obsahující:
  - Číslo kanálu.
  - Vysílací frekvenci.
  - Vlastnosti kanálu (Channel type), které obsahuje typ modulace, vysílací pásmo atd. Zajímavý je příznak **Passive**, který nastavený na 1 by signalizoval, že rozhraní umožňuje jen naslouchání.
- ◆ Dále pak následují parametry přijímaného signálu a šumu. Např. pole Antenna specifikuje anténu (0 = 1. anténa).
- ◆ Pole 802.11 FCS obsahuje kontrolní součet a informaci, jestli je kontrolní součet Ok.

+ Frame 19 (106 bytes on wire, 106 bytes captured)

- Radiotap Header v0, Length 24

Header revision: 0

Header pad: 0

```
Header length: 24
- Present flags: 0x000058ee
    .... .... .... .... .... .0 = TSFT: False
    .... .... .... .... .... .1. = Flags: True
    .... .... .... .... .... .1.. = Rate: True
    .... .... .... .... .... 1... = Channel: True
    .... .... .... .... .... .0 .... = FHSS: False
    .... .... .... .... .... .1. .... = DBM Antenna Signal: True
    .... .... .... .... .... .1.. .... = DBM Antenna Noise: True
    .... .... .... .... .... 1.... = Lock Quality: True
    .... .... .... .... .... .0 .... = TX Attenuation: False
    .... .... .... .... .... .0. .... = DB TX Attenuation: False
    .... .... .... .... .... 0.... = DBM TX Attenuation: False
    .... .... .... .... .... 1.... = Antenna: True
    .... .... .... .... .... .1 .... = DB Antenna Signal: True
    .... .... .... .... .... .0. .... = DB Antenna Noise: False
    .... .... .... .... .... 1.... = FCS in header: True
    0.... .... .... .... .... = Ext: False
- Flags: 0x10
    .... .0 = CFP: False
    .... .0. = Preamble: Long
    .... .0.. = WEP: False
    .... 0... = Fragmentation: False
    ...1 .... = FCS at end: True
    ..0. .... = Data Pad: False
Data Rate: 54.0 Mb/s
Channel: 12
Channel frequency: 2467
- Channel type: 802.11g (pure-g) (0x00c0)
    .... .... .0 .... = Turbo: False
    .... .... .0. .... = Complementary Code Keying (CCK): False
    .... .... .1.. .... = Orthogonal Frequency-Division Multiplexing (OFDM): True
    .... .... 1.... .... = 2 GHz spectrum: True
    .... .... 0 .... .... = 5 GHz spectrum: False
    .... .0. .... .... = Passive: False
    .... .0.. .... .... = Dynamic CCK-OFDM: False
    .... 0.... .... .... = Gaussian Frequency Shift Keying (GFSK): False
SSI Signal: -71 dBm
SSI Noise: -100 dBm
Signal Quality: 82
Antenna: 0
SSI Signal: 29 dB
802.11 FCS: 0xbc909102 [correct]
+ IEEE 802.11
+ Logical-Link Control
+ Internet Protocol, Src: 82.77.99.19 (82.77.99.19), Dst: 10.0.0.1 (10.0.0.1)
+ User Datagram Protocol, Src Port: 16360 (16360), Dst Port: 53605 (53605)
+ Data (18 bytes)
```



## Kapitola 4

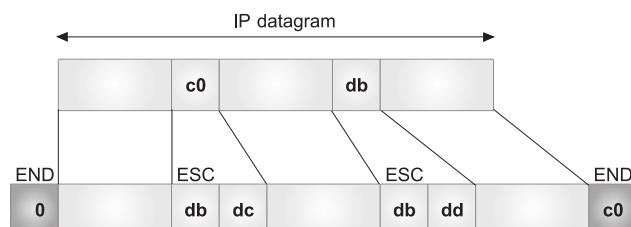
# Linková vrstva

Při prvním čtení této kapitoly doporučujeme se zastavit u protokolu SLIP a dál pokračovat až protokolem Ethernet. Při druhém čtení této kapitoly je pak významné pochopit protokol HDLC, byť jej třeba nikdy používat nebudete. Je to proto, že principy protokolu HDLC jsou základem mnoha linkových protokolů. Např. protokol PPP je přímo odvozen od protokolu HDLC. Samotný protokol PPP je probrán velice podrobně, proto jej doporučujeme číst až při třetím čtení s protokolem Frame Relay. Kapitola o protokolu Frame Relay má zvláštní postavení – jde za rámec rodiny protokolů TCP/IP. Je to příklad protokolu, který slouží operátorům poskytujícím přenosové služby. Uživatelé jim svěřují své IP datagramy a očekávají, že jim je poskytovatel přenese do požadované lokality.

Linkových protokolů je velké množství. My se zastavíme u protokolů SLIP, HDLC, PPP, Frame Relay, Ethernet a WLAN.

## SLIP

Protokol **Serial Line IP** (dále jen SLIP) je bezesporu nejjednodušším možným linkovým protokolem. SLIP totiž vkládá IP pakety přímo do sériové linky. Pro řízení linky jsou mezi data vkládány tzv. Esc-sekvence (analogicky jako při komunikaci počítače s terminálem či tiskárnou). Protokol SLIP je specifikován normou RFC-1055.



Obrázek 4.1: Rámec protokolu SLIP

Každý rámec protokolu SLIP je ukončen křídlovou značkou označovanou v případě protokolu SLIP též jako znak END ( $c0_{16}$ ). Většina implementací protokolu SLIP však znak END umisťuje navíc i na počátek rámce. Jestliže se vyskytne znak  $c0_{16}$  v přenášených datech, je nahrazen tzv. SLIP Esc-sekvencí: dvojicí  $db_{16},dc_{16}$  (ASCII Esc-sekvence je  $1b_{16}$ ). A konečně znak  $db_{16}$  je nahrazován dvojicí  $db_{16},dd_{16}$ .

Protokol SLIP je velice jednoduchý, ale nezabezpečuje:

- ◆ **Detekci chyb při přenosu.** Je proto výhodné použít detekci chyb alespoň na úrovni modemů, např. podle doporučení V.42, protože jinak by detekce chyb nemusela být zabezpečena vůbec (IP protokol má kontrolní součet pouze ze záhlaví a kontrolní součet v protokolu UDP je nepovinný). Je proto nebezpečné umisťovat za linky s protokolem SLIP např. DNS servery nebo NFS servery, které nemívají zapnut kontrolní součet v UDP datagramu.
- ◆ Rámec protokolu **SLIP neneše informaci o přenášeném protokolu** síťové vrstvy. Je proto možné přenášet vždy pouze jeden síťový protokol, tj. není možné na jedné lince mixovat např. pakety protokolu IPv4 s pakety protokolu IPv6.
- ◆ **Není možné, aby se oba konce např. informovaly o své IP adrese či jiných konfiguračních parametrech.**
- ◆ **Nelze jej použít pro synchronní linky.**

Protokol SLIP má díky své jednoduchosti i jednu výhodu. Díky tomu, že neposkytuje téměř žádné služby, přenáší minimum služebních informací, takže je poměrně oblíben na méně poruchových pomalých sériových linkách.

## HDLC

Protokol HDLC provádí detekci chyb i řízení toku dat. HDLC je standardizován mj. normami: ISO-3309, ISO-4335, ISO-6159 a ISO-6256.

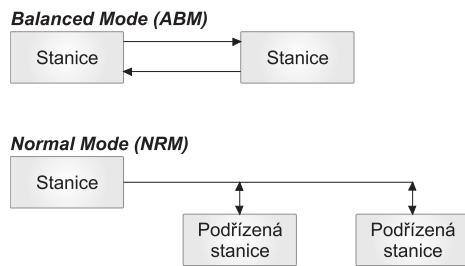
Protokol HDLC vznikl z protokolu SDLC firmy IBM, který byl určen pro synchronní přenos. Dnes se protokol SDLC vesměs chápe jako podmnožina protokolu HDLC, i když ne všechny možnosti protokolu SDLC byly do HDLC zahrnuty.

Později byla norma HDLC rozšířena i pro asynchronní přenos. Odlišnosti asynchronní varianty si ukážeme na příkladu protokolu PPP, který je odvozen od protokolu HDLC a zpravidla se s ním setkáváme na asynchronních linkách. Nadále budeme v této kapitole předpokládat synchronní bitově orientovaný přenos na fyzické úrovni.

Protokol HDLC rozeznává tzv. módy:

- ◆ **ABM (Asynchronous balanced mode)** je určen pro propojení dvou stanic plně duplexním spojem, tj. obě stanice mohou vysílat současně, aniž by si vzájemně na lince překážely. Většinou se dnes setkáváme právě s tímto módem.
- ◆ **NRM (Normal response mode)** odpovídá víceméně protokolu SDLC. Jedná se o situaci, kdy je propojeno více stanic na poloduplexní spoji (tzv. přepínaný duplex mezi vysíláním a příjemem). Pro vysílání i příjem slouží společné přenosové médium, tj. v jednom okamžiku lze buď přijímat, nebo vysílat. Jedna stanice je označena jako řídicí stanice a ostatní jako podřízené stanice. Je definován tzv. *pooling*, tj. řízení, kdy smí která stanice vysílat. Bez povolení smí vysílat pouze řídicí stanice. Ostatní stanice mohou vysílat jen tehdy, když jim to řídicí stanice povolí. Povolení se nastavuje bitem P/F v řídicím poli rámce HDLC. Pouze řídicí stanice může vydávat příkazy – podřízené stanice mohou jen odpovídat. Tento mód je v Internetu používán jen výjimečně. Nebudeme se jím zabývat – uvádíme jej zejména proto, abychom vysvětlili význam bitu P/F.

- ◆ **ARM (Asynchronous response mode)** je dnes málo běžný režim. Opět je určený pro situaci, kdy je propojeno více stanic pomocí sdíleného přenosového média. Na rozdíl od NRM mohou podřízené stanice vysílat bez povolení od nařízené stanice. Nadřízená stanice je zodpovědná za inicializaci, ošetřování chyb apod.



**Obrázek 4.2:** Módy ABM a NTM

Formát HDLC rámce je znázorněn na obr. 4.3.



**Obrázek 4.3:** Rámcem protokolu HDLC

### Křídlová značka (Flag)

Křídlová značka uvozuje a ukončuje datový rámcem, tj. každý HDLC-rámeček začíná a končí právě křídlovou značkou. Jdou-li dvě křídlové značky po sobě, uvozují prázdný rámcem, který se nezpracovává.



**Poznámka:** Není-li po jistou dobu po přenosové lince třeba vysílat žádná data (je-li linka v klidovém stavu), pak se na linku čas od času vkládají posloupnosti samotných křídlových značek, aby se signalizovalo, že linka je živá (tj. že druhý konec je dostupný).

Křídlová značka se skládá z osmi bitů: 0111 1110. Šest po sobě následujících jedniček určuje právě křídlovou značku. Okamžitě můžete namítnout: Vždyť přenášený znak se může skládat i z více než šesti jedniček! Jenže právě bitově orientovaná synchronní verze HDLC používá trik. Na vstupu, kdykoliv, když data obsahují více než pět jedniček za sebou, tak se za těchto pět jedniček automaticky vloží jedna nula. Analogicky na výstupu, je-li v přenášených datech za pěti jedničkami 0, pak se tato nula vypouští. Není-li za pěti jedničkami nula, ale jednička, jedná se o křídlovou značku. Tato technika se též označuje jako *bit stuffing*.

Tato technika je možná jen u bitově orientovaného přenosu, kde se přenáší řada bitů. U znakově orientovaného přenosu tato technika není možná, protože počet přenášených bitů musí být dělitelný délkom znaku (zpravidla 7 nebo 8 bitů). Vložení bitu by pak toto pravidlo porušilo.

## Adresní pole

Adresní pole je dlouhé 8 bitů. Vyjadřuje adresu stanice, které je paket určen. Je vcelku evidentní, že toto pole má své opodstatnění u módu NRM (resp. protokolu SDLC), kdy spolu komunikují často více než dvě stanice. Je však striktně vyžadováno, proto je přítomno ve všech mutacích protokolu HDLC. Pro úplnost uvedeme, že protokol PPP používá např. hodnotu 1111 1111, tj. oběžník.

Adresní pole v rámci HDLC nesouvisí s IP adresou. Jedná se o linkovou adresu!

## Datové pole a typ přenášeného protokolu

Datové pole obsahuje přenášená data (*payload*). Všimněte si, že záhlaví HDLC-rámce neposkytuje možnost specifikace protokolu vyšší vrstvy, tj. neumožňuje např. mixovat rámce protokolu IPv4 s protokolem IPv6. Volba protokolu se přitom určuje v počátečním inicializačním dialogu.

Toto omezení platí pro tzv. číslované rámce. U nečíslovaných rámčů je možné na počátek datového pole zadat specifikaci protokolu.

## Řídicí pole

Řídicí pole je nejsložitější pole. Podle nejnižších dvou bitů řídicího pole rozlišujeme 3 typy rámčů HDLC:

- ◆ **Informační rámce neboli I-rámce (v nejnižším bitu je 0)**, které jsou primárně určeny pro přenos dat. Mohou však ve svém řídicím poli přenášet i některé řídicí informace (např. pozitivní potvrzení přijatých rámčů).
- ◆ **Nečíslované rámce neboli U-rámce (v nejnižších dvou bitech je 11)**, které se používají nejen pro přenos dat, ale i pro mnohé řídicí funkce (úvodní inicializační dialog, řízení linky a diagnostiku).
- ◆ **Rámce upervizitu neboli S-rámce (v nejnižších dvou bitech je 10)**. Používají se pro řízení toku dat (požadavek na vysílání, potvrzování I-rámčů atd.). S-rámce mohou být používány, až když je linka inicializována, tj. když mohou být používány I-rámce. S-rámce zpravidla neobsahují datové pole.

Řídicí pole je u U-rámčů osmibitové. U I-rámčů a S-rámčů může být buď osmibitové, nebo šestnáctibitové. Na následujících obrázcích je použito šestnáctibitové řídicí pole, tj. jedná se o tzv. rozšířený mód.

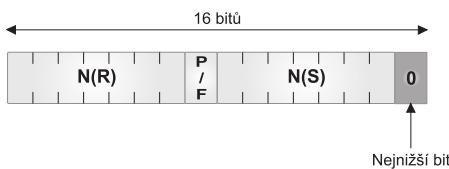


**Poznámka:** Označení módů ABM a NRM je většinou míněno jako příslušný mód s osmibitovým řídicím polem. Rozšířené módy s šestnáctibitovými řídicími poli se někdy označují jako ABME a NRME.

Co se v osmibitovém řídicím poli ušetří? Použijí se pouze tři bity pro číslování N(S) i N(R), tj. rámce se nečíslují modulo 128, ale jen modulo 8.

### I-rámeček

V I-rámci slouží pole  $N(S)$  a  $N(R)$  pro číslování rámců. Čísluje se od nuly do 127 (nejvyšší možné číslo v sedmi bitech), po dosažení čísla 127 se opět pokračuje od nuly.  $N(S)$  určuje číslo odeslaného rámce. Naopak pole  $N(R)$  slouží pro potvrzení přijatého rámce. Jelikož je komunikace obousměrná, potvrzují se v protisměru správně přijaté rámce.

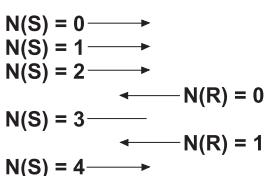


Obrázek 4.4: I-rámeček

V případě, že není třeba v protisměru posílat data, použije se k potvrzení přijatých dat S-rámeček (s příkazem RR). V případě, že přijatý rámec byl po přepočítání kontrolního součtu shledán jako chybný, je pomocí S-rámce (příkazem REJ) vyžádáno opakování přenosu – tzv. negativní potvrzení. Tento S-rámeček ve svém poli  $N(R)$  zopakuje číslo posledního správně přijatého rámce.

Je možné potvrzovat rámce postupně, po jednom. To ovšem prodlužuje odezvu, protože se musí u každého rámce čekat na jeho potvrzení. Proto se zpravidla potvrzuje rámce pomocí tzv. okna.

Je-li např. okno rovno třem (viz obr. 4.5), pak se čeká po odeslání tří paketů na potvrzení prvního z nich. Po potvrzení prvního se odešle čtvrtý, po potvrzení druhého se odešle pátý... Ve vyrovnávací paměti odesilatele je nutné udržovat celé okno nepotvrzených rámů pro případ vyžádání chybného nebo ztraceného rámce.



Obrázek 4.5: Příklad dialogu při okně o velikosti 3

Bit P/F je důležitý pro NRM-mód protokolu HDLC. V NRM-módu řídící stanice nastavením tohoto bitu na P (=Pool) povoluje podřízené stanici vysílat data. Podřízená stanice nechává při vysílání tento bit nastaven. Tím signalizuje, že chce ve vysílání pokračovat. U posledního vysílaného rámce tento bit shodí (nastaví ho na F=Final).

### S-rámeček

S-rámeček může potvrzovat správně přijatý rámec. Dále v poli „Příkaz“ může nést následující příkazy, resp. odpovědi:

- ◆ **RR** (*Receiver Ready* = přijímač připraven). Informuje protějšek, že přijímač je připraven akceptovat I-rámce. Dále se používá jako signalizace, že linka je opět volná (poté co tomu tak nebylo). Ještě musíme připomenout, že se též může použít (jak bylo uvedeno u popisu I-rámce) k potvrzení čísla správně přijatého rámce.

**Obrázek 4.6:** S-rámcem

- ◆ **RNR** (*Receiver Not Ready* = přijímač nepřipraven). Informuje protějšek o dočasné neschopnosti přijímat I-rámce a zároveň potvrzuje dosud přijaté rámce.
- ◆ **REJ** (*Reject* = odmítnutí). Přijetí chybného rámce, tj. používá se jako příkaz nebo jako odpověď pro zopakování vysílání.

### **U-rámcem**

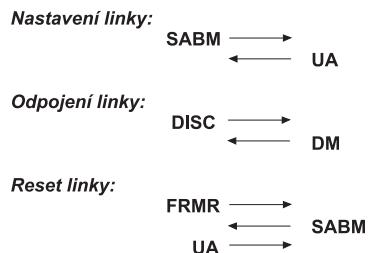
U-rámce mohou jak přenášet data, tak i příkazy a odpovědi, např.:

- ◆ **SABM** (*Set Asynchronous Mode* = nastavení módu ABM). Příkaz nastavuje linku do módu ABM s osmibitovým řídicím polem.

**Obrázek 4.7:** U-rámcem

- ◆ **SABME** (*Set Asynchronous Mode* = nastavení módu ABM). Příkaz nastavuje linku do módu ABM s šestnáctibitovým řídicím polem – jedná se tedy o tvar protokolu HDLC zobrazovaný v této kapitole.
- ◆ **SNRM** (*Set Normal Response Mode* = nastavení módu NRM). Příkaz nastavuje linku do módu NRM s osmibitovým řídicím polem.
- ◆ **SNRME** (*Set Normal Response Mode* = nastavení módu NRM). Příkaz nastavuje linku do módu NRM s šestnáctibitovým řídicím polem.
- ◆ **UA** (*Unnumbered Acknowledgment* = nečíslované potvrzení). Používá se pro potvrzení SABM, SABME, SNRM, SNRME a DISC.
- ◆ **DISC** (*Disconnect* = odpojení). U komutovaných linek znamená požadavek na zavěšení. U pevných linek umožňuje zrušit aktuální operační mód (aktuální nastavení).
- ◆ **DM** (*Disconnect Mod*). Pozitivní potvrzení příkazu DISC.
- ◆ **FRMR** (*Frame Reject* = odmítnutí rámce). Používá se k indikaci přijetí vadného rámce. Přitom není možné dosáhnout opravy opakováním vysílání (retransmisi). Po obdržení FRMR se začíná znova od nastavení módu linky, tj. jedním z příkazů SABM, SABME, SNRM či SNRME. Počátek datové části paketu obsahuje 2–3 bajtová pole s kódem chyby (např. chyba v řídicím poli rámce, chyba v informačním poli, překročení kapacity rámce, chyba v sekvenci přijatých rámci atp.).

- ◆ **XID** (*Exchange Station Identification* = výměna konfiguračních informací). Příkazy a odpovědi XID se používají k úvodní inicializační sekvenci, kdy se stanice domlouvají na délce kontrolního součtu, přenášeném protokolu vyšší vrstvy atd.
- ◆ **UI** (*Unnumbered Information* = nečíslované datové rámce). Používá se pro přenos nečíslovaných (nepotvrzovaných) datových rámciů.



**Obrázek 4.8:** Jednoduché dialogy protokolu HDLC

### Kontrolní součet (*Frame Check Sequence – FCS*)

Z přenášených dat, adresního a řídicího pole se počítá kontrolní součet, který je zpravidla buď 32bitový, nebo 16bitový. Adresát z přijatého rámce rovněž spočte kontrolní součet, který porovná s kontrolním součtem v přijatém rámci. Jsou-li shodné, pak považuje přijatý rámec za správně přenesený. V opačném případě si u číslovaných rámci může vyžádat zopakování přenosu.

Určení, jaký kontrolní součet se bude používat, je součástí úvodního dialogu stanic při inicializaci spojení (pomocí příkazu XID).

### Základní vlastnosti protokolu HDLC

Protokol HDLC je vyspělý linkový protokol umožňující:

- ◆ Pomocí kontrolního součtu (FCS) zjišťovat chybné rámce.
- ◆ Při přijetí chybného číslovaného rámce je možné vyžádat zopakování vysílání rámce. To neplatí pro nečíslované rámce (nečíslované chybné rámce se zahazují).

Linka se v protokolu HDLC nachází v následujících stavech:

- ◆ V odpojeném stavu.
- ◆ Ve stavu nastavování linky, kdy se mohou používat pouze U-rámce.
- ◆ Ve stavu přenosu informací, tj. za běžných okolností se přenášejí pouze I a S-rámce (nepoužívají-li se k přenosu dat U-rámce).
- ◆ Odpojování linky, opět se přenáší jen U-rámce.

### Cisco HDLC (cHDLC)

Implementace protokolu HDLC firmou CISCO se označuje jako cHDLC. Tato implementace má následující odlišnosti pro implementaci HDLC podle standardů ISO popisované v předchozích odstavcích:

- ◆ Adresní pole obsahuje:
  - Šestnáctkově 0F pro unicast.
  - Šestnáctkově 8F pro multicast (oběžník).
- ◆ Řídicí pole obsahuje nulu (00 šestnáctkově).
- ◆ Naprosto zásadní změnou je, že za řídicím polem následuje ještě dvojbajtové pole nesoucí identifikaci protokolu vyšší vrstvy (viz protokol PPP nebo protokol Ethernet II). Toto pole tak umožňuje protokolu cHDLC jednou linkou přenášet více protokolů vyšší vrstvy současně (např. mixovat datagramy protokolů IPv4 a IPv6).
- ◆ Dále zavádí protokol *Serial Line Address Resolution Protocol* (SLARP), který je obdobou protokolu ARP, ale pro sériové linky.

## PPP

Téměř každý uživatel Internetu se setká s protokolem PPP. S největší pravděpodobností je to totiž ten protokol, který vás potrápí, když chcete PC připojit pomocí telefonického připojení k Internetu. Protokol PPP využívá rámce tvaru protokolu HDLC, nevyužívá však zdaleka všechny možnosti protokolu HDLC. Snad o to více zavádí dalších vlastností. Základní vlastnosti protokolu PPP jsou:

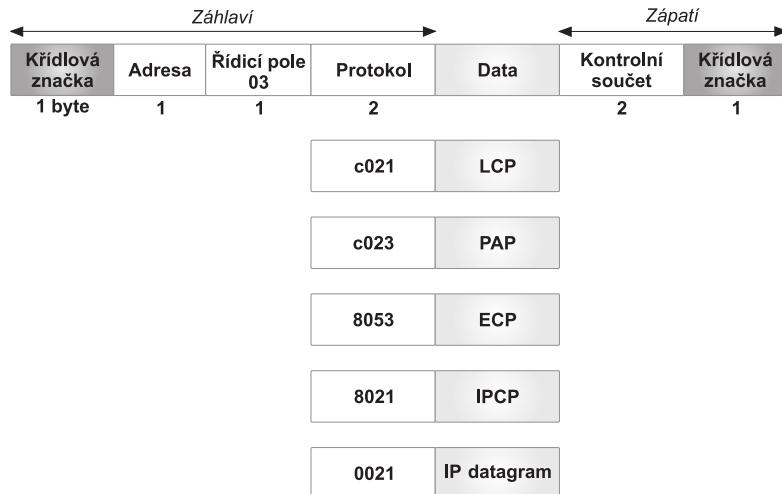
- ◆ Na fyzické úrovni je schopen používat rozhraní podle doporučení V.24, V.35 atp. Nevyžaduje žádné řídicí signály (RTS, CTS, DCD, DTR atp.). Řídicí signály však mohou být využity pro zvýšení efektivity.
- ◆ Může používat jak asynchronní, tak bitově či znakově synchronní přenos dat.
- ◆ Pro asynchronní přenos použije 1 start-bit, 8 datových bitů a 1 stop-bit (bez parity).
- ◆ Vyžaduje plně duplexní dvojbodové spoje (*point-to-point*), které mohou být pevné i komutované.
- ◆ Využívá zpravidla 16 nebo 32 bitů pro kontrolní součet, aby mohl zjistit, zda nebyl rámec během přenosu poškozen.
- ◆ Cílem protokolu PPP je umožnit po jedné lince přenos více síťových protokolů současně (mixovat protokoly). Nepoužívá I-rámce, ale přenos provádí pouze pomocí U-rámců. Nemůže tedy použít číslování rámců, a tedy ani možnost opakování rámce v případě zjištění chybného rámce.
- ◆ Na počátku datového pole umisťuje osmi nebo šestnáctibitovou identifikaci přenášeného síťového protokolu.

Protokol PPP specifikuje RFC-1661. Tvar rámce protokolu PPP v zapouzdření na způsob protokolu HDLC specifikuje RFC-1662. Na obr. 4.9 jsou znázorněny tvary rámce protokolu PPP. Pole protokol specifikuje přenášený protokol.



**Poznámka:** Kromě zapouzdření rámce protokolu PPP na způsob protokolu HDLC jsou specifikovány i jiné způsoby zapouzdření, např. zapouzdření FrameRelay (str. 109) nebo zapouzdření Ethernet (viz RFC-2516).

Vraťme se však k zapouzdření HDLC (obr. 4.9 – porovnejte s obr. 4.3). Rámec protokolu PPP obsahuje v poli „Adresa“ hodnotu ff<sub>16</sub> (oběžník) a v řídicím poli vždy 03<sub>16</sub> (U-rámce s nastaveným bitem



**Obrázek 4.9:** Rámce protokolu PPP v zapouzdření HDLC

P/F na 0). Pokud se na lince vyskytují rámce pouze s touto adresou a řídicím polem, pak oba konce linky mohou použít kompresi *Address-and-Control-Field-Compression*. Při této kompresi se prostě obě pole vypustí.

Nyní pár slov ke křídlové značce (*flag*): Křídlovou značkou je uvozen i ukončen každý rámec protokolu PPP. Křídlová značka obsahuje binárně 0111 1110, tj. znak  $7e_{16}$ . Co ale když je znak 7e přenášen v datech? U binárně synchronních linek jsme si popsali techniku *bit stuffing*. Jenže co si počít u asynchronních linek?

Pro asynchronní spoje (též pro znakově synchronní linky) se použijí Esc-sekvence (analogicky jako u protokolu SLIP). Znak 7e se nahradí dvojicí 7d 5e. A znak 7d se nahradí dvojicí 7d 5d.

Implicitně se uvozuji escape sekvencí 7d i všechny řídicí znaky ASCII (tj. znaky z tabulky ASCII s kódem desítkově menším než 32). Navíc se k hodnotě těchto znaků připočte desítkově 32 (tj. 20 šestnáctkově). Např. místo znaku 03 se přenáší  $7e23_{16}$ . Takže ani terminálový ovladač nemůže přenášeným znakům uškodit tím, že by je chybě interpretoval např. jako zvonek, BACKSPACE atp. Možná vás zarazilo slovo implicitně v úvodu tohoto odstavce. Ale obě stanice se mohou pomocí volby *Async-Control-Character-Map* (ACCM) dohodnout na tabulce znaků, které se budou uvozovat sekvencí escape a které nikoliv.

Na binárně synchronní lince se escape sekvence nepoužívají. Ale není tomu tak vždy. S sekvencemi escape je možné se setkat i na binárně synchronních linkách. Proč? V případě, že je třeba konvertovat přenos z asynchronních na bitově synchronní (a naopak), sekvence escape pak přejdou z asynchronního přenosu do synchronního jako znaky. Při odpovědi musí naopak synchronní strana obohatit synchronní data o sekvence escape, aby bylo po konverzi možné komunikovat s protějškem. Takže i synchronní stanice mohou použít volbu *Async - Control - Character - Map*. Takováto konverze se používá, např. když máme synchronní linku připojenou do PC a používáme autosynchronní modemu, tj. komunikace vychází z PC asynchronně a v autosynchronním modemu se mění na synchronní.

Součástí protokolu PPP je pět typů služebních protokolů:

- ◆ Protokol LCP sloužící k navázání spojení.
- ◆ Protokoly sloužící pro autentizaci (protokoly PAP, CHAP, EAP apod.).
- ◆ Protokol pro zpětné volání.
- ◆ Další protokoly: protokoly pro šifrování přenosu, pro komprimaci dat, pro rozložení zátěže do více linek (*Multilink Protocol*), pro dynamické rozšiřování přenosového pásma do více linek (*Bandwidth Allocation Protocol*) atp.
- ◆ Skupina protokolů NCP. Důležité je množné číslo. Každý síťový protokol, který bude využívat linkový protokol PPP, má definovanou vlastní normu pro protokol typu NCP. Součástí této normy je vždy i číslo protokolu, které se použije v poli protokol rámce, a to jak pro příslušný protokol NCP (číslo začíná číslicí 8), tak i pro datové rámce (číslo začíná číslicí 0). Mj. máme následující protokoly NCP:
  - **Protokol IPCP** (číslo protokolu  $8021_{16}$ ) je variantou protokolu NCP pro IP protokol verze 4, IPCP je specifikován RFC-1332. Datové rámce používají v poli protokol hodnotu  $0021_{16}$ .
  - **Protokol IPv6CP** (číslo protokolu  $8057_{16}$ ) je variantou protokolu NCP pro IP protokol verze 6 (RFC-2023). Datové rámce přenášené protokolem IP verze 6 používají číslo protokolu  $0057_{16}$ .
  - **Protokol SNACP** (číslo protokolu  $804d_{16}$ ), tj. protokol NCP pro IBM SNA (RFC-2043). Datové rámce používají číslo protokolu  $004d$ .
  - **Protokol DNCP** (číslo protokolu  $8027_{16}$ ), tj. protokol NCP pro DECnet Phase IV (RFC-1762). Datové rámce používají číslo protokolu  $0027_{16}$ .
  - **Protokol IPXCP** (číslo protokolu  $802b_{16}$ ), tj. protokol NCP pro IPX (RFC-1552). Datové rámce používají číslo protokolu  $002b_{16}$ .
  - **Protokol OSINLCP** (číslo protokolu  $8023_{16}$ ), tj. protokol NCP pro OSI protokoly, tj. např. protokoly ES-IS, IS-IS atd. (RFC-1377). Datové rámce používají číslo protokolu  $0023_{16}$ .

## Vytáčení telefonní linky

Ještě předtím, než začneme popisovat jednotlivé protokoly z rodiny protokolů PPP, zastavíme se u vlastního vytáčení linky. Mnohdy nastávají totiž praktické problémy ještě před tím, než je samotné spojení protokolem PPP navázáno.

Ve Windows jsme měli volby, které umožňovaly spustit tzv. terminálové okno. Přitom terminálové okno bylo možné spustit ve dvou okamžicích:

- ◆ Předtím než modem začne vytáčet. V tomto okamžiku může uživatel do terminálového okna zadávat příkazy pro ovládání modemu (např. příkaz AT). Uživatel např. může „ručně“ zadat vytáčení telefonního čísla (např. ATDT123456789 v případě telefonního čísla 123456789) a sledovat činnost modemu.
- ◆ Když je telefonní okruh sestaven, tj. když jsou modemy na obou stranách přepnuty do datového režimu. Uživatel se v tomto okamžiku může ocitnout ve dvou situacích:
  1. Druhá strana očekává pouze komunikaci protokolu PPP (na terminálovém okně se uživateli zpravidla zobrazuje „rozsypáný čaj“, tj. zobrazují se mu tam rámce s příkazem Configure-Request, které mají binární tvar). S tímto případem se setkáváme např. při navazování komunikace se servery Windows. Jedinou rozumnou akcí je v takovém případě uzavřít terminálové okno a dále pokračovat v komunikaci protokolem PPP.

2. Druhá strana očekává terminálový dialog. Tato situace je častá např. na serverech s operačním systémem UNIX nebo boxech CISCO. Uživatel je vyzván k zadání jména uživatele a hesla. Opět mohou nastat dva případy:
- ◆ Uživatel zadá jméno uživatele, který může interaktivně pracovat v daném systému, pak se objeví příkazový řádek tohoto operačního systému a uživatel může pracovat tak, jak je běžně zvyklý pracovat v tomto systému. Může také příkazem `ppp` aktivovat protokol PPP (opět se objeví „rozsypaný čaj“), uzavřít terminálové okno a přenechat svému počítači linku ke komunikaci protokolem PPP.
  - ◆ Uživatel nemůže interaktivně pracovat v uvedeném operačním systému – po zadání správného hesla se rovnou aktivuje protokol PPP. Pak uživateli po jeho autentizaci nezbývá než ukončit terminálové okno a pokračovat v komunikaci protokolem PPP.

V obou případech došlo k autentizaci uživatele. Avšak k autentizaci za využití terminálového dialodu, tj. nikoliv k autentizaci prostředky protokolu PPP.



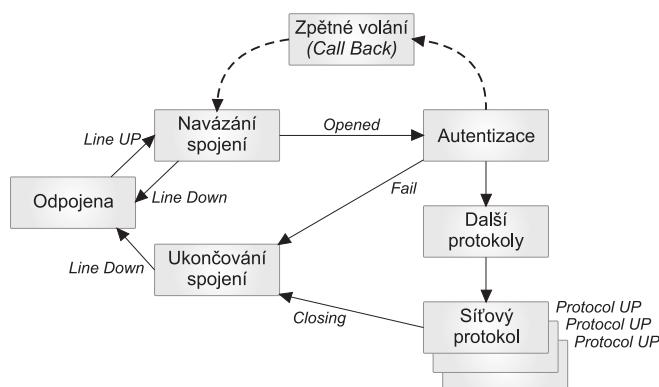
**Poznámka:** Ve Windows Vista došlo k zásadní změně tím, že program HyperTerminal už není součástí Windows. Pokud chceme komunikovat s modemem před vytvořením (před sestavením okruhu), pak musíme použít jiný program – např. putty, který se často používá jako klient protokolu SSH. Tento program má však i volbu serial, ve které je možné zvolit příslušný COM-port. A začít do něj zadávat „modemové“ příkazy AT.



**Poznámka:** Ve Windows Vista stále můžeme otevírat terminálové okno po sestavení telefonního okruhu nastavením ve vlastnostech telefonického připojení (viz obr. 4.13 – prostřední okno).

## Protokol LCP

Protokol LCP je určen pro navázání spojení, ukončení spojení, k dohodě na autentizačním algoritmu apod. LCP je společný protokol (na rozdíl od protokolů NCP) pro všechny síťové protokoly a používá se ještě před tím, než se vůbec uvažuje o tom, jaký síťový protokol na lince poběží. Linka se nachází postupně ve fázi navazování spojení, autentizace, síťový protokol a ukončování spojení, jak je znázorněno na obr. 4.10.



Obrázek 4.10: Jednotlivé fáze linky v protokolu PPP

**Linka odpojena** je fáze, u které se vždy začíná a končí. Když dojde k nějaké externí události (např. modemy ztratí mezi sebou spojení nebo síťový administrátor vydá příkaz k ukončení spojení), přechází linka do této fáze.

Z fáze **linka odpojena** se přechází do fáze **navazování spojení**, jejímž cílem je ustanovit konektivitu mezi oběma konci spojení. Navazování spojení se provádí výměnou konfiguračních paketů. Datové pakety se během této fáze nepřenášejí. V případě výskytu datového paketu během navazování spojení se takový paket zahazuje.

**Autentizace** je fáze, kdy klient prokazuje svou totožnost. Slovo klient jsem použil záměrně. Asi jste si položili otázku, kdo je to klient? Klientem je ta strana (stanice), která je vyzvána k prokázání své totožnosti. Po prokázání totožnosti jedné stanice si mohou stanice svou roli vyměnit a k prokázání své totožnosti může být vyzvána druhá strana. V praxi většinou prokazuje svou totožnost jen jedna strana (např. uživatel PC proti poskytovateli Internetu). Fáze je nepovinná, tj. může být přeskočena.

Protokol LCP nepopisuje žádné autentizační algoritmy, pouze přenáší data používaná k vlastnímu prokazování totožnosti, která jsou pak následně využita autentizačními protokoly. Jako autentizační protokol se používá např. protokol PAP, protokol CHAP atp. Zpravidla je možná i terminálová autentizace.

Během autentizace opět nemohou být ještě přenášeny datové pakety (síťový protokol).

Po fázi autentizace se mohou oba konce pomocí **protokolu zpětného volání** dohodnout na zpětném volání, po kterém následuje opět navázání linky a případně znova i fáze autentizace.

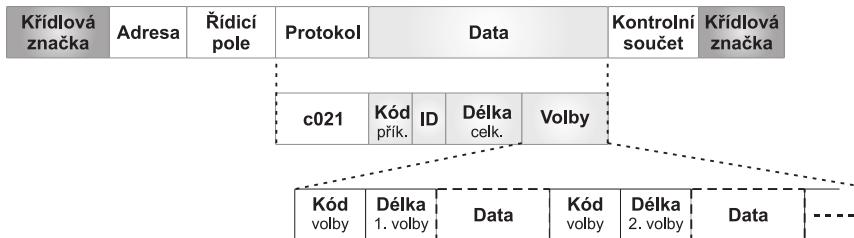
Jestliže je spojení navázáno, mohou být aktivovány další protokoly. Nejčastěji se však aktivují těsně po fázi autentizace. Pomocí těchto protokolů se mohou oba konce např. dohodnout na šifrování přenosu, komprimaci dat, rozložení zátěže do více linek (*Multilink Protocol*), na dynamickém rozširování přenosového pásma do více linek (*Bandwidth Allocation Protocol*) atp.

Fáze, která je na obr. 4.10 označena jako **síťový protokol**, v sobě může obsahovat celou řadu kroků. V tomto okamžiku přicházejí ke slovu jednotlivé protokoly NCP. Každý síťový protokol, který chce linku využívat, si ji musí přivést pomocí svého protokolu NCP do otevřeného stavu pro tento protokol. Pokud se objeví datové pakety síťového protokolu, pro který není linka otevřena, pak se tyto pakety zahodí.

Např. mají-li se na lince přenášet pakety protokolu IP verze 4 a protokolu IS-IS, musí se linka v této fázi otevřít dvakrát, jednou pomocí protokolu IPCP a podruhé pomocí OSINLCP. Po otevření linky pro konkrétní síťový protokol se teprve mohou začít přenášet data konkrétního síťového protokolu. Linka může být otevřena pro více síťových protokolů současně.

Poslední fází je fáze **ukončování spojení**. Během této fáze jsou všechny jiné pakety než protokolu LCP již zahazovány. Fyzické vrstvě je signalizováno ukončení spojení. Fyzická vrstva může reagovat např. zavěšením komutované linky.

Formát rámce protokolu LCP je vyjádřen na následujícím obrázku 4.11. Je třeba zdůraznit, že všechny řídicí protokoly z rodiny protokolů PPP budou mít obdobné rámce skládající se z kódu příkazu a voleb (tj. parametrů příkazu).



Obrázek 4.11: Formát rámce protokolu LCP

Osmibitové pole Kód specifikuje typ příkazu (resp. odpovědi) protokolu LCP:

Kód	Název (anglicky)	Význam
1	Configure-Request	Konfigurační paket nesoucí požadavky na změnu implicitních parametrů linky.
2	Configure-Ack	Konfigurační paket s kladným potvrzením požadavků na změnu implicitních parametrů linky, tj. všechny požadované změny parametrů jsou rozeznány a akceptovány.
3	Configure-Nak	Konfigurační paket s odpovědí. Protější strana sice všechny volby rozeznala, ale neakceptuje všechny volby. Ty, které neakceptuje, jsou v tomto paketu specifikovány s případným návrhem na jinou hodnotu.
4	Configure-Reject	Konfigurační paket odmítající všechny požadavky. Může být důsledkem i chybného kódu požadavku (kódu volby), tj. nerozpoznaného požadavku.
5	Terminate-Request	Požadavek na ukončení spojení.
6	Terminate-Ack	Potvrzení požadavku na ukončení spojení.
7	Code-Reject	Odmítnutí požadavku z důvodu neznámého kódu příkazu. Může být způsobeno i tím, že protější stanice používá jinou verzi protokolu.
8	Protokol-Reject	Protější strana nepodporuje uvedený protokol.
9	Echo-Request	Podpora testovací smyčky na linkové úrovni.
10	Echo-Reply	Povinná odpověď na Echo-Request.
11	Discard-Request	Zahod' tento paket. Používá se pro testování linky při zátěži, tj. odesílatel generuje pomocí téhoto paketu umělou zátěž linky.
12	Identification	Jedná se o rozšiřující volbu, která obsahuje čtyřbajtové „kouzelné“ číslo a text proměnné délky. Význam „kouzelného“ čísla je obdobný jako u volby Magic-Number (viz níže). Text může obsahovat např. typ hardwaru, verzi softwaru atp.

Osmibitové pole **ID** je identifikace požadavku. Odesílatel do tohoto pole vygeneruje identifikaci požadavku a adresát ji zkopiuje do své odpovědi. Pomocí tohoto pole se určuje příslušnost odpovědi k danému požadavku.

Šestnáctibitové pole **Délka** obsahuje číslo udávající součet délek polí: kód, ID, délka a volby.

Pole **Volby** obsahuje jednotlivé požadavky (resp. odpovědi) na změnu implicitních parametrů linky. Toto pole se skládá z jedné nebo více voleb. Jednotlivé volby jsou ukládány sekvenčně za sebou, jak je znázorněno na obr. 4.11.

Následující tabulka uvádí některé volby. Pole volba a délka jsou osmibitová.

Kód	Název volby (anglicky)	Význam volby
1	Maximum-Receive-Unit	Pomocí této volby se stanice mohou dohodnout na délce rámce (MTU) delší než 1 500 bajtů (všechny stanice jsou povinny přenášet rámce délky alespoň 1 500 bajtů). Datová část volby pak specifikuje délku rámce.
2	ACCM (Async Control Character Map)	Ve čtyřech bajtech datové části volby, tj. 32 bitech, je specifikováno, které řídicí znaky (prvních 32 znaků z tabulky ASCII) se uvodí sekvencí escape ( $7d_{16}$ ) a hodnota samotného znaku se zvětší o 32 (tj. o $20_{16}$ ). Je-li např. 5. bit ve 32bitovém řetězci nastaven na 1, pak se místo znaku $05_{16}$ bude přenášet $7d25_{16}$ . Pokud nehrozí dezinterpretace, pak se zpravidla obě strany dohodnou na ACCM skladajícím se ze samých nul.
3	Authentication-Protocol	Požadavek na autentizaci konkrétním autentizačním protokolem – např. pro protokol PAP obsahuje datová část volby šestnáctkové c023, pro protokol CHAP c223, pro EAP c227 atd. V případě protokolu CHAP volba kromě identifikace protokolu (c223) nese ještě identifikátor jednocestné funkce: 05 pro algoritmus MD-5; 8016 v případě protokolu MS CHAP verze 1 (používá algoritmus MD-4); 8116 v případě protokolu MS CHAP verze 2.
5	Magic-Number	Obsahuje náhodné čtyřbajtové „kouzelné číslo“ sloužící k detekci zpětné vazby („smyčky“) na lince. Pokud příjemce obdrží Configure-Request s vyplněným kouzelným číslem, porovnává, jestli předchozí Configure-Request náhodou neměl shodné kouzelné číslo. Pokud ano, jedná se pravděpodobně o zpětnou vazbu. V takovém případě do odpovědi vygeneruje jiné kouzelné číslo. V případě, že se nejedná o zpětnou vazbu, zkópiuje do odpovědi původní kouzelné číslo. Některé implementace PPP opakují první rámec na lince tak dlouho, než dostanou první odpověď. Uvedením shodného kouzelného čísla signalizují, že se jedná o opakování rámce.
7	Protocol-Field-Compression	„Komprese vedoucích nul v čísle protokolu.“ Rámec protokolu PPP obsahuje dvoubajtové pole specifikující typ protokolu (např. 0021 pro IP datagramy). Při zahájení komunikace se musí toto pole používat vždy dvoubajtové. Po potvrzení Protocol-Field-Compression se používá toto pole jenom jednobajtové, tj. vypustí se první bajt, pokud obsahuje nulu.
8	Address-and-Control Field-Compression	V rámci protokolu PPP je vždy v poli adresa hodnota ff a v řídicím poli hodnota 03. Po potvrzení Address-and-Control-Field-Compression odesílatel tato pole vypouští a příjemce je zpět automaticky doplňuje.
13	Call-Back	Požadavek na zpětné volání. Po potvrzení tohoto požadavku by se měla provést autentizační fáze a dialog protokolu LCP by měl spět k zavěšení linky, po kterém naváže spojení druhý konec linky. Datová část této volby nese typ telefonního čísla (typ telefonní adresy). Zpravidla obsahuje hodnotu 06, tj. „bude vyjednáno až protokolem CBCP“ (viz RFC-1700).
17	Multilink-MRRU	Potvrzením této volby se signalizuje, že systém má implemetován protokol MP. Datová část volby obsahuje maximální velikost paketu, který je systém schopen rekonstruovat (viz protokol MP).
18	SSNH	Požadavek na zkrácení záhlaví MP ze čtyřech na dva bajty (viz protokol MP).
19	Multilink-Endpoint-Discriminator	Jednoznačná identifikace počítače jako konce spojení pro protokol MP (viz protokol MP).
23	Link-Discriminator for BAP	Jednoznačná identifikace linky v rámci jednoho počítače (viz protokol BAP).

## Příklad

Význam jednotlivých polí snadno pochopíme na následujícím výpisu rámce v programu Wireshark, který nese příkaz Configure-request, kterým server Windows specifikuje své požadavky na spojení pomocí vytáčené linky:

```
Ethernet II
    Destination: 20:53:45:4e:44:0e ( SEND_ )
    Source: 20:53:45:4e:44:0e ( SEND_ )
    Type: PPP Link Control Protocol (0xc021)
PPP Link Control Protocol
    Code: Configuration Request (0x01)
    Identifier: 0x00
    Length: 44
    Options: (40 bytes)
        Async Control Character Map: 0x00000000 (None)
        Authentication protocol: 4 bytes
            Authentication protocol: Password Authentication Protocol (0xc023)
        Magic number: 0x06147a40
        Protocol field compression
        Address/control field compression
        Callback: 3 bytes
            Operation: Location is determined during CBCP negotiation (0x06)
        Multilink MRRU: 1614
        Multilink endpoint discriminator: 9 bytes
            Class: IEEE 802.1 globally assigned MAC address (3)
            Address: 00:d0:59:0f:eb:1d
        Link discriminator for BAP: 0x000d
```

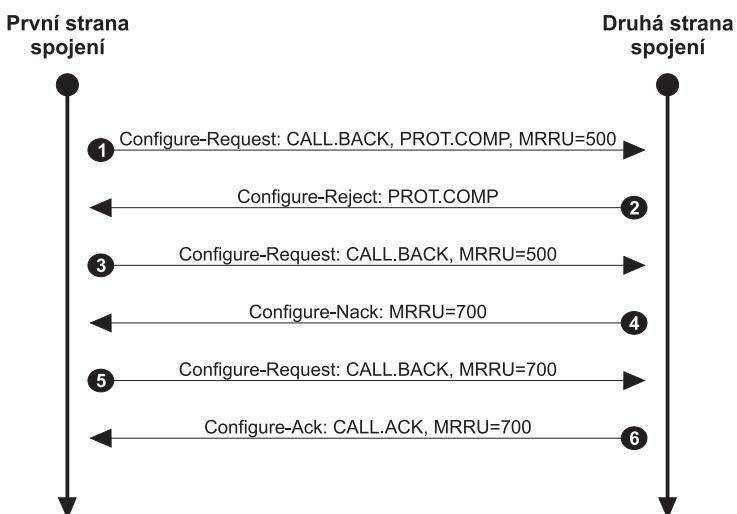
Předchozí výpis obsahuje rámec protokolu PPP, který Windows „přepočetly“ na tvar rámce Ethernet II. To poznáme z ethernetové adresy „SEND\_“, kterou jsem získal konverzí hexadecimálního řetězce 20:53:45:4e:44:0e do textového tvaru. K významu jednotlivých polí:

- ◆ Code – kódu příkazu (1=Configuration Request).
- ◆ Identifier – identifikátor (=0).
- ◆ Length – délka (44 bajtů).
- ◆ Options – volby:
  - Async Control Character Map: Tato volba specifikuje, které z prvních 32 znaků tabulky ASCII se mají uvozovat sekvencí escape. Jelikož v našem případě datová část této volby obsahuje samé nuly, nebudou se uvozovat žádné znaky. To znamená, že se jedná o linku, která se nepoužívá např. jako terminálová, proto nehrází dezinterpretace řídicích znaků.
  - Authentication protocol: „Požaduji od tebe autentizaci protokolem PAP.“
  - Magic number: Tato volba obsahuje „kouzelné“ číslo.
  - Protocol field compression: Vypuštění vedoucích nul v číslech protokolů.
  - Address/control field compression: „Požaduji Address-and-Control-Field-Compression“. Tj. požaduji nepoužívání těchto polí.
  - Callback: „Požaduji aktivovat protokol zpětného volání.“
  - Multilink MRRU: „Podporuješ protokol MP? Jak velký paket jsi schopen zpětně sestavit?“

- **Multilink endpoint discriminator:** „Moje jednoznačná identifikace konce spojení (počítače) v protokolu MP je:“ Hodnota identifikátoru na tomto výpisu není zobrazena – je však v šestnáctkovém výpisu.
- **Link discriminator for BAP:** Jednoznačná identifikace linky v rámci počítače.

Na obr. 4.12 je znázorněn demonstrativní (smyšlený) příklad dialogu protokolem LCP:

1. První strana zašle příkazem Configure-Request své požadavky na spojení.
2. Druhá strana nezná volbu PROT.COMP (Protocol field compression), což signalizuje příkazem Configure-Reject s volbou Protocol field compression.
3. První strana proto odesílá příkaz, o kterém si myslí, že mu již druhá strana rozumí.
4. Druhá strana mu rozumí, ale preferuje volbu MRRU=700 (Multilink MRRU), což signalizuje příkazem Configure-Nack s volbou „Multilink MRRU“=700.
5. První strana odesílá příkaz s volbami, o kterých si myslí, že budou pro druhou stranu přijatelné.
6. Druhá strana vše kladně potvrzuje příkazem Configure-Ack.



Obrázek 4.12: Příklad dialogu protokolu LCP

## Autentizace

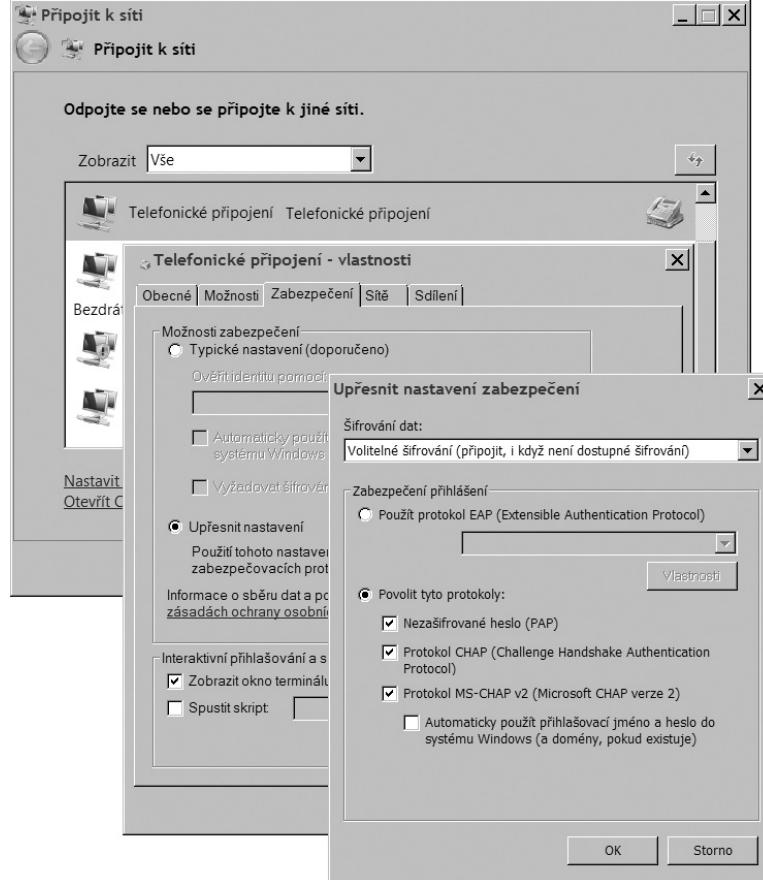
Totožnost lze v protokolu PPP prokazovat dvojím způsobem (nepovažujeme-li za třetí možnost eventualitu, že autentizace je zcela vynechána):

- ◆ **Terminálovým dialogem.** Tato možnost byla popsána v části 4.4.1 – Vytáčení telefonní linky.
- ◆ Autentizačními protokoly z rodiny protokolů PPP, zejména protokoly:
  - **Password Authentication Protocol (PAP).** Tento protokol je obdobou autentizace pomocí terminálového dialogu. To znamená, že uživatel prokazuje svou totožnost také pomocí jména

uživatele a hesla, avšak obě hodnoty nevkládá přímo do terminálové linky, ale do protokolu PAP (RFC-1334).

- **Challenge Handshake Authentication Protocol (CHAP).** Tento protokol specifikuje RFC-1994. Je považován za dokonalejší. Oba konce spojení sdílí společné tajemství. Stanice, která autentizaci inicializuje, vygeneruje náhodný řetězec jako dotaz (challenge), který odesle druhé straně. Druhá strana tento řetězec sloučí se sdíleným tajemstvím a z výsledku spočte kontrolní součet jednocestným algoritmem (např. MD-5). Výsledek tvořící „jednorázové přístupové heslo“ odesle zpět. Stanice, která autentizaci inicializovala, tak obdržela jednorázové heslo. Jelikož však zná sdílené tajemství a challenge, je schopna také spočítat jednorázové heslo, které porovná s jednorázovým heslem, které obdržela. Jsou-li obě stejná, potvrdí protějšku úspěšný výsledek autentizace. Firma Microsoft pak zavedla zdokonalení protokolu CHAP pod označením **MS CHAP verze 1 a 2**.

- **Protokolem EAP.**



Obrázek 4.13: Konfigurace autentizačních protokolů a šifrování ve Windows Vista

### Protokol PAP

Protokol PAP je jednoduchý protokol. Komunikace se skládá zpravidla ze dvou paketů. Autentizovaný se autentizuje příkazem Authenticate-Request a druhý konec buď autentizaci potvrdí příkazem Authenticate-Ack, nebo odmítne příkazem Authenticate-Nak.

Kód	Název (anglicky)	Význam volby
1	Authenticate-Request	Paket, kterým začíná komunikace protokolem PAP. Nese volbu s kódem 1, která obsahuje jméno a heslo.
2	Authenticate-Ack	Kladné potvrzení autentizace.
3	Authenticate-Nak	Autentizace selhala.

### Příklad

Příklad dialogu protokolu PAP – odchyceno programem Wireshark (samotné heslo pak uvidíme ve spodním rámu po poklepnutí na položku Password – v následujícím výpisu není vidět):

```
+Frame 9 (41 bytes on wire, 41 bytes captured)
+Ethernet II, Src: 20:52:45:43:56:0e, Dst: 20:52:45:43:56:0e
-PPP Password Authentication Protocol
  Code: Authenticate-Request (0x01)
  Identifier: 0x05
  Length: 27
  -Data (23 bytes)
    -Peer ID length: 13 bytes
      Peer-ID (13 bytes)
    -Password length: 8 bytes
      Password (8 bytes)
```

### Protokoly typu CHAP

Výhodou protokolu CHAP je skutečnost, že oba konce sdílí stejné tajemství – je tak snadno možné provádět oboustranně autentizaci. Sdílení tajemství je současně nevýhodou protokolu CHAP, protože nelze zabránit zneužití tohoto tajemství druhou stranou (na rozdíl od autentizace heslem, kde druhá strana má přístup pouze k heslu znehodnocenému jednocestnou funkcí).

Protokol CHAP komunikuje ve třech krocích. V prvním kroku příkazem Challenge je autentizovanému odeslána výzva obsahující náhodný řetězec. Autentizovaný prožene sdílené tajemství spojené s výzvou jednocestnou funkcí (např. algoritmem MD-5). Výsledek vloží do odpovědi (Response), kterou odešle. Reakcí je pak potvrzení autentizace (Success) nebo zamítnutí (Failure).

Kód	Název (anglicky)	Význam volby
1	Challenge	Nese volbu výzva (challenge).
2	Response	Nese volbu odpověď (response).
3	Success	Kladné potvrzení autentizace.
4	Failure	Autentizace selhala.

Nevýhodou protokolu CHAP je, že oba konce musí mít k dispozici sdílené tajemství v otevřeném tvaru. To ovšem není zcela běžné v mnoha operačních systémech. V běžných operačních systémech

si uživatel určí heslo, ale toto heslo není v otevřeném tvaru uloženo v databázi uživatelů operačního systému. Operační systém heslo „znehodnotí“ jednocestnou funkcí a teprve výsledek této operace ukládá do databáze uživatelů. V případě autentizace uživatele uživatel předá systému heslo, systém na heslo aplikuje zmíněnou jednocestnou funkci a až výsledek této operace se porovnává s údaji v databázi uživatelů.

Microsoft zavedl modifikovaný protokol CHAP označovaný jako MS CHAP verze 1 (RFC-2433). V operačním systému jsou udržována hesla, na která byla aplikována jednocestná funkce MD-4. Proto na uživatelské heslo je na straně klienta také nejprve aplikován algoritmus MD-4. Uživatelské heslo znehodnocené algoritmem MD-4 tak tvoří sdílené tajemství. Software klienta pak aplikuje opět algoritmus MD-4 na sdílené tajemství spojené s náhodnou výzvou.

Základní výhodou protokolu MS CHAP verze 1 byla zpětná kompatibilita se systémy LAN Manager. Nemohlo však dojít k oboustranné autentizaci, a zejména byla potíž se šifrováním přenášených dat. Více se dozvím v kapitole o dalších protokolech, kterými jsou např. protokoly zajišťující šifrování. Přitom šifrovací klíče se zpravidla odvozují od dat vyměněných již ve fázi autentizace. Protokol MS CHAPv1 umožňuje odvodit tyto klíče pouze od sdíleného tajemství, tj. po dobu, po kterou uživatel používá stejně heslo, používá i stejně šifrovací klíče. Tyto neduhy odstranil protokol MS CHAP verze 2 (RFC-2759), který již ale není zpětně kompatibilní se systémy typu LAN Manager.

Připomeňme, že to, který z protokolů CHAP (CHAP, MS CHAP verze 1, MS CHAP verze 2 atd.) se bude používat, se vyjedná již ve fázi navazování spojení protokolem LCP. Přičemž volbou Authentication-Protocol se v případě protokolů typu CHAP specifikují dvě hodnoty. První hodnota specifikuje, že se jedná o autentizační protokol typu CHAP, a druhá hodnota specifikuje konkrétní autentizační protokol.

## Protokol EAP

I když z hlediska tvaru rámce je protokol EAP velice podobný protokolu CHAP, přichází se zcela jinou filozofií. Zatímco u protokolů PAP, CHAP či MS CHAP docházelo k vyjednání autentizačního protokolu během navazování spojení protokolem LCP, v případě protokolu EAP dojde v této fázi pouze k dohodě na tom, že se použije protokol EAP. Protokol EAP specifikuje RFC-2284.

Skutečnost, že mezi oběma konci spojení dojde k dohodě na použití protokolu EAP, ještě nepředurčuje použití konkrétního autentizačního algoritmu – ten se vyjedná až protokolem EAP.

Protokol EAP tak umožňuje používat libovolný autentizační mechanismus, stačí jej pouze implementovat na obou stranách spojení. Pokud se použije protokol EAP, skládá se fáze autentizace jednak z dohody na konkrétním autentizačním algoritmu (tzv. autentizačním schématu) a teprve následně na vlastní autentizaci.

Je tak možné implementovat autentizační schéma využívající nejrůznější autentizační kalkulátory pro generování jednorázových hesel; je možné využívat autentizaci protokolem TLS (RFC-2716) atp. Minimálně by mělo být implementováno schéma EAP-MD5, které je obdobou protokolu CHAP:

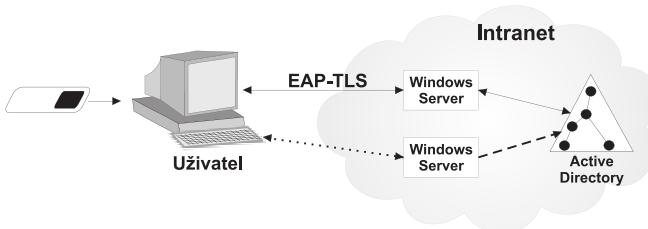
1. Strana, která ověřuje totožnost druhé strany, zašle zprávu EAP-Request, ve které žádá druhou stranu o prokázání její totožnosti.
2. Pokud autentizovaná strana souhlasí s touto autentizací, signalizuje to ve zprávě EAP-Response.
3. Strana, která ověřuje totožnost druhé strany, zašle zprávu EAP-Request výzvu (Challenge).

4. Autentizovaná strana spojí sdílené tajemství („heslo“) s výzvou, na což aplikuje jednocestnou funkci MD-5. Výsledek vloží do odpovědi EAP-Response.
5. Strana, která ověřuje totožnost druhé strany, potvrdí/zamítne autentizaci zprávou EAP-Success/EAP-Failure.

Velice zajímavé je autentizační schéma využívající protokol TLS, schéma EAP-TLS (protokoly SSL i TLS jsou podrobně popsány v publikaci „Velký průvodce PKI a technologií elektronického podpisu“). Toto schéma využívá autentizaci serveru na základě certifikátu serveru a autentizaci uživatele na základě osobního certifikátu uživatele.

Windows (od 2000 výše) přímo podporují využívání čipových karet, na kterých je uložen soukromý klíč příslušející k veřejnému klíči z certifikátu. Využití čipových karet a schématu EAP-TLS pak přináší bezpečnostně řádově větší možnosti umožňující přistupovat např. i přes telefonní linku na servery v intranetu. Prakticky to vypadá následovně: uživatel spustí svůj notebook, avšak není přihlášen do systému. Uživatel vloží čipovou kartu do počítače a zadá PIN. Systém za využití protokolu PPP naváže telefonické spojení se serverem. Autentizace vůči serveru proběhne protokolem EAP-TLS. V případě kladné autentizace je uživatel přihlášen do systému na notebooku a má navázáno spojení se serverem. Stejnou čipovou kartu může uživatel pak použít i k přihlášení k běžnému počítači na LAN v intranetu (to však již nemá nic společného s protokolem PPP). Další výhodou schématu EAP-TLS je, že se na obou stranách spojení vytvoří tzv. hlavní tajemství, ze kterého lze odštipovat šifrovací klíče, sdílená tajemství zajišťující integritu přenášených dat atd.

Nevýhodou schématu EAP-TLS je skutečnost, že v databázi uživatelů musí být udržovány i certifikáty uživatelů, tj. vyžaduje to mít vybudovanou infrastrukturu PKI. V případě Windows to pak konkrétně vyžaduje mít k dispozici ActiveDirectory. A tak je schéma EAP-TLS spíše určeno k sofistikovaným řešením, která se opírají o dobře propracovaný projekt (řešení není určeno pro samostatná PC!).



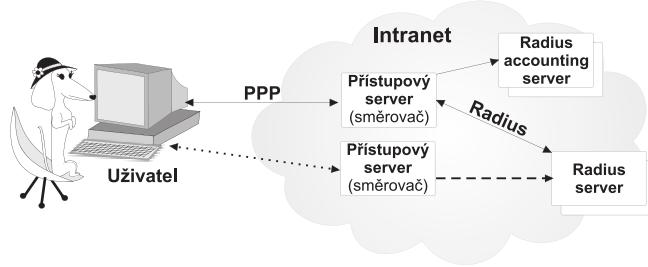
**Obrázek 4.14:** Uživatel se autentizuje za využití čipové karty a schématu EAP-TLS ve Windows

Promyšlený projekt může přinést bezpečnost připojování pracovníků na cestách (resp. z domova) do vnitřní sítě přes telefonické připojení, která může být i bezpečnější než použití VPN.

### Protokol Radius

Problém s autentizací spočívá v tom, že se klient bude chtít přihlašovat nikoliv stále na jeden přístupový server, ale na různé přístupové servery. Klasickým případem je připojení k poskytovatele Internetu, který má své přístupové body v různých městech. V takovém případě by autentizační informace musely být udržovány na každém přístupovém serveru. Myšlenka spočívá v centralizaci autentizačních informací. V síti je jeden nebo i více záložních serverů, které udržují autentizační informace o každém uživateli. Kromě autentizačních informací mohou být udržovány i konfigurační

informace (např. IP adresa uživatele, přístupové filtry atd.). Přístupový server pak vůči takovému serveru vystupuje jako klient, který požaduje službu: prověření autentizační odpovědi či poskytnutí IP adresy, kterou má protokolem IPCP předat uživateli atd. Jako protokol mezi přístupovým servery a serverem s autentizačními a konfiguračními informacemi se dnes často používá protokol RADIUS. Protokol RADIUS je aplikační protokol.



**Obrázek 4.15:** Protokoly Radius a Radius Accounting

Kromě protokolu RADIUS existuje ještě RADIUS Accounting Protocol. Tímto protokolem mohou přístupové servery předávat informace o připojování a odpojování uživatelů. RADIUS Accounting Server soustředí tyto informace, které pak mohou být využity např. pro zúčtování přístupu uživatelů do Internetu.



**Poznámka:** V prostředí Microsoft je Radius součástí IAS (Internet Authentication Service).

## Protokol řízení zpětného volání

Protokol řízení zpětného volání (*Call Back Control Protocol – CBCP*) slouží k dohodě na zpětném volání. V zásadě jsou dvě eventuality zpětného volání:

1. Klient vytvoří spojení se serverem, kterému předá telefonní číslo, na něž mu má server zpět zavolat. Server volá zpět, aniž by číslo kontroloval. Tato eventualita bezpečnostně nepřináší nic nového. Klient pouze nemusí platit telefonní poplatky. Vylepšením této eventuality je možnost, že server v databázi uživatelů ověří, jestli je klientem určené telefonní číslo přípustné.
2. Klient vytvoří spojení se serverem, kterému nepředá žádné telefonní číslo. Server telefonní číslo nalezne v databázi uživatelů (nezapomínejme, že vyjednávání zpětného volání proběhne až po autentizaci klienta, takže server ví, s kým má tu čest).

Je třeba zdůraznit, že zpětné volání, ve kterém server kontroluje telefonní číslo, na něž volá zpět, má velice silný bezpečnostní aspekt. V bezpečnostním slangu se označuje jako „dva zámky“. Případný útočník by totiž musel prolomit nejenom autentizační protokol (což v případě takového EAP-TLS se soukromým klíčem na čipové kartě je opravdu obtížné), ale musel by současně prolomit bezpečnostní systém telefonní ústředny, která by musela telefonní okruh sestavit k hackerovi místo ke klientovi. Pokud namítnete, že hacker se může napojit na telefonní svorkovnici na klientově domě nebo mu odcizit mobilní telefon, pak hacker musí překonat nikoliv bezpečnost telefonní ústředny, ale fyzickou ochranu klientova sídla (opět „druhý zámek“).

Protokol CBCP má paket stejného tvaru jako protokol LCP (pouze s tím rozdílem, že číslo protokolu není c021, ale c029). Používá příkazy CB-Request (kód 1), CB-Response (kód 2) a CB-Ack (kód 3).

Protokol CBCP využívá následující volby:

Kód	Název volby (anglicky)	Význam volby
1	No CallBack	Bez zpětného volání nebo zpětné volání na klientem zadané číslo.
2	CallBack to user-specified number	Zpětné volání na klientem zadané číslo.
3	CallBack to administrator-defined number	Zpětné volání na číslo uvedené v databázi uživatelů.
4	CallBack to any of a list of numbers	Server zavolá na jedno číslo ze seznamu čísel.

Dialog v případě, že klient zadává telefonní číslo:

- ◆ Server odešle příkaz CB-Request s volbou „No CallBack“.
- ◆ Klient odpoví příkazem CB-Response s volbou „CallBack to user-specified number“, která obsahuje tři údaje:
  1. Časové prodlení ve volání.
  2. Typ telefonního čísla, např. 01 pro klasické analogové síť (PSTN) nebo síť ISDN.
  3. Telefonní číslo.
- ◆ Server příkazem CB-Ack kladně stvrď výsledek.

Eventualitou též je, že klient odmítne zadat telefonní číslo a navázané spojení pokračuje dále bez zpětného volání.

Dialog v případě, že server použije telefonní číslo uvedené v databázi uživatelů:

- ◆ Server odešle příkaz CB-Request s volbou „CallBack to administrator-defined number“.
- ◆ Klient odpoví příkazem CB-Response s volbou „CallBack to administrator-defined number“.
- ◆ Server příkazem CB-Ack kladně stvrď výsledek.

## Další protokoly

### Multilink Protocol (MP)

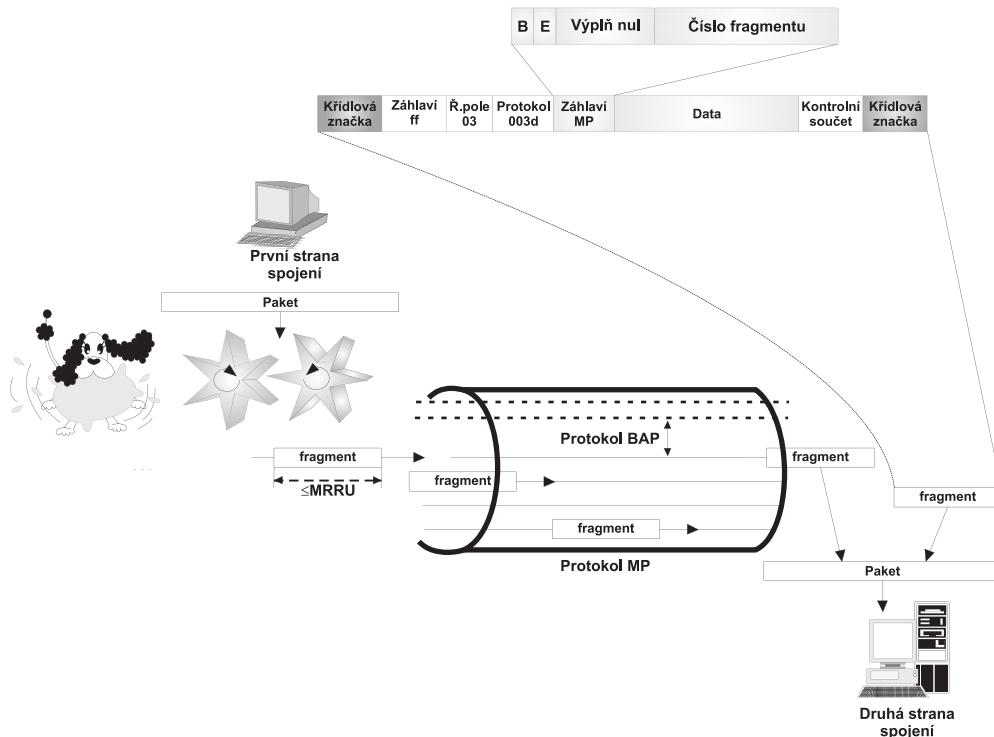
Protokol MP, specifikovaný v RFC-1990, popisuje, jak společně využít několik fyzických linek mezi dvěma počítači. Výsledkem je, že se několik linek spojí v jeden virtuální svazek – jeden tlustý drát. Přitom jednotlivé linky, které tvoří svazek, mohou být fyzicky realizovány na různých technologiích (vytáčené linky, pevné linky, X.25 atp.). Klasickým případem je spojení obou B-kanálů ISDN (*Basic Rate*).

Na obr. 4.16 je schematicky znázorněna komunikace v jednom směru. Na obrázku se předpokládá, že jednotlivé linky už navázaly protokolem LCP spojení a protokolem LCP též vyjednaly podporu protokolu MP. Ta se vyjedná tím, že si strany vzájemně potvrdí podporu volby MRRU, jejíž součástí je maximální velikost paketu, který je strana schopna sestavit (minimum je 1 500 bajtů). Další volbou Multilink-Endpoint-Discriminator si konce spojení vymění svou jednoznačnou identifikaci.

Podle této identifikace pak protokol PPP pozná, že určité linky vedou na stejný počítač (protějšek na těchto linkách zaslal stejnou identifikaci), a tudíž je může vkládat do téhož svazku. Identifikace konce spojení se skládá ze dvou částí:

1. Z třídy, která specifikuje typ adresy (2=IP adresa, 3=ethernetová adresa, 5=telefonní číslo).
2. Z příslušné adresy. Windows používají ethernetovou adresu, takže pokud v okamžiku, kdy jsou aktivní např. linky telefonického připojení, provedete příkaz ipconfig s přepínačem /all, uvidíte, že systém vymyslel etherntové adresy i pro sériové linky.

Máme-li svazek vytvořen, mohou se do něj vkládat pakety. Jak je znázorněno na obrázku, odesilatel vezme paket protokolu PPP skládající se z čísla protokolu a dat a vkládá jej do svazku. Jenže jak to udělat, když je svazek tvořen více linkami?



**Obrázek 4.16:** Protokoly MP a BCP (spojení je obousměrné, i když na obrázku je jen jeden směr komunikace)

Proces, který odesilatel provádí, se nazývá fragmentace (nezaměňovat s fragmentací IP!). Paket může být i (ale nemusí) rozřezán na menší části – fragmenty. Fragmenty se následně vkládají do jednotlivých linek. Fragment má formát úplně normálního PPP-rámce, jak je znázorněn na obr. 4.9, s tím, že používá číslo protokolu  $003d_{16}$ . Datová část je pak uvozena záhlavím MP. Příjemce opačným postupem z fragmentů sestavuje původní pakety protokolu PPP skládající se z čísla protokolu a dat.

Záhlaví MP obsahuje číslo fragmentu a příznaky B a E. Záhlaví slouží ke dvěma účelům:

1. Aby z fragmentů mohly být sestaveny původní pakety.

2. Aby pakety byly sestavovány v tom pořadí, v jakém byly do virtuálního svazku vkládány. Uvědomme si, že nejsme na síťové vrstvě, a tudíž protokol PPP nic netuší např. o vlastnostech protokolu IP, který nepočítá se zachováním pořadí rámci. Navíc jiným protokolům záměna pořadí rámci vadit může.

Číslování v záhlaví MP slouží k zachování pořadí fragmentů (čísluje se striktně po jedné). Příznaky B (begin – začátek) a E (end – konec) se signalizuje, že fragment nese začátek, resp. konec paketu. Pokud má fragment nastaveny oba bity B a E, jedná se o fragment, který nese celý (nerozřezaný) paket.

Záhlaví MP je dlouhé 4 bajty s tím, že číslo fragmentu je dlouhé 24 bitů a výplň dlouhá 6 bitů obsahuje nuly. Pomocí volby SSNH (*Short Sequence Number Header*) je možná dohoda na záhlaví dlouhém pouze 2 bajty. V tomto případě je výplň pouze dvoubitová a ve zbylých dvanácti bitech je číslo fragmentu.

Zmíněné volby jsou volbami protokolu LCP s kódy 17 až 19, tj. k dohodě na těchto volbách dochází během fáze navazování spojení.

### **Protokoly Bandwidth Allocation Protocol (BAP) a Bandwidth Allocation Control Protocol (BACP)**

Tyto protokoly specifikované RFC-2125 umožňují dynamicky přidávat/ubírat jednotlivé linky do/ze svazku linek (viz protokol MP). Pokud jsou implementovány protokoly BAP a BACP, pak si každá linka, která navazuje spojení, během fáze navazování spojení specifikuje volbou „Link-Discriminator for BAP“ protokolu LCP svou identifikaci (v rámci počítače). Máme tak dvě různé identifikace vyjednané protokolem LCP během fáze navazování spojení:

- ◆ „Multilink-Endpoint-Discriminator“, který identifikuje celý počítač. Tento identifikátor používá protokol MP.
- ◆ „Link-Discriminator for BAP“, který identifikuje konkrétní linku v rámci svazku, přičemž každá strana spojení používá nezávislé identifikace.

Poté se aktivuje protokol BACP. Jedná se o jednoduchý řídicí protokol, jehož funkce spočívá v řešení případu, kdyby se náhodou stalo, že oba konce budou současně požadovat přidat/ubrat linku do/ze svazku. Problém se řeší tak, že obě strany si hodí kostkou a ta, které padne menší číslo, vyhrává (její požadavky budou mít přednost). Když padne stejně číslo, hod se opakuje. Ve skutečnosti musím napsat: Protokol BACP je protokol, jehož pakety mají obdobný tvar jako pakety protokolu LCP (číslo protokolu c02b<sub>16</sub>), tj. používá příkazy Configure-Request, Configure-Ack (padlo jiné číslo), Configure-Nack (padlo stejné číslo) a případně Configure-Reject. Tyto příkazy obsahují jedinou volbu „preferovaná strana“ (*Favored-Peer*), která nese čtyřbajtové nenulové náhodné číslo. Preferovanou se stane ta strana, jejíž generátor náhodných čísel vygeneroval menší číslo. A nehází šestibokou kostkou, ale kostkou která má 2<sup>32</sup> boků.

Vlastní protokol BAP, který přidává a ubírá linky, pak používá příkazy:

Kód	Název příkazu (anglicky)	Význam volby
1	Call-Request	„Začal bych navazovat spojení další linkou“ (před tím, než strana začne přidávat linku, to musí být potvrzeno od druhé strany).
2	Call-Response	Potvrzení požadavku na přidání linky.
3	CallBack-Request	„Přidal bych linku, ale vytočit ji musíš ty. Co ty na to?“

Kód	Název příkazu (anglicky)	Význam volby
4	CallBack-Response	Potvrzení požadavku na přidání linky zpětným voláním.
5	Link-Drop-Query-Request	„Ubral bych linku.“
6	Link-Drop-Query-Response	Potvrzení požadavku na ubrání linky.
7	Call-Status-Indication	Po kladném potvrzení požadavku na přidání linky se volbou <i>Call-Status-Indication</i> sděluje, jak přidání dopadlo.
8	Call-Status-Response	Odpověď na <i>Call-Status-Indication</i> .

Všimněte si, že všechny odpovědi jsou typu „potvrzení požadavku na...“. Odpověď tedy neříká, jestli potvrzení je kladné nebo záporné, to se dozvímme až z datové části odpovědi. Datová část odpovědi začíná jedním bajtem (osmi bitů), který obsahuje chybový kód:

- ◆ 00000000 – Request-Ack
- ◆ 00000001 – Request-Nack
- ◆ 00000010 – Request-Reject
- ◆ 00000011 – Request-Full-Nak (strana požadovanou akci principiálně odmítá; např. bylo dosaženo maximálního/minimálního počtu linek ve svazku)

Příkazy mohou nést následující volby:

Kód	Název příkazu (anglicky)	Význam volby
1	Link-Type	Rychlosť a typ linky. Typy jsou např.: 0=ISDN, 1=X.25, 2=analogová atd.
2	Phone-Delta	Obsahuje informace potrebné pro vytvorenie telefónneho čísla (napr. telefónne číslo).
3	No-Phone-Number-Needed	Informace pre vytvorenie telefónneho čísla sa neprepravujú. Napr. sú uložené v konfigurácii a z bezpečnostných dôvodov si to neprejeme.
4	Reason	Dôvod pridania/odebrania linky (textový reťazec).
5	Link-Discriminator	Používa sa pri odebíraní linky. Obsahuje identifikáciu linky, ktorá sa bude odebírať (viz volba „Link-Discriminator for BAP“ protokolu LCP).
6	Call-Status	Informace o tom, ako dopadlo pridanie ďalšej linky (status), a akce, ktorá bude následovať v prípade, že sa vytvorenie nepovedlo. Kódy pre status sú podľa Q.931 (0 napr. znamená, že sa vytvorenie úspešne zdařilo) a akce mohou byť: 0=vytáčenie sa nebude opakovat, 1=vytáčenie sa bude opakovat.

### Compression Control Protocol (CCP)

Protokol CCP (specifikovaný v RFC-1962) slouží k vyjednání algoritmu pro kompresi dat. Rámce protokolu PPP nesoucí pakety protokolu CCP používají číslo protokolu 80fd<sub>16</sub> a opět mají formát odvozený od protokolu LCP. V případě, že je spoj realizován více linkami (MP) a jedná se o individuální kompresi na konkrétní lince, použije se číslo protokolu 80fb<sub>16</sub>. Dlouho jsem nemohl pochopit, proč dvě čísla protokolu. Ale stačí si prohlédnout obrázek 4.16 a uvědomit si, že se může komprimovat paket nebo fragment. Doslovně podle obrázku 4.16: komprese může probíhat buď nad

ozubenými koly (komprese celého paketu), nebo pod ozubenými koly (komprese fragmentu). Problém je v tom, že komprese pod ozubenými koly, pak teoreticky na některých linkách lze provádět kompresi a na jiných nikoliv (používají např. kompresi na fyzické vrstvě protokolem V.42bis).

Datové rámce nesoucí komprimované pakety pak používají číslo protokolu  $00\text{fd}_{16}$  (resp.  $00\text{fb}_{16}$ ), tj. komprimovaný datagram. Pokud se komprese nezdaří (např. komprimovaný paket by byl větší než původní paket), pošle se nekomprimovaný paket. Příjemce pozná podle čísla protokolu, zda má provádět dekomprese, či nikoliv.

Protokol CCP slouží obecně k vyjednání komprimačního algoritmu. Jeho příkazy jsou obdobné příkazům protokolu LCP: Configure-Request, Configure-Ack, Configure-Nak, Configure-Reject, Terminate-Request, Terminate-Ack a Code-Reject s kódy 1 až 7. Zvláštností protokolu CCP jsou dva speciální příkazy:

- ◆ Reset-Request (kód=14). Je totiž doporučováno, aby komprimovaná data byla periodicky doplnována o kontrolní součet. Pokud příjemce zjistí, že komprimace selhala, tj. kontrola na kontrolní součet selhala, pak příkazem Reset-Request vydá požadavek, aby odesílatel inicializoval (vynuloval) všechny své komprimační čítače, slovníky atd., tj. aby komprimace začala probíhat znova od počátku.
- ◆ Reset-Ack (Kód=15). Když odesílatel vynuluje všechny své čítače a slovníky, pak o tom příkazem Reset-Ack informuje příjemce, který musí také provést vynulování, aby byl schopen korektně dekomprimovat následující data.

Volby slouží pro dohodu na konkrétním algoritmu. Nepodaří-li se konkrétní algoritmus dohodnout, nevadí, komunikace běží dále nekomprimovaně. Kódy volby pak odpovídají jednotlivým algoritmům. Např. pro „Stac LZS“ je kód 17, pro MPPC kód 18 atp.

Konkrétní dohodu na konkrétním komprimačním algoritmu specifikují jednotlivé následné normy:

- ◆ RFC-1967: „PPP LZS-DCP Compression Protocol (LZS-DCP).“
- ◆ RFC-1974: „PPP Stac LZS Compression Protocol.“
- ◆ RFC-2118: „Microsoft Point-To-Point Compression (MPPC) Protocol.“

Když jsou data zkomprimována, můžeme přejít k jejich šifrování. Opačný postup není příliš efektivní, protože šifrovaná data („rozsypaný čaj“) se příliš komprimovat nedají.

### **PPP Encryption Control Protocol (ECP)**

Protokol ECP (specifikovaný v RFC-1968) slouží k vyjednání šifrovacího algoritmu. Tento protokol **neslouží** k výměně šifrovacích klíčů. Jeho syntaxe je zcela obdobná syntaxi protokolu CCP, a to včetně příkazů Reset-Request a Reset-Ack. Na rozdíl od protokolu CCP když se oba konec nedohodnou na konkrétním algoritmu, tak v závislosti na konfiguraci jednotlivých stran může být komunikace ukončena. Např. na obr. 4.13 je právě zvoleno, že v takovém případě má být komunikace ukončena.

Následné normy konkretizují dohody na konkrétních algoritmech. Např. RFC-3078: „Microsoft Point-To-Point Encryption (MPPE) Protocol“ používá šifru RC-4. Součástí protokolu MPPE je též dohoda na délce šifrovacího klíče (40, 56 nebo 128 bitů).

## Stanovení šifrovacích klíčů

Jsme-li dohodnuti na šifrovacím algoritmu, schází nám ke spokojenosti jen šifrovací klíče. Protokol PPP odvozuje šifrovací klíče od informací, které si obě strany vyměnily již během fáze autentizace. Uvědomme si, že pro autentizaci např. některým z protokolů CHAP musejí obě strany spojení sdílet společné tajemství. Přitom společné tajemství je svou podstatou něco jako symetrický šifrovací klíč. Avšak použít sdílené tajemství přímo jako symetrický šifrovací klíč by mohlo zjednodušit jeho vyzrazení, proto se společné tajemství přemele jednocestnými funkcemi a do mlýnku se často jako koření přidává i řetězec náhodné výzvy (challenge).

RFC-3079 „Deriving Keys for use with Microsoft Point-to-Point Encryption (MPPE)“ právě specifikuje odvození šifrovacích klíčů v případě autentizace protokoly MS CHAP verze 1 a MS CHAP verze 2.

RFC-3079 dále specifikuje odvození šifrovacích klíčů i v případě autentizace EAP-TLS. Pokud použijeme autentizaci EAP-TLS, již máme na obou stranách vytvořeno tzv. hlavní tajemství, od kterého odsekváme jednotlivé šifrovací klíče (pro každý směr komunikace jiný). Poté můžeme začít zabezpečovat komunikaci tak, jak to dělá protokol TLS. Pokud je systém promyšleně nasazen, nastavuje zajištění pomocí EAP-TLS bezpečnostní laťku značně vysoko. Bez zkušeností však přináší spíše trampoty.

## Protokol IPCP

Poté co je mezi oběma konci navázáno spojení, proběhla autentizace a došlo k případným dohodám na použití více linek, dohodě na kompresi, dohodě na šifrování atd., tak do hry vstupují jednotlivé síťové protokoly. Otevření komunikace si musí každý vyjednat svým řídicím protokolem.

Protokol IPCP je protokol typu NCP pro protokol IP verze 4 (viz RFC-1332), tj. protokol IPCP je řídicím protokolem protokolu IP. Formát rámce protokolu IPCP je obdobný formátu rámce protokolu LCP s tím, že číslo protokolu je 8021<sub>16</sub> (viz obr. 4.9).

Kód	Název kódu (anglicky)	Význam volby
1	Configure-Request	Konfigurační paket nesoucí požadavky na změnu implicitních parametrů.
2	Configure-Ack	Konfigurační paket s kladným potvrzením požadavků na změnu implicitních parametrů, tj. všechny požadované změny parametrů jsou akceptovány.
3	Configure-Nak	Konfigurační paket s odpovědí. Avšak protější strana neakceptuje všechny požadavky na změnu parametrů linky. Ty, které neakceptuje, jsou v tomto paketu specifikovány. Ostatní požadavky jsou akceptovány (tj. požadavky nespecifikované v paketu Configure-Nak jsou akceptovány).
4	Configure-Reject	Konfigurační paket odmítající všechny požadavky. Může být i důsledkem chybného kódu požadavku.
5	Terminate-Request	Požadavek na ukončení spojení (ukončení přenosu protokolu IP).
6	Terminate-Ack	Potvrzení požadavku na ukončení spojení.
7	Code-Reject	Odmítnutí požadavku z důvodu neznámého kódu. Může být způsobeno i tím, že protější stanice používá jinou verzi protokolu.

Protokol IPCP používá následující volby:

Kód	Název kódu (anglicky)	Význam volby
2	IP Compression-Protocol	Komprese záhlaví protokolu IP. Datová část této volby obsahuje číselnou identifikaci kompresního protokolu, např. šestnáctkově 002d pro „klasickou“ kompresi podle RFC-1144 popsanou v části CSLIP. Pro Van Jacobsonovu kompresi zdokonalenou v RFC-2507 až 2509 datová část obsahuje šestnáctkově 0061. Za číslem kompresního protokolu následují parametry komprese. Např. pro klasickou Van Jacobsonovu kompresi nese dva jednobajtové parametry: „Max Slot ID“ (nejvyšší číslo slotu bude od nuly do tohoto čísla) a „Comp Slot Id“ (je-li nastaveno na 0, pak číslo slotu nesmí být vypouštěno ani v po sobě následujících komprimovaných paketech téhož datového toku). Pakety komprimované podle RFC-2507 mají dle RFC-2507 parametry: TCP_SPACE, NON_TCP_SPACE atd.
3	IP Address	Předání IP adresy protějšku. Takto je možno dynamicky přidělovat IP adresy. Chce-li strana použít jinou IP adresu, odpoví paketem Configure-Nak, kde tuto adresu specifikuje.
129	Primary-DNS-Address	Pole data obsahuje čtyřbajtovou IP adresu primárního jmenného serveru.
130	Primary-NBNS-server-address	Adresa primárního WINS-serveru.
131	Secondary-DNS-Address	Adresa sekundárního jmenného serveru.
132	Secondary-NBNS-server-address	Adresa sekundárního WINS-serveru.

V rámci protokolu PPP se pro přenos Van Jacobsonovou metodou nekomprimovaných datových paketů s protokolem IP verze 4 použije identifikace protokolu 0021<sub>16</sub>. V případě komprese je situace komplikovanější, protože ne všechny pakety mají komprimované záhlaví. Je tedy nutné rozlišovat v přenášených paketech pakety s komprimovaným záhlavím TCP/IP a pakety s nekomprimovaným záhlavím (např. první pakety toku). Proto v záhlaví PPP-rámce v poli protokol mají nekomprimované pakety identifikaci 002f<sub>16</sub> (pole protokol v IP záhlaví mají nahrazeno číslem slotu) a pakety s komprimovaným IP záhlavím identifikaci 002d<sub>16</sub>.

## Příklad

Příklad rámce protokolu PPP nesoucí příkaz Configure-Ack protokolu IPCP zobrazený programem Wireshark:

```
+Frame 39 (36 bytes on wire, 36 bytes captured)
+Ethernet II, Src: 20:53:45:4e:44:09, Dst: 20:53:45:4e:44:09
-PPP IP Control Protocol
    Code: Configuration Ack (0x02)
    Identifier: 0x08
    Length: 22
    Options: (18 bytes)
        -IP compression: 6 bytes
            IP compression protocol: VJ compression (0x002d)
            Max slot id: 15 (0x0f)
            Compress slot id: yes (0x01)
```

IP address: 10.1.1.3  
 Primary DNS server IP address: 194.149.105.18

Všimněte si, že je potvrzena Van Jacobsonova (VJ) komprese IP & TCP záhlaví (RFC-1144), dále IP -adresa klienta (10.1.1.3) a IP adresa serveru DNS (194.149.105.18).

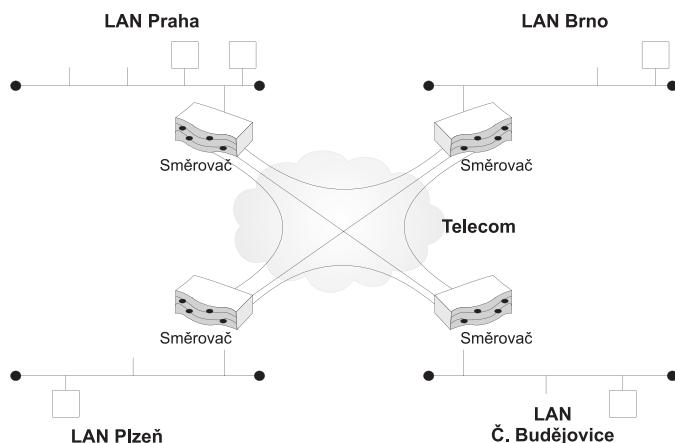
## Frame Relay

Frame Relay je protokol linkové vrstvy pro rozsáhlé sítě. Je specifikován v normách: I.122, I.441 a ANSI TI.606, a zejména v normách skupiny Frame Relay Forum (FRF).

Frame Relay používá zpravidla pevné virtuální okruhy (komutované okruhy jsou také teoreticky možné, ale v praxi jsem se s nimi u protokolu Frame Relay dosud nesetkal). Pevný virtuální okruh je obdobou pevné linky. Uživatel si od provozovatele sítě Frame Relay pronajímá virtuální okruhy mezi svými jednotlivými lokalitami.

Frame Relay je datagramová nespojovaná služba, tj. jsou jí přenášeny nečíslované rámce. Doručení rámce obecně není provozovatelem garantováno. Každý rámec obsahuje kontrolní součet, lze tedy ověřovat, došlo-li během přenosu k narušení paketu. Narušený paket se zahazuje.

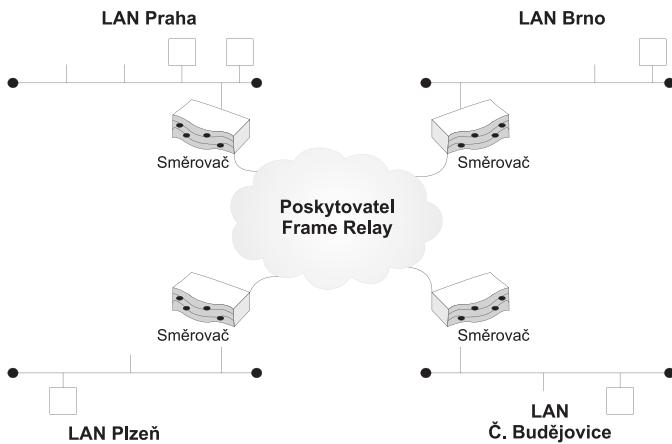
Základním parametrem virtuálního okruhu je množství dat, které může uživatel v časovém intervalu  $T_C$  předat virtuálnímu okruhu. Tuto veličinu budeme dále označovat jako šíře přenosového pásmá (*bandwidth interval*) a označovat ji budeme  $B_C$ . Častěji se však používá poměr  $B_C/T_C$ , který se označuje jako CIR (*Committed Information Rate*). CIR vyjadřuje, kolik dat může uživatel předat sítí Frame Relay za jednotku času. CIR je abstrakce, protože nikdy nelze přesně za 1 sekundu předat tolik bajtů, kolik vyjadřuje zprůměrovaný CIR. Data se totiž sítí předávají v celých rámcích – nikoliv v jejich částech. Jedná se o průměr.



**Obrázek 4.17:** WAN založená na pevných linkách

Zajímavou vlastností je tzv. šíře pásmá podle potřeby. Jinými slovy: uživatel se dohodne s poskytovatelem na CIR (např. 64 kb/s). Jenže v případě špičky může tuto hranici i překročit. Vše je pochopitelně za určitý poplatek, takže kromě CIR se uživatel dohodne i na další veličně  $B_E$  (*Excess Burst Rate*).  $B_E$  vyjadřuje, o kolik bajtů je možné v časovém intervalu  $T_C$  veličinu  $B_C$  překročit.

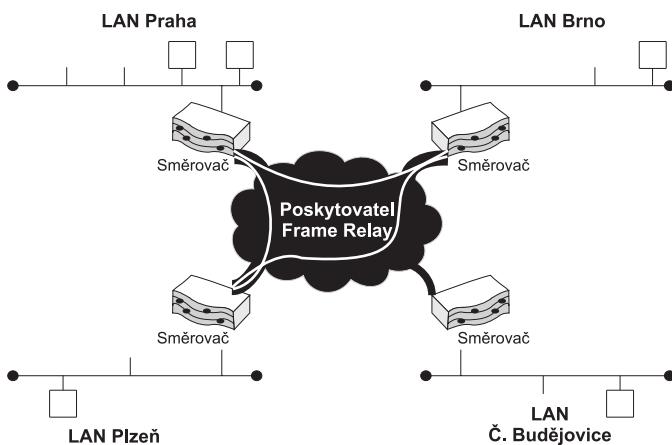
Frame Relay je určen pro rychlosti od 56 kb/s do 2 Mb/s. Je však efektivní ještě při rychlostech okolo 100 Mb/s.



**Obrázek 4.18:** WAN založená Frame Relay

Pronajmeme-li si pevné linky od provozovatele veřejné telefonní sítě, potřebujete pro propojení např. čtyř lokalit vzájemně mezi sebou čtyři pevné linky. V každé lokalitě budete mít tři modemy a obsazená tři rozhraní na směrovači. Obrázek 4.17 znázorňuje rozsáhlou síť hypotetické firmy, která je situována v Praze, Plzni, Č. Budějovicích a Brně.

Naproti tomu poskytovatel sítě Frame Relay provozuje vlastní datovou síť (obr. 4.18), na kterou je uživatel (zákazník) napojen v jednotlivých lokalitách zpravidla jednou linkou o vyšší kapacitě. Po této lince svěřuje své datové rámce provozovateli sítě Frame Relay a očekává, že je obdrží v jiné své lokalitě. Provozovatelská síť se skládá z tzv. přepínačů Frame Relay, které si mezi sebou předávají svěřené rámce. Uživateli vcelku nezájímá, přes jaké přepínače jde jeho rámec. Dokonce jej ani nezájímá, zdali uvnitř provozovatelského sítě používá protokol Frame Relay nebo ne.



**Obrázek 4.19:** Virtuální okruhy

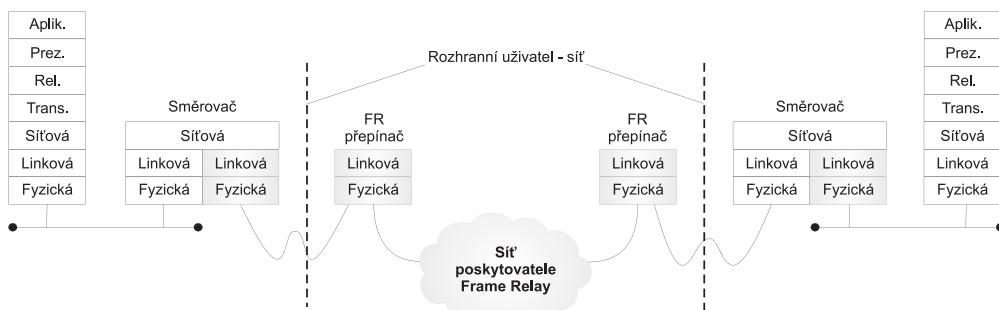
Pro propojení jednotlivých lokalit stačí v každé lokalitě jedno rozhraní na směrovači a jedna linka na nejbližší přístupový bod poskytovatele Frame Relay. Může jít např. o pevnou linku (např. E1), radioreléový nebo družicový spoj.

V datové síti poskytovatele jsou tedy vytvořeny virtuální okruhy mezi jednotlivými lokalitami uživatele, v našem případě okruhy Praha–Brno, Plzeň–Praha, Plzeň–Brno atd. (obr. 4.19). Vzniklá síť má topologii shodnou se sítí, kde by byly jednotlivé lokality propojeny pevnými linkami. Rozdíl od pevných linek spočívá v tom, že zde fyzická linka spojující např. lokalitu Praha se sítí Frame Relay slouží současně dvěma virtuálním okruhům.

Provozovatel sítě Frame Relay vytvořil na přání zákazníka pomocí virtuálních okruhů rozsáhlou privátní síť. Avšak provozovatel Frame Relay má více zákazníků a pro každého na jeho přání vytváří jinou privátní WAN.

Klasický případ připojení zákazníka k síti Frame Relay není, že by se k síti Frame Relay připojoval přímo počítač. K Frame Relay se připojuje směrovač a jednotlivé počítače jsou v lokalitě se směrovačem připojeny pomocí lokální sítě.

Mezi směrovačem uživatele a nejbližším přepínačem Frame Relay je definováno rozhraní uživatel-sítě (obr. 4.20). Na tomto rozhraní svěřuje své rámce uživateli provozovateli sítě. Právě vůči tomuto rozhraní má Frame Relay vlastnosti, o kterých se zmínujeme. Co je uvnitř, to uživatele nezajímá. Je to zcela obdobné jako u datových sítí na bázi protokolu X.25 (X.25 je také jen rozhraní mezi uživatelem a provozovatelem sítě).

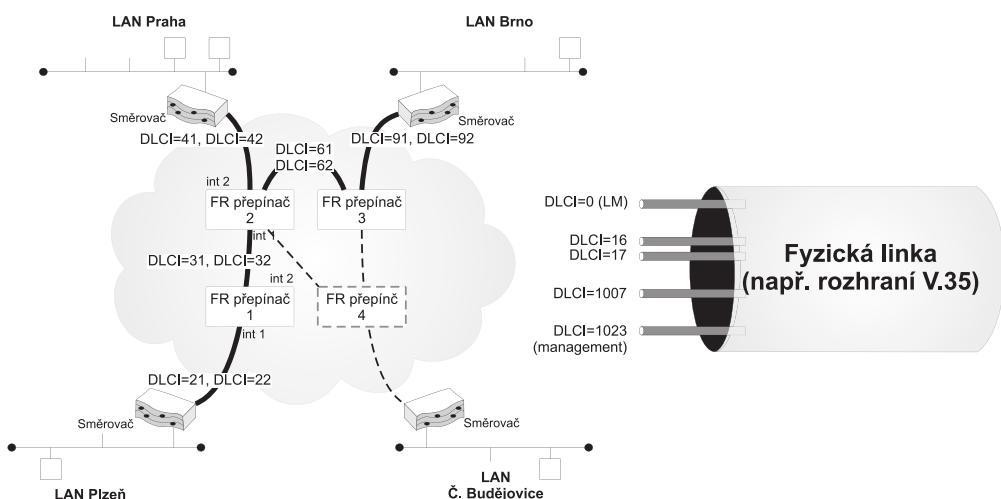


**Obrázek 4.20:** Připojení k poskytovateli Frame Relay

Na fyzické vrstvě se u sítí Frame Relay používá rozhraní V.35, X.21 apod. Používají se okruhy se synchronním přenosem (hodiny generuje síť Frame Relay).

Rámec prochází na cestě po virtuálním okruhu od zdroje k cíli celou řadou linek. Mezi uživatelem a provozovatelem je linka (rozhraní uživatel-sítě). Dále má na cestě jednotlivé linky od jednoho přepínače Frame Relay ke druhému. Na druhém konci následuje opět rozhraní uživatel-sítě. Každý virtuální okruh je identifikován tzv. DLCI (*Data Link Connection Identifier*). DLCI je součástí záhlaví rámce Frame Relay. Jdou-li z nějaké lokality uživatele např. dva virtuální okruhy do různých lokalit, pak se jednotlivé lokality rozlišují v záhlaví rámce pomocí DLCI. Záhlaví rámce protokolu Frame Relay má zpravidla 10 bitů pro DLCI, tj. DLCI nabývá hodnot 0 až 1 023.

Na obr. 4.21 rámc na cestě z Plzně do Brna postupně mění své DLCI. Uživatel jej v Plzni odevzdá poskytovateli sítě Frame Relay s vyplňeným DLCI=22. Přepínač číslo 1 má nakonfigurováno ve své tabulce, že příchozí rámec z rozhraní 1 s DLCI=22 má předat na rozhraní 2 s tím, že změní DLCI na 32 („Brněnská lokalita našeho zákazníka přes přepínač 2“). Přepínač 2 je nakonfigurován tak, že mění DLCI=32 na DLCI=62 („Brněnská lokalita našeho zákazníka přes přepínač 3“). Přepínač 3 je zase nakonfigurován tak, že změní DLCI=62 na DLCI=92 a rámec předá uživateli. Pokud uživatel odevzdá v Plzni rámec s DLCI=21, pak přepínač číslo 1 nezmění DLCI na 32, jak tomu bylo při směrování do Brna, ale na hodnotu 31. Přepínač 2 změní pak hodnotu DLCI na 41 a rámec je přijmut v Praze.



Obrázek 4.21: DLCI

Z hlediska zákazníka je situace ale jednodušší. Když chce z Plzně poslat rámec do Brna, tak v Plzni vyplní v záhlaví rámce DLCI=22 a ví, že jej v Brně obdrží s DLCI=92. A odevzdá-li v Plzni rámec s DLCI=21, pak jej obdrží v Praze s DLCI=41. Obráceně: odevzdá-li v Praze rámec s DLCI=41, pak jej obdrží v Plzni s DLCI=21.

DLCI se mohou používat v síti Frame Relay:

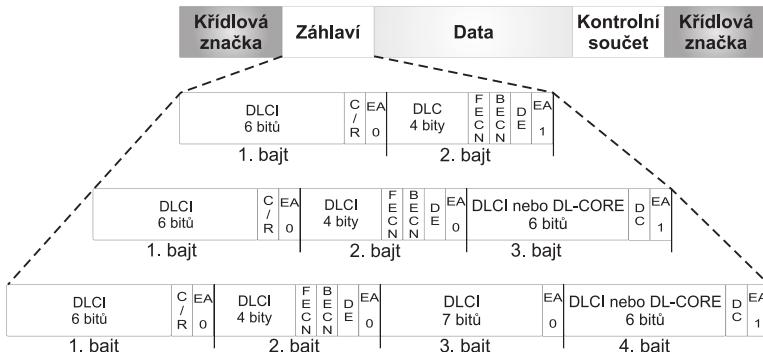
- ◆ Jednoznačná v rámci celé sítě provozovatele Frame Relay.
- ◆ Jednoznačná v rámci jednoho přepínače Frame Relay.
- ◆ Jednoznačná v rámci jednoho rozhraní přepínače Frame Relay.

### Rámec protokolu Frame relay

Rámec protokolu Frame Relay na rozdíl od protokolu HDLC nemá samostatné adresní a řídicí pole. Má společné pole záhlaví, které obsahuje identifikaci virtuálního okruhu (DLCI) a další řídicí informace. Záhlaví je dlouhé jeden až čtyři bajty. V praxi se nejčastěji setkáváme se záhlavím dlouhým 2 bajty (tj. 10 bitů pro DLCI).

Záhlaví může být dlouhé dva, tři nebo čtyři bajty. Každý bajt záhlaví obsahuje bit EA, který určuje, zdali následující bajt je ještě součástí záhlaví, nebo už přenášených dat. Je-li EA=0, je i následující bajt součástí záhlaví, je-li EA=1, je tento bajt posledním bajtem záhlaví.

Pole **DLCI** (*Data Link Connection Identifier*) je identifikací virtuálního okruhu. DLCI může být jednoznačné pouze v rámci konkrétního sítového rozhraní, v rámci uzlu nebo v rámci celé sítě. Nejčastěji se setkáváme s jednoznačností pouze v rámci konkrétního rozhraní.



**Obrázek 4.22:** Rámcem formátu Frame Relay

Bit **C/R** určuje, zda jde o příkaz (C) nebo odpověď (R). U záhlaví dlouhého tří nebo čtyři bajty máme ještě bit **DC**, který pokud je nastaven na 0, pak specifikuje, že posledních šest bitů je součástí DLCI. V případě, že je nastaven na 1, nese posledních 6 bitů DL-CORE.

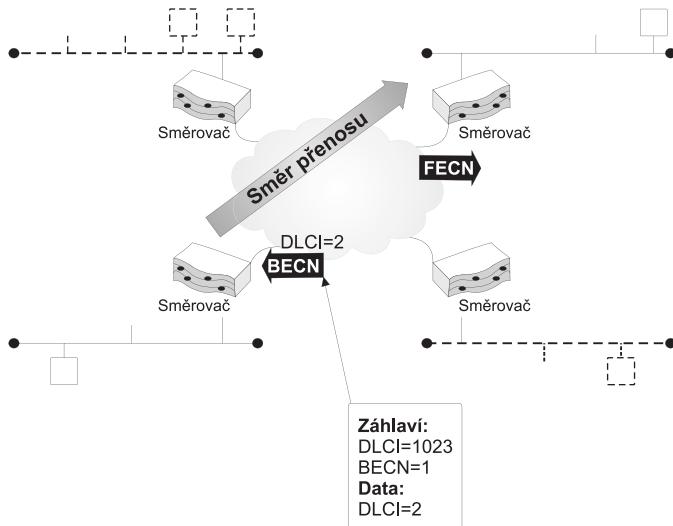
Nastavením bitu **DE** (*Discard Eligibility*) se signalizuje, že rámec je kandidátem na zahození (takto nastavený rámec má menší důležitost). Např. rámcům vysílaným nad sjednanou průměrnou kapacitu linky (CIR) se může nastavit bit DE na jedničku. V případě, že síť není schopna přenést všechny rámce, zahazuje přednostně rámce s nastaveným bitem DE.

Zbývají bity **BECN** (*Backward Explicit Congestion Notification*) a **FECN** (*Forward Explicit Congestion Notification*). I když je nastavování těchto bitů nepovinné, zastavíme se u nich. Jejich pomocí se řeší problém zahlcení virtuálního okruhu. Posílají-li se data virtuálním okruhem z jednoho konce na druhý, nevadí, když se nějaký paket ztratí (např. na úrovni protokolu TCP se přenos zopakuje). Problém je ale se zahlcením spoje, tj. je-li na cestě k cíli nějaké úzké místo, které není schopno rámce požadovanou rychlosť předávat dále. Rámce se v takovém uzlu hromadí v jeho vyrovnávací paměti (ve frontě), až dojde k vyčerpání paměti a další rámce se musí zahazovat. Tuto situaci nazýváme zahlcením linky.

Ztráta rámců pro vyšší vrstvy znamená, že musí vyžadovat opakování přenosu paketů vyšší vrstvy, tj. zopakování vysílání ztracených rámců. Nebo to dokonce může znamenat ztrátu spojení, tj. spojení je nutné obnovovat. V každém případě to znamená zvýšení objemu přenášených dat.

Dojde-li k zahlcení linky, každé další i drobné zvýšení provozu zahlcení ještě prohlubuje a linka se stává čím dál méně průchodnou. Na úrovni protokolu TCP se musí stále znova a znova obnovovat komunikace, až postupně dojde k rozpadnutí spojení. Koncový uživatel je rozlolen – myslí si, že nastala porucha sítě.

Řešení spočívá v tom, že se zvýší doba odezvy na virtuálním okruhu, tj. virtuální okruh se už na vstupu bude tvářit, že má menší propustnost. Virtuální okruh tedy odebírá rámce od uživatele sice menší rychlostí, ale odebrané rámce se snaží doručit. Uživateli se virtuální okruh jeví jako pomalejší, ale spojení mu nepřipadá porouchané.



**Obrázek 4.23:** Signalizace zahlcení virtuálního okruhu

V případě zahlcení virtuálního okruhu signalizuje síť odesilateli zahlcení nastavením bitu BECN a příjemci signalizuje zahlcení nastavením bitu FECN (odesilatelem a příjemcem je myšlen uživatelův směrovač na rozhraní uživatel-síť). Viz obr. 4.23. Odesílání rámců s nastaveným bitem BECN (resp. FECN) síť provádí od okamžiku, kdy se blíží stav, že síť bude muset zahazovat rámce, nebo dokonce k zahazování rámčů již dochází. Síť může zahlcení i předpovídat, když při kontrole front nahromaděných rámců na uzlech sítě Frame Relay zjistí, že některá fronta je blízká vyčerpání.

Nastavení bitů BECN a FECN se neprovádí u rámců běžných datových okruhů (datových DLCI). Pro tuto signalizaci je rezervováno služební DLCI=1 023, jehož rámce síť Frame Relay odesílá uživateli na rozhraní uživatel-síť. V datové části má takovýto služební rámec strukturu odvozenou od rámce XID protokolu HDLC (U-rámec XID protokolu HDLC slouží pro příkazy a odpovědi nesoucí konfigurační informace). V našem případě rámec XID nese číslo DLCI, které odpovídá zahlcenému virtuálnímu okruhu. Obrázek 4.23 vyjadřuje situaci, kdy postižený virtuální okruh má u odesilatele DLCI=2.

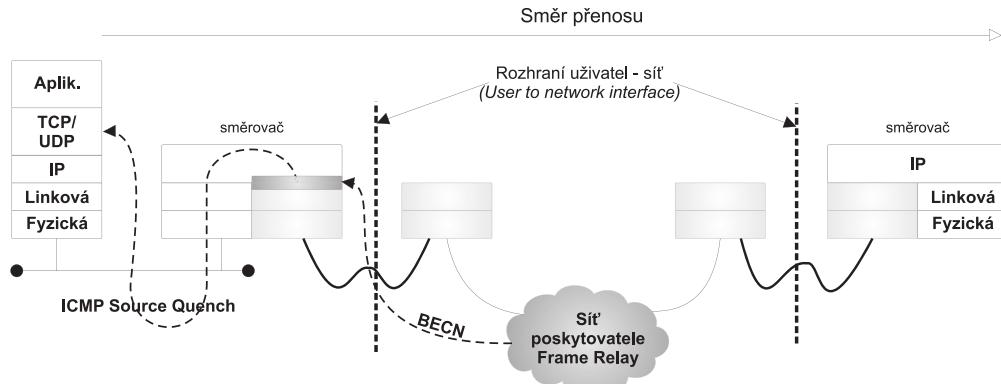
Problém je v tom, že uživatelův přístupový směrovač dostane pomocí bitu BECN informaci, že konkrétní virtuální okruh je zahlcen. Jak ale může směrovač snižovat zátěž linky? Musí být dostatečně inteligentní, aby vyšší vrstvě signalizoval zahlcení.

V případě Internetu se jako vyšší vrstva používá protokol IP (na obr. 4.24 to je nejvyšší vrstva směrovače). Směrovač v sobě musí obsahovat podporu, která je na pomezí Frame Relay a IP.

Podpora ošetření zahlcení okruhu může spočívat v různých mechanismech. Protokol IP má vlastní mechanismus pro ošetření zahlcení linek pomocí protokolu ICMP. Směrovač, který je nucen zahodit IP paket z důvodu zahlcení, o tom informuje odesilatele IP paketu ICMP datagramem *Source Quench*. Příjemce po obdržení ICMP datagramu *Source Quench* snižuje rychlosť příslušného TCP spoje. (Poznamenejme, že signalizace *Source Quench* se u protokolu UDP nepoužívá.)

Směrovač rovněž v pravidelných intervalech zjišťuje na každém virtuálním okruhu podíl přijatých rámců s nastaveným bitem BECN. Je-li výskyt rámců s nastaveným bitem BECN významný, začne

na příslušném virtuálním okruhu na příchozí pakety generovat odpovědi *Source Quench* v protokolu ICMP (viz obr. 4.24).



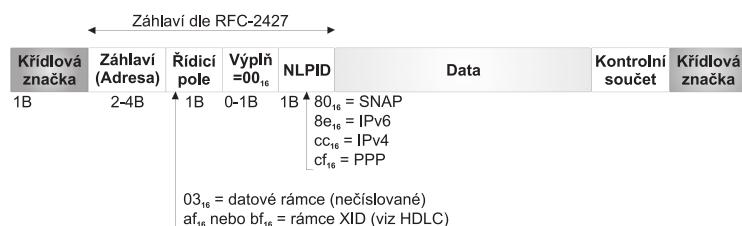
**Obrázek 4.24:** Signalizace BECN

Jsou i jiné možnosti. Např. přístupový směrovač rámce s nastaveným příznakem BECN vůbec nezpracovává. Pak na úrovni protokolu TCP/IP může docházet až ke ztrátám segmentů TCP. Spoj se může jevit jako poruchový.

Další možností je, že přístupový směrovač uživatele umí pracovat nejen s třetí vrstvou (IP protokol), ale i se čtvrtou vrstvou (protokol TCP). Potom by směrovač mohl přímo v záhlaví segmentů TCP opravit délku okna nebo mohl uměle pozdržet odpovědi od protější strany. Zdroj by si tedy myslel, že linka k cíli má delší odezvu, a automaticky by snížil rychlosť odesílání TCP paketů. Zásah do protokolu TCP na směrovači se však obecně považuje za nekorektní.

## IP přes Frame Relay

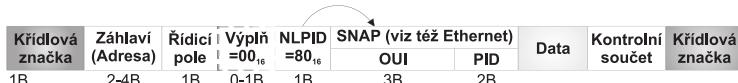
Prohlédneme-li si obrázek 4.22 s rámcem protokolu Frame Relay, bude nám v něm scházet pole nesoucí identifikaci protokolu vyšší vrstvy. Pomocí takového pole lze pak jednoduše a efektivně vkládat do linkového rámce (v našem případě do rámce Frame Relay) pakety různých síťových protokolů. Tento problém řeší RFC-2427 (*Multiprotocol over Frame Relay*). Rámec podle RFC-2427 je znázorněn na obr. 4.25.



**Obrázek 4.25:** Rámec Frame Relay dle RFC-2427 (pole záhlaví viz Obrázek 4.22)

Tento rámec je doplněn o:

- ◆ Řídicí pole odvozené od řídicího pole protokolu HDLC, tj. jelikož Frame Relay používá nečíslované rámcce, nese řídicí pole v případě datových rámců hodnotu 3.
- ◆ Výplň obsahující binární nuly zajišťuje sudou délku celého záhlaví podle RFC-2427. Takže se použije v případě, že pole záhlaví (adresa) má 3 bajty.
- ◆ Pole NLPID (*Network Layer Protocol ID*) obsahuje protokol vyšší vrstvy. Je bohužel jednobajtové, proto se ani nepočítá, že by všechny protokoly vyšší vrstvy měly přiděleno číslo. Z toho důvodu má přidělenou identifikaci  $80_{16}$  i protokol SNAP (*Sub-network Access Protocol*), který vkládá další záhlaví SNAP nesoucí identifikaci protokolu vyšší vrstvy (viz obr. 4.26).



**Obrázek 4.26:** Rámec Frame Relay dle RFC-2427 se záhlavím SNAP

Záhlaví SNAP se skládá z pole OUI s identifikací organizace, která čísla protokolů vyšších vrstev pro pole PID (*Protocol Identifier*) přiděluje. Např. pro OUI obsahující samé nuly nese v poli PID stejná čísla, která např. používá protokol Ethernet II (např.  $0800_{16}$  pro protokol IP atd.). Jelikož záhlaví SNAP je dlouhé tři bajty, použije se pole výplně jen v případě, když je pole záhlaví (adresa) dlouhé dva bajty.

Pro protokol IP můžeme teoreticky použít tři formáty rámce Frame Relay:

- ◆ Standardní formát rámce podle obrázku 4.22, který neobsahuje ani řídicí pole, ani pole NLPID. IP datagram se vkládá přímo do datové části.
- ◆ Rámec podle RFC-2427 (bez SNAP), kdy pole NLPID obsahuje hodnotu  $cc_{16}$  (protokol IPv4).
- ◆ Rámec podle RFC-2427 se záhlavím SNAP, kde pole NLPID obsahuje hodnotu  $80_{16}$ . Pole OUI obsahuje samé nuly a pole PID hodnotu  $800_{16}$ .

Existuje však ještě další varianta. Při této variantě vkládáme do rámců Frame Felay protokol PPP (*PPP in Frame Relay*) a teprve do něj protokol IP. Tato alternativa je specifikována v RFC-1973. Vychází z RFC-2427 (bez SNAP), kdy pole NLPID nese hodnotu  $cf_{16}$ . Jedná se o variantu protokolu PPP, která nepoužívá zapouzdření HDLC, ale zapouzdření Frame Relay.

## LMI

Síť Frame Relay můžeme používat tak, že mechanicky vkládáme rámce do virtuálního okruhu a na druhé straně je z virtuálního okruhu odebíráme a o nic dalšího se nezajímáme. Síť Frame Realy je nám však většinou schopna poskytnout další informace, jako jsou různé statistiky, účtovací informace, informace o tom, jestli je konkrétní rozhraní připojeno, odpojeno, nakonfigurováno atd. Tato komunikace se provádí protokolem LMI (*Local Management Interface*).

Pro tyto informace se používá služební DLCI s číslem nula, kde si uživatel se sítí Frame Relay vyměňuje pakety nesoucí rámce LMI s příslušnou informací. Při konfiguraci směrovačů pak uvádíme typ implementace protokolu LMI.

## Závěr k protokolu Frame Relay

Zákazník se s poskytovatelem Frame Relay zpravidla dohaduje na:

- ◆ Lokalitách, mezi kterými se vytváří jednotlivé virtuální okruhy.
- ◆ Na velikosti přenosových intervalů (CIR) na těchto okruzích a na veličině  $B_e$ , určující, o kolik se může ve špičkách CIR překračovat.
- ◆ Na nastavení bitů BECN a FECN. Uživatel si musí rozmyslet, jak je využije – zda uživatelský směrovače umí tyto bity využít.
- ◆ Na lince spojující uživatele a poskytovatele (pevná linka, radioreléový spoj atd.). Většinou poskytovatel Frame Relay dodává i linky spojující uživatele s jeho sítí. Pak je velice důležité použít fyzického rozhraní (V.35, X.21...), abyste věděli, jakou si máte koupit propojovací šnúru ke směrovači.

Častým dotazem je, jaký je rozdíl mezi Frame Relay a veřejnou sítí X.25. Frame Relay je protokol pouze linkové vrstvy (X.25 je protokol síťové vrstvy), tj. uživatelé Frame Relay zpravidla nemají nějakou celosvětově jednoznačnou adresu. Druhou odlišností je, že Frame Relay je datagramovou službou, tj. obecně není garantováno doručení rámce. Protokol X.25 si ukládá data, která nestačí zpracovávat do paměti, a postupně je předává dále. Společnou vlastností je, že oba protokoly vytvářejí pro uživatele virtuální okruhy.

## Ethernet

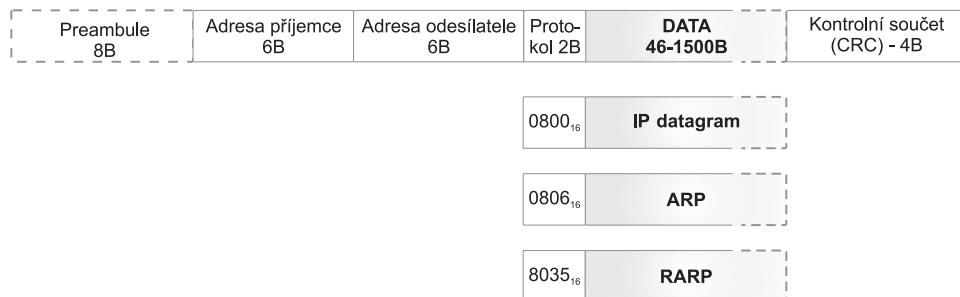
Protokol Ethernet byl původně vyvinut firmami DEC, Intel a Xerox (pod označením Ethernet II). Později byl Ethernet standardizován institutem IEEE jako norma IEEE 802.3 (byl i přejat jako ISO 8802-3).

Názvem Ethernet je nazývána celá rodina protokolů. Ty se liší:

- ◆ Na fyzické vrstvě použitým přenosovým médiem (koaxiální kabel, kroucená dvojlinka či optická vlákna) a pro uživatele pak hlavně přenosovou kapacitou.
- ◆ Na linkové vrstvě tvarem rámce. Pro všechny nejrůznější fyzické vrstvy máme výběr ze dvou typů rámce. Jeden typ linkového rámce se označuje jako Ethernet II a druhý jako IEEE 802.3.



**Poznámka:** Formát linkových rámců podle normy Ethernet II se odlišuje od formátu rámců IEEE 802.3.



Obrázek 4.27: Ethernet II

## Ethernet II

Rámec protokolu Ethernet II (obr. 4.27) má na počátku synchronizační preambuli (součást fyzické vrstvy), na které se synchronizují všechny stanice při přijímání rámce. Na konci rámce je kontrolní součet, ze kterého lze zjistit, nebyl-li rámec přenosem poškozen. Dále rámec obsahuje:

- ◆ šestibajtové linkové adresy příjemce a odesilatele,
- ◆ pole specifikující protokol vyšší vrstvy (tj. IP vrstvy nebo obecně síťové vrstvy),

Protokol	Protokol šestnáctkově	Protokol desítkově
Vyhrazeno pro délku dat protokolu IEEE 802.3	0000 až 05DC	0 až 1500
IPv4	0800	2048
ARP	0806	2054
RARP	0835	2101
IPv6	86DD	34525
TCP/IP s komprimovaným záhlavím dle RFC-1144	876B	34667
PPP	880B	34827
PPPoE Discovery	8863	34915
PPPoE Session	8864	34916

- ◆ datové pole, které musí být minimálně 46 bajtů dlouhé, takže v případě, že je potřeba přenášet méně dat, se datové pole zpráva doplní bezvýznamnou výplní,
- ◆ kontrolní součet (CRC) zajišťující integritu rámce.

Protokoly Ethernet II i IEEE 802.3 používají šestibajtovou linkovou adresu. Těchto 6 bajtů se obecně dělí na dvě poloviny (obr. 4.28):

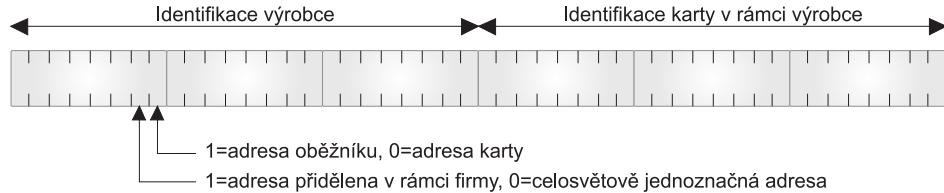
- ◆ První polovina je určena pro identifikaci výrobce karty (*Organizationally Unique Identifiers – OUI*), přičemž nejnižší 2 bity prvního bajtu mají specifický význam (viz obr. 4.28).
- ◆ Druhou polovinu je pak identifikace konkrétní karty v rámci výrobce.

Šestibajtové linkové adresy mohou být:

- ◆ Globálně (tj. světově) jednoznačné. Toho se dociluje tím, že první tři bajty linkové adresy specifikují výrobce síťové karty a zbylé tři bajty kartu v rámci výrobce. Tato šestibajtová adresa karty je uložena výrobcem do permanentní paměti síťové karty a nahrává se při inicializaci karty do ovladače karty.
- ◆ Jednoznačné v rámci firmy. Pak si firma může přidělovat tyto adresy podle vlastní politiky. To znamená, že při inicializaci karty se ovladači sdělí, aby nepoužíval tuto adresu, ale adresu jinou. V rámci firmy tak lze používat vlastní systém linkových adres. To je dobrá zpráva i pro nejrůznější útočníky.
- ◆ Oběžník, který je určen více adresátům. Rozlišujeme dva typy oběžníků:
  - Všeobecný oběžník (*Broadcast*), jehož adresa se skládá z 48 jedniček. Všeobecný oběžník je určen pro všechny stanice na LAN.
  - Adresný oběžník (*Multicast*), jehož adresa má nastaven nejnižší bit prvního bajtu na jedničku. Adresný oběžník je určen pouze některým stanicím na LAN stanicím, které akceptují uvedenou adresu.



**Poznámka:** Linkovou adresu karty nalezneme v odchycených rámcích jako linkovou adresu odesilatele rámce. Adresu oběžníku zase budeme zásadně nalézat jako adresu příjemce.



**Obrázek 4.28:** Linková adresa příjemce

Nultý a první bit prvního bajtu linkové adresy mají specifický význam (viz obr. 4.28):

- ◆ Nultý bit specifikuje, zda se jedná o jednoznačnou adresu nebo adresu oběžníku.
- ◆ První bit specifikuje, zda se jedná o globálně jednoznačnou adresu.

### Příklad

Uveďme si příklad výpisu rámce protokolu Ethernet II zachycený programem Wireshark:

```
+Frame 442 (60 bytes on wire, 60 bytes captured)
-Ethernet II, Src: 00:19:d2:29:51:53, Dst: 01:00:5e:00:00:fc
  -Destination: 01:00:5e:00:00:fc (01:00:5e:00:00:fc)
    Address: 01:00:5e:00:00:fc (01:00:5e:00:00:fc)
      .... .1 ..... .... .... = IG bit: Group address
      .... .0 ..... .... .... = LG bit: Globally unique address
  -Source: 00:19:d2:29:51:53 (00:19:d2:29:51:53)
    Address: 00:19:d2:29:51:53 (00:19:d2:29:51:53)
      .... .0 ..... .... .... = IG bit: Individual address (unicast)
      .... .0 ..... .... .... = LG bit: Globally unique address
  Type: IP (0x0800)
  Trailer: AE941D5EAACB000000000023FF53
+Internet Protocol, Src: 10.0.0.2, Dst: 224.0.0.252
+Internet Group Management Protocol
```

Všiměte si položky Trailer. Tu na obr. 4.27 nenajdete. Odchycený paket má totiž menší datovou část než 46B, takže musí být doplněn o výplň – ta je zde označena jako Trailer, protože je až na konci rámce (před CRC).



**Obrázek 4.29:** Rámec protokolu IEEE 802.3

### Ethernet IEEE 802.3

Situace u protokolu IEEE 802.3 je poněkud složitější. Datový rámec protokolu IEEE 802.3 (obr. 4.29) se sice liší od rámce Ethernet II pouze v jediném poli Délka, ale důsledky tohoto rozdílu jsou značné. Rámec protokolu IEEE 802.3 totiž neobsahuje pole s identifikací protokolu vyšší vrstvy. Pole Protokol obsahující identifikaci protokolu vyšší vrstvy je podstatné, aby na LAN mohly být vyměnovány rámce s různými protokoly vyšší vrstvy (např. IPv4, IPv6 a ARP).

V provozu síť však nemůže dojít k záměně rámců protokolu Ethernet II a IEEE 802.3, protože délka dat je nejvýše 1 500 B a specifikace protokolů pro normu Ethernet II jsou vyjadřovány vyššími čísly než 1 500.

Dále budeme pokračovat až na str. 124 v kap. LLC (IEEE 802.2), protože IEEE 802.2 je společný pro Ethernet a Wireless.

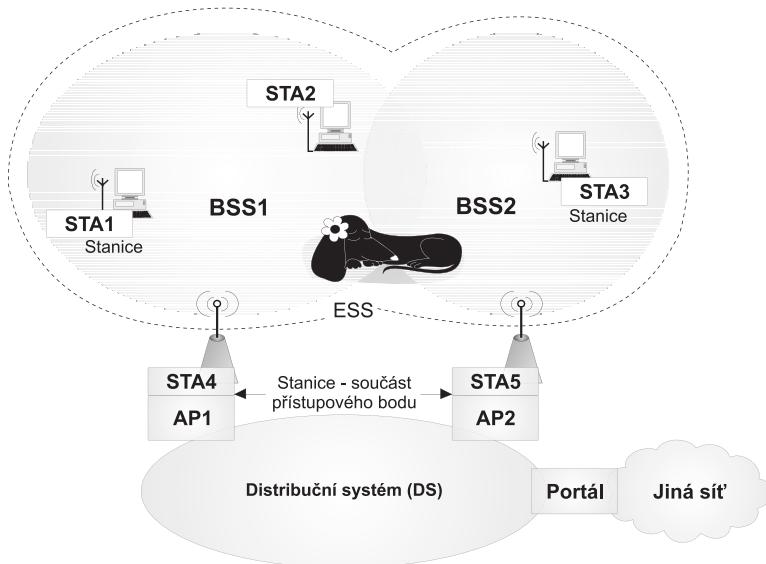
## Bezdrátové lokální sítě WLAN – IEEE 802.11

Podobně jako máme Ethernet, Fast Ethernet, Gigabitový Ethernet a vše ještě na nejrůznějších médiích, tak pro bezdrátové sítě máme rovněž několik protokolů (IEEE 802.11, 802.11a, 802.11b atd.). Avšak podobně jako v případě Ethernetu na linkové vrstvě jsou i v případě WLAN rozdíly zcela minimální.

Když si spustíte Wireshark na běžném bezdrátovém rozhraní, pak se vám komunikace na WLAN bude jevit jako komunikace na Ethernetu v protokolu Ethernet II. My jsme si ale obstarali jednoduchý hardwarový doplněk k programu Wireshark – sledovací síťové rozhraní AirPcap. A najednou se nám otevřel úplně jiný svět – svět protokolu IEEE 802.11. A musíme říci, že je to hodně odlišný svět, který používá celou řadu linkových rámců. A ještě ke všemu se nám v jednotlivých bajtech záhlaví mnohdy bity zobrazují v opačném pořadí...

Základní architektura WLAN (obr. 4.30) se skládá z následujících prvků:

- ◆ **Stanice (STA)** – jakékoli zařízení s rozhraním do bezdrátové sítě IEEE 802.11.
- ◆ **Basic Service Set (BSS)** – skupina stanic sdílejících stejné bezdrátové médium. V rámci jedné BSS mohou stanice komunikovat buď ad hoc mezi sebou (viz str. 71), nebo skrze přístupový bod



**Obrázek 4.30:** Infrastruktura WLAN

(*Access point* – AP). Stanice ale nemohou tyto typy komunikace kombinovat v rámci téhož bezdrátového média.

- ◆ **Distribuční systém (DS)** je základním termínem standardu IEEE 802.11. Jedná se o abstraktní síť, která ve skutečnosti buď ani fyzicky neexistuje (případ sítě s jedním AP, které simuluje celý DS), nebo je realizována nějakou jinou sítí (případ sítě s více AP). Zajímavé je, že z hlediska protokolu IEEE 802.11 není důležité, jakou sítí je distribuční systém realizován.
- ◆ **Přístupový bod (AP)** je stanice zajišťující komunikaci mezi BSS a distribučním systémem (DS). AP je zpravidla pevná (nepohybující se) stanice s rozhraním do distribučního systému. Z hlediska přenosu dat je AP mostem.
- ◆ Portál zprostředkovávající komunikaci mezi distribučním systémem a obecně jinou sítí. Portál nebývá samostatným zařízením, ale další funkčností přístupového bodu (AP).
- ◆ Oblast rozšířených služeb (*Extended Services Set* – ESS) začleňující více BSS tak, aby se stanice mohly jakoby volně pohybovat v rámci celé ESS. ESS se zpravidla skládá z více BSS.

V rámci WLAN jsou poskytovány následující služby:

- ◆ **Autentizace / Deautentizace.** Autentizace je prokazování identity protější stanici, se kterou chceme komunikovat. Deautentizaci je informování protějšku o ukončení stavu autentizace.
- ◆ **Asociace / Disasociace / Reasociace.** Asociací je míňeno přiřazení stanice ke konkrétnímu AP. Distribuční systém totiž musí znát, skrze které AP je konkrétní stanice dostupná. Disasociaci se rozumí informování protějšku o zrušení asociace. Reasociace se provádí při přesunu stanice z jednoho BSS do druhého BSS v rámci ESS.
- ◆ **Vlastní výměna rámců** v rámci BSS.
- ◆ **Distribuce** – přenos rámců mezi jednotlivými AP v rámci ESS.
- ◆ **Integrace** – přenos rámců skrze Portál.
- ◆ **Soukromí** – šifrování přenášených dat.

Každá stanice se může nacházet v jednom z následujících stavů:

1. **Počáteční stav**, kdy je neautentizovaná a neasociována.
2. **Autentizovaný stav**, kdy je autentizována, ale není asociována (s AP).
3. **Asociovaném stavu**, kdy je nejenom autentizována, ale i asociována.

Linkové rámce protokolu IEEE 802.11 jsou trojího typu:

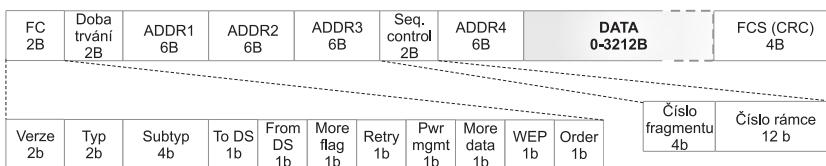
- ◆ Datové rámce, které v datové části zpravidla nesou záhlaví IEEE 802.2 následované daty vyšší vrstvy.
- ◆ Ovládací rámce (*Control Frames*), kterými se řídí komunikace v rámci BSS. Jedná se o rámce ACK, CTS, RTS apod., které mají jen minimální záhlaví IEEE 802.11 a zpravidla nenesou žádná data.
- ◆ Řídicí rámce (*Management Frames*), které v datové části nesou informace pro řízení WLAN, tj. za záhlavím IEEE 802.11 následuje obsah řídicího rámců (*IEEE 802.11 wireless LAN management frame*). Pomocí těchto rámců se uskutečňuje např. asociace stanice s přístupovým bodem. Při monitorování WLAN pak často narazíme na Majákové rámce (*Beacon frames*), kterými přístupový bod v pravidelných intervalech synchronizuje čas stanic v rámci BSS a též nabízí v síti své služby – stále mechanicky vysílá rámce podobně jako bliká maják na moři.

Důležité je, že v jednotlivých fázích je povoleno používat jen konkrétní typy rámců:

Typ rámce	Subtyp rámce	Počáteční stav	Autentizovaný stav	Asociovaný stav
Ovládací (Control)	Request to send (RTS)	ano	ano	ano
	Clar to send (CTS)	ano	ano	ano
	Acknowledgement (ACK)	ano	ano	ano
	CF-End+ACK	ano	ano	ano
	CF-End	ano	ano	ano
	PS-Poll	-	-	ano
Řídící (Management)	Probe request/response	ano	ano	ano
	Beacon	ano	ano	ano
	Authentication	ano	ano	ano
	Deauthentication	ano	ano	ano
	Announcement traffic	ano	ano	ano
	Association request/response	-	ano	ano
	Reassociation request/response	-	ano	ano
	Disassociation	-	ano	ano
	Deauthentication (způsobující i disasociaci)	-	-	ano
Datový	Datové rámce s příznaky „To DS“ a „From DS“ nastavenými na nulu	ano	ano	ano
	Libovolné datové rámce	-	-	ano

### Obecný formát rámce

Na obr. 4.31 je znázorněna nejdelší forma linkového rámce protokolu IEEE 802.11 se čtyřmi poli pro linkové adresy. Většina typů rámců ale využívá jen 1 až 3 adresy.



Obrázek 4.31: Rámec IEEE 802.11

Význam jednotlivých polí linkového rámce IEEE 802.11 je následující:

- ◆ **Frame Control (FC)** – Řídící pole. Význam jednotlivých bitů tohoto pole:
  - **Verze** standardu IEEE 802.11. T.č. se používá hodnota 0.
  - **Typ** rámce: Ovládací, datový či řídící.
  - **Subtyp** rámce. Např. ovládací rámce RTS, CTS, ACK atd.
  - Zbylých 8 bitů tvoří pole příznaků (*Flags*).
- ◆ **To DS** – rámec má být dopraven do distribučního systému.

- ◆ **From DS** – rámec přichází z distribučního systému.
- ◆ **More Fragment** – rámec je rozdělen do více fragmentů. Hodnota 0 znamená, že se jedná o poslední fragment rámce.
- ◆ **Retry** – jedná se o opakování vysílání nepotvrzeného rámce. Hodnota 1 znamená, že se jedná o opakováný rámec.
- ◆ **Pwr mgmt (Power Management)** – stanice po přenesení rámce přejde do úsporného režimu.
- ◆ **More Data** – stanice má ve vyrovnávací paměti více rámců.
- ◆ **WEP** – datová část rámce je zabezpečena protokolem WEP.
- ◆ **Doba trvání** – doba platnosti rámce. V případě rámců Power Save (PS-Poll) toto pole obsahuje identifikaci stanice.
- ◆ Rámce jsou číslovány tak, že pole Seq.control obsahuje číslo rámce (12 bitů). V případě, že je rámec fragmentován, pak všechny fragmenty mají téže číslo, ale liší se v čísle fragmentu (4 byty). U nefragmentovaných rámců je číslo fragmentu 0.
- ◆ Pole ADDR1 až ADDR4 obsahují linkové adresy, přitom v linkovém rámci může být uvedena 1 až 4 adresy.

IEEE 802.11 využívá stejný formát linkové adresy jako Ethernet, to znamená, že linkové adresy jsou rovněž 6 B dlouhé. Ale překvapením je, že zde máme 5 druhů linkových adres:

- ◆ Adresa příjemce (*Destination Address - DS*)
- ◆ Adresa odesilatele (*Source Address - SA*)
- ◆ Adresa stanice vysílající rámec (*Transmitting Station Address - TA*)
- ◆ Adresa stanice přijímající rámec (*Receiving Station Address - RA*)
- ◆ Identifikace konkrétního BSS jako celku (*Basic Services Set Identifier - BSS Id*). Zjednodušeně by se tato adresa dala označit jako „adresa konkrétního bezdrátového média“. Jako BSS Id se nejčastěji používá linková adresa AP.

## Ovládací rámce

Všeobecně známým problémem bezdrátových sítí je tzv. problém „skryté stanice“. Jeho princip spočívá v tom, že dvě stanice mohou být sice v dosahu společného přístupového bodu (AP), ale vzájemně se nevidí, tj. nejsou ve vzájemném dosahu. Důsledkem toho je, že nelze použít protokol CSMA/CD, ale „jen“ protokol CSMA/CA. Jeho princip je zjednodušeně následující:

- ◆ Stanice naslouchá, pokud je bezdrátové médium volné, pak ještě náhodný krátký čas vyčká.
- ◆ Poté stanice začne vysílat.

**Clear To Send (CTS), Acknowledgment (ACK):**

FC 2B	Doba trvání 2B	Receiving Station Address - RA 6B	FCS (CRC) 4B
----------	----------------------	---	-----------------

**Request To Send (RTS):**

FC 2B	Doba trvání 2B	Receiving Station Address - RA 6B	Transmitting Station Address - TA - 6B	FCS (CRC) 4B
----------	----------------------	---	--	-----------------

**Obrázek 4.32:** Formát rámců RTS, CTS a ACK

- ◆ Pokud přijímací stanice rámc přijme, zkontroluje kontrolní součet (CRC) a potvrdí přijetí rámce odesláním ovládacího rámce ACK (*Acknowledgement*).
- ◆ Pokud odesílací stanice nedodrží rámc ACK, pak opakuje vysílání.

Program Wireshark nám pak zobrazí rámc ACK následovně:

```
Radiotap Header v0, Length 24
IEEE 802.11
Type/Subtype: Acknowledgement (0x1d)
Frame Control: 0x00D4 (Normal)
Version: 0
Type: Control frame (1)
Subtype: 13
Flags: 0x0
Duration: 0
Receiver address: 00:13:02:91:2e:8b
Frame check sequence: 0xdfe385b5 [correct]
```

Zajímavé je, že rámc ACK nepotvrzuje příjem rámce konkrétního čísla. Jen říká: „pokračuj v komunikaci“.

Dále zde máme k dispozici jednoduchý mechanismus rezervace bezdrátového média:

- ◆ Stanice, která chce odeslat více rámců, může odeslat ovládací paket RTS (*Request To Send*), který žádá o rezervaci média na konkrétní časový interval.
- ◆ Cílová stanice odpoví řídicím rámcem CTS (*Clear To Send*) obsahujícím dobu, po kterou je médium rezervováno pro následující přenos.
- ◆ Stanice slyšící tuto komunikaci se na dobu specifikovanou v paketu CTS odmlčí.



**Poznámka:** Vzpomenete si, kde jsme se se zkratkami RTS a CTS už setkali?

## Řídicí rámc

Řídicí rámc jsou nejkomplikovanějšími rámc WLAN. Na obr. 4.33 je zobrazen pouze rámc spodní vrstvy, tj. obálka řídicího rámc. Tajemství řídicích rámců se totiž skrývá v poli DATA, tj. v datové části.



**Obrázek 4.33:** Řídicí rámc protokolu IEEE 802.11

Pole DATA nese vlastní obsah řídicího rámc (tj. nese *IEEE 802.11 wireless LAN management frame*), pomocí kterého jsou poskytovány jednotlivé služby WLAN. Jak vypadá pole DATA, tj. vnitřek toho řídicího rámc? Nikoho asi nepřekvapí, že každý subtyp má opět svůj tvar pole DATA. Ale přece. Obecně lze říci, že obsah pole DATA má pevné záhlaví skládající se z několika položek, následované modulární částí skládající se z jednotlivých modulů (*Tagged parameters*). Každý modul pak nese nějakou konkrétní informaci. Výrobci si dokonce mohou zařazovat i své vlastní moduly.

Obecně se každý modul skládá ze tří částí (obr.4.34): Identifikace modulu (*Tag Number*), Délky informace (*Tag Length*) a vlastní informace (*Tag Interpretation*).



**Obrázek 4.34:** Obecný tvar modulu (Tagged parametr)

Např. identifikace 0 je pro modul nesoucí jméno SSID, identifikace 1 je pro modul nesoucí podporované rychlosti atd.

### SSID (Service Set IDentifier)

Doposud jsme používali jen „hardwareovou“ identifikaci BSS ve tvaru linkové adresy, kterou je BSS ID. Nyní nám přibude ještě jméno SSID, které je 1 až 32 znaků dlouhé. Zdánlivě to vypadá, jako by toto jméno sloužilo je ke zjednodušení konfigurace – podobně jako jméno DNS. Ale pozor! Rozdíl pochopíme v okamžiku, kdy si uvědomíme, že jeden přístupový bod (AP) může podporovat více SSID současně. Výsledkem je, že v rámci jednoho fyzického AP můžeme vytvořit více virtuálních AP, tj. v rámci fyzicky jednoho bezdrátového média můžeme vytvářet více virtuálních médií. Můžeme tak vytvářet virtuální WLANy.



**Tip:** Na jedné infrastruktuře můžeme mít jeden zcela otevřený WLAN pro návštěvníky a druhý s autentizací IEEE 802.1X pro zaměstnance.

SSID je nesen ve stejnojmenném modu pole DATA.

### Majákové rámce (Beacon frame)

Tyto řídicí rámce vysílá přístupový bod v pravidelných intervalech. Součástí tohoto rámce je zejména časové razítko, které slouží k časové synchronizaci stanic v rámci BSS. Avšak tento rámec nese i další informace o WLAN, takže veřejnosti je chápán jako veřejné nabízení služeb přístupového bodu. Někteří se proto snaží potlačit vysílání těchto rámčů. Z bezpečnostního hlediska to ale nemá valný smysl, protože tytéž informace stejně získáte sledováním sítě.

Majákový rámec nese v pevném záhlaví:

- ◆ Časové razítko
- ◆ *Beacon interval* – od tohoto intervalu se pak odvozují další důležité časové intervaly protokolu IEEE 802.11.
- ◆ Přehled parametrů podporovaných AP – *Capability information*

V modulární části pak nese:

- ◆ SSID
- ◆ Podporované rychlosti
- ◆ Další parametry související s fyzickou vrstvou atd.

V následujícím výpisu rámce odchyceného programem Wireshark si všimněte, že Majákový rámec (*Beacon frame*) je opravdu vnořen do rámce IEEE 802.11, a také si všimněte posledního modulu, který tam vložil výrobce zařízení:

```
Radiotap Header v0, Length 24
IEEE 802.11
    Type/Subtype: Beacon frame (0x08)
    Frame Control: 0x0080 (Normal)
        Version: 0
        Type: Management frame (0)
        Subtype: 8
        Flags: 0x0
    Duration: 0
    Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
    Source address: 00:4f:62:0f:fc:89 (00:4f:62:0f:fc:89)
    BSS Id: 00:4f:62:0f:fc:89 (00:4f:62:0f:fc:89)
    Fragment number: 0
    Sequence number: 1227
    Frame check sequence: 0x6c1a2831 [correct]
IEEE 802.11 wireless LAN management frame
    Fixed parameters (12 bytes)
        Timestamp: 0x000008118D7E819B
        Beacon Interval: 0,512000 [Seconds]
        Capability Information: 0x0401
            .... .... .... .1 = ESS capabilities: Transmitter is an AP
            .... .... .... ..0. = IBSS status: Transmitter belongs to a BSS
            .... ..0. .... 00.. = CFP participation capabilities: No point
            .... .... ..0 .... = Privacy: AP/STA cannot support WEP
            .... .... .0 .... = Short Preamble: Short preamble not allowed
            .... .... 0.... .... = PBCC: PBCC modulation not allowed
            .... .... 0.... .... = Channel Agility: Channel agility not in use
            .... .... 0 .... .... = Spectrum Management: dot11SpectrumManagement
                                Required
            .... .1.. .... .... = Short Slot Time: Short slot time in use
            .... 0.... .... .... = Automatic Power Save Delivery: apsd not
                                implemented
            ..0. .... .... .... = DSSS-OFDM: DSSS-OFDM modulation not allowed
            .0... .... .... .... = Delayed Block Ack: delayed block ack not
                                implemented
            0.... .... .... .... = Immediate Block Ack: immediate block ack not
    Tagged parameters (46 bytes)
        SSID parameter set: "airlive"
            Tag Number: 0 (SSID parameter set)
            Tag length: 7
            Tag interpretation: airlive
        Supported Rates: 1,0(B) 2,0(B) 5,5(B) 11,0(B) 6,0 9,0 12,0 18,0
            Tag Number: 1 (Supported Rates)
            Tag length: 8
            Tag interpretation: Supported rates: 1,0(B) 2,0(B) 5,5(B) 11,0(B) 6,0
                                9,0 12,0 18,0 [Mbit/sec]
        DS Parameter set: Current Channel: 12
            Tag Number: 3 (DS Parameter set)
```

```

Tag length: 1
Tag interpretation: Current Channel: 12
Traffic Indication Map (TIM): DTIM 0 of 1 bitmap empty
Tag Number: 5 (Traffic Indication Map (TIM))
TIM length: 4
DTIM count: 0
DTIM period: 1
Bitmap Control: 0x00 (mcast:0, bitmap offset 0)
ERP Information: no Non-ERP STAs, do not use protection, long preambles
Tag Number: 42 (ERP Information)
Tag length: 1
Tag interpretation: ERP info: 0x4
Extended Supported Rates: 24,0 36,0 48,0 54,0
Tag Number: 50 (Extended Supported Rates)
Tag length: 4
Tag interpretation: Supported rates: 24,0 36,0 48,0 54,0 [Mbit/sec]
Vendor Specific: 00:e0:4c
Tag Number: 221 (Vendor Specific)
Tag length: 7
Vendor: 00:e0:4c
Tag interpretation: Not interpreted

```

### Otukávací rámce (Probe frames)

Otukávání je mechanismus, kterým se stanice dotazuje přístupového bodu (AP) na jeho možnosti. Mohli bychom také říci, že stanice tímto mechanismem žádá majákový rámec. Mechanismus používá dva rámce:

- ◆ *Probe Request* – rámec, kterým se stanice dotazuje přístupového bodu (AP). Tento rámec zpravidla obsahuje pouze SSID a podporované rychlosti.
- ◆ *Probe Response* – rámec, kterým AP odpovídá. Formát tohoto rámce je velice podobný majákovému rámci (*Beacon frame*).

### Asociace

Asociace je služba, kterou stanice žádá AP o asociaci s konkrétním SSID. O asociaci stanice žádá rámcem *Association Request*, AP následně odpovídá rámcem *Association Response*. Důležité je, že rámec *Association Response* obsahuje výsledkový kód (*Status code*), který vypovídá o tom, jestli asociace proběhla nebo nikoliv.

### Příklad

Příklad žádosti o asociaci s SSID jménem „airlive“ (Association Request):

```

Radiotap Header v0, Length 24
IEEE 802.11
IEEE 802.11 wireless LAN management frame
Fixed parameters (4 bytes)
    Capability Information: 0x0401
    Listen Interval: 0x000a
Tagged parameters (25 bytes)
    SSID parameter set: "airlive"
    Supported Rates: 1,0(B) 2,0(B) 5,5(B) 11,0(B) 6,0 9,0 12,0 18,0
    Extended Supported Rates: 24,0 36,0 48,0 54,0

```

### Příklad

Odpověď na předešlou žádost o asociaci (všimněte si, že odpověď obsahuje výsledkový kód, ale neobsahuje už jméno SSID):

```
Radiotap Header v0, Length 24
IEEE 802.11
IEEE 802.11 wireless LAN management frame
    Fixed parameters (6 bytes)
        Capability Information: 0x0401
        Status code: Successful (0x0000)
        Association ID: 0x00c1
    Tagged parameters (16 bytes)
        Supported Rates: 1,0(B) 2,0(B) 5,5(B) 11,0(B) 6,0 9,0 12,0 18,0
        Extended Supported Rates: 24,0 36,0 48,0 54,0
```

### Reasociace

Reasociace je služba, která umožňuje pohyb stanic mezi BSS v rámci ESS. Stanice žádá o reasociaci s novým AP pomocí rámce *Reassociation Request*. Tento rámec navíc obsahuje pole s adresou aktuálního AP. Odpověď je pak rámec *Association Response*, který opět obsahuje výsledkový kód.

### Disasociace

Je jen notifikace o disasociaci. Druhá strana disasociaci nemůže ani povolit, ani zamítнуть. Může ji vzít pouze na vědomí. Rámec *Disassociation* obsahuje pouze důvod disasociace.

### Autentizace

Autentizace je složitější problém, protože máme k dispozici několik typů autentizace. Součástí samotného standardu IEEE 802.11 jsou dva způsoby autentizace:

- ◆ Otevřená síť – bez autentizace.
- ◆ Autentizace pomocí sdíleného tajemství.

Autentizace (podobně jako asociace) končí rámcem obsahujícím výsledkový kód vypovídající o tom, jestli autentizace byla úspěšná, nebo nikoliv.

### Datové rámce

Nyní jsme se propracovali až k „obyčejným“ datovým rámcům určeným pro dopravu paketů protokolů vyšších vrstev. Datový rámec může obsahovat až 4 linkové adresy (obr. 4.31), zpravidla však obsahuje pouze 3 linkové adresy, ADDR1 až ADDR3. Přičemž naplnění polí ADDR1 až ADDR4 závisí na hodnotách příznaků „To DS“ a „From DS“:

To DS	From DS	ADDR1	ADDR2	ADDR3	ADDR4	Příklady využití
0	0	DA	SA	BSS Id	-	Ad Hoc sítě
0	1	DA	BSS Id	SA	-	Infrastrukturní sítě
1	0	BSS Id	SA	DA	-	Infrastrukturní sítě
1	1	RA	TA	DA	SA	Point-to-Point

Programem Wireshark si opět vypíšeme příklad – tentokrát datového rámce IEEE 802.11 (jedná se o opakované vysílání rámce):

### Příklad

```
Frame 129 (171 bytes on wire, 171 bytes captured)
Radiotap Header v0, Length 24
IEEE 802.11
    Type/Subtype: Data (0x20)
    Frame Control: 0x0908 (Normal)
        Version: 0
        Type: Data frame (2)
        Subtype: 0
        Flags: 0x9
            DS status: Frame from STA to DS via an AP (To DS: 1 From DS: 0) (0x01)
            .... .0.. = More Fragments: This is the last fragment
            .... 1... = Retry: Frame is being retransmitted
            ...0 .... = PWR MGT: STA will stay up
            ..0. .... = More Data: No data buffered
            .0.. .... = Protected flag: Data is not protected
            0.... .... = Order flag: Not strictly ordered
    Duration: 44
    BSS Id: 00:4f:62:0f:fc:89 (00:4f:62:0f:fc:89)
    Source address: 00:13:02:91:2e:8b (00:13:02:91:2e:8b)
    Destination address: 00:1a:92:ca:e6:b9 (00:1a:92:ca:e6:b9)
    Fragment number: 0
    Sequence number: 1145
    Frame check sequence: 0x85a920f0 [correct]
Logical-Link Control
Internet Protocol, Src: 10.0.0.1 (10.0.0.1), Dst: 192.35.17.14 (192.35.17.14)
User Datagram Protocol, Src Port: 2737 (2737), Dst Port: 53 (53)
Domain Name System (query)
```

## WEP

Protokol WEP (*Wired Equivalent Privacy*) je součást normy IEEE 802.11. Jeho cílem je zajistit soukromí autorizovaným uživatelům. Toho dociluje šifrováním komunikace.

Bezpečnost protokolu WEP implementovaného podle IEEE 802.11 je problematická, protože z odpo- slechnutých dat je možné šifru poměrně snadno zlomit pomocí běžně dostupných programů, jako je WEPcrack či AirSnort. Díky bezpečnostní díře v algoritmu tvorby klíčů zlomí program AirSnort šifru po jednodenním pasivním odposlechu na síti během několika sekund.



**Poznámka:** Velmi často se WLAN používá na tzv. Point-to-Multipoint připojení klientů do Internetu. V takovém případě si musí jednotliví účastníci připojení ke stejněmu připojněmu bodu (AP) uvědomit, že jsou jakoby na „společném segmentu Ethernetu“ a používají stejný sdílený klíč. Doporučujeme uživatelům, aby použili k ochraně svých dat před „sousedy“ další prostředky, např. IPsec či zabezpečení na aplikační vrstvě (SSL/TLS, S/MIME, SSH atd.).



**Poznámka:** Je sice pravdou, že bezpečnost protokolu WEP je teoreticky sporná, ale pro zabezpečení běžných uživatelů je mnohdy dostačující. Pokud máme pochybnosti o zabezpečení čehokoliv, tak je dobré provést alespoň jednoduchou analýzu rizik, tj. odpovědět si na otázku, zda cena námi přenášených dat je opravdu větší než náklady útočníka na příslušný útok. Do nákladů musíme pak zahrnout nejenom rádové tištíčekoruny za hardware, ale zejména cenu úsilí kvalifikovaného hackera.

## IEEE 802.1X

Firemní síť a rozsáhlejší bezdrátové sítě vůbec se zásadně zabezpečují pomocí protokolu IEEE 802.1X. Jedná se o standard pro autentizaci uživatelů v sítích skupiny IEEE 802. Protokol IEEE802.1X je jen obálkou, která balí protokol EAP (viz str. 93) do rámců rodiny protokolů IEEE 802. Jak již víme z kap. 4, protokol EAP je výhradně určen pro autentizaci, tj. nezabezpečuje (nešifruje) přenos datových rámců. Na druhou stranu ale během autentizace protokolem EAP (zejména protokolem EAP-TLS) dojde k výměně kvalitního kryptografického materiálu, který je následně možné využít k šifrování přenášených dat.

Protokol IEEE802.1X není určen pro zajištění malých sítí (např. domácích sítí). Vyžaduje totiž nějaký autentizační server – např. RADIUS server.

## IEEE 802.11i

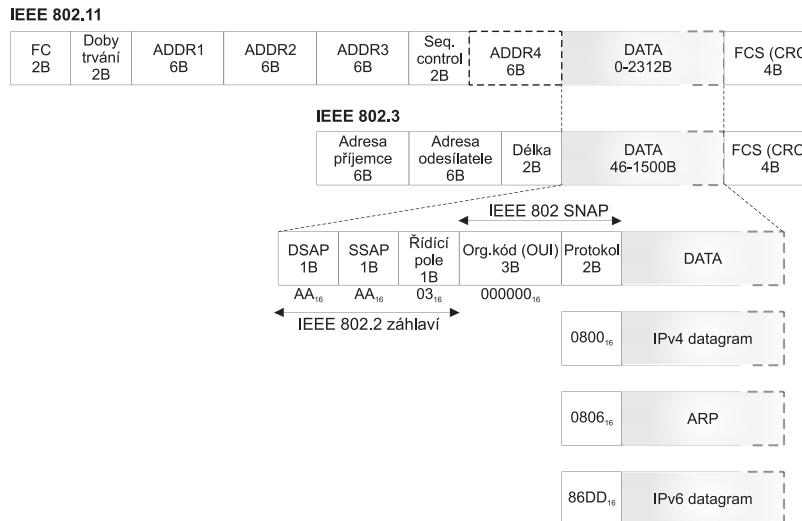
Tento dodatek standardu IEEE 802.11, využívající i IEEE802.1X, řeší bezpečnost WLAN komplexně: řeší autentizaci, soukromí (šifrování algoritmem AES) přenášených dat i integritu přenášených dat.

## LLC (IEEE 802.2)

Standard IEEE 802.2 má více než 250 stran. To je sice proti takovému IEEE 802.3 tenká knížečka, ale my z něj pro potřebu rodiny protokolů TCP/IP použijeme jen střípek. Standard totiž řeší i doručování rámců. My se potřebujeme dobrat k tomu, abychom v rámci mohli specifikovat protokol vyšší vrstvy, a tak v rámci jednoho média (drátového, optického, bezdrátového atd.) dopravovat pakety různých protokolů vyšších vrstev, např. protokolů ARP, IPv4, IPv6 atd.

Protokol IEEE 802.2 má rovněž své rámce, které se balí do transportních rámců jednotlivých protokolů IEEE 802.x (např. do rámců protokolů IEEE 802.3 nebo IEEE 802.11). Záhlaví IEEE 802.2 (viz obr. 4.35) připomíná záhlaví protokolu HDLC, oproti kterému ale má zase dvě adresní pole. Celkově má záhlaví IEEE 802.2 tři pole:

- ◆ DSAP (*Destination service access point address*) – adresa služby vyšší vrstvy příjemce. Protokol SNAP má např. vyhrazenu adresu AA<sub>16</sub>. Pole je dlouhé 1 B.
- ◆ SSAP (*Source service access point address*) – adresa služby vyšší vrstvy odesilatele. Pole je dlouhé 1 B.
- ◆ Řídicí pole, které je obdobou řídicího pole protokolu HDLC. Budeme používat jen U-rámce s bitem P/F nastaveným na 0, takže řídicí pole bude mít vždy hodnotu 03. Řídicí pole používáme jednobajtové.



Obrázek 4.35: Záhlaví IEEE 802.2 se SNAP, zapouzdřené do IEEE 802.11 nebo IEEE 802.3

Záhlaví IEEE 802.2 je sice pěkné, ale nepřineslo nám kýžené pole, do kterého bychom umístili identifikaci protokolu vyšší vrstvy. K tomuto účelu ve standardu IEEE 802 máme protokol SNAP (*Sub-network Access Protocol*). Na obr. 4.35 máme záhlaví i tohoto protokolu, které je 5 B dlouhé. První tři bajty mají stejný význam jako první tři bajty v linkové adrese, které jsou určeny pro identifikaci výrobce karty (*Organizationally Unique Identifier – OUI*). To je nám vcelku k ničemu, takže tyto bajty plníme hodnotou 000000, která je oficiálně pro tyto účely vyhrazena jako „Zapouzdřovaný Ethernet“. Teprve zbývající dva bajty konečně nesou identifikátor protokolu vyšší vrstvy. Dobrou zprávou je, že se používají stejné hodnoty identifikace jako pro protokol Ethernet II. Výsledkem tak je, že datová část protokolů IEEE 802.3, IEEE 802.11 atd. vždy začíná řetězcem AA-AA-03-00-00-00, tedy každopádně v případě, že přenáší protokoly rodiny TCP/IP.

Pro ukázkou si vypišme jeden rámec programem Wireshark:

```
Frame 81 (116 bytes on wire, 116 bytes captured)
Radiotap Header v0, Length 24
IEEE 802.11
Logical-Link Control
    DSAP: SNAP (0xaa)
    IG Bit: Individual
    SSAP: SNAP (0xaa)
    CR Bit: Command
    Control field: U, func=UI (0x03)
        000. 00.. = Command: Unnumbered Information (0x00)
        .... .11 = Frame type: Unnumbered frame (0x03)
    Organization Code: Encapsulated Ethernet (0x000000)
    Type: IPv6 (0x86dd)
Internet Protocol Version 6
Internet Control Message Protocol v6
```

## Domácí cvičení

Programem Wireshark odchytě rámce:

1. Protokolu Ethernet
2. Protokolu WiFi

Všimněte si, že s největší pravděpodobností jedny rámce budou obsahovat záhlaví IEEE 802.2 a druhé nikoliv. Proč?

(Protože na Ethernetu nejspíš odchytíte rámce protokolu Ethernet II a ten žádné záhlaví IEEE 802.2 nezná.)

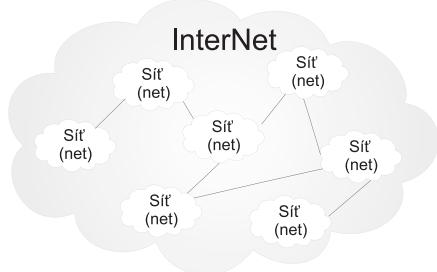
## Kapitola 5

# IPv4

Některé linkové protokoly jsou určeny pro dopravu dat v rámci lokální sítě, jiné linkové protokoly dopravují data mezi sousedními směrovači rozlehlé sítě. IP protokol na rozdíl od linkových protokolů dopravuje data mezi dvěma libovolnými počítači v Internetu, tj. i přes mnohé LAN.

Data jsou od odesilatele k příjemci dopravována (směrována) přes směrovače (*router*). Na cestě od odesilatele k příjemci se může vyskytnout celá řada směrovačů. Každý směrovač řeší samostatně směrování k následujícímu směrovači. Data jsou tak předávána od směrovače k směrovači. Z angličtiny se počeštil v tomto kontextu termín „následující hop“ (*next hop*) – jako následující uzel, kam se data předávají. Hopem se rozumí buď následující směrovač, nebo cílový stroj.

IP protokol je protokol umožňující spojit jednotlivé lokální sítě do celosvětového Internetu. Od protokolu IP dostal také Internet své jméno. Zkratka IP totiž znamená InterNet Protocol, tj. protokol propojující jednotlivé sítě. Později se místo InterNet začalo psát Internet a Internet byl na světě.



Obrázek 5.1: InterNet

IP protokol je tvořen několika dílčími protokoly:

- ◆ Vlastním protokolem IP.
- ◆ Služebním protokolem **ICMP** (*Internet Control Message Protocol*) sloužícím zejména k signifikaci mimořádných stavů.
- ◆ Služebním protokolem **IGMP** (*Internet Group Management Protocol*) sloužícím pro dopravu adresních oběžníků.
- ◆ Služebními protokoly **ARP** (*Address Resolution Protocol*) a **RARP** (*Reverse Address Resolution Protocol*), které jsou často vyčleňovány jako samostatné, na IP nezávislé protokoly, protože jejich pakety nejsou vkládány do IP datagramu, ale přímo do linkového rámce.

Zatímco v linkovém protokolu mělo každé síťové rozhraní (*network interface*) svou fyzickou (tj. linkovou) adresu, která je v případě LAN zpravidla šestibajtová, v IP protokolu má každé síťové

rozhraní alespoň jednu IP adresu, která je v případě IP protokolu verze 4 čtyřbajtová, v případě IP-protokolu verze 6 šestnáctibajtová.



**Obrázek 5.2:** Linková adresa a IP adresa stanice připojené k LAN

Základním stavebním prvkem rozlehlé sítě (WAN) je směrovač (anglicky *router*), kterým se vzájemně propojují jednotlivé LAN do rozlehlé sítě. Jako směrovač může sloužit běžný počítač s více síťovými rozhraními a běžným operačním systémem nebo specializovaná skříňka (*box*), do které nebyvá běžně zapojen ani monitor ani klávesnice. Tyto specializované skříňky se u nás v Česku mezi odbornou veřejností nazývají routery a v tiskovinách směrovače. Slovo směrovač má dva významy. V prvním obecném významu se směrovačem míní funkce počítače (ať klasického počítače nebo specializované skříňky) předávat datové pakety mezi dvěma síťovými rozhraními, v druhém, praktickém, smyslu se jím označuje specializovaná skříňka pracující jako směrovač.

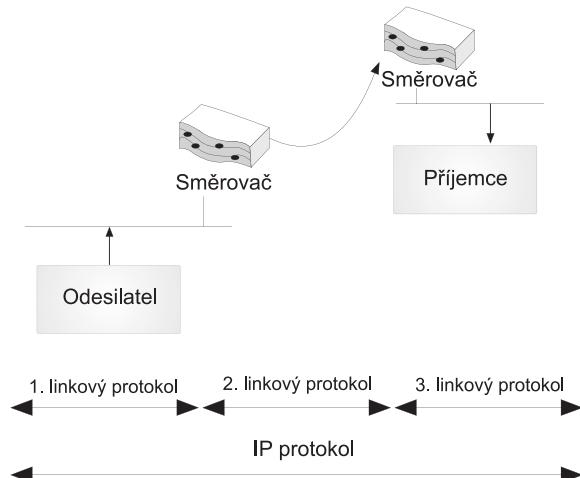
Schopnost předávat datové pakety mezi síťovými rozhraními směrovače se nazývá předávání (*forwarding*). Zatímco u směrovačů je tato funkce požadována, u počítačů s klasickým operačním systémem (UNIX, Windows apod.) se někdy vyskytne otázka, jak přinutit jádro operačního systému předávání zakázat.

Základní otázkou je: „Proč jsou třeba dva protokoly: linkový protokol a protokol IP? Proč nestačí pouze linkový protokol?“ Linkový protokol slouží pouze k dopravě dat v rámci LAN, tj. k dopravě IP datagramu k nejbližšímu směrovači. Ten z linkového rámce IP datagram „vybalí“ a „přebalí“ jej do jiného linkového rámce. Na každém rozhraní směrovače může být použit jiný linkový protokol. A nenechte se mylit případem, kdy směrovač na svých rozhraních používá stejný linkový protokol, např. Ethernet. I v tomto případě dochází k „přebalování“ – stačí si uvědomit, že ethernetový rámec používá před přebalením jiné fyzické adresy než po přebalení.

Pádným argumentem na otázkou „proč dva protokoly“ jsou zejména vlastnosti protokolů, které používají k dopravě dat pouze linkovou vrstvu, tj. jednotliví účastníci komunikace mají pouze linkové (šestibajtové) adresy. Takovými protokoly byly dnes již opuštěné protokoly NetBEUI (Microsoft) či LAT (Digital). Tyto protokoly byly jednoduché a opravdu asi rychlejší při tvorbě a zpracování paketů. Avšak díky tomu, že lze příjemce adresovat pouze v rámci LAN, tak těmito protokoly nebylo možné odeslat data příjemci za směrovačem – tj. ve WAN. Proto se tyto protokoly označovaly jako nesměrovatelné. Byly použitelné pouze v rámci lokální sítě, nikoliv mimo ni.

Obrázek 5.3 znázorňuje, že linkový protokol dopravuje datové rámce pouze k následujícímu směrovači, kdežto IP protokol dopravuje data mezi dvěma vzdálenými počítači rozlehlé sítě (WAN). Zatímco obálka (tj. záhlaví + zápatí), kterou jsou na linkové vrstvě data obalena, je na každém směrovači vždy zahozena a vytvořena nová, IP datagram není směrovačem změněn. Směrovač dokonce ani nesmí změnit obsah IP datagramu. Výjimkou je položka TTL (*Time To Live*) ze záhlaví IP datagramu. Položku TTL je naopak každý směrovač povinen zmenšit alespoň o jedničku, v případě změny na

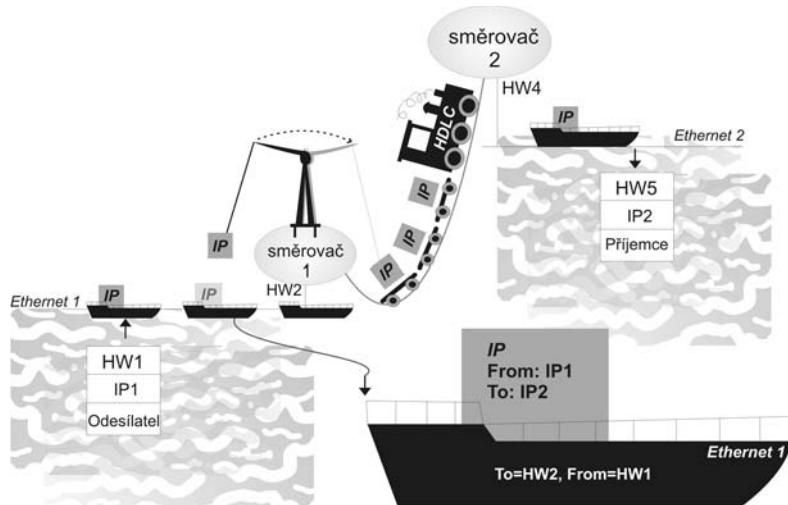
nulu se IP datagram zahazuje. Tímto mechanismem se Internet snaží zabránit nekonečnému toulání paketů Internetem. Existují i další výjimky, ke kterým se také později dostaneme (např. fragmentace nebo volitelné položky záhlaví).



**Obrázek 5.3:** Linkové protokoly a IP protokol

Zatímco u linkových protokolů jsme základní přenášené kvantum dat označovali jako **linkový rámec**, u IP protokolu je základní jednotkou přenášených dat **IP datagram**.

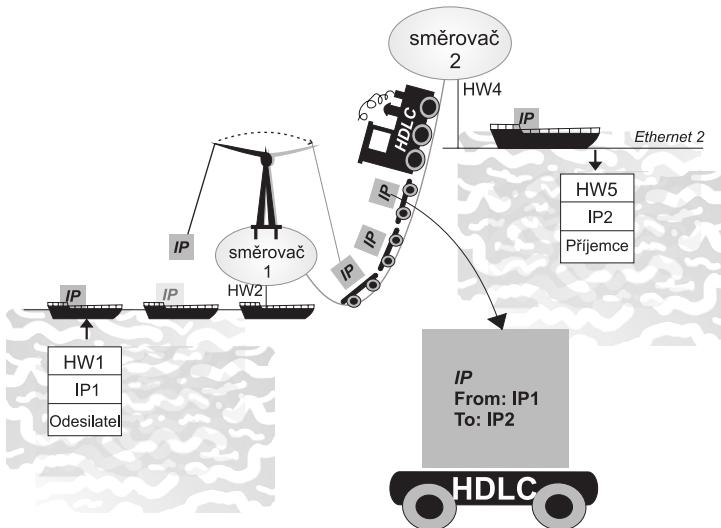
Proberme si případ z obr. 5.4, kdy odesilatel z lokální sítě **Ethernet 1** odesílá IP datagram příjemci na síti **Ethernet 2**. IP adresu odesilatele a příjemce jsme na obrázku 5.4 pro jednoduchost označili slovy **From** a **To**, jak je zvykem u elektronické pošty. Obdobně jsme označili i linkové adresy. Např. odesilatel má na obrázku linkovou adresu HW1.



**Obrázek 5.4:** Odesilatel odesílá IP datagram v rámci protokolu Ethernet

Odesilatel chce odeslat IP datagram příjemci s IP adresou IP2. Vytvoří IP datagram, ale aby jej mohl vložit do lokální sítě, musí jej vložit do linkového rámce (v našem případě Ethernet). Docela výstižné je přirovnání, že „IP datagram byl naložen na loď Ethernet 1“. Linkovým protokolem však mohou tato data putovat jen na **směrovač 1**, který IP datagram vybalí z ethernetového rámce a podívá se na IP adresu příjemce. Podle IP adresy příjemce se rozhodne, kterým svým rozhraním pošle IP datagram dále – tj. „na jaký linkový protokol se provede překladka IP datagramu“.

Rozhodování to však není jednoduché, směrovač se rozhoduje na základě svých **směrovacích tabulek (routing table)**, kterým se budeme také podrobně věnovat. Předpokládejme, že směrovač se rozhodl pro linku **HDLC**.

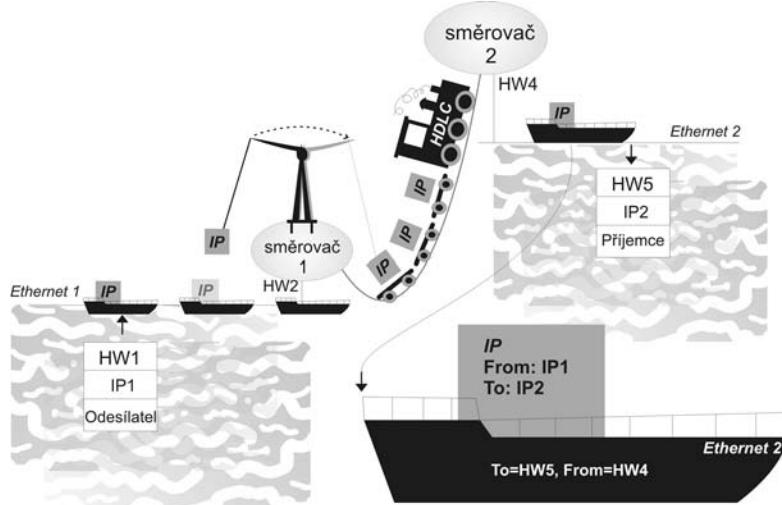


**Obrázek 5.5:** IP datagram byl vložen do rámce protokolu HDLC

Směrovač sníží hodnotu položky TTL alespoň o jedničku a vloží nás IP datagram do jiného linkového protokolu, kterým je v tomto případě protokol HDLC – viz obr. 5.5. Přirovnáme-li protokol HDLC k železniční dopravě, pak „nás IP datagram byl přeložen z lodi Ethernet 1 na vlak společnosti HDLC“.

Protokolem HDLC je nás IP datagram dopraven na následující směrovač, který opět IP datagram vybalí z HDLC-obálky, sníží hodnotu položky TTL a po obalení ethernetovou obálkou jej vloží do cílové LAN.

Záměrně jsem si na obou LAN vybral stejný linkový protokol (Ethernet) – aby bylo vidět, že pokud se jedná o zcela jiný linkový rámec. Na LAN odesilatele má ethernetový rámec adresu příjemce HW2 a odesilatel HW1, kdežto na LAN příjemce se sice také jedná o Ethernet, ale linková adresa příjemce je HW5 a odesilateli HW4.



Obrázek 5.6: IP datagram je opět vložen do rámce protokolu Ethernet

## IP datagram

Při výkladu protokolů TCP/IP je zvykem vše znázorňovat v tabulce, jejíž řádek má 4 bajty, tj. byty 0 až 31. I my budeme často používat toto znázornění.

IP datagram se skládá ze záhlaví a přenášených dat. Záhlaví má zpravidla 20 bajtů, ale může obsahovat i volitelné položky, které záhlaví prodlouží. Struktura IP datagramu je na obrázku 5.7.

0	8	16	24
Verze IP 4 bitů	Délka záhlaví	Typ služby 8 bitů	Celková délka IP datagramu 16 bitů
Identifikace IP datagramu 16 bitů	Příznaky (flags)	Posunutí fragmentu od počátku (fragment offset) - 13 bitů	
Doba života datagramu (TTL) - 8 bitů	Protokol vyšší vrstvy (protocol) - 8 bitů	Kontrolní součet z IP záhlaví (checksum) 16 bitů	
IP adresa odesílatele (source IP address) 32 bitů			
IP adresa příjemce (destination IP address) 32 bitů			
Volitelné položky záhlaví			
Přenášená data (nepovinné)			

Obrázek 5.7: IP datagram

Ještě než začneme popisovat jednotlivé položky záhlaví, odchytneme si nějaký IP datagram pomocí programu Wireshark.

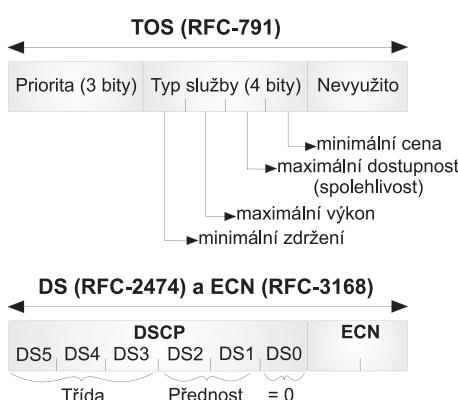
```
No.      Time      Source          Destination        Protocol Info
4 7.053899  10.0.0.138       10.0.0.1         ICMP      Echo (ping) reply

Frame 4 (74 bytes on wire, 74 bytes captured)
Ethernet II, Src: 00:1a:92:ca:e6:b9 (00:1a:92:ca:e6:b9), Dst: 00:13:02:91:2e:8b
(00:13:02:91:2e:8b)
Internet Protocol, Src: 10.0.0.138 (10.0.0.138), Dst: 10.0.0.1 (10.0.0.1)
    Version: 4
    Header length: 20 bytes
    Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
    Total Length: 60
    Identification: 0xf6cf (63183)
    Flags: 0x00
    Fragment offset: 0
    Time to live: 64
    Protocol: ICMP (0x01)
    Header checksum: 0x6f67 [correct]
    Source: 10.0.0.138 (10.0.0.138)
    Destination: 10.0.0.1 (10.0.0.1)
Internet Control Message Protocol
```

Jedině tak bude okamžitě vidět, zda síť chodí opravdu to, co popisujeme. Nyní již můžeme začít s popisováním významu jednotlivých položek záhlaví IP datagramu:

**Verze (version)** je první položkou záhlaví IP datagramu. Tato položka dlouhá 4 byty (půl bajtu) obsahuje verzi IP protokolu. V této kapitole hovoříme o IP protokolu verze 4 (IPv4), tudíž je tato položka v našem případě rovná hodnotě 4.

**Délka záhlaví (header length)** obsahuje délku záhlaví IP datagramu. V případě našeho odchyceného IP -datagramu je délka záhlaví 20 bajtů, jak je vidět z hexadecimálního výpisu z programu Wireshark. Pokud se podíváte v programu Wireshark do oblasti výpisu paketu, pak s překvapením zjistíte, že položka délka záhlaví nabývá hodnoty 5 (nikoliv 20). Vysvětlení je prosté. Délka se totiž neuvádí v bajtech, ale v čtyřbajtech a  $5 \times 4 = 20$ . Délka záhlaví musí tak být i v případě použití volitelných položek násobkem čtyř. V případě, že by záhlaví nevyšlo na násobek čtyř, doplní se na násobek čtyř nevýznamnou výplní.



**Obrázek 5.8:** Význam jednotlivých bitů položky Typ služby (nahoře podle RFC-791 a dole podle RFC-2474)

Maximální délka záhlaví IP datagramu je tedy omezena tím, že položka délka záhlaví má k dispozici pouze 4 byty ( $1111_2 = F_{16} = 15_{10}$ ). Délka záhlaví IP datagramu je tedy maximálně 60 B (=15x4). Jelikož povinné položky mají 20 B, zbývá na volitelné položky maximálně 40 B.

**Typ služby (type of service – TOS)** – tato položka měnila význam v průběhu let. Důvod je velice prostý. Pomocí těchto osmi bitů se pokouší řešit problém zajištění šířky přenosového pásma. Přitom v rodině protokolů TCP/IP je to opravdu problém. Již v první kapitole jsme uvedli, že rodina protokolů TCP/IP je založena na paketovém přenosu, který je přímo ze své podstaty protikladem protokolů garantujících šíři přenosového pásma.

Význam jednotlivých bitů položky Typ služby – TOS je na obr. 5.8. V horní části obrázku 5.8 je původní význam této položky podle RFC-791 a v dolní části je pak tato položka popsána pomocí novějších standardů RFC-2474 a RFC-3168.

Blíže se touto problematikou budeme zabývat na konci této kapitoly v odstavci QoS.

**Celková délka IP datagramu (total length)** obsahuje celkovou délku IP datagramu v bajtech. Jelikož je tato položka pouze dvoubajtová, je maximální délka IP datagramu 65 535 bajtů.

**Identifikace IP datagramu (identification)** obsahuje identifikaci IP datagramu, kterou do IP datagramu vkládá operační systém odesilatele. Tato položka je společně s položkami **příznaky (flags)** a **posunutí fragmentu (fragment offset)** využívána mechanismem fragmentace datagramu.

Do češtiny se názvy bitů pole příznaky překládají v negaci (viz obr. 5.9). Je-li bit DF nastaven na 1, je fragmentace zakázána. Nastavení na 0 naopak znamená, že fragmentace je možná. Je-li nastaven bit MF na jedničku, vyjadřuje, že není posledním fragmentem.

### 3 byty příznaků

0	DF	MF
---	----	----

DF - Don't Fragment (fragmentace možná)  
MF - More Fragments (poslední fragment)

**Obrázek 5.9:** Příznaky

**Doba života datagramu (time to live – TTL)** slouží k zamezení nekonečného toulání IP datagramu Internetem. Každý směrovač kladnou položku TTL snižuje alespoň o jedničku. Není-li už možné hodnotu snížit, IP datagram se zahazuje a odesilateli IP datagramu je tato situace signalizována protokolem ICMP.

Jak se hodnota položky TTL nastavuje? U příkazů ping a traceroute je možné ji explicitně nastavit. Obecně se však jedná o parametr jádra operačního systému, pokud ji tvůrci programu nenastaví explicitně. Od Windows 2000 lze TTL měnit klíčem DefaultTTL v registru HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters. Musíme si jej do registru přidat (je typu REG\_DWORD).

**Protokol vyšší vrstvy (protocol)** obsahuje číselnou identifikaci protokolu vyšší vrstvy, který využívá IP datagram ke svému transportu. V praxi se nesetkáváme s případem, že by se komunikovalo přímo IP protokolem. Vždy je použit protokol vyšší vrstvy (TCP nebo UDP) nebo jeden ze služebních protokolů ICMP či IGMP. Protokoly ICMP a IGMP jsou sice formálně součástí protokolu IP, avšak chovají se jako protokoly vyšší vrstvy, tj. v přenášeném paketu je záhlaví IP protokolu následováno záhlavím protokolu ICMP (resp. IGMP).

Čísla protokolů vyšších vrstev přiřazuje tvůrcům protokolů vyšších vrstev organizace IANA. Přiřazená čísla lze najít na <http://www.iana.org/numbers.html>. Pro zajímavost jsou některá čísla protokolů popisovaných v této publikaci uvedena v tab. 5.1.

**Tabulka 5.1:** Čísla některých protokolů vyšší vrstvy

Číslo protokolu vyšší vrstvy (desítkově)	Protokol
1	ICMP
2	IGMP
6	TCP
17	UDP
89	OSPF

Jako protokol vyšší vrstvy může být i protokol, který je tunelován přes Internet (*encapsulation*). Tunelováním se rozumí situace, kdy pakety jednoho síťového protokolu jsou vkládány jako data do paketů jiného síťového protokolu. Tunelovány mohou být např. protokoly, které Internet nepodporuje, jako je např. protokol IPX. Častým je i tunelování protokolu IPv6 přes síť IPv4. Nebo může být tunelován sám protokol IPv4 (*IPv4 over IPv4*), což se na první pohled jeví jako nesmyslné plýtvání. Avšak v případě, že přes Internet chceme přenášet data mezi dvěma částmi privátní sítě o adrese 10.0.0.0, je takový tunel nezbytností. Navíc je možné vnitřní IP datagramy zabezpečit šifrováním a vznikne nám tak jednoduchá virtuální privátní síť (VPN).

**Tabulka 5.2:** Čísla pro některé tunelované protokoly

Číslo protokolu vyšší vrstvy (desítkově)	Protokol
4	IP over IP
97	Ethernet within IP
111	IPX in IP

Pokud je třeba transportovat datagramy protokolu IP verze 6 přes síť podporující pouze IP protokol verze 4, pak také nezbývá opět nic jiného než tunelování. V tab. 5.2 jsou uvedena některá čísla pro tunelované protokoly.

**Kontrolní součet z IP záhlaví** (*header checksum*) obsahuje kontrolní součet, avšak pouze ze záhlaví IP datagramu, a nikoliv z datagramu celého. Jeho význam je tedy omezený. Bližší informace o výpočtu kontrolního součtu lze nalézt v normách RFC-1071 a RFC-1141.

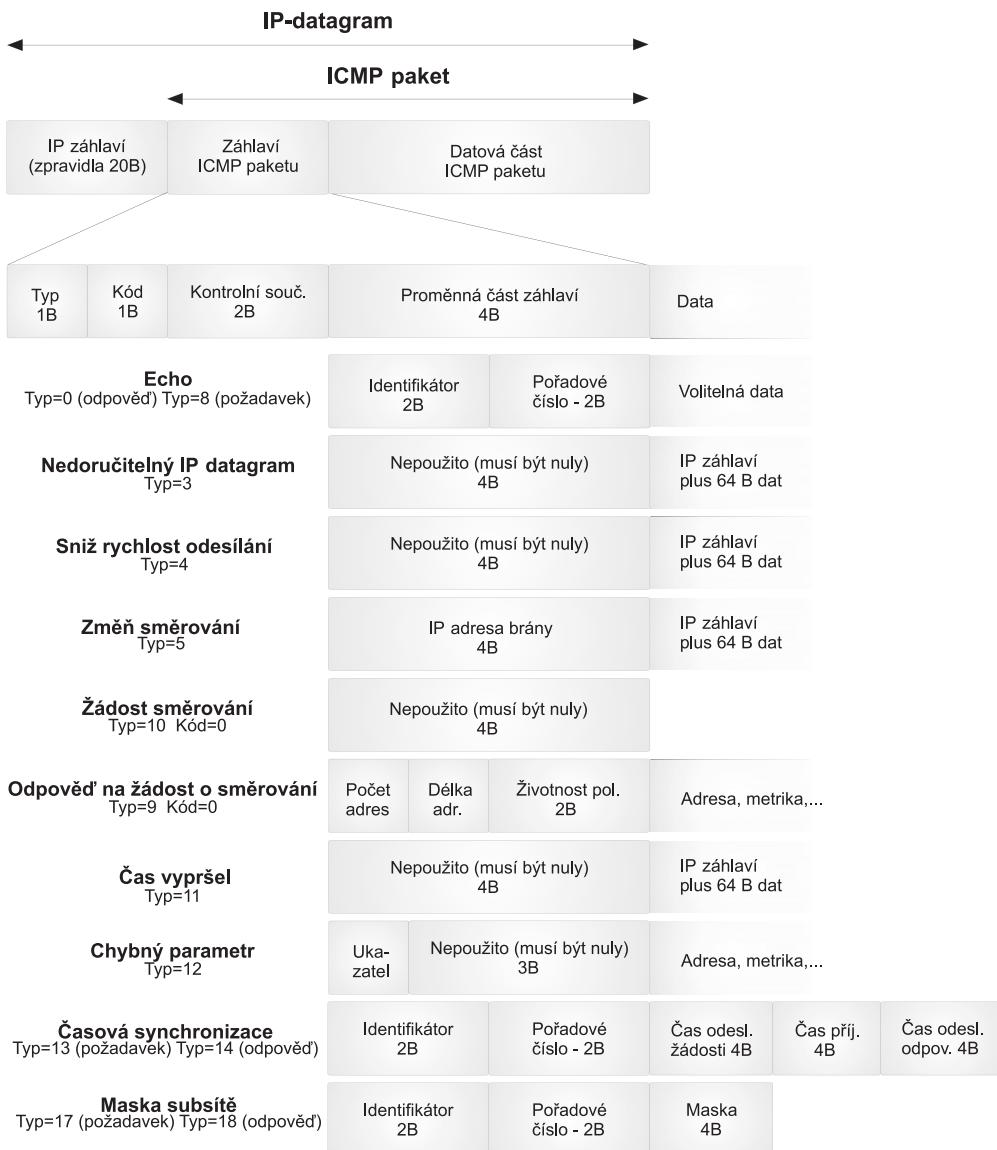
Problém s kontrolním součtem spočívá v tom, že když směrovač změní nějakou položku v záhlaví IP datagramu (např. TTL změnit musí), musí změnit i hodnotu kontrolního součtu, což vyžaduje jistou režii směrovače.

**IP adresa odesilatele a IP adresa příjemce** (*source and destination address*) obsahuje čtyřbajtovou IP adresu odesilatele a příjemce IP datagramu.

**Volitelné položky** jsou využívány ojediněle a směrovače bývají zpravidla nakonfigurovány tak, aby IP datagramy s použitými volitelnými položkami byly bez okolků zahrozeny. Volitelným položkám je věnována samostatná část (str. 145).

## Protokol ICMP

Protokol ICMP je služební protokol, který je součástí IP protokolu. Protokol ICMP slouží k signalizaci mimořádných událostí v síťích postavených na IP protokolu. Protokol ICMP svoje datové pakety balí do IP protokolu, tj. pokud budeme prohlížet přenášené datagramy, pak v nich najdeme za linekovým záhlavím záhlaví IP protokolu následované záhlavím paketu ICMP.



Obrázek 5.10: ICMP paket

Protokolem ICMP je možné signalizovat nejrůznější situace. Skutečnost je však taková, že konkrétní implementace TCP/IP podporují vždy jen jistou část těchto signalizací a navíc z bezpečnostních důvodů mohou být na směrovačích mnohé ICMP signalizace zahazovány.

Záhlaví ICMP paketu je vždy osm bajtů dlouhé (viz obr. 5.10). První čtyři bajty mají vždy stejnou strukturu a obsah zbylých čtyř závisí na typu ICMP paketu.

První čtyři bajty záhlaví obsahují vždy typ zprávy, kód zprávy a šestnáctibitový kontrolní součet. Formát zprávy závisí na hodnotě pole Typ. Pole Typ je hrubým dělení ICMP paketů. Pole Kód pak specifikuje konkrétní problém (jemné dělení), který je signalizován ICMP protokolem.

Přehled jednotlivých typů a kódů uvádí tabulka 5.3.

**Tabulka 5.3:** Přehled zpráv protokolu ICMP (OS=Operační systém)

Typ	Kód	Popis	Co signalizuje	Kdo zpracovává
0	0	Echo	Odpověď už. aplikaci	Uživ. aplikace
3	<b>Nedoručitelný IP datagram (Destination unreachable)</b>	0 Nedosažitelná síť ( <i>Network unreachable</i> ) 1 Nedosažitelný uzel ( <i>Host unreachable</i> ) 2 Nedosažitelný protokol ( <i>Protocol unreachable</i> ) 3 Nedosažitelný port protokolu UDP ( <i>Port unreachable</i> ) 4 Fragmentace zakázána, avšak pro další přenos by byla nutná ( <i>Fragmentation needed but don't fragment bit set</i> ) 5 Explicitní směrování selhalo ( <i>Source route failed</i> ) 6 Adresátova síť je neznámá ( <i>Destination network unknown</i> ) 7 Adresátův uzel je neznámý ( <i>Destination host unknown</i> ) 9 Adresátova síť je administrativně uzavřena ( <i>Destination network administratively prohibited</i> ) 10 Adresátův uzel je administrativně uzavřen ( <i>Destination host administratively prohibited</i> ) 11 Síť nedosažitelná pro uvedený typ služby ( <i>Network unreachable for TOS</i> ) 12 Uzel nedosažitelný pro uvedený typ služby ( <i>Host unreachable for TOS</i> ) 13 Komunikace administrativně uzavřena filtrování ( <i>Communication administratively prohibited by filtering</i> )	Chyba	Uživ. aplikace
4	0	<b>Sniž rychlosť odesílání (Source quench)</b>	Chyba	1. Jádro OS pro TCP 2. Zahazuje se pro UDP
5	<b>Změň směrování (Redirect)</b>	0 Změň směrování pro síť ( <i>Redirect for network</i> ) 1 Změň směrování pro uzel ( <i>Redirect for host</i> ) 2 Změň směrování pro síť pro daný typ služby ( <i>Redirect for TOS and network</i> ) 3 Změň směrování pro uzel pro daný typ služby ( <i>Redirect for TOS and host</i> )	Chyba	Jádro OS

Typ	Kód	Popis	Co signalizuje	Kdo zpracovává
8	0	<b>Žádost o echo (Echo request)</b>	Dotaz už. aplikace	Jádro OS
9	0	<b>Odpověď na žádost o směrování (router advertisement)</b>	Odpověď už. aplikaci	Uživ.proces
10	0	<b>Žádost o směrování (router solicitation)</b>	Dotaz už. aplikace	Uživ.proces
11	Čas vypršel (time exceeded)	0 Čas vypršel během transportu ( <i>TTL equals 0 during transit</i> ) 1 Vypršel čas na sestavení IP datagramu z jeho fragmentů ( <i>time to live equals 0 during reassembly</i> )	Chyba	Uživ.proces
12	Chybý parametr (parameter problem)	0 Chybné IP záhlaví ( <i>IP header bad</i> ) 1 Schází požadovaný volitelný parametr ( <i>required option missing</i> )	Chyba	Uživ.proces
13	0	<b>Požadavek na časovou synchronizaci (timestamp request)</b>	Dotaz už. aplikace	Uživ.proces
14	0	<b>Odpověď na časovou synchronizaci (timestamp reply)</b>	Odpověď už. aplikaci	Jádro OS
17	0	<b>Žádost o masku subsítě (address mask request)</b>	Dotaz už. aplikace	Uživ. proces
18	0	<b>Odpověď na žádost o masku (address mask reply)</b>	Odpověď už. aplikace	Jádro OS

Nyní se zastavme u jednotlivých typů zpráv.

## Echo

Je to jednoduchý nástroj protokolu ICMP, kterým můžeme testovat dosažitelnost jednotlivých uzlů v Internetu. Žadatel vysílá ICMP paket „Žádost o echo“ a cílový uzel je povinen odpovědět ICMP -paketem „Echo“.

Všechny operační systémy podporující protokol TCP/IP obsahují program ping, kterým uživatel může na cílový uzel odeslat žádost o echo. Program ping pak zobrazuje odpověď.

Význam pole identifikátor v záhlaví ICMP paketu spočívá ve spárování žádosti s odpovědí (aby se dalo zjistit, ke které žádosti patří příslušná odpověď).

Např. ve Windows chceme zjistit dostupnost uzlu 194.149.105.18:

```
D:\>ping 194.149.105.18
```

```
Pinging 194.149.105.18 with 32 bytes of data:
```

```
Reply from 194.149.105.18: bytes=32 time<10ms TTL=63
```

Systém odeslal čtyřikrát žádost o echo. Odpověď měla 32 bajtů dlouhou datovou část a získal ji do 10 ms. V odpovědi měla položka záhlaví TTL hodnotu 63.

### Nedoručitelný IP datagram

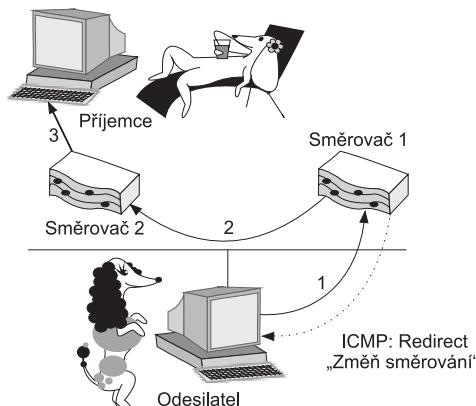
Nemůže-li být IP datagram předán dále směrem k adresátovi, pak je zahozen a odesilatel je o tom uvědomen protokolem ICMP zprávou „Nedoručitelný IP datagram“. Jednotlivé důvody jsou uvedeny v tabulce 5.3.

### Sniž rychlosť odesílání

Jestliže je síť mezi odesilatelem a příjemcem v některém místě přetížena, pak směrovač, který není schopen předávat dále všechny IP datagramy, signalizuje odesilateli „Sniž rychlosť odesílání“. Odesilatel pak v případě, že používá protokol TCP, snižuje rychlosť odesílání TCP segmentů. V případě protokolu UDP se zprávy „Sniž rychlosť odesílání“ ignorují. S touto zprávou jsme se setkali u protokolu FrameRelay (viz obr. 4.24).

### Změň směrování (Redirect)

Pomocí tohoto ICMP paketu se provádí dynamické změny ve směrovací tabulce.



**Obrázek 5.11:** Redirect

Na obr. 5.11 má směrovač 1 předávat IP datagram na stejné síťové rozhraní, kterým IP datagram přišel, pak jej sice předá, avšak upozorní adresáta ICMP paketem „Redirect“, aby si změnil vlastní směrovací tabulkou a takovou podivnou službu už více nežádal.

Tato situace nejčastěji nastává tehdy, když máme na lokální síti více směrovačů, ale jednotlivé počítače na LAN mají po svém startu pouze položku *default* ukazující na jeden ze směrovačů.

## Žádost o směrování

Jedná se o poměrně novou záležitost, pomocí které nemusíme do směrovací tabulky počítačů na LAN ručně konfigurovat vůbec žádnou položku *default*. Počítač po svém startu vyšle oběžníkem „Žádost o směrování“ a směrovač mu odpoví ICMP paketem „Odpověď na žádost o směrování“, která obsahuje: počet adres směrovače, délku adresy a pak dvojice IP adresa a preference. Z odpovědi může počítač automaticky vygenerovat položku *default*.

Čím má preference vyšší hodnotu, tím je IP adresa více preferována. Hodnota preference 8000000016 signalizuje, že tato adresa se má ze směrovací tabulky vypustit.



**Poznámka:** V oběžnících „Odpověď na žádost o směrování“ se adresuje buď všeobecným oběžníkem (255.255.255.255), nebo adresným oběžníkem „Všem systémům LAN“ (224.0.0.1).

Směrovače odpovídají na žádost o směrování, avšak v náhodném intervalu mezi 450 a 600 sekundami by měly oběžníkem samy do lokální sítě generovat ICMP pakety „odpověď na žádost o směrování“.

Položka „doba života“ udává čas, po který je informace platná, tj. po který má být položka ve směrovací tabulce udržována.

## Čas vypršel (time exceeded)

Tento typ zahrnuje dva velmi odlišné případy.

- ◆ Pro Kód=0 signalizuje, že položka TTL by byla na směrovači snížena na nulu, tj. že je podezření, že IP datagram v Internetu zabloudil, proto bude zlikvidován.
- ◆ Pro Kód=1 signalizuje, že počítač adresáta není schopen v daném čase sestavit z fragmentů celý IP datagram.

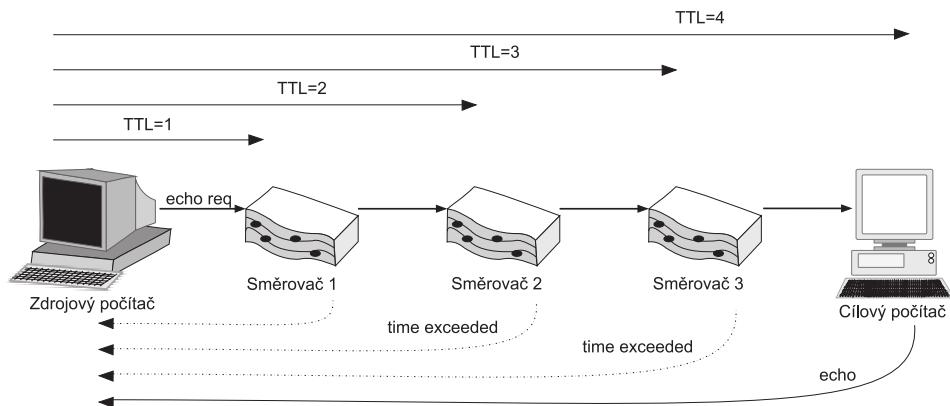
ICMP paket čas vypršel Kód=0 využívá ke své činnosti program traceroute (UNIX) i program tracert (Microsoft).

Program tracert odesílá ze zdrojového počítače na cílový počítač ICMP pakety „Žádost o echo“, avšak v prvním paketu nastaví položku TTL na jedničku. První směrovač na cestě paket zahodí a vrátí ICMP paket „Čas vypršel“, protože musí zmenšit TTL alespoň o jedničku, ale tímto zmenšením už dostane nulu.

Zdrojový počítač tak od prvního směrovače na cestě dostane v IP datagramu ICMP paket „Čas vypršel“. Z položky adresa odesilatele v IP záhlaví lze zjistit adresu prvního směrovače na cestě. Změří se časový interval od odeslání po příjem paketu a zjistí se tak čas procházky paketu od odesilatele k příjemci a zpět. Toto se opakuje třikrát a všechny tři časy se zobrazí. Na konec řádku ještě zobrazí jméno směrovače a v hranatých závorkách jeho IP adresu. Jméno získá z reverzního překladu v DNS.

Nezíská-li v časovém limitu odpověď, zobrazí místo času hvězdičku (\*).

Poté vše opakuje s hodnotou TTL=2 atd. Svou činnost ukončí, když od cílového uzlu obdrží ICMP -zprávu „Echo“. K ukončení může pochopitelně také dojít, když nějaký směrovač nezná cestu k cílovému počítači; pak zdrojovému počítači zašle zprávu „Nedoručitelný IP datagram“.



**Obrázek 5.12:** Příkaz tracert

```
D:\>tracert kula.usp.ac.fj
Tracing route to kula.usp.ac.fj [144.120.8.11] over a maximum of 30 hops:
```

```
1 <10 ms 10 ms <10 ms cbuN002e00.pvt.net [194.149.104.193]
2 10 ms 10 ms 10 ms phucbu.pvt.net [194.149.96.13]
3 601 ms 561 ms 641 ms 951.Hssi5-0.GW1.NYC2.ALTER.NET [157.130.0.117]
4 591 ms 571 ms 571 ms 143.ATM2-0.XR1.EWR1.ALTER.NET [146.188.177.50]
5 591 ms 581 ms 571 ms 193.ATM1-0-0.BR1.EWR1.ALTER.NET [146.188.176.49]

6 400 ms 381 ms 360 ms sl-pen-11-h3.sprintlink.net [137.39.44.130]
7 811 ms 591 ms 661 ms sl-bb10-pen-0-1.sprintlink.net [144.232.5.5]
8 500 ms 651 ms 731 ms sl-bb22-stk-6-0.sprintlink.net [144.232.8.178]
9 871 ms 831 ms 932 ms sl-bb23-stk-8-0.sprintlink.net [144.232.4.110]
10 691 ms 650 ms 611 ms sl-bb10-sj-6-0.sprintlink.net [144.232.8.193]
11 811 ms 771 ms 771 ms sl-gw2-sj-0-0-155M.sprintlink.net [144.232.3.38]
12 641 ms 651 ms 641 ms sl-cais-1.sprintlink.net [144.228.111.18]
13 801 ms 811 ms 861 ms hssi9-0-0.hk-T3.hkt.net [202.84.128.253]
14 801 ms * 811 ms f5-0.yck06.hkt.net [205.252.130.201]
15 821 ms 831 ms 822 ms a6-0.tmh08.hkt.net [205.252.130.81]
16 1402 ms 1342 ms 1362 ms s4-3b.tmh08.hkt.net [205.252.128.158]
17 1381 ms 1362 ms 1352 ms 202.84.251.6
18 1362 ms 1362 ms 1352 ms 202.62.120.6
19 1422 ms 1372 ms 1392 ms 202.62.125.134
20 1412 ms 1382 ms 1412 ms kula.usp.ac.fj [144.120.8.11]
```

Trace complete.

Program traceroute pracuje na obdobném principu, avšak neodesílá ICMP pakety „Echo request“, ale generuje protokolem UDP datagramy (UDP port je možné modifikovat parametrem -p).



**Poznámka:** Je-li na cestě k cílovému počítači použita filtrace na směrovači, pak vhodnou volbou čísla UDP portu lze mnohdy nalézt „díru“ ve filtru a objevit cestu až k cílovému počítači. Dobrým typem je pro takový případ číslo portu 53 (-p 53), který používá DNS.

```
$ /usr/sbin/traceroute -p 20000 libor.pvt.net
traceroute to libor.pvt.net (194.149.104.198), 30 hops max, 40 byte packets
1 cbuN003f00.pvt.net (194.149.105.17) 1 ms 1 ms 1 ms
2 Libor.pvt.net (194.149.104.198) 1 ms 1 ms 1 ms
```

Cílový počítač zpravidla odpoví ICMP paketem „Port unreachable“ (typ=3, kód=3). Kromě času a hvězdičky program traceroute ještě může vypsat !H (nedostupný uzel), !N (nedostupná síť), !A (síť administrativně uzavřena) či !S (explicitní směrování selhalo).

## Žádost o masku

Tímto ICMP paketem může bezdisková stanice žádat o masku své sítě poté, co protokolem RARP obdržela svou IP adresu.

Tento mechanismus je v praxi dnes již málo běžný. Stanice může získat masku své sítě protokolem BOOTP, kterým získá i další informace. Avšak i protokol BOOTP je dnes vytlačován protokolem DHCP, který je komplexnější, tj. poskytuje více informací. Protokoly BOOTP a DHCP jsou aplikace-ní protokoly.

## Časová synchronizace

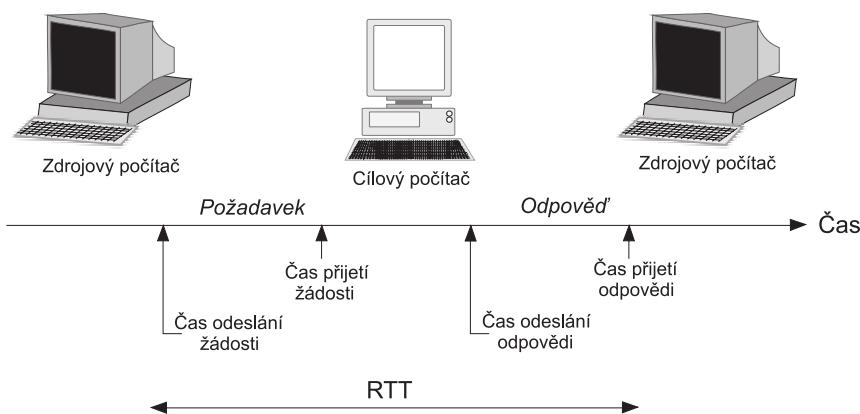
Tímto ICMP paketem se žádá cílový počítač o čas. Mechanismus je znázorněn na obr. 5.13.

Zdrojový počítač do ICMP paketu „**Požadavek na časovou synchronizaci (timestamp request)**“ vyplní čas odeslání žádosti.

Cílový počítač vyplní do své odpovědi „**Odpověď na časovou synchronizaci (timestamp reply)**“ dva časy:

- ◆ Čas přijetí žádosti.
- ◆ Čas odeslání odpovědi.

Zdrojový počítač si zjistí čas přijetí odpovědi (ten se pochopitelně nepřepravuje v žádném ICMP-paketu). Odečtením času odeslání požadavku od času přijetí odpovědi se získá doba procházky od zdrojového počítače k cílovému a zpět (anglicky *Round Trip Time – RTT*).



**Obrázek 5.13:** Časová synchronizace

Čas se udává v milisekundách od poslední půlnoci Světového času – GMT.

## Fragmentace

IP datagramy jsou baleny do linkových rámců. Linkové protokoly umožňují přenášet ve svých datových rámcích data pouze do určité maximální velikosti. Tato maximální velikost dat, kterou lze vložit do jednoho linkového rámce, se označuje MTU (*Maximum Transfer Unit*).

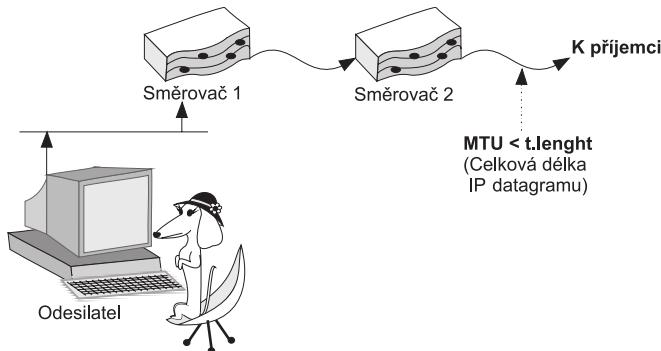
**Tabulka 5.4:** MTU pro některé linkové protokoly

Linkový protokol	MTU
Ethernet	1500
WiFi	2312
FDDI	4478

Z tabulky 5.4 je zřejmé, že linkové protokoly mají nejčastěji MTU řádově v jednotkách KB. Na linkách spojujících vzdálené lokality se někdy dokonce setkáváme s MTU menším než 1 KB. Pole celková délka IP datagramu je však dlouhé 16 bitů, takže teoreticky je možné vytvořit IP datagram až 64 KB dlouhý.

Co se však stane, když IP datagram na své pouti od odesilatele k příjemci dorazí na směrovač (na obrázku 5.14 směrovač 2), z něhož směrem k příjemci vede linka, která má menší MTU než je velikost našeho IP datagramu?

Směrovač není schopen takový IP datagram poslat dále. Směrovač se rozhoduje co dále na základě



**Obrázek 5.14:** Délka IP datagramu < MTU

příznaku „Fragmentace možná“ (*DF bit*) v záhlaví IP datagramu (ponecháváme stranou možnost, že k příjemci vede ještě jiná linka, byť s horší metrikou). Příznak „Fragmentace možná“ může být buď nastaven, nebo ne. Jsou tedy dvě možnosti:

1. Fragmentace je možná, pak se provede fragmentace, jak je popsáno dále v této kapitole.
2. Fragmentace není možná, pak směrovač IP datagram zahodí a odesílatele o tom informuje ICMP signalizací „Fragmentace zakázána, avšak pro další přenos by byla nutná (*Fragmentation needed but don't fragment bit set*)“.

Zakážeme-li příznakem (*DF-bit*) fragmentaci, můžeme zjistit, jaké nejmenší MTU je mezi odesilateli a příjemcem, tj. jak velké IP datagramy nebude nutné fragmentovat.

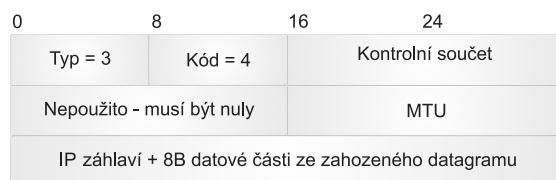
Může nám k tomu posloužit např. příkaz ping. Implementace příkazu ping firmy Microsoft umožňuje zakázat fragmentaci pomocí parametru *-f* a nastavit délku IP datagramu pomocí parametru *-l*. Takže příkaz

```
C:\> ping -f -l 2000 příjemce
```

buďto sdělí, že příjemce je funkční, a zobrazí nám čas procházky odesilatel-příjemce-odesilatel (RTT), nebo naopak zobrazí chybovou hlášku, takže se dozvíme, zdali na cestě byla nutná fragmentace pro IP datagram dlouhý 2 000 B. V případě, že fragmentace byla nutná, můžeme velikost odesílaného IP datagramu zmenšit a pozorovat, zda tentokrát bude fragmentace nutná, či nikoliv. Tak můžeme postupovat tak dlouho, až zjistíme hranici, od které je fragmentace nutná.

Podstatně jednodušší by bylo, kdyby ICMP signalizace obsahovala hodnotu MTU, která platí pro inkriminovanou linku. Původně se s touto možností nepočítalo, avšak později byl ICMP paket pro tento případ doplněn o pole MTU. Tato možnost je jen zřídka implementována.

V ICMP paketu byly využity druhé dva bajty z nevyužitých čtyř bajtů záhlaví. Struktura ICMP paketu je znázorněna na obr. 5.15.



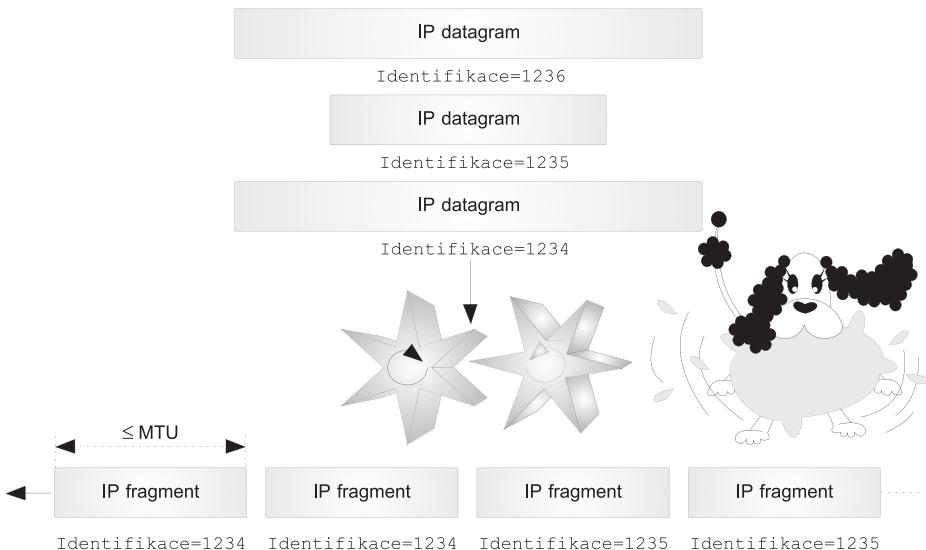
**Obrázek 5.15:** Rozšíření ICMP zprávy „fragmentace zakázána“

Nyní se vraťme k situaci, kdy je v IP paketu nastaveno, že fragmentace je možná. Směrovač pak dělí delší IP datagramy na fragmenty, jejichž celková délka (*total length*) je menší nebo rovná MTU následující linky – viz obr. 5.16.

Každý IP datagram má ve svém záhlaví svou identifikaci, kterou dědí i jeho fragmenty. Díky identifikaci příjemce pozná, ze kterých fragmentů má datagram složit. Nikdo jiný než příjemce není oprávněn z fragmentů skládat původní datagram, tj. ani např. směrovač, ze kterého vede linka s takovým MTU a do kterého by se celý datagram již vešel. Důvod je prostý: Internet negarantuje, že jednotlivé fragmenty půjdou stejnou cestou (ani negarantuje pořadí, v jakém dojdou). Takže směrovač, který by se pokoušel datagram sestavit, by mohl být na závadu spojení, protože fragmentů, které by šly jinou cestou, by se nikdy nedočkal.



**Poznámka:** Žádný směrovač na cestě není oprávněn sestavovat IP datagramy z jednotlivých fragmentů.



**Obrázek 5.16:** Kroužní IP datagramu na fragmenty

Identifikace IP datagramů může být jednoznačná pouze v rámci jednoho protokolu vyšší vrstvy, protože záhlaví IP datagramu obsahuje ještě pole „Protokol vyšší vrstvy“. Globální identifikace může být brána jako zřetězení polí identifikace a protokol vyšší vrstvy (+ pochoopitelně IP adresa odesilateli a příjemce). Teoreticky tak mohou být za sebou odeslány dva IP datagramy o stejné identifikaci, jeden však nese TCP paket a druhý UDP paket.

Každý fragment tvoří samostatný IP datagram. Při fragmentaci je nutné vytvořit pro každý fragment nové IP záhlaví. Některé údaje (jako protokol vyšší vrstvy či IP adresa odesilatele a příjemce) se získají ze záhlaví původního IP datagramu.

Při fragmentaci vstupuje do hry pole „Posunutí fragmentu od počátku IP datagramu (*fragment offset*)“, které vyjadřuje, kolik bajtů datové části původního IP datagramu bylo vloženo do předchozích fragmentů. Pole „Celková délka IP datagramu (*total length*)“ obsahuje délku fragmentu, nikoliv délku původního IP datagramu. Aby příjemce poznal, jak je původní datagram dlouhý, je poslední fragment opatřen příznakem „Poslední fragment“. Celý mechanismus je znázorněn na obr. 5.17.

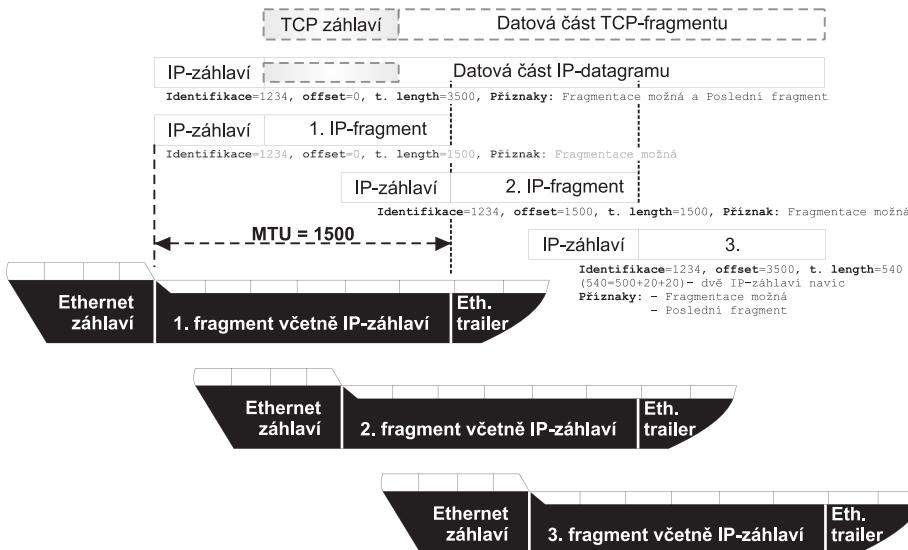
Síť nerozlišuje mezi přenosem fragmentu a přenosem celého (nefragmentovaného) IP datagramu. Nefragmentovaný IP datagram není nic jiného než fragment s posunutím nula a příznakem „Poslední fragment“. Proto se často slova IP datagram a fragment zaměňují.

Mechanismus fragmentace umožňuje i fragmentovat fragment, dorazí-li na směrovač, jehož odchozí linka má ještě menší MTU.

Důležité je, že každý další fragment znamená zatížení o minimálně 20 B jeho záhlaví. Pro zajímavost je na obrázku 5.17 znázorněn také TCP segment, který je vložen do IP datagramu. Co je na tom zajímavého?



**Upozornění:** Záhlaví TCP segmentu je dopravováno pouze v prvním IP fragmentu!



**Obrázek 5.17:** Fragmentace IP datagramu

Zajímavé je to, že TCP záhlaví je obsaženo pouze v prvním IP fragmentu. Takže pokud se provádí na směrovači filtrace IP datagramů na základě nejen informací z IP záhlaví, ale též na základě informací z TCP záhlaví, lze filtrovat pouze první fragment a ostatní se propouští. Příjemce pak po určeném časovém intervalu zjistí, že mu schází první fragment z IP datagramu, a signalizuje to příjemci ICMP zprávou „Vypršel čas na sestavení IP datagramu z jeho fragmentů (*time to live equals 0 during reassembly*)“. Při filtraci TCP paketů je tedy nutné nezapomenout v protisměru filtrovat i tyto ICMP pakety, pokud útočníkovi nechceme poskytnout informaci o tom, že se chráníme filtrací.

Fragmentace je považována za jakési nutné зло. Aplikace vyžadující extrémně bezpečnou komunikaci fragmentaci zakazují.

## Volitelné položky IP záhlaví

Volitelné položky v záhlaví IP datagramu patří k zajímavostem protokolů TCP/IP. Ukážeme si, jak nebezpečné může být využití těchto položek a proč mnozí poskytovatelé Interentu IP datagramy s některými volitelnými položkami zahazují. Z hlediska protokolu TCP/IP je však toto jednání poskytovatelů neomluvitelné (byť v dobré víře) a lze je přirovnat k doporučení, aby každý občan s sebou nosil berly pro případ, že by si zlomil nohu.

Pokud příjemce obdrží IP datagram s některou z voleb, měl by v odpovědi rovněž použít tuto volbu. Volitelné položky rozšiřují IP záhlaví. Vzhledem k omezené délce IP záhlaví na 60 B (z čehož je 20 B povinných) jsou volitelné položky shora omezeny 40 B. Tč. existuje několik možností rozšíření IP záhlaví:

1. Zaznamenávej směrovače (*record route*).
2. Zaznamenávej čas (*timestamp*).

3. Explicitní směrování (*loose source routing*).
4. Striktní explicitní směrování (*strict source routing*).
5. Upozornění pro směrovač (*IP Router Alert Option*).
6. Bezpečnostní omezení podle normy RFC-1108.

Volitelné položky v IP záhlaví následují povinné položky. Volitelné položky mají obecně formát znázorněný na obr. 5.18.



**Obrázek 5.18:** Volitelné položky záhlaví IP datagramu

Pokud je bit **Kopírovat** nastaven na 1, pak tato volitelná položka má být kopírována do všech fragmentů vzniklých z tohoto IP datagramu. Je-li bit nastaven na 0, kopíruje se pouze do záhlaví prvního fragmentu.

Dva bity tvořící pole **Třída** nabývají:

- ◆ Hodnoty 0 v případě, že se jedná o IP datagram nesoucí běžná data nebo data určená pro řízení sítě.
- ◆ Hodnoty 2 (=10<sub>2</sub>) v případě, že se jedná o IP datagram sloužící k ladění nebo měření sítě.

Pole **Číslo volby** pak specifikuje konkrétní volbu. Často používané kódy jsou uvedeny v tabulce 5.5.

**Tabulka 5.5**

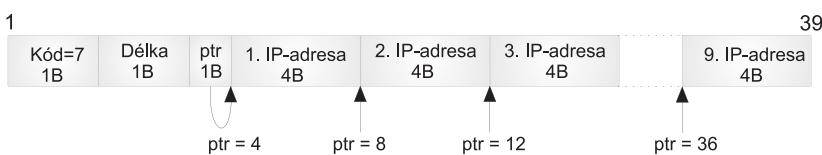
Kód	Šestnáctkově	Desítkově	Délka	Volba
0 00 00000	00	0	Není	Konec voleb ( <i>End of options list</i> ). Použije se v případě, že volby nekončí s koncem IP záhlaví. Pole délka a data se nepoužijí.
0 00 00001	01	1	Není	Prázdná volba – výplň záhlaví na násobek čtyř bajtů. Pole délka a data se nepoužijí.
0 00 00111	07	7	Proměnná	Zaznamenávej směrovače ( <i>Record route</i> ).
0 10 00100	44	68	Proměnná	Zaznamenávej čas ( <i>Timestamp</i> ).
1 00 00011	83	131	Proměnná	Explicitní směrování ( <i>Loose source routing</i> ).
1 00 01001	89	137	Proměnná	Explicitní striktní směrování ( <i>Strict source routing</i> ).
1 00 10100	94	148	4	Upozornění pro směrovač ( <i>IP Router Alert Option</i> ).



**Poznámka:** Na všechny volitelné položky IP záhlaví zbývá jen 40 bajtů.

## Zaznamenávej směrovače

Obsahuje-li záhlaví volbu s vyplněným polem kód=7, všechny směrovače na cestě k příjemci IP -datagramu doplní do IP záhlaví IP adresu svého výstupního rozhraní. Jednotlivá čtyřbajtová pole v záhlaví IP datagramu pro IP adresy se nazývají sloty. Do IP záhlaví je možno vložit až 9 slotů pro IP adresy.



**Obrázek 5.19:** Volitelná položka záhlaví „Zaznamenávej směrovače“

Pole délka obsahuje celkovou délku rozšíření „Zaznamenávej směrovače“ a pole ptr ukazuje na první volný slot, který je možno vyplnit (další směrovač vždy zapíše novou IP adresu a zvětší položku ptr o 4).

Datagram na své pouti od odesilatele k příjemci nasbírá do svých slotů odchozí IP adresy. Pokud odesilatelův stroj rovněž podporuje tuto volbu, pak ji ve své odpovědi rovněž použije, a navíc do odesílaného datagramu nejprve zkopíruje všechny sloty z datagramu přijatého.

Příkazem ping podporujícím volbu „zaznamenej směrovače“ zjistíme tedy seznam odchozích adres nejen při cestě datagramu od odesilatele k příjemci, ale i cestou zpět.

V příkazu ping implementovaném firmou Microsoft lze rozšíření „zaznamenávej směrovače“ vytvořit pomocí parametru -r následovaného počtem vytvořených slotů. Např.

```
D:\>ping -r 5 ns.pvt.net
```

Vygeneruje ICMP paket s rozšířením „zaznamenávej směrovače“ s pěti sloty pro IP adresy. Přičemž odesilatel vytvoří sice pět slotů, ale ani jeden není naplněn, takže ukazatel prvního volného slotu ptr ukazuje na první slot.

IP záhlaví je rozšířeno o  $3 + 5 \times 4 = 23$  bajtů, jak je vidět z odchyceného IP datagramu:

```
+ FRAME: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP: DOD Internet Protocol
  IP: ID = 0x673D; Proto = ICMP; Len: 84
    IP: Version = 4 (0x4)
    IP: Header Length = 44 (0x2C)
    + IP: Service Type = 0 (0x0)
    IP: Total Length = 84 (0x54)
    IP: Identification = 26429 (0x673D)
    + IP: Flags Summary = 0 (0x0)
    IP: Fragment Offset = 0 (0x0) bytes
    IP: Time to Live = 32 (0x20)
    IP: Protocol = ICMP - Internet Control Message
    IP: Checksum = 0xCB51
    IP: Source Address = 194.149.104.198
    IP: Destination Address = 194.149.105.18
```

```

IP: Option Fields = 7 (0x7)
IP: Record Route Option = 7 (0x7)
IP: Option Length = 23 (0x17)
IP: Next Slot Pointer = 4 (0x4)
IP: Route Traveled = 0 (0x0)
IP: End of Options = 0 (0x0)
IP: Data: Number of data bytes remaining = 40 (0x0028)
+ ICMP: Echo,      From 194.149.104.198 To 194.149.105.18

```

Uživateli se přitom zobrazí odpověď:

```
Pinging ns.pvt.net [194.149.105.18] with 32 bytes of data:
```

```

Reply from 194.149.105.18: bytes=32 time<10ms TTL=63
  Route: 194.149.105.17 ->
           194.149.105.18 ->
           194.149.104.193

```

Na cestě mezi odesilatelem a příjemcem (194.149.105.18) a zpět je tedy jeden směrovač, který má směrem k příjemci adresu 194.149.105.17 a směrem k odesilateli adresu 194.149.104.193 (odesilatelem rozumíme uživatele, který vydal příkaz ping).

Tato odpověď vznikla z ICMP paketu Echo. Odchycená odpověď s vyplněnými sloty je následující:

```

+ FRAME: Base frame properties
+ ETHERNET: ETYPE = 0x0800 : Protocol = IP: DOD Internet Protocol
  IP: ID = 0x2DD8; Proto = ICMP; Len: 84
    IP: Version = 4 (0x4)
    IP: Header Length = 44 (0x2C)
  + IP: Service Type = 0 (0x0)
    IP: Total Length = 84 (0x54)
    IP: Identification = 11736 (0x2DD8)
  + IP: Flags Summary = 0 (0x0)
    IP: Fragment Offset = 0 (0x0) bytes
    IP: Time to Live = 63 (0x3F)
    IP: Protocol = ICMP - Internet Control Message
    IP: Checksum = 0x3334
    IP: Source Address = 194.149.105.18
    IP: Destination Address = 194.149.104.198
    IP: Option Fields = 7 (0x7)
      IP: Record Route Option = 7 (0x7)
      IP: Option Length = 23 (0x17)
      IP: Next Slot Pointer = 16 (0x10)
      IP: Route Traveled = 194 (0xC2)
        IP: Gateway = 194.149.105.17
        IP: Gateway = 194.149.105.18
        IP: Gateway = 194.149.104.193
      IP: End of Options = 0 (0x0)
    IP: Data: Number of data bytes remaining = 40 (0x0028)
+ ICMP: Echo Reply, To 194.149.104.198 From 194.149.105.18

```

## Zaznamenávej čas

Tato volba je obdobou volby zaznamenávej směrovače. Každý směrovač zapíše do záhlaví IP datagramu časové razítka, kdy datagram směrovačem prochází. Čas se opět zaznamenává v milisekundách od poslední půlnoci GMT (obdobně jako u časové synchronizace ICMP).

1	Kód=44 1B	Délka 1B	ptr 1B	O F F L	1. Časové razítka	2. Časové razítka	3. Časové razítka	.....	Časové razítka	40
---	--------------	-------------	-----------	------------	----------------------	----------------------	----------------------	-------	-------------------	----

OF - overflow (4b) - směrovač nastaví toto pole, když už nemá volný slot na přidání razítka.

FL - flaga (4b)

**FL = 0** Každý směrovač do slotu zapiše pouze čas, nikoliv IP adresu (slot 4B dlouhý).

**FL = 1** Každý směrovač do slotu zapiše čas i IP adresu (slot 8B dlouhý). Odesílatele inicializuje až čtyři slotové dlouhé 8B, tj. pro IP-adresu i pro časové razítka. Odesílatele vyplní IP adresy směrovačů; když pak datagram prochází přes směrovač uvedený v tomto seznamu, tento směrovač do slotu doplní časové razítka.

**Obrázek 5.20:** Volitelná položka IP záhlaví „Zaznamenávej čas“

Pole kód pro tuto volbu má hodnotu  $44_{16} = 68_{10}$ . Formát této volby je rozšířen o dvě čtyřbitová pole OF a FL.

Příkaz ping implementovaný firmou Microsoft pomocí parametru -s vygeneruje ICMP paket s požadavkem „zaznamenávej čas“. Číslo uvedené za parametrem -s udává počet alokovaných slotů. V případě, že se požaduje jak časové razítka, tak IP adresa, je možno alokovat max. 4 slotové.

D:\>ping -s 3 194.149.105.18

Vygeneruje IP datagram (zkrácený výpis):

```
...
IP: Option Fields = 68 (0x44)
IP: Internet Timestamp Option = 68 (0x44)
IP: Option Length = 28 (0x1C)
IP: Time pointer = 5 (0x5)
IP: ....0001 = Both time stamps and IP addresses
IP: Missed stations = 0 (0x0)
IP: Time Route = 0 (0x0)
IP: Gateway = 0.0.0.0
IP: Time Point = 0 (0x0)
IP: Gateway = 0.0.0.0
IP: Time Point = 0 (0x0)
IP: Gateway = 0.0.0.0
IP: Time Point = 16792576 (0x1003C00)
IP: Data: Number of data bytes remaining = 40 (0x0028)
...
```

Uživateli se pak zobrazí IP adresy a časová razítka. Milisekundy je nutno převést na hodiny, minuty, sekundy a nesmí se také zapomenout na letní čas.

Pinging 194.149.105.18 with 32 bytes of data:

```
Reply from 194.149.105.18: bytes=32 time<10ms TTL=63
  Timestamp: 194.149.105.17 : 52251609 ->
    194.149.105.18 : 52531841 ->
      194.149.104.193 : 52251610
```

Což přinesl IP datagram (zkrácený výpis):

```
...
  IP: Option Fields = 68 (0x44)
  IP: Internet Timestamp Option = 68 (0x44)
  IP: Option Length = 28 (0x1C)
  IP: Time pointer = 29 (0x1D)
  IP: ....0001 = Both time stamps and IP addresses
  IP: Missed stations = 0 (0x0)
  IP: Time Route
    IP: Gateway = 194.149.105.17
    IP: Time Point = 52251609 (0x31D4BD9)
    IP: Gateway = 194.149.105.18
    IP: Time Point = 52531841 (0x3219281)
    IP: Gateway = 194.149.104.193
    IP: Time Point = 52251610 (0x31D4BDA)
  IP: Data: Number of data bytes remaining = 40 (0x0028)
...

```

## Explicitní směrování

Explicitní směrování (*source routing*) umožňuje explicitně zadat, přes které směrovače má být IP -datagram Internetem dopravován. Je to dobrá zpráva pro hackera, protože pro něj to znamená, že pomocí explicitního směrování lze odklonit dopravu IP datagramů podle jeho potřeby.

Rozeznáváme dva typy explicitního směrování:

1. Explicitní směrování (kód = 83<sub>16</sub>), kdy je IP datagram dopravován přes vyjmenované směrovače. Výjmenovat se však nemusí všechny směrovače, přes které je IP datagram dopravován.
2. Explicitní striktní směrování (kód = 89<sub>16</sub>), kdy seznam směrovačů musí obsahovat všechny směrovače, přes které je IP datagram směrován. V případě, že by bylo nutné směrovat datagram přes jiný směrovač, směrování selže.

1							39
Kód 1B	Délka 1B	ptr 1B	1. IP adresa 4B	2. IP adresa 4B	3. IP adresa 4B	...	9. IP adresa 4B

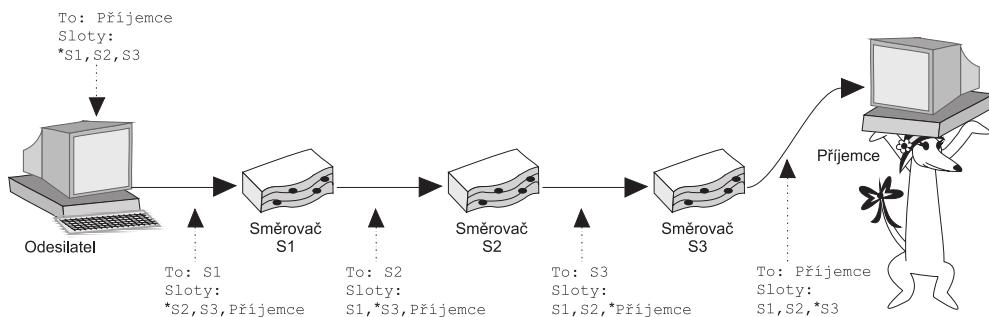
Obrázek 5.21: Volitelná položka „Explicitní směrování“

Mechanismus explicitního směrování je poměrně komplikovaný. Jednotlivé zúčastněné směrovače opravují nejen pole ptr, ale dokonce i adresu příjemce v IP datagramu.

I když odesilatel adresuje z aplikace přímo příjemce, ve skutečnosti adresa příjemce v IP datagramu vždy obsahuje následující směrovač (následující hop) ze seznamu směrovačů. Celý proces automaticky zabezpečuje vrstva IP protokolu, která na odesilatele stroji vezme z prvního slotu IP adresu

a nahradí jí původní adresu příjemce. Obsah jednotlivých slotů posune vlevo (první slot se uvolnil po vložení adresy do pole pro adresu příjemce). Do posledního (volného) slotu uschová původní adresu příjemce. Ukazatel ptr ukazuje na slot s IP adresou ob jeden hop dále.

Obdobně postupují i následující směrovače. Celý proces je znázorněn na obrázku 5.22, kde hvězdička vyjadřuje slot, na který ukazuje pole ptr:



**Obrázek 5.22:** Explicitní směrování

Příkaz ping implementovaný firmou Microsoft umožnuje specifikovat explicitní směrování pomocí parametrů -j pro explicitní směrování a parametru -k pro explicitní striktní směrování. Parametr je následován seznamem IP adres, přes které má být směrováno.

### Příklad

```
D:\>ping -j 195.47.1.1 10.1.1.1
```

Výpis:

```
...
IP: Source Address = 194.149.104.198
IP: Destination Address = 195.47.1.1
IP: Option Fields = 131 (0x83)
IP: Loose Source Routing Option = 131 (0x83)
IP: Option Length = 7 (0x7)
IP: Routing Pointer = 4 (0x4)
IP: Route To Go
IP: Gateway = 10.1.1.1
IP: End of Options = 0 (0x0)
IP: Data: Number of data bytes remaining = 40 (0x0028)
...
```

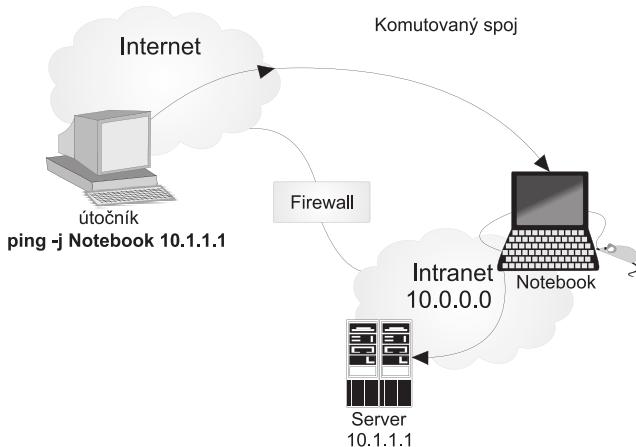
Uživatel obdrží např. hlášku (aby vše nebylo bez chyby):

```
Pinging 172.17.101.1 with 32 bytes of data:
Reply from 194.149.104.193: Invalid source route specified.
```

Tato hláška je důsledkem skutečnosti, že na uvedeném směrovači bylo zakázáno explicitní směrování.

Proč se explicitní směrování zakazuje? Důvody jsou bezpečnostní. Explicitní směrování lze zneužít dvěma způsoby:

- ◆ Pomocí explicitního směrování lze odklonit dopravu IP datagramů přes jiný směrovač, na kterém se budou data sledovat, případně měnit.
- ◆ Pomocí explicitního směrování je možné útočit z Internetu dovnitř do intranetu, i když intranet používá adresaci pro privátní síť (např. 10.0.0.0/8). Tyto privátní sítě nejsou z Internetu přímo adresovatelné – je to jedna z možných ochran intranetů. Pro průchod se použije buď přímo firewall, pokud umožňuje explicitní směrování (méně pravděpodobná varianta). Schůdnější je cesta přes počítač, který z nějakých důvodů má přímou konektivitu do Internetu. Jedná se např. o notebook zaměstnance, který se může v případě, že není na pracovišti, přímo přes komutovanou linku připojit do Internetu. Pro práci na pracovišti má pak síťovou kartu pro práci na LAN. Stačí když naváže spojení na obě strany, bude mít v operačním systému povoleno předávání (*IP forwarding*) a bude podporovat explicitní směrování. Mechanismus je znázorněn na obr. 5.23.



Obrázek 5.23: Útok proti intranetu za využití explicitního směrování

### Upozornění pro směrovač (*IP Router Alert Option*)

IP datagram je dopravován Internetem přes řadu směrovačů. Směrovač se za normálních okolností obsahem dopravovaného IP datagramu příliš nezabývá, podobně jako při dopravě pošty se poštovní úředníci nezabývají obsahem přepravovaných dopisů.

Jenže kromě běžných IP datagramů jsou Internetem dopravovány i datagramy směrovacích protokolů, které jsou směrovačům určeny. Na cestě k cílovému směrovači ale mohou být další směrovače, pro které je přenášené informace rovněž užitečná. Takže informace nesená v těchto IP-dataframech může být zajímavá i pro směrovače na cestě, tj. ty, kterým není přímo adresována. Za normálních okolností se směrovač na cestě ani nedozví, že předával informace, které mu mohly být užitečné. Odesílatel zase nevěděl, že na cestě je nějaký směrovač, kterému by měl takovou informaci přímo poslat.

Upozornění pro směrovač je volba v záhlaví IP datagramu, která říká všem směrovačům na cestě: „Tento IP datagram není sice určen přímo pro tebe, ale přenáší informace, které mohou být i pro tebe zajímavé. Pokud to umíš, tak se na tyto informace podívej a využij je.“

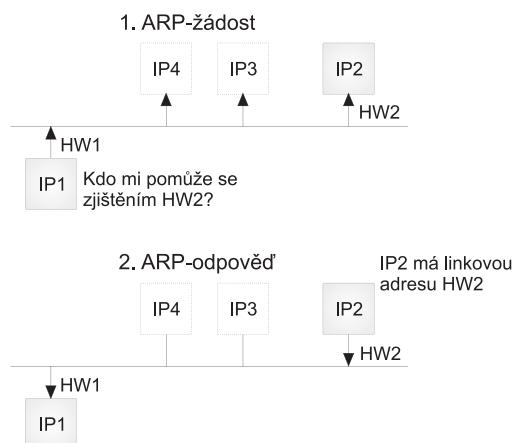


Obrázek 5.24: Volitelná položka „Upozornění pro směrovač“

## Protokoly ARP a RARP

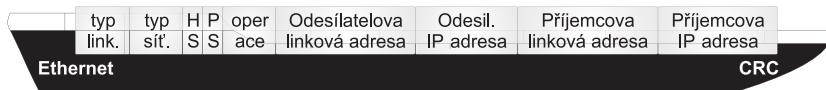
Jsem-li stanice na lokální síti a chci protokolem IP komunikovat s jinou stanicí na téže síti, pak ji v protokolu IP adresuji čtyřbajtovou IP adresou. Pro komunikaci znám IP adresu odesilatele (svou IP adresu) a IP adresu příjemce. Jsem tedy schopen sestavit IP datagram. Jenže potíž je v tom, že tento IP datagram musí být zabalen do linkového rámce – např. do ethernetového rámce. Abych vytvořil ethernetový rámec, potřebuji linkovou (6B) adresu příjemce i odesilatele. Odesilatel jsem já a svou linkovou adresu znám, avšak neznám linkovou adresu příjemce. Jak takovou adresu zjistím? To řeší protokol ARP.

Protokol ARP (*Address Resolution Protocol*) řeší problém zjištění linkové adresy protější stanice ze znalosti její IP adresy. Řešení je jednoduché (viz obr. 5.25) – do LAN vyšle linkový oběžník (linková adresa FF:FF:FF:FF:FF) s prosbou: „Já stanice o linkové adrese HW1 a IP adrese IP1 chci komunikovat se stanicí o IP adrese IP2 kdo mi pomůže s nalezením linkové adresy stanice o IP adrese IP2?“ Stanice IP2 takovou žádost uslyší a odpovídá. V odpovědi uvede svou linkovou adresu HW2.



Obrázek 5.25: Protokol ARP

ARP-paket (obr. 5.26) je balen přímo do Ethernetu, tj. nepředchází mu žádné IP záhlaví. Protokol ARP je vlastně samostatný, na IP nezávislý protokol. Proto jej mohou používat i jiné protokoly, které s protokoly TCP/IP nemají nic společného.



**Obrázek 5.26:** Paket protokolu ARP

Pole **typ linkového protokolu** specifikuje linkový protokol používaný na LAN. Linkovému protokolu Ethernet II je vyhrazeno číslo 1. Seznam přidělených čísel je uveřejněn na <http://www.iana.org>.

**Typ síťového protokolu** – používají se stejná čísla jako pro pole protocol v protokolu Ethernet II, tj. IP -protokol má přiděleno číslo 800\_16.

Pole **HS** určuje délku linkové adresy a pole **PS** délku síťové adresy. Standardně je tedy HS=6 a PS=4.

Pole **operace** určuje, o jakou operaci jde. Žádost (ARP *request*) má hodnotu 1 a odpověď (ARP *reply*) má hodnotu 2. Toto pole je definováno rovněž pro reverzní překlad (protokol RARP), kdy žádost RARP používá hodnotu 3 a odpověď RARP hodnotu 4.

Pak již následuje linková adresa odesilatele, IP adresa odesilatele, linková adresa příjemce (v dotazu vyplňena nulami) a IP adresa příjemce.

Žádost je posílána linkovým oběžníkem a v poli příjemcova linková adresa má vyplněny nuly. Odpověď pak má již vyplněna všechna pole a nemusí být odesílána oběžníkem. Je třeba zdůraznit, že v odpovědi dojde k výměně příjemce a odesilatele. Vše je patrné z následujícího příkladu.

## Příklad

C:\ > ping 194.149.104.126

Tento příkaz, než může vyslat první IP datagram (ICMP paket echo request), musí zjistit pomocí směrovací tabulky, zdali je příjemce na LAN nebo za směrovačem (musí zjistit následující hop). Pokud je příjemce za směrovačem, hledá linkovou adresu směrovače. Pokud příjemce není za směrovačem, hledá přímo linkovou adresu příjemce (náš případ).

Nyní již víme, že příjemce má IP adresu 194.149.104.126 a je přímo na LAN. Je třeba zjistit jeho linkovou adresu. Operační systém odesilatele vygeneruje ARP-žádost:

```
+ FRAME: Base frame properties
ETHERNET: ETYP = 0x0806 : Protocol = ARP: Address Resolution Protocol
+ ETHERNET: Destination address : FFFFFFFFFFFF
+ ETHERNET: Source address : 0020AFFA2589
ETHERNET: Frame Length : 42 (0x002A)
ETHERNET: Ethernet Type : 0x0806 (ARP: Address Resolution Protocol)
ETHERNET: Ethernet Data: Number of data bytes remaining = 28 (0x001C)
ARP_RARP: ARP: Request, Target IP: 194.149.104.126
ARP_RARP: Hardware Address Space = 1 (0x1)
ARP_RARP: Protocol Address Space = 2048 (0x800)
ARP_RARP: Hardware Address Length = 6 (0x6)
ARP_RARP: Protocol Address Length = 4 (0x4)
ARP_RARP: Opcode = 1 (0x1)
ARP_RARP: Sender's Hardware Address = 0020AFFA2589
ARP_RARP: Sender's Protocol Address = 194.149.104.121
```

```
ARP_RARP: Target's Hardware Address = 000000000000
ARP_RARP: Target's Protocol Address = 194.149.104.126
```

Příjemce okamžitě odpoví paketem:

```
+ FRAME: Base frame properties
ETHERNET: ETYPE = 0x0806 : Protocol = ARP: Address Resolution Protocol
+ ETHERNET: Destination address : 0020AFFA2589
+ ETHERNET: Source address : 00603E1D9001
ETHERNET: Frame Length : 60 (0x03C)
ETHERNET: Ethernet Type : 0x0806 (ARP: Address Resolution Protocol)
ETHERNET: Ethernet Data: Number of data bytes remaining = 46 (0x02E)
ARP_RARP:
ARP_RARP: Hardware Address Space = 1 (0x1)
ARP_RARP: Protocol Address Space = 2048 (0x800)
ARP_RARP: Hardware Address Length = 6 (0x6)
ARP_RARP: Protocol Address Length = 4 (0x4)
ARP_RARP: Opcode = 2 (0x2)
ARP_RARP: Sender's Hardware Address = 00603E1D9001
ARP_RARP: Sender's Protocol Address = 194.149.104.126
ARP_RARP: Target's Hardware Address = 0020AFFA2589
ARP_RARP: Target's Protocol Address = 194.149.104.121
ARP_RARP: Frame Padding
```

Z tohoto paketu si systém do pracovní paměti (*ARP cache*) automaticky doplní položku říkající, jaká linková adresa přísluší udané IP adrese. Při příští komunikaci s počítačem 194.149.104.126 se již dotaz ARP nebude generovat, ale použije se tato položka. ARP-cache můžeme vypsat příkazem:

```
D:\> arp -a
Interface: 194.149.104.121
      Internet Address          Physical ADDRESS      Type
        194.149.104.126           00-60-3e-1d-90-01    dynamic
          10.1.1.1                00-01-11-11-ff-08    static
```

V ARP-cache mohou být položky získané ARP-dotazem, ty jsou typu *dynamic*. Do ARP cache můžeme také zapsat položky explicitně příkazem arp. Takové položky jsou typu *static*. Rovněž je možné položky ARP-cache příkazem arp rušit.

### **Příklad**

Příklad vložení statické položky:

```
D:\> arp -s 10.1.1.1 00-01-11-11-ff-08
```

### **Příklad**

Příklad zrušení položky:

```
D:\> arp -d 10.1.1.1
```

Jak dlouho zůstávají dynamické položky v ARP-cache? Tento interval je parametrem jádra operačního systému. Nejčastěji mají položky dobu života 2 minuty, pokud nebyly podruhé použity. Při každém dalším použití se jejich životnost zvětší o další dvě minuty. To se však neděje do nekonečna. Životnost je zpravidla možné prodlužovat až na maximální životnost, která je zpravidla 10 minut. Do tabulky však některé systémy zaznamenávají i negativní odpovědi, ty mají zpravidla dobu života

3 minuty. Než dojde k negativnímu závěru, opakuje se ARP-žádost po 5,5 sekundě a ještě po dalších 24 sekundách znovu.

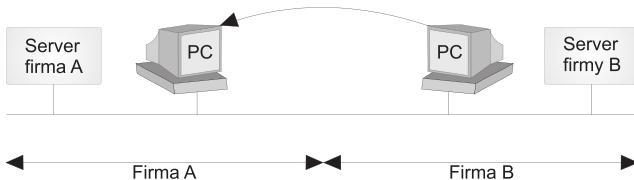
Od Windows 2000 lze měnit uvedené konstanty změnou klíčů v Registry. Klíče pro ovládání ARP-cache jsou uloženy v registru `HKEY_LOCAL_MACHINE\ SYSTEM\ CurrentControlSet\ Services\ Tcpip\ Parameters`. Do tohoto registru můžeme přidat mj. následující klíče typu REG\_DWORD:

- ◆ `ArpCacheLife` obsahující dobu života nepoužité položky v ARP-cache v sekundách.
- ◆ `ArpCacheMinReferenceLife` obsahující maximální dobu života položky v ARP-cache v sekundách.

Protokolem ARP je také možné odeslat žádost s vyplněnou IP adresou odesilatele i příjemce a také s oběma vyplněnými linkovými adresami. Takovou žádost je možné chápat jako: „Neexistuje náhodou na LAN ještě jiná stanice, která používá stejnou IP adresu jako já? V případě, že se obdrží odpověď, signalizuje se uživateli zpráva: „*Duplicate IP address sent from Ethernet address xx:xx:xx:xx:xx:xx*.“ To pochopitelně signalizuje chybu v konfiguraci jedné ze stanic používajících tuto adresu.

## Filtrace ARP

Filtrace ARP není žádnou filtrací, ale její účinek má podobný efekt jako filtrace. Používá se v případě, že na jedné LAN jsou dvě firmy (resp. dvě samostatné části firmy).



**Obrázek 5.27:** Filtrace ARP

Problém spočívá v zamezení zaměstnanci firmy B v přístupu na server firmy A, tj. aby zaměstnanec firmy B nemohl prohlásit svůj PC za PC zaměstnance firmy A.

Řešení spočívá v tom, že na serveru firmy A staticky naplníme ARP-cache. Server pak bude odpovídat stále na linkovou adresu PC zaměstnance firmy A, aniž by použil ARP-protokol, takže hacker má smůlu.

Význam filtrace ARP je ale omezený, protože zaměstnanec firmy B může podvrhnout i linkovou adresu. To však není tak triviální a začínajícího hackera to odradí.

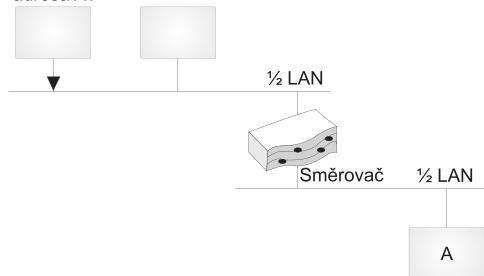
## Proxy ARP

Protokol ARP pracuje pouze v rámci LAN, tj. mezi dotazovaným a odpovídajícím počítačem nemůže být směrovač. Důvod je prostý: v dotazu je adresou příjemce všeobecný oběžník, který směrovače nešíří.

Co si však počít, když mám dvě nebo více částí LAN oddělených směrovačem? Řešením je proxy ARP. Proxy ARP běží na směrovači.

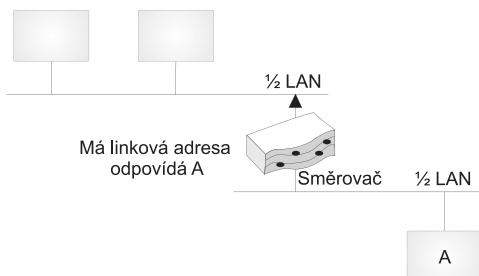
Počítač chce ARP-dotazem zjistit linkovou adresu k IP adrese A, která leží na druhé polovině LAN za směrovačem.

Jaká je linková  
adresa A?



**Obrázek 5.28:** LAN, jejíž část je za jiným rozhraním směrovače

Směrovač nemůže propustit takový dotaz, avšak pokud je nakonfigurován jako proxy ARP, pak odpovídá, že IP adrese A odpovídá linková adresa samotného směrovače.



**Obrázek 5.29:** Proxy ARP

Počítač pak, když chce odeslat linkový rámec A, adresuje směrovač, který IP datagram předá cílovému počítači A.

## RARP

Zatímco protokol ARP slouží k překladu IP adres na linkové adresy, reverzní ARP, označované jako RARP, slouží k překladu linkové adresy na IP adresu. Avšak proč takový překlad provádět?

Smysl protokolu RARP je u bezdiskových stanic. Bezdisková stanice po svém zapnutí nezná nic jiného než svou linkovou adresu (tu má uloženu výrobcem v paměti ROM). Po svém zapnutí se potřebuje dozvědět svou IP adresu. Proto do LAN vyšle oběžník s prosbou: „Já mám linkovou adresu HW1; kdo mi řekne, jakou mám IP adresu?“ Na LAN pak musí být RARP-server, který jí IP adresu přidělí a sdělí v odpovědi. Protokol RARP používá stejný formát paketu jako protokol ARP. Pouze hodnota pole operace je zvětšena o jedničku. V RARP žádosti pochopitelně není vyplněna ani IP-adresa žadatele.

Protokol RARP se v praxi téměř nepoužívá, nahradil jej protokol DHCP, který je komplexnější.

## IGMP

Protokol IGMP je podobně jako protokol ICMP služebním protokolem (podmnožinou) protokolu IP. Pakety IGMP-protokolu jsou baleny do IP datagramů.

Protokol IGMP slouží k šíření adresných oběžníků (*multicasts*). Nyní je aktuální protokol IGMP verze 2 podle normy RFC-2236.

Struktura IGMP-paketu verze 2:

IP-záhlaví	Typ 1B	MRT 1B	Kont. součet 2B	IP-adresa ardr. oběžníku 4B
------------	-----------	-----------	--------------------	--------------------------------

**Obrázek 5.30:** Struktura IGMP-paketu verze 2

Pole **typ** nabývá hodnot:

**Tabulka 5.6**

Hodnota (šestnáctkově)	Význam
11	Dotaz směrovače: „Jsou na LAN ještě nějací členové?“ ( <i>Membership query</i> )
16	Požadavek na členství ve skupině ( <i>Membership report</i> )
17	Opuštění skupiny ( <i>Leave group</i> )
12	Požadavek IGMP v1 na členství ve skupině ( <i>Version 1 membership report</i> )

Pole **MRT** (*Maximum response time*) se používá pouze v dotazu směrovače a specifikuje v desetinách sekundy čas, do kterého musí členové skupiny opakovat požadavky na členství ve skupině. Ve všech ostatních případech má pole MRT hodnotu 0.

Kontrolní součet se počítá stejně jako u protokolu ICMP. Pole IP adresa adresného oběžníku je nulové u všeobecného dotazu, v ostatních případech specifikuje konkrétní IP adresu adresného oběžníku.

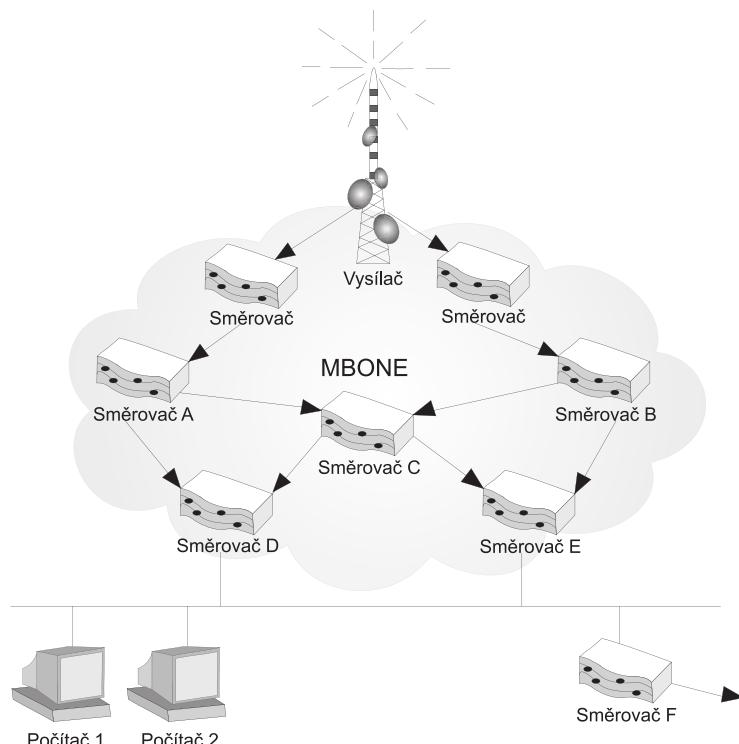
IP adresy adresných oběžníků jsou v intervalu 224.0.0.0 až 239.255.255.255. Interval 224.0.0.0 až 224.0.0.255 je určen pro vyhrazené účely na LAN (viz tab. 5.7). Jelikož jsou oběžníky s těmito adresami určeny výhradně pro LAN, mívají v položce TTL nastavenou hodnotu 1.

**Tabulka 5.7**

IP adresa	Vyhrazeno pro adresaci
224.0.0.1	Všechny systémy na LAN
224.0.0.2	Všechny směrovače na LAN
224.0.0.4	Distance Vector Multicast Routing Protocol – viz RFC-1075
224.0.0.5	OSPF All Routers – viz RFC-1583
224.0.0.6	OSPF Designated Routers – viz RFC-1583
224.0.0.9	RIP-2 atd.

Všechny IGMP, pakety mají v IP záhlaví nastavenou položku TTL=1. Pakety protokolu IGMP verze 2 používají volbu (rozšíření) IP záhlaví „Upozornění pro směrovač (IP Router Alert Option)“.

Jedním z jader Internetu je tzv. Mbone (zkráceno z *Mulicast Backbone*), kde je zabezpečeno šíření adresných oběžníků. Že to není jednoduché, je vidět z obrázku 5.31, kdy zdroj adresných oběžníků – např. internetová rozhlasová stanice – šíří svá data pomocí adresných oběžníků. Kdyby se oběžníky šířily nekontrolovaně lavinovitě, mohla by se data postupně duplikovat. Např. na směrovač C by tatáž data mohla dorazit ze směrovače A i ze směrovače B.



**Obrázek 5.31:** Šíření adresných oběžníků

Protokol IGMP řeší šíření adresných oběžníků v rámci LAN.

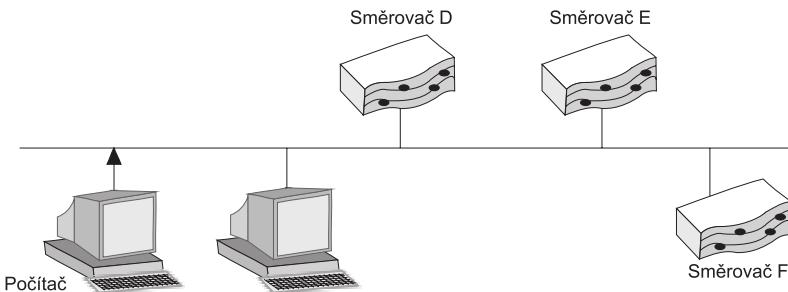
Představme si situaci, kdy jsme na LAN a některé směrovače přijímají adresné oběžníky z Mbone a řeší otázku, zdali je mají šířit dále na LAN. Obecně pokud žádný počítač na LAN adresné oběžníky nepotřebuje, je zbytečné je šířit – pouze by se zvětšilo zatížení LAN.

Jsme tedy v situaci, kdy některé směrovače na LAN mohou LAN zásobovat adresnými oběžníky, ale nečiní tak, protože adresné oběžníky na LAN nejsou vyžadovány.

Pro každou IP adresu adresného oběžníku se na LAN definuje tzv. skupina členů adresného oběžníku. Směrovače udržují seznam skupin. V případě, že se nějaký počítač na LAN přihlásí do konkrétní skupiny, pak směrovače začnou daný oběžník na LAN šířit.

Když skupinu opustí poslední člen, šíření adresného oběžníku na LAN se zastaví. Existence skupiny tedy znamená šíření oběžníků. Přítom není důležité, kolik má skupina členů, ale jestli má alespoň jednoho člena.

Spustí-li se na počítači aplikace, která chce poslouchat rozhlasovou stanici, např. 226.1.1.1, vyšle počítač požadavek na členství ve skupině 226.1.1.1 – viz obr. 5.32.



**Obrázek 5.32:** Počítač na LAN odesílá požadavek na členství ve skupině

Vyplnění polí požadavku:

IP záhlaví:

TTL=1

IP adresa odesilatele=Počítač

IP adresa příjemce=224.0.0.2 (všem směrovačům na LAN)

IGMP-paket:

Typ=16<sub>16</sub>

MRT=0

IP adresa oběžníku=226.1.1.1

Pokud na LAN dosud nejsou šířeny adresné oběžníky 226.1.1.1, pak se s jejich šířením začne.

Pokud počítač odebírá adresné oběžníky a je posledním členem skupiny, může šíření zastavit odesláním obdobného IGMP-paketu, ale s polem Typ=17<sub>16</sub>.

Jenže co když na počítači není aplikace řádně ukončena? Např. pokud je počítač vytažen ze zásuvky a nemá šanci vyslat „vypínací paket“? Představme si, že adresné oběžníky na LAN šíří směrovač E (viz obr. 5.33). Směrovač E, aby zjistil, zdali je třeba naši skupinu stále ještě šířit, poše čas od času na LAN IGMP-paket „Jsou na LAN ještě nějací členové?“ (*Membership query*), tj. pole typ=11<sub>16</sub>. Tento paket má dvě varianty:

1. Všeobecný dotaz (pole IP adresa oběžníku je vyplněno nulami), který se ptá na všechny skupiny. Jednotlivé počítače musí postupně zopakovat své požadavky na členství v každé skupině do MTR deseti sekundy. V opačném případě se chápe, že skupinu opustily.
2. Adresný dotaz na konkrétní skupinu (pole IP adresa oběžníku je v IGMP-paketu vyplněno). Všichni členové uvedené skupiny musí do MTR deseti sekundy zopakovat požadavek na členství.

Kde:

IP záhlaví:

TTL=1

IP adresa odesilatele=Směrovač E

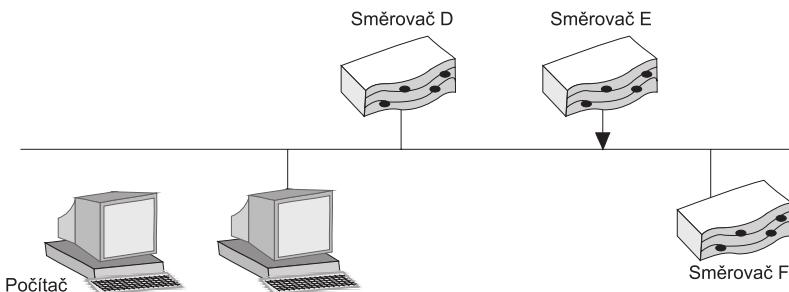
IP adresa příjemce=224.0.0.1 (všem systémům na LAN)

IGMP-paket:

Typ=1116

MRT>0 pro verzi 2, =0 pro verzi 1 (pak je konstantně 10 s)

IP adresa oběžníku=226.1.1.1 (adresný dotaz), 0.0.0.0 (všeobecný dotaz)



**Obrázek 5.33:** Směrovač E: „Jsou na LAN ještě nějací členové?“

Otázkou zůstává, jak se mezi sebou dohodnou jednotlivé směrovače na LAN v případě, že jich je tam více než jeden. Směrovače vzhledem k protokolu IGMP pracují ve dvou režimech:

1. Dotazovač, který posílá na LAN dotazy na členství ve skupinách.
2. Posluchač, který není aktivní, pouze naslouchá provozu, a pokud je na LAN nějaký dotazovač, nevstupuje do hry.

Směrovač po svém zapnutí začíná pracovat jako dotazovač, zjistí-li však, že se na LAN vyskytuje i dotazy směrovače s vyšší IP adresou, pak se přepne do režimu posluchače.

## Oběžníky a linkový protokol

Zatím jsme popisovali odesílání všeobecných oběžníků, ale problém na LAN spočívá v určení linkové adresy jejich příjemce.

Protokol ARP určil jednoznačný vztah mezi jednoznačnou IP adresou příjemce (*unicast*) a linkovou adresou příjemce. To je možné tehdyn, když mezi IP adresami a linkovými adresami existuje jednoznačný vztah. Tento vztah se anglicky nazývá *mapping*, česky se (asi ne zcela správně) označuje jako mapování IP adres na linkové adresy.

Jiným případem na LAN je všeobecný oběžník (*broadcast*), který se posílá všem systémům na LAN. Pro tyto účely slouží linkovému protokolu všeobecný oběžník, který pro Ethernet, FDDI apod. je ff:ff:ff:ff:ff:ff.

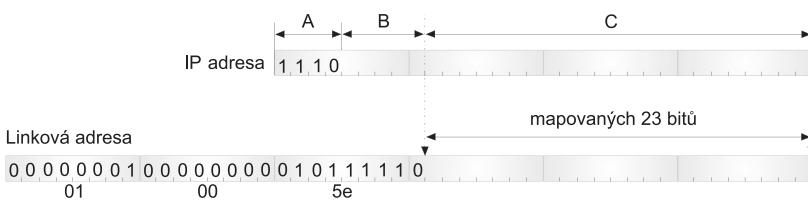
Jenže jak to udělat na LAN v případě adresného oběžníku (*multicast*), který není určen jednomu adresátovi na LAN ani všem systémům na LAN, nýbrž několika konkrétním adresátům?

V čem je problém? Příjemce za normálních okolností zpracovává pouze rámce, které jsou všeobecnými oběžníky nebo jsou adresovány příjemcovou linkovou adresou. (Je možné síťovou kartu realizující síťové rozhraní přepnout do tzv. promiskuitního módu, kdy přijímá vše, ale tento případ není považován za běžný.)

Linkové protokoly umožňují též linkové adresné oběžníky (*multicast*). Jsou to takové linkové adresy, kde nejnižší bit prvního bajtu linkové adresy je nastaven na 1, tj. všeobecný linkový oběžník je zvláštním případem takového adresného oběžníku. Jenže jak mapovat IP adresu adresného oběžníku na linkový adresný oběžník?

Není to tak jednoduché, jak to vypadá na první pohled. Šestibajtová linková adresa se skládá ze tří bajtů specifikujících výrobce a tří bajtů čísla karty v rámci výrobce.

IANA (nejvyšší autorita Internetu) se nechala zaregistrovat jako fiktivní výrobce síťových karet a obdržela pro sebe identifikaci 00:00:5e. První polovinu těchto adres použila pro mapování adres- ných IP oběžníků na adresné linkové oběžníky (viz obr. 5.34). Bohužel tato polovina má pouze 23 bitů, takže mapování nemůže být jednoznačné.



**Obrázek 5.34:** Mapování adresných oběžníků na linkové adresy

První bajt linkové adresy musí mít nastaven nejnížší bit na jedničku, protože se jedná o adresný linkový oběžník. Prefix tak ve skutečnosti nebude 00:00:5e, ale 01:00:5e.

Část A IP adresy specifikuje adresný oběžník, je tedy vždy konstantní. Část B není mapována.

Pokud se tedy dva adresné oběžníky liší pouze v části B, pak jsou mapovány na stejnou linkovou adresu. Např. IP adresy 224.0.1.1, 224.128.1.1 a 225.0.1.1 jsou mapovány vždy na 01:00:5e:00:01:01.

Linková vrstva počítače akceptuje linkové rámce:

- ◆ Adresované jednoznačnou linkovou adresou počítače (*unicast*).
  - ◆ Všeobecné linkové oběžníky (*broadcast*).
  - ◆ Adresné linkové oběžníky, jejichž seznam je linkové vrstvě předán vyššími vrstvami.

Seznam přijímaných adresních linkových oběžníků zahrnuje adresu 224.0.0.1 a s každým oběžníkem i všechny adresné oběžníky, které vznikly díky nejednoznačnosti mapování. Nadbytečné oběžníky musí odfiltrovat IP protokol. Některé implementace softwaru přepnou síťovou kartu do promiskuitního módu pro interval všech adresních oběžníků a vše ponechají na IP protokolu, to však zbytečně zvyšuje zatížení operačního systému.

QoS

Zejména s příchodem požadavků na přenos audia a videa přes Internet přišly požadavky na zajištění garance šíře přenosového pásma v rodině protokolů TCP/IP. Pro tyto účely se v záhlaví IP datagramu využívá položka TOS (obr. 5.35).

Pro zajištění šíře pásmo se používají dvě odlišné strategie:

- ◆ „Integrované služby“ (*Integrated Services*) – jedná se o starší, dnes méně populární strategii, u které je opravdu možné hovořit o garanci šíře pásmo (QoS). Myšlenka spočívá v tom, že všechny směrovače na cestě od odesilatele k příjemci se dohodnou na rezervaci šíře pásmo. To znamená, že všechny směrovače na cestě musí podporovat „Integrované služby“ a navíc každá aplikace, která potřebuje garanci šíře pásmo, má možnost si provést individuální rezervaci zdrojů na celé cestě od odesilatele k příjemci. K dispozici je Resource ReSerVation Protocol (RSVP) specifikovaný v RFC-2205. Pomocí tohoto protokolu si aplikace zpravidla každých 30 vteřin žádá o rezervaci zdrojů na cestě. Jedná se vlastně o vytváření jakéhosi virtuálního okruhu. V operačních systémech Microsoft je k dispozici dokonce příkaz „`pathping -R`“, pomocí kterého můžeme takovou cestu testovat. Problémem je právě fakt, že „všechny“ směrovače musí podporovat „Integrované služby“, což je v rozsáhlejších sítích problém.
- ◆ „Rozdílné služby“ (*Differentiated services*). Nehraje se na individuální rezervaci zdrojů, ale každý IP datagram je klasifikován a na základě této klasifikace je pak označen v položce TOS, tj. je zařazen do příslušné třídy. Přitom jednotlivých tříd je omezené množství. Důležité je, že směrovače odbavují (předávají) IP datagramy na základě jejich příslušnosti do konkrétní třídy. Přitom odbavování jednotlivých tříd je nezávislé na odbavování jiných tříd.



**Obrázek 5.35:** Položka TOS v záhlaví IP datagramu (viz též obrázek 5.8)

Zastavme se u „Rozdílných služeb“, které využívají nejvyšších šesti bitů položky TOS, které tvoří pole DSCP (*Differentiated Services Code Point*) – viz obr. 5.35. Chování směrovačů vůči předávanému IP -datagramu je pak závislé na nastavení těchto nejvyšších šesti bitů DS0 – DS5 .

Aby to nebylo jednoduché, tak nastavení bitů DS0 – DS5 specifikuje několik standardů, které se v podstatě vzájemně doplňují (přesněji: vzájemně se nevylučují):

- ◆ RFC-2474 specifikuje význam bitů DS3 až DS5, které určují třídu, ve které bude IP datagram zpracováván:

DSCP binárně	DSCP desítkově	Význam
111 000	7	Směrovací pakety „Keep alive“
110 000	6	IP směrování
101 110	5	„Expresní předávání“
100 000	4	Třída 4
011 000	3	Třída 3
010 000	2	Třída 2
001 000	1	Třída 1
000 000	0	Standardní předávání

Kromě již zmíněných tříd 1 až 4 jsou zde zavedeny vysoce prioritní třídy pro směrování (5 až 7). Je to logické, protože bez zajištění směrování by se komunikace rozpadla. O třídě „Expresní předávání“ se ještě zmíníme a třída „Standardní předávání“ vyjadřuje jen fakt, že položka TOS nemusí být vůbec využívána, což je asi ta nejběžnější situace, kdy QoS vůbec neřešíme.

- ◆ RFC-2597 rozvádí mechanismus tříd. Opět využívá 4 třídy (všimněte si, že bity DS3 až DS5 mají shodné s třídami dle RFC-2474), ale využívá i bity DS1 a DS2 proto, aby se umožnilo zjednodušit celý mechanismus.

IP datagramy různých tříd jsou zpracovávány nezávisle. Naopak IP datagramy téže třídy jsou zpracovávány společně, avšak navíc pomocí bitů DS1 a DS2 je možné vyznačit, kterým IP datagramům má být dána přednost před jinými. Avšak v rámci zpracování jedné třídy nesmí docházet ke změně pořadí IP datagramů v rámci mikrotoku IP datagramů.

	Třída 1	Třída 2	Třída 3	Třída 4
<b>Nízká přednost</b>	AF11 001 010	AF21 010 010	AF31 011 010	AF41 100 010
<b>Střední přednost</b>	AF12 001 100	AF22 010 100	AF32 011 100	AF42 100 100
<b>Vysoká přednost</b>	AF13 001 110	AF23 010 110	AF33 011 110	AF43 100 110

- ◆ RFC-3246 specifikuje „Expresní předávání“, které má rezervovanou hodnotu DSCP: 101 110. Jedná se o třídu zajišťující stejnoměrný tok dat s garantovanou šíří pásma a minimální ztrátou IP datagramů.

## ECN

ECN (*Explicit Congestion Notification*) je mechanismem bránícím zahlcování sítě, který mj. využívá dva nejnižší bity na obr. 5.35.

Zatímco QoS slouží pro řízení toku odesilatelem již odeslaných IP datagramů, tak cílem ECN je předmět odesilatele, aby v případě potencionálního zahlcení sítě odesílal méně IP datagramů. Princip je následující:

1. Odesilatel IP datagramu sítí signalizuje nastavením bitů ECN na hodnotu '01' nebo '10', že podporuje mechanismus ECN. V případě, že ECN nepodporuje, jsou tyto bity nastaveny na '00'.
2. Síť v případě zahlcení nastaví bity ECN na hodnotu '11'.
3. IP datagram s takto nastavenými bity ECN dorazí k příjemci, který informuje o této skutečnosti odesilatele nastavením příznaku ECE v záhlaví TCP segmentu odesílaného v protisměru.

Princip je podobný mechanismu BECN, se kterým jsme se setkali v případě protokolu Frame Relay.

## Domácí cvičení

1. Odchyťte IP datagram programem Wireshark a výsledek porovnejte s obr. 5.7.
2. Programem Wireshark odchyťte komunikaci programu tracert vůči vašemu oblíbenému serveru.
3. Programem Wireshark odchyťte ARP komunikaci s vašim místním směrovačem, vypište obsah ARP, paměti vašeho počítače.
4. Pomocí programu nmap zjistěte:
  - ◆ všechny sousedy na vaší LAN (tj. provedte „ARP-ping“),
  - ◆ všechny počítače vaší oblíbené vzdálené sítě odpovídající protokolem ICMP „Echo-Reply“.



## Kapitola 6

# IPv4 – adresa

Protokol IP verze 4 používá IP adresu o délce čtyři bajty. IP adresa adresuje jednoznačně síťové rozhraní systému. Anglicky se takováto jednoznačná adresa nazývá **unicast**. Pokud má systém více síťových karet (více síťových rozhraní) a na všech je provozován protokol IP, pak má každé rozhraní svou IP adresu. Je to podobné jako s adresou domu. Pokud má dům vchod ze dvou ulic, má každý vchod svou orientační adresu.

Je možná i opačná varianta, kdy na jedné síťové kartě (fyzicky jednom síťovém rozhraní) podporujeme několik IP adres. První adresa se obvykle nazývá primární, další adresy pak sekundární nebo aliasy. Využití sekundárních IP adres je běžné např. pro WWW servery, kdy na jednom počítači běží WWW servery několika firem a každý se má tvářit jako samostatný WWW server.

V praxi se však využívání sekundárních IP adres pro WWW servery považuje za plýtvání – používají se tzv. virtuální WWW servery, kdy mnoha WWW serverům stačí jedna společná IP adresa. Specifikace serveru se pak provádí na aplikační úrovni v protokolu HTTP (pomocí hlavičky *host*).

Jelikož má většina počítačů jedno síťové rozhraní, říká se přeneseně místo IP adresa rozhraní IP -adresa počítače.

Vedle jednoznačných IP adres (**unicast**) máme v IPv4 ještě následující adresy:

- ◆ Všeobecný oběžník (**broadcast**), který je určen všem stanicím v síti (LAN). Všeobecný oběžník se šíří jen v rámci LAN, tj. směrovače jej nepředávají do dalších sítí (pokud nejsou směrovače speciálně konfigurovány).
- ◆ Adresný oběžník (**multicast**), který je určen (adresován) konkrétním stanicím. Obecně je adresné oběžníky možné šířit i do dalších sítí.
- ◆ Programová smyčka (**loopback**) o IP adrese 127.0.0.1. Tato IP adresa nikdy neopouští systém (počítač).

Adresa IPv4 je tvořena čtyřmi bajty. Adresa IPv4 se zapisuje notací, kde se jednotlivé bajty mezi sebou oddělují tečkou. Rozeznáváme:

- ◆ Dvojkovou notaci, kde je každý ze čtyř bajtů adresy zapsán číslem ve dvojkové soustavě, např.: 10101010.01010101.11111111.11111000.
- ◆ Desítkovou notaci, kde se čtyři osmiceferná dvojková čísla (z dvojkové notace) převedou do desítkové soustavy, tj. pro náš příklad: 170.85.255.248.
- ◆ Šestnáctkovou notaci, kde jsou jednotlivé bajty IP adresy vyjádřeny šestnáctkově (hexadecimálně), tj. náš příklad: aa.55.ff.f8.

IP adresa se skládá ze dvou částí:

1. Adresy (lokální) sítě.
2. Adresy počítače v (lokální) síti, která je určena první částí.

Problém je v tom, jak zjistit, která část IP adresy je adresou sítě a která adresou počítače. Dokonce i význam slova síť se postupně měnil a kromě slova síť se zavedly pojmy subsíť a supersíť. K tomu však musíme dospět postupně.

## Síť – historická epocha I

Tato epocha trvala od počátku Internetu až do roku 1993. V této epoše bylo slovo síť specifikováno normou RFC-796 (J. Postel, 1.9.1981). Těchto dvanáct let je poznamenáno představou, že čtyři bajty na IP adresy přece musí stačit.



**Obrázek 6.1:** Struktura IP adresy

IP adresa se dělí na adresu sítě a adresu počítače v rámci této sítě (viz obr. 6.1).

Kolik bajtů z IP adresy tvoří adresu sítě, určují počáteční bity prvního bajtu IP adresy. IP adresy se dělí do pěti tříd (sledujte tab. 6.1):

- ◆ **Třída A**, kde nejvyšší bit prvního bajtu má hodnotu 0. Zbylých 7 bitů prvního bajtu tvoří adresu sítě a zbytek (24 bitů) je určen pro adresu počítače v rámci sítě. Ve třídě A máme sítě 1 až 126 (sítě 0 a 127 mají zvláštní význam). V každé z těchto sítí je  $2^{24}-2$  adres pro počítače (adresy tvořené samými nulami a samými jedničkami mají zvláštní význam).
- ◆ **Třída B**, kde nejvyšší dva bity prvního bajtu mají hodnotu  $10_2$ . Zbylých 6 bitů a následující druhý bajt je určen pro adresy sítí. Můžeme tedy mít celkem  $2^{14}$  sítí a v každé síti  $2^{16}-2$  počítačů.
- ◆ **Třída C**, kde nejvyšší tři bity prvního bajtu mají hodnotu  $110_2$ . Zbylých 5 bitů a následující dva bajty jsou určeny pro adresu sítě. Můžeme tedy mít  $2^{22}$  sítí a v každé síti  $128-2$  počítačů.
- ◆ **Třída D**, kde nejvyšší čtyři bity prvního bajtu mají hodnotu  $1110_2$ . Zbytek IP adresy se pak už nedělí na adresu sítě a adresu počítače. Tyto adresy jsou určeny pro tzv. adresné oběžníky (*multicast*).
- ◆ **Třída E** tvořící zbytek adres je t.č. rezervou.

**Tabulka 6.1:** Třídy IP adres

Třída	1. bajt IP adresy	2. bajt IP adresy	3. bajt IP adresy	4. bajt IP adresy
A	$0sssssss$ $1-127_{10}$	adresa počítače		
B	$10ssssss$ $128-191_{10}$	$ssssssss$	adresa počítače	
C	$110ssssss$ $192-223_{10}$	$ssssssss$	$ssssssss$	adresa počítače
D	$1110mmmm$ $224-239_{10}$	$mmmmmmmm$	$mmmmmmmm$	$mmmmmmmm$
E	$>239_{10}$			

Jednotlivé třídy adres jsou shrnuty v tab. 6.1, kde s vyznačuje bity používané pro adresu sítě a m byt používané pro adresný oběžník.

Z tabulky je dále patrné, že síti třídy A může být celkem  $128-2 = 126$  a v každé může být  $2^{8+8+8} = 16$  M adres. Obdobně síti třídy B může být 14 K a každá může obsahovat až 64 K adres. A konečně síti třídy C může být 2 M a každá může obsahovat až 256 adres. Některé adresy jsou však vyhrazeny pro speciální účely.

## Speciální IP adresy

IP adresa byla v této epoše obecně tvaru:

*síť.počítač*

kde síť je v případě třídy A tvořena jedním bajtem, v případě třídy B tvořena dvěma bajty a v případě třídy C tvořena třemi bajty.

Jsou-li na místě síť nebo počítače binárně samé nuly (00...0), pak se to vyjadřuje slovem „tento“. Jsou-li tam naopak samé jedničky (11...1), pak se to vyjadřuje slovem „všichni“ (či oběžník).

Přehled speciálních adres ve dvojkové notaci je uveden v tab. 6.2.

**Tabulka 6.2:** Speciální IP adresy

Typ adresy	Význam
0.0.0.0	Tento počítač na této síti
00...0.počítač	Počítač na této síti
síť.00...0	Adresa síť jako takové
síť.11...1 (samé jedničky na místě adresy počítače)	Všeobecný oběžník ( <i>broadcast</i> ) zasílaný do sítě sí – možno poslat i na vzdálenou síť
11...1 (samé jedničky, tj. desítkově 255.255.255.255)	Všeobecný oběžník na lokální síti ( <i>limited broadcast</i> ) – směrovače jej nepředávají dále
127.cokoliv (desítkově)	Programová smyčka ( <i>loopback</i> ) – nikdy neopouští počítač, zpravidla se používá adresa 127.0.0.1
224.0.0.1 (desítkově)	Adresný oběžník ( <i>multicast</i> ) adresovaný všem systémům na LAN
224.0.0.2 (desítkově)	Adresný oběžník ( <i>multicast</i> ) adresovaný všem směrovačům na LAN

Každé síťové rozhraní má alespoň jednu jednoznačnou adresu (*unicast*). Kromě toho celý systém má jednu adresu programové smyčky 127.0.0.1. Adresa 127.0.0.1 není v Internetu jednoznačná, protože ji má každý počítač (*host*).



**Tip:** Prvním testem nainstalovaného TCP/IP na každém počítači by měl vždy být příkaz „ping 127.0.0.1“.

## Příklad

Síť 192.168.6.0 je síť třídy C. Zjistěte, jaké jsou všechny běžící počítače na této síti. Řešení je jednoduché. Všeobecný oběžník (*broadcast*) na této síti má IP adresu 192.168.6.255. Po vydání příkazu:

```
ping 192.168.6.255
```

všechny běžící počítače na této síti odpoví ICMP paketem echo. Implementace příkazu ping firmy Microsoft bohužel nezobrazí všechny odpovědi, většina ostatních implementací nám všechny odpovědi zobrazí, takže zjistíme, které počítače na síti běží. V Linuxu je třeba u příkazu ping použít přepínač `-b`.

Obdobně lze příkazem ping (s TTL=1) zjistit, které počítače na LAN zpracovávají které adresné oběžníky. Např.:

```
ping 224.0.0.1
```

Zde však nebylo třeba nastavovat TTL na 1 (což se provádí v Linuxu přepínačem `-t` a v MS přepínačem `-i`), protože adresný oběžník 224.0.0.1 se šíří jen v rámci LAN.



**Poznámka:** Poslední dobou je bohužel snaha tyto užitečné nástroje potlačovat. Je to přitom nesmysl, protože běžící počítače na LAN můžeme snadno zjistit ARP-pingem (viz str. 40).

## Sítová maska

Sítová maska je nástroj pro zjištění (vyříznutí) adresy sítě z IP adresy. To znamená, že sítová maska určuje, které byty v IP adresě tvoří adresu sítě. Sítová maska je, obdobně jako IP adresa, čtyřbajtová. Bity sítové masky, které patří adrese sítě, jsou v sítové masce nastaveny na 1 a ostatní byty na 0.



**Poznámka:** Sítová maska je tvořena zleva souvislou řadou jedniček.

Jednotlivé třídy mají tak následující masky ( $11111111_2 = 255_{10}$ ):

- ◆ Třída A má masku 255.0.0.0
- ◆ Třída B má masku 255.255.0.0
- ◆ Třída C má masku 255.255.255.0
- ◆ Ostatní třídy masky nemají

Tyto sítové masky budeme označovat jako „standardní sítové masky“.



**Obrázek 6.2:** IP adresa a její sítová maska

Princip síťové masky se dobře pochopí na příkladu, když používáme dvojkovou notaci:

### Příklad

Určete adresu sítě, na které leží počítač o IP adrese:

170.85.255.248, tj. dvojkově 10101010.01010101.11111111.11111000

Řešení je jednoduché: Nejprve se podíváme do tabulky tříd (tab. 6.1) a zjistíme, že naše adresa je třídy B, protože o prvním bajtu zadané IP adresy platí:  $127 < 170 < 192$ . Používáme-li standardní síťovou masku, pak maska pro třídu B je 255.255.0.0, tj. dvojkově:

11111111.11111111.00000000.00000000

Napíšeme-li nyní IP adresu a její síťovou masku podtrhneme, pak vynásobením bitu po bitu získáme adresu sítě:

$$\begin{array}{r} 10101010.01010101.11111111.11111000 \\ \times \quad 11111111.11111111.00000000.00000000 \\ \hline 10101010.01010101.00000000.00000000 \end{array}$$

Výsledek převedeme do desítkové soustavy a zjistíme, že počítač leží na síti 170.85.0.0.

Tato metoda určení adresy sítě se může zdát až příliš komplikovanou v případě, že se používají standardní síťové masky. Význam síťové masky doceníme až v následující historické epoše.

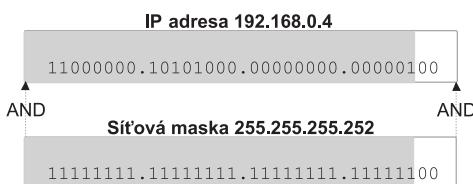


**Tip:** Pro ty, kteří nemají rádi převádění do dvojkové soustavy: stačí použít kalkulačku s operací „AND“. Pak zadáme desítkově bajt IP adresy „AND“ desítkově bajt síťové masky – a je to!

## Síť – historická epocha II

V roce 1993 vyšly normy RFC-1517 až 1520 (*Classless Inter-Domain Routing – CIDR*). Tyto dnes již málo citované normy od základu změnily pohled na slovo síť, jak je chápáno v Internetu. Přestalo se na síť hledět přes třídy, ale výhradně přes síťové masky.

**Síť chápeme souvislou řadu IP adres, které mají společnou síťovou masku.**



**Obrázek 6.3:** IP adresa 192.168.0.4 s nestandardní maskou 255.255.255.252

A tak můžeme mít např. síť 192.168.0.0 nejenom se standardní maskou 255.255.0.0, ale např. s maskou 255.255.255.252 apod. Zajímavější než síť 192.168.0.0 bude ale např. síť 192.168.0.4 také s maskou 255.255.255.252 (obr. 6.3). Jelikož dvojkové vyjádření síťové masky je tvořeno zleva souvislou řadou

třiceti jedniček, tak se místo vyjádření „síť 192.168.0.4 s maskou 255.255.255.252“ častěji zkracuje na síť „192.168.0.4/30“, kde číslo 30 vyjadřuje počet jedniček masky.

Pokud zapomeneme na třídy a budeme používat libovolné masky, pak už nestačí mluvit jen o síti, ale vždy k ní musíme dodat i síťovou masku, abychom vyjádřili, co touto sítí míníme (masku totiž již nelze odvozovat od prvních bitů IP adresy).



**Poznámka:** Rozdíl oproti sítím minulé epochy spočívá vlastně jen v tom, že počet jedniček v síťové masce není omezen jen na 8, 16 nebo 24, tzn. že dělení IP adresy na adresu síti a adresu počítače nemusí být na hranici bajtu.

## Subsítě a supersítě

Máme-li přidělenu nějakou síť, pak si ji můžeme rozdělit na subsítě. Základní pravidla pro subsítě jsou následující:

- ◆ Subsítě je rovněž síť:
  - má opět masku tvořenou souvislou řadou jedniček,
  - maska subsítě má více jedniček než původní síť.
- ◆ IP adresy subsítě jsou souvislou podmnožinou IP adres původní síti.
- ◆ Cílem je rozdělit síť na subsítě tak, aby:
  - se subsítě vzájemně nepřekrývaly (tj. byly vzájemně disjunktní),
  - subsítě vyčerpaly všechny adresy původní síti.

Kromě subsítí se používají supersítě, u kterých je počet jedniček masky menší než u standardní síťové masky. Jako příklad je v tab. 6.3 uvedeno dělení síti 192.168.0.0 na subsítě i supersítě s různými maskami (standardní maska je zobrazena tučně). Adresy s maskami majícími méně jedniček než standardní maska se nazývají adresy supersítí (v tabulce 6.3 nahoře) a adresy s maskami o více jedničkách, než má standardní maska, se nazývají adresy subsítí (dolní část tabulky 6.3). Supersítě se používají k agregaci sítí v rámci poskytovatele Internetu. Později se dozvíme, že tuto agregaci lze dělat i přes celé kontinenty. Např. supersítě 194.0.0.0/7 je jedním z intervalů IP adres přidělovaných evropským poskytovatelům Internetu.

**Tabulka 6.3:** Příklad aggregace/dělení síti 192.168.0.0 na supersítě/subsítě

Maska	Počet jedniček v masce (zleva)	Síť je tvořena intervalom IP adres	Zkrácený zápis síti (včetně masky)
255.248.0.0	13	192.168.0.0 až 192.175.255.255	192.168.0.0/13
255.252.0.0	14	192.168.0.0 až 192.171.255.255	192.168.0.0/14
255.254.0.0	15	192.168.0.0 až 192.169.255.255	192.168.0.0/15
255.255.0.0	16	192.168.0.0 až 192.168.255.255	192.168.0.0/16
255.255.248.0	21	192.168.0.0 až 192.168.7.255	192.168.0.0/21
255.255.252.0	22	192.168.0.0 až 192.168.3.255	192.168.0.0/22
255.255.254.0	23	192.168.0.0 až 192.168.1.255	192.168.0.0/23

Maska	Počet jedniček v masce (zleva)	Síť je tvořena intervalom IP adres	Zkrácený zápis sítě (včetně masky)
<b>255.255.255.0</b>	24	<b>192.168.0.0 až 192.168.0.255</b>	<b>192.168.0.0/24</b>
255.255.255.128	25	192.168.0.0 až 192.168.0.127	192.168.0.0/25
255.255.255.192	26	192.168.0.0 až 192.168.0.63	192.168.0.0/26
255.255.255.224	27	192.168.0.0 až 192.168.0.31	192.168.0.0/27
255.255.255.240	28	192.168.0.0 až 192.168.0.15	192.168.0.0/28
255.255.255.248	29	192.168.0.0 až 192.168.0.7	192.168.0.0/29
255.255.255.252	30	192.168.0.0 až 192.168.0.3	192.168.0.0/30
<b>255.255.255.254</b>	<b>31</b>	<b>192.168.0.0 až 192.168.0.1</b>	<b>192.168.0.0/31</b>
255.255.255.255	32	Adresa samostatného počítače ( <i>host address</i> )	192.168.0.0/32



**Poznámka:** Síť s maskou /31 (tj. 255.255.255.254) se nepoužívají. Proč? Protože jsou absurdní, neboť obsahují jen adresu sítě a všeobecného oběžníku – na samotné počítače na takovýchto síťích už totiž nezbýlo místo!



**Poznámka:** Síťová maska nerozlišuje mezi částí IP adresy určené pro síť a pro subsítě – všude jsou jedničky. I u subsítě se používají speciální IP adresy (tab. 6.4). Filosofie však zůstává stejná.

**Tabulka 6.4:** Speciální adresy

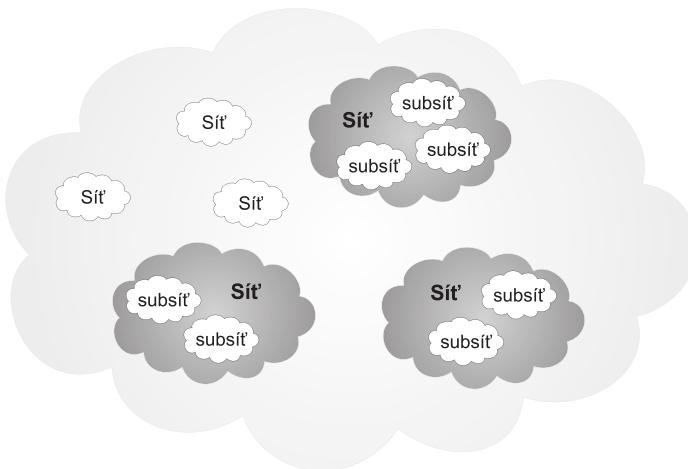
sít.subsít.00...0	Adresa subsítě jako takové
sít.00...0.00...0	Adresa sítě
sít.subsít.11...1	Oběžník na subsítě
sít.11...1.11...1	Pozor, toto je oběžník pro všechny subsítě dané sítě

Problém je ale se subsítě, která má na místě pro subsítě nuly, protože pak se těžko rozlišuje mezi adresou sítě (se standardní maskou) a subsítě. Rovněž u případu, kdy na místě pro subsítě jsou samé jedničky, je nejednoznačnost, zda oběžník je oběžníkem na subsítě, nebo na všechny subsítě. Proto se tyto subsítě snažíme nepoužívat. Mnohý software takové subsítě vůbec nepodporuje, jiný software je v případě použití takovýchto subsítě třeba speciálně konfigurovat.

Avšak oběžník na všechny subsítě dané sítě je stejně jen teorie. Nesetal jsem se s případem, že by to šlo využít, protože směrovač nemá informaci, na jaké subsítě je vzdálená síť dělena.

Subsítě se používají v rámci firem na konfiguraci jednotlivých lokálních sítí. Vzhledem k nedostatku IP -adres má dnes většina firem přiděleno jen subsítě sítě třídy C. Tuto subsítě si pak dělí na ještě menší subsítě.

Subsítě je část Internetu odpovídající jedné firmě nebo části firmy.



**Obrázek 6.4:** Internet je tvořen sítěmi, sítě se mohou dělit na subsítě

### Příklad

Mám za úkol připojit firmu k Internetu. Získal jsem adresu třídy C (např. 194.149.115.0 desítkově, tj. 11000010.10010101.1110011.00000000 dvojkově), která má standardní síťovou masku 255.255.255.0. Měl jsem tedy štěstí, že jsem dostal celou adresu třídy C.

Problém je v tom, že můj podnik má složitější strukturu – jeho síť se skládá z řady menších lokálních sítí (LAN) a řady sériových linek propojujících tyto LAN. Je tedy nutné rozdělit přidělenou síť na subsítě. Navenek se pochopitelně bude celý podnik tvářit jako jedna síť se standardní maskou.

Vzhledem k tomu, že první tři bajty přidělené IP adresy zůstávají stále stejné, budu v dalších úvahách psát jen poslední bajt v IP adresě (první tři bajty jsou konstantně např. 194.149.115).

Pro rozdělení sítě na subsítě se na první pohled naskytá možnost nepoužít standardní síťovou masku pro adresu třídy C, tj. 255.255.255.0, ale pro všechny subsítě stejnou – **konstantní síťovou masku** 255.255.255.240 (dvojkově 11111111.11111111.11111111.11110000 – všimněte si, že první polovina posledního bajtu slouží pro adresu subsítě), která nám umožní rozdělit přidělenou adresu třídy C na 16 subsítí a každá subsíť může mít 16 adres. O konstantní síťové masce hovoříme, když všechny subsítě mají stejný počet jedniček v síťové masce.

**Tabulka 6.5:** Konstantní síťová maska

Subsíť dvojkově (poslední bajt z IP adresy 194.149.115.0)	Síťová maska (dvojkově)	Adresa subsítě (desítkově)	Síťová maska (desítkově)	Max. počet počítačů v subsítě (bez adresy subsítě a oběžníku)
00000000 až 00001111	11110000	.0	.240	0 (nejednoznačná subsíť)
00010000 až 00011111		.16		14
00100000 až 00101111		32		14
00110000 až 00111111		.48		14

<b>Subsít dvojkově (poslední bajt z IP adresy 194.149.115.0)</b>	<b>Síťová maska (dvojkově)</b>	<b>Adresa subsítě (desítkově)</b>	<b>Síťová maska (desítkově)</b>	<b>Max. počet počítačů v subsítě (bez adresy subsítě a oběžníku)</b>
01000000 až 01001111		.64		14
01010000 až 01011111		.80		14
01100000 až 01100000		.96		14
01110000 až 01111111		.112		14
10000000 až 10001111		.128		14
10010000 až 10011111		.144		14
10100000 až 10101111		.160		14
10110000 až 10111111		.176		14
11000000 až 11001111		.192		14
11010000 až 11011111		.208		14
11100000 až 11101111		.224		14
11110000 až 11111111		.240		0 (nejednoznačná subsít)

Každá subsít má 16 adres, ale použitelných je pouze 14, protože dvě adresy mají speciální význam:

- ◆ Samé nuly v části subsítě označují jak adresu subsítě, tak i celé síť. Např. adresa 194.149.115.0 je jak adresou síti 194.149.115.0/24, tak i subsítě 194.149.115.0/28.
- ◆ Samé jedničky zase označují adresu všeobecného oběžníku. Problémem je, že není jednoznačně určeno, zda 194.149.155.255 je oběžník pro všechny subsítě síti 194.149.155.0/24, nebo jen subsítě 94.149.115.240/28.

Proto se subsítím 94.149.115.0/24 a 94.149.115.240/28 raději vyhneme

V praxi často nepotřebujeme rozdělit přidělenou adresu na stejné části. Např. pro sériové linky je 14 adres na subsítě zbytečný luxus a naopak pro mnohé LAN je to nedostatečné. Pro rozdělení síti na různě dlouhé subsítě používáme **variabilní síťovou masku**. Např. viz tab. 6.6.

**Tabulka 6.6:** Variabilní síťová maska

<b>Subsít dvojkově (poslední bajt)</b>	<b>Síťová maska (dvojkově)</b>	<b>Adresa subsítě (desítkově) 194.149.115.</b>	<b>Síťová maska (desítkově)</b>	<b>Max. počet počítačů v subsítě (bez adresy subsítě a oběžníku)</b>
00000000 až 00000011	11111100	.0	.252	0 (nejednoznačná subsít)
00000100 až 00000111	11111100	.4/30	.252	2
00001000 až 00001111	11111000	.8/29	.248	6
00010000 až 00011111	11110000	.16/28	.240	14
00100000 až 00111111	11100000	.32/27	.224	30
01000000 až 01111111	11000000	.64/26	.192	62
10000000 až 10111111	11000000	.128/26	.192	62

<b>Subsít dvojkově (poslední bajt)</b>	<b>Síťová maska (dvojkově)</b>	<b>Adresa subsítě (desítkově) 194.149.115.</b>	<b>Síťová maska (desítkově)</b>	<b>Max. počet počítačů v subsítě (bez adresy subsítě a oběžníku)</b>
11000000 až 11011111	11100000	.192/27	.224	30
11100000 až 11101111	11110000	.224/28	.240	14
11110000 až 11100011	11111000	.240/29	.248	6
11111000 až 11111011	11111100	.248/30	.252	2
11111100 až 11111111	11111100	.252/30	.252	0 (nejednoznačná subsít)

Z předchozí tabulky je vidět, že nejdelší subsít má 64 prvků; potřebujeme-li na jednu LAN více jak 62 síťových rozhraní, pak je dobré použít celou adresu třídy C.

Nyní si můžeme dát nový příklad.

### Příklad

Určete adresu sítě, na které leží počítač o IP adrese 10.0.0.239, používáme-li síťovou masku 255.255.255.240.

IP adresu i masku převedeme do dvojkové soustavy, napíšeme pod sebe, podtrhneme a bit po bitu vynásobíme:

$$\begin{array}{l}
 \begin{array}{l} 00001010.00000000.00000000.11101111 \text{ tj. } 10.0.0.239 \\
 \times \quad 11111111.11111111.11111111.11110000 \text{ tj. } 255.255.255.240 \end{array} \\
 \hline
 00001010.00000000.00000000.11100000 = 10.0.0.224
 \end{array}$$

Adresa leží na síti 10.0.0.224. Ale může to být adresa počítače? Ne. Proč? Oddělíme adresu sítě a adresu počítače:

$$\begin{array}{l}
 00001010.00000000.00000000.1110|1111 \\
 \leftarrow \text{--- síť ---} \rightarrow | \langle \text{poč} \rangle
 \end{array}$$

Adresa je tvaru síť.jedničky, nejdříve se tedy o adresu počítače, ale o adresu oběžníku na síti 10.0.0.224!

### Supersítě a autonomní systémy

Zatímco subsítě se používají v konfiguracích jednotlivých síťových rozhraní (síťových karet), tak supersítě se používají pro agregace IP adres. Agregace IP adres je výhodná pro směrování a pro administrativu při přidělování IP adres.

V současné době je při pohledu z velké vzdálenosti (z Měsíce) Internet soustavou vzájemně propojených poskytovatelů Internetu. Poskytovatel Internetu (*provider*) poskytuje připojení k Internetu buď pro komerční, nebo nekomerční účely. Kromě poskytovatelů tvoří Internet ještě několik organizací, které se zabývají správou a vývojem v této oblasti, avšak ze síťového hlediska se od poskytovatelů neliší.

Poskytovatelé dopravují IP datagramy buď v rámci sebe, nebo mezi sebou. Dva poskytovatelé si mohou vyměňovat IP datagramy mezi sebou, existují však i tranzitní poskytovatelé, kteří přes sebe IP datagramy tranzituji.

Neříká se, že se Internet dělí na poskytovatele, ale z hlediska dopravy IP datagramů se Internet dělí na **autonomní systémy** (AS). Každý poskytovatel má totiž přidělen jeden nebo více autonomních systémů. Autonomní systém je reprezentován dvoubajтовým číslem.

Internet je tedy z hlediska směrování (tj. dopravy IP paketů) rozdělen na AS. Pro směrování mezi AS se dříve používal protokol EGP, později se přešlo na protokol BGP.

Poskytovatelé Internetu jsou správci autonomního systému. Správci AS žádají o intervaly IP adres, které přidělují sobě a svým zákazníkům (kap. 13). Jednotliví poskytovatelé jsou pak i se svými zákazníky součástí konkrétního AS.

Proč je snaha v rámci autonomního systému používat intervaly adres? Důvod je prostý. Interval adres je možné agregovat do jedné adresy supersítě. Ve směrových tabulkách směrovačů vzdálených autonomních systémů může celý interval adres vystupovat jako jedna položka, čímž se šetří paměť směrovače a zjednoduší se správa.

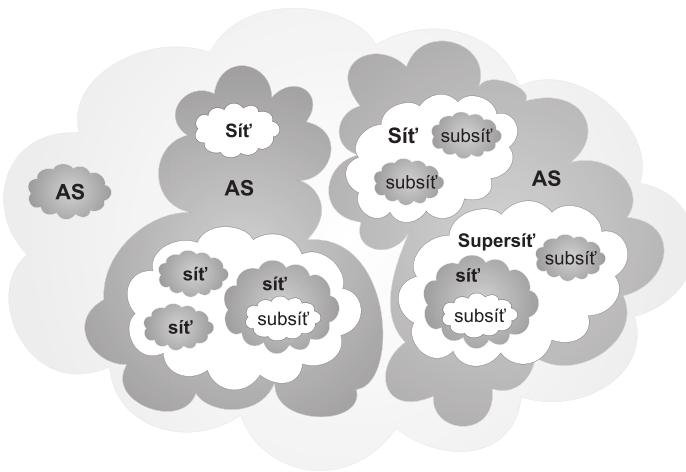
Agregace je jednoduchá. Např. je-li přidělen interval adres sítí 62.177.64.0 až 62.177.127.0, pak je možné jej agregovat na adresu supersítě 62.177.64.0/18.

Zatímco při dělení sítě na subsítě obsahovala maska více jedniček, při agregaci je tomu naopak, ale princip je týž. O aggregovaných sítích se hovoří jako o supersítích. Z pohledu vzdáleného AS se supersíť jeví jako jeden aggregovaný celek.



**Poznámka:** Pro správce AS je síť jeden celek. A pro správce sítě jsou zase jednotlivými celky subsítě.

Problém ale může nastat, když firma přejde od jednoho poskytovatele Internetu k jinému, který je navíc v jiném autonomním systému. Pak musí od nového poskytovatele získat nové IP adresy a následně všechny používané sítě přečíslovat. Jména počítačů (tj. i e-mailová adresa) však mohou zůstat zachována.



**Obrázek 6.5:** Internet se dělí na autonomní systémy, autonomní systémy se mohou dělit na supersítě, supersítě na sítě a sítě se mohou dělit na subsítě

Existují i adresy na poskytovateli nezávislé (*provider independent*), které jsou poskytovány pro firmy připojené k více poskytovatelům současně nebo LAN tvořící jádro **NIX** (*Neutral Internet eXchange*), kde si jednotliví poskytovatelé kolektivně mezi sebou předávají IP datagramy.

Představme si hypotetický příklad: Naší firmě byl přidělen interval IP adres 62.177.90.0/24. Firma byla situována do několika lokalit, a tak správce sítě přidělil lokalitě v Českých Budějovicích rozsah 62.177.90.176/28. V této lokalitě je počítač o IP adrese 62.177.90.190. Tento počítač chce komunikovat se serverem na Fidži (stáhnout webovou-stránku <http://www.fijimuseum.org.fj/>). IP datagramy jsou doprovázeny Internetem na Fidži. Nyní nám chce fidžiský server odpovědět. Zformuluje odpověď a zkoumá adresu 62.177.90.190, kudy odpovědět (do jakého podmořského kabelu odpověď strčit).

Adresa 62.177.90.190 patří do intervalu 62.0.0.0/8 přiděleného pro RIPE (organizace přidělující IP -adresy v Evropě a okolních teritoriích). Z hlediska serveru na Fidži je adresát někde mezi Kanárskými ostrovy, norským Svalbardem a Alma Atou. Na Fidži by se tedy teoreticky ani nemuseli zabývat otázkou, do jakého autonomního systému adresa 62.177.90.190 patří, v případě, že směrem do Evropy strkají všechno do jednoho kabelu. Teoreticky by jim tak stačilo mít ve směrovací tabulce jen jednu položku:

62.0.0.0 s maskou 255.0.0.0.

Z Fidži je to do Evropy geograficky přibližně stejně daleko směrem ze západu, na východ, na jih i na sever, avšak náš IP datagram tč. pošlou do USA. V Americe už to není tak jednoduché, z Ameriky do Evropy vede řada spojů, je tedy nutné nejprve zjistit, směrem ke kterému spoji to v Americe dopravit a pak do kterého spoje to v Americe strčit (během dopravy může dojít i ke změně názoru na to, do jakého spoje do Evropy nás IP datagram strčit). Američané zjistí, že adresa 62.177.90.190 patří do intervalu 60.177.64.0/18 patřícího k autonomnímu systému AS6706. Ve směrovací tabulce svých směrovačů již pro každý interval adres, který má přidělen autonomní systém AS6706, musí mít jednu položku. V našem případě:

62.177.64.0 s maskou 255.255.192.0.

Důležité směrovače ležící v USA na hranici autonomních systémů musí tedy mít pro každý interval IP adres přidělený nějakému poskytovateli kdekoli na Zemi jednu položku. Říkáme, že takový směrovač má úplné směrovací tabulky Internetu. Takovéto směrovače s úplnými směrovacími tabulkami jsou nutné jako hraniční směrovače tzv. tranzitních autonomních systémů, tj. autonomních systémů, přes které se IP datagramy dopravují do dalších autonomních systémů. Jestliže náš IP datagram se v USA objevil na západním pobřeží, tak pravděpodobně bude muset být dopraven skrze tranzitní autonomní systémy USA na východní pobřeží, odkud pravděpodobně povedou podmořské kably do Evropy.

Předávání IP datagramů mezi autonomními systémy nezávisí pouze na technických faktorech, tj. technicky nejvýhodnějším spojení směrem k adresátovi, ale také na směrovací politice (*routing policy*) mezi jednotlivými autonomními systémy – lidově řečeno jestli druhá strana platí, nebo ne. V případě výskytu nějakých překážek může být náš IP datagram dopravován komplikovanější cestou nebo směrování může být i administrativně zakázáno.

Náš IP datagram mezitím už dorazil z Ameriky na hraniční směrovač autonomního systému AS6706. Nyní již musí směrovač zkoumat IP adresu podrobněji a zjistí, že IP adresa 62.177.90.190 patří do intervalu 162.177.90.0/24, který byl přidělen naší firmě.

Ve směrovacích tabulkách směrovačů uvnitř autonomního systému AS6706, je položka:

62.177.90.0 s maskou 255.255.255.0.

IP datagram poskytovatel dopraví na hraniční směrovač naši firmy. Směrovač naši firmy zkoumá, kam má IP datagram dopravit v rámci naši firmy, proto zkoumá adresu 62.177.90.190 a zjistí, že má položku směrovací tabulky (LAN v Českých Budějovicích):

62.177.90.176 s maskou 255.255.255.240

Náš IP datagram je dopraven do Českých Budějovic na směrovač. Ten zjistí, že síť 62.177.90.176/28 je síť přímo připojená na jeho lokální rozhraní. Protokolem ARP zjistí šestibajtovou linkovou adresu příjemce (pokud ji nemá v ARP-cache) a příjemci dopraví náš IP datagram. Příjemce zahodí IP -záhlaví a z TCP záhlaví zjistí, že informace je určena pro webový prohlížeč, zahodí se TCP záhlaví a obsah v protokolu HTTP se zobrazí (interpretuje) na obrazovce. Právě jsem to vyzkoušel a tě. celý tento proces mezi serverem v Suvu (Fidži) a klientem v Českých Budějovicích trvá cca 0,35 sekundy. To nevypovídá jen o rychlosti a propustnosti přenosových linek, ale i o ohromném výkonu hraničních směrovačů, které musí ohromnou rychlosť vyhledávat ve směrovacích tabulkách, proto často bývají vybaveny specializovanými procesory, jež se zabývají obsluhou směrovacích tabulek.

Čísla autonomních systémů přidělují mezinárodní regionální agentury pro přidělování IP adres (*Internet Registry*). V Evropě je takovou agenturou RIPE. Tyto agentury udržují ve svých databázích informace o intervalech přidělených IP adresám i o číslech přidělených autonomním systémům.

Na serveru <http://www.isc.org> lze nalézt nástroj *prtraceroute*, který v sobě volá příkaz *traceroute*, avšak z výstupu příkazu *traceroute* dohledává informace v databázích regionálních agentur příkazem *whois* (kap. 13). Takže např. cesta na Fidži z hlediska autonomních systémů vypadá takto (všimněte si posledního řádku obsahující cestu přes autonomní systémy):

```
$ prtraceroute www.fijimuseum.org.fj
1 [AS6706] lo0.adsl-plus-jhc.adsl.vol.cz (212.20.125.148) 85 ms 14 ms 13 ms
2 [AS6706] adsl-plus-1.adsl.ctc-ptp04.vol.cz (212.20.125.141) 99 ms 14 ms 15 ms
3 [AS6706] ge3-42.c17.prg.vol.cz (195.122.209.37) 15 ms 15 ms 14 ms
4 [AS6706] ge3-42.c17.prg.vol.cz (195.122.209.37) 14 ms 14 ms 16 ms
5 [AS6706] ge5-1.tr3.prg.vol.cz (195.122.207.119) 16 ms 19 ms 15 ms
6 [AS8447] AUX1-Czech-Online.highway.telekom.at (195.3.102.209) 26 ms 26 ms 25 ms
7 [AS8447] IIX2-WARSSW02.highway.telekom.at (195.3.70.196) 29 ms 28 ms 27 ms
8 [AS3356] 212.73.202.122 ms 20 ms 24 ms
9 [AS3356] so-4-0-0.mpl1.Vienna1.Level3.net (4.68.112.77) 22 ms 23 ms 22 ms
10 [AS3356] ae-0-0.bbr2.NewYork1.Level3.net (64.159.1.42) 114 ms 121 ms 113 ms
11 [AS3356] ae-11-51.car1.NewYork1.Level3.net (4.68.97.20) 114 ms 122 ms 114 ms
12 [AS3356] mci-level13-oc48.NewYork1.Level3.net (4.68.111.30) 118 ms 118 ms 115 ms
13 [AS701] 0.so-6-0-0.XL1.NYC4.ALTER.NET (152.63.21.78) 119 ms 118 ms 122 ms
14 [AS701] 0.so-7-0-0.XL1.SAC1.ALTER.NET (152.63.53.249) 207 ms 206 ms 205 ms
15 [AS701] POS6-0.IG3.SAC1.ALTER.NET (152.63.54.121) 241 ms 206 ms 207 ms
16 [AS701] fintelfiji4-gw.customer.alter.net (157.130.214.186) 338 ms 335 ms 337 ms
17 [AS9241] 202.170.33.15335 ms 336 ms 333 ms
18 [AS9241] 202.137.176.253336 ms 332 ms 336 ms
19 [AS9241] juniper1.is.com.fj (210.7.20.2) 338 ms 335 ms *
20 [AS9241] www.fijimuseum.org.fj (202.62.120.2) 339 ms 337 ms 335 ms

Path taken:
AS6706 AS8447 AS3356 AS701 AS9241
```

První sloupec nám uvádí číslo hopu, druhý sloupec číslo autonomního systému (v desítkové soustavě předcházené řetězcem AS), třetí sloupec uvádí název rozhraní na směrovači, čtvrtý IP adresu rozhraní směrovače a další pak čas – viz příkaz traceroute.

Dále pak následuje samostatný řádek vypisující cestu, kde se hopem nerozumí směrovač, ale celý autonomní systém.

Příkaz prtraceroute je velice užitečný pro správce autonomních systémů, protože není-li někam spojení, pak příkazem prtraceroute zjistí, do jakého autonomního systému je až spojení. Následně Uniovým příkazem whois (nebo na webu jednotlivých RIR – viz kap. 13 Mezinárodní a národní organizace Internetu) můžeme zjistit informace o jednotlivých AS. Např. na [www.apnic.net](http://www.apnic.net) pomocí formuláře WHOIS vyhledáme kontaktní osobu pro případ, kdy by nám komunikace vázla:

```
aut-num: AS9241
as-name: FINTEL-FJ
descr: Fiji International Telecommunications Ltd
country: FJ
remarks: www.fintelfiji.com
admin-c: IK136-AP
tech-c: LN21-AP
mnt-by: MAINT-FJ-LNAKACIA
changed: lmnakacia@fintelfiji.com 20040316
source: APNIC

person: Ioane N Koroivuki
nic-hdl: IK136-AP
e-mail: inkoroivuki@fintelfiji.com
address: FINTEL
address: PO Box 59
address: Suva
address: Fiji
phone: +679-3312933
fax-no: +679-3300750
country: FJ
changed: lmnakacia@fintelfiji.com 20030128
mnt-by: MAINT-FJ-FINTEL
source: APNIC

person: Laisiasa Nakacia
address: C/- Fintel
address: PO Box 59
address: Suva, Fiji
country: FJ
phone: +679-312933
fax-no: +679-300750
e-mail: lmnakacia@fintelfiji.com
nic-hdl: LN21-AP
mnt-by: MAINT-NEW
changed: lmnakacia@fintelfiji.com 20010310
source: APNIC
```

## IP adresy v intranetu

Použití technologie Internetu uvnitř uzavřené firemní sítě se nejprve označovalo internet (s malým i), později se objevilo slovo intranet, které se uchytilo (Němci si z toho dělají legraci, že to bylo kvůli nim, protože v němčině byl internet s malým počátečním písmenem i nepřekonatelným trnem v oku němčinářům).

IP adresy musí být v Internetu přidělovány celosvětově jednoznačně. Před lety mnohé podniky budovaly svou uzavřenou podnikovou síť na bázi protokolu TCP/IP a ani ve snu je nenapadlo, že by se někdy připojovaly k Internetu. I zvolily si naprostě libovolné adresy vlastních sítí, které často kolidovaly se sítěmi v Internetu. Dnes chtějí tyto sítě propojit přes firewall do Internetu a zjišťují, že stejné adresy už v Internetu někdo používá. Jsou nuceny své sítě přečíslovat, což je velice nepříjemná operace.

Většinou firmy používající v intranetu adresy, které kolidují s adresami v Internetu, zpočátku hledají nějaká netradiční řešení, jak se vyhnout přečíslování intranetu. Takovým řešením je např. NAT (*Network Address Translator*), avšak tato řešení přináší jiná negativa, proto po zbytečně vynaloženém úsilí firmy stejně nakonec přistoupí k přeadresování celého intranetu.

Pro uzavřené podnikové sítě si zásadně zvolte IP adresy sítí podle RFC1918 uvedené v tab.6.7.

**Tabulka 6.7:** IP adresy uzavřených sítí („Privátní adresy“)

Třída A	10.0.0.0/8	10.0.0.0 až 10.255.255.255
Třída B	172.16.0.0/12	172.16.0.0 až 172.31.255.255
Třída C	192.168.0.0/16	192.168.0.0 až 192.168.255.255

Použití těchto adres navíc zvyšuje bezpečnost, protože v Internetu jsou nepoužitelné (stovky podniků je používají na svých uzavřených sítích). O přidělení adres v těchto rozsazích není třeba nikoho žádat.



**Poznámka:** Častou otázkou je, jak to poskytovatelé Internetu dělají, že tyto adresy nelze použít, oni je nějak filtroují? Filtrace není třeba, oni je prostě jen nemají ve směrovacích tabulkách, takže je nemohou doprovádat.

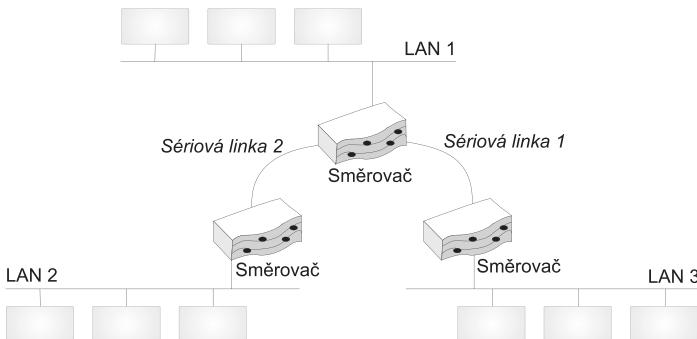
Je také zajímavé si prohlédnout doporučení RFC 3330 (*Special-Use IPv4 Addresses*), které specifikuje další intervaly IP adres pro speciální použití:

- ◆ 169.254.0.0/16: Blok adres pro výhradně lokální komunikaci mezi počítači. Tyto adresy využívá např. Microsoft, když se počítači žádným jiným způsobem nepodaří získat IP adresu (ani např. přes DHCP).
- ◆ 192.0.2.0/24: Blok přiřazený jako TEST-NET pro použití v dokumentaci a příkladech. Tyto adresy se nesmí objevit na Internetu.
- ◆ 198.18.0.0/15: Blok pro použití v testech výkonnosti (*benchmark tests*) – viz též RFC-2544.

Pro intranety jsou vyhrazena i čísla autonomních systémů 64512 až 65535 (viz RFC-1930).

## Nečíslované sítě

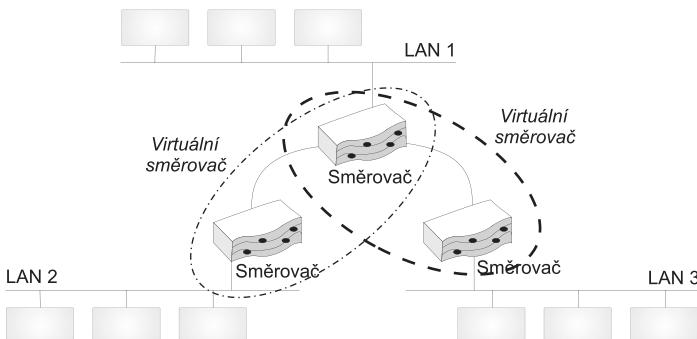
Zamysleme se nyní nad sériovými linkami spojujícími LAN. Pro každou linku potřebujeme subsíť o minimálně čtyřech IP adresách (adresa sítě, oběžník na síti a dvě adresy pro síťová rozhraní na směrovačích).



**Obrázek 6.6:** LAN propojené sériovými linkami

Z obrázku 6.6 je patrné, že kromě tří intervalů IP adres pro lokální sítě budeme potřebovat další adresy pro sítě tvořené sériovými linkami. Na první pohled je vidět, že by bylo efektivní, aby pro sériové linky nebyla nepotřeba další adresa sítě.

Současné směrovače umí na sériových linkách vytvořit „nečíslovanou“ síť (*unnumbered interface*), tj. protější směrovače se chovají jako jeden virtuální směrovač. Každý fyzický směrovač pak tvoří polovinu virtuálního směrovače. Virtuální směrovač má pouze dvě rozhraní – jedno pro každou LAN.



**Obrázek 6.7:** Nečíslované sítě

Pro sériové linky tak není třeba plýtvat IP adresami.

## Dynamicky přidělované adresy

Má-li síť již interval IP adres přidělen, pak můžeme začít s přidělováním adres jednotlivým síťovým rozhraním na této síti. Jsou dvě možnosti:

- ◆ Staticky (trvale) přidělit IP adresu.
- ◆ Dynamicky (na dobu připojení) přidělit IP adresu.

Dynamické přidělování přináší výhodu i v tom, že je potřeba jen tolik IP adres, kolik je současně přihlášených uživatelů.

Dynamické přidělování adres dnes řeší aplikační protokol DHCP. Protokol DHCP vychází ze zkušeností a částečně v sobě zahrnuje i podporu starších protokolů z této oblasti, tj. protokolů RARP, DRARP a BOOTP. Blíže viz RFC-1531.

V protokolu DHCP žádá klient DHCP-server o přidělení IP adresy (případně o další služby). DHCP-server může být realizován jako proces na počítači s operačním systémem UNIX, Windows Server atp. Nebo DHCP-server může být realizován i jako součást směrovače, přepínače nebo přístupového bodu WiFi.

Zatímco přidělování IP adres na LAN je v současné době doménou protokolu DHCP, pro přidělování IP adres počítačům za linkou do WAN (např. zákazníkům poskytovatele Internetu) se zpravidla přidělují IP adresy pomocí protokolu PPP.

Protokol PPP neumožňuje takové služby jako protokol DHCP, avšak přidělit stanici IP adresu a IP adresu serveru DNS umí. Stejně pro připojení jednotlivého uživatele k Internetu nebývá více třeba.

Dynamické přidělování IP adres může být ještě kombinováno s nečíslovanými sítěmi. V případě, že budou sériové linky označeny jako nečíslované sítě (obr. 6.7), pak směrovač dynamicky přidělující IP adresy zařídí, že počítače se budou javit, jako by byly přímo na lokální síti poskytovatele (na té, která je z pohledu zákazníka za WAN). Z hlediska vzdálenějších lokalit se pak počítače na LAN poskytovatele i na sériových linkách poskytovatele tváří jako jeden celek – jedna supersíť.

## Adresní plán

Každá firma, která se chce připojit k Internetu, si musí nejprve udělat adresní plán. Ten se obvykle skládá ze dvou částí. Jednak ze schematického znázornění propojení jednotlivých LAN do WAN a jednak ze seznamu jednotlivých LAN s odhadovaným počtem síťových rozhraní na LAN. Adresní plán by měl obsahovat rezervu s výhledem na příští a přespříští rok. Jako rezerva se běžně bere dvojnásobek současného stavu. Adresní plán se pak zasílá jako požadavek poskytovateli Internetu, kterého tím žádáme o příslušný počet IP adres.

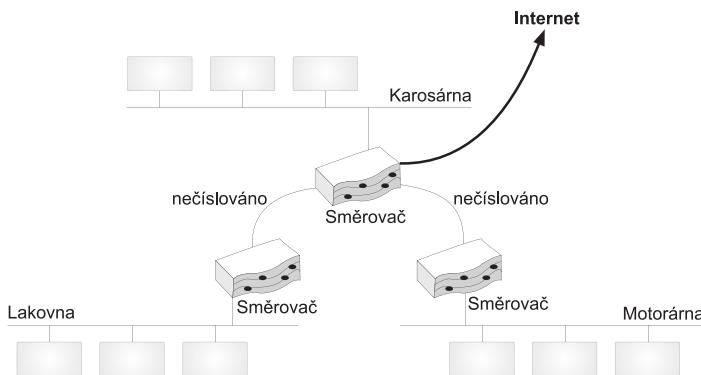
### Příklad

Máme připojit k Internetu firmu používající 3 lokální sítě: karosárna, lakovna a motorárna (nikdy se poskytovatel nespokojí s žádostí typu: tři sítě A, B a C – vždy se musí jednat o konkrétní požadavek).

V karosárně máme 8 počítačů s výhledem na 16, v motorárně 9 počítačů s výhledem na 18 a v lakovně je 20 počítačů s výhledem na 40 počítačů.

**Tabulka 6.7:** Současné a výhledové požadavky

LAN	Současný stav	Příští rok	Za 2 roky	Nejbližší možnost pro LAN	Požadováno
Karosárna	8	10	16	32 (16 nelze, protože je třeba adresa pro subsíť a oběžník)	<b>32</b>
Lakovna	9	15	18	32	<b>32</b>
Motorárna	20	35	40	64	<b>64</b>



**Obrázek 6.8:** Síť fiktivní firmy

Požadujeme na poskytovatele, aby přidělil 128 IP adres pro tři subsítě. V případě, že by těchto 128 adres mělo tvořit jeden celek – „supersíť“, pak nemůžeme požadovat supersíť o 128 adresách, protože jedna LAN by využívala nejednoznačnou subsíť C, v takovém případě je třeba žádat celou síť třídy C, tj. 256 IP adres. Aby všechny LAN z hlediska poskytovatele tvořily jeden celek („supersíť“), je vyžadováno zejména v případě, kdy firma využívá pro připojení k Internetu komutovaný spoj. Přitom komutovaným spojem může být zálohována i pevná linka (*dialup backup*).

Příklad neřešil problém sériové linky propojující firmu s Internetem. To je třeba projednat vždy s poskytovatelem.

Možná, že se vám zdá divné, proč připojovat jednotlivé provozy do Internetu. Větší a velké firmy se vyznačují tím, že nepotřebují více než 16 IP adres. Používají totiž firewall, který odděluje intranet od Internetu. A na intranetu používají zásadně privátní adresy (tab. 6.7). Nejvýše tak potřebují IP adresy pro:

- ◆ „internetovou“ stranu firewallu,
- ◆ síť pro demilitarizovanou zónu, kde je např. firemní WWW server (např. realizovaný jako reverzní proxy, tj. Access Manager).

## Více než 254 rozhraní na LAN

Někdy se stane, že na lokální síti je např. 300 počítačů. Nestačí tedy jedna síť třídy C. Přidělí se dvě síť třídy C. Je zde nebezpečí chybnej konfigurace v případě, že použijeme dvě samostatné sítě. Pak na LAN musí být i směrovač, který směruje mezi těmito dvěma sítěmi (nebo se použije proxy ARP). Počítače místo aby komunikovaly na síti přímo mezi sebou, tak musí využívat služeb směrovače. Horší na tom je, že data jdou sítí dvakrát. To je při třech stech počítačích na LAN obzvláště nepříjemné. Jednou od odesilatele na směrovač a podruhé ze směrovače k adresátovi.

Rozumné je v tomto případě použít supersíť skládající se ze dvou sítí třídy C, tj. supersíť s maskou /23, tj. 255.255.128.0.

## Domácí cvičení

Vytvořte adresní plán vaší firmy, školy apod. Pokud nevíte, jak zjistit, jaké veřejné IP adresy má vaše firma přidělena, pak s tímto úkolem posečkejte až po studiu kap. 13.

## Kapitola 7

# Směrování

Směrování IP datagramů (**IP routing**) a předávání IP datagramů (**IP forwarding**) jsou dva procesy, na kterých Internet stojí.

Základní schéma směrování je zobrazeno na obr. 7.1. Představme si, že jsme právě na obr. 7.1 přijali linkový rámec nesoucí IP datagram pomocí prvního síťového rozhraní. Jeho zpracování pak prochází následujícími kroky:

1. IP datagram je vybalen z linkového rámce a předán do vstupní fronty ke zpracování. Linková adresa + IP adresa odesilatele mohou být uloženy do paměti ARP pro další využití.
2. IP datagram je vybrán ze vstupní fronty ke zpracování.
3. Zpracování začíná vyhodnocením volitelných položek záhlaví IP datagramu. Začíná se zpracováním volitelných položek Explicitní směrování. V případě, že IP datagram byl pouze explicitně odkloněn přes tento systém, je přímo předán subsystému směrování (7) k odeslání dále.
4. Nyní se zjišťuje, je-li IP datagram adresován této stanici. V případě, že nikoliv, pak je rovněž předán subsystému směrování (7) k odeslání dále. (Klasickým případem, kdy IP datagram není adresován této stanici, je, když tato stanice slouží jako směrovač.)
5. Nyní se zkoumá, nenese-li IP datagram zprávu ICMP. V takovém případě se IP datagram předá subsystému ICMP pracujícímu na IP vrstvě ke zpracování (5). Obdobně se postupuje i v případě IGMP-paketů či paketů směrovacích protokolů. To již však není na obr. 7.1 znázorněno.
6. Nyní jsme již vyčerpali možnosti zpracování IP datagramu na vrstvě IP, tj. IP datagram v sobě nese buď TCP segment, nebo UDP datagram. Ty jsou vybaleny z IP datagramu a předány vyšší vrstvě ke zpracování (6).
7. Všechny odchozí IP datagramy jsou nejprve předány subsystému Směrování (7). Cílem tohoto subsystému je nalézt síťové rozhraní, kterým bude IP datagram zabalený do linkového rámce odeslán. K tomu subsystému Směrování pomáhají Směrovací tabulky.
8. Nyní se zkoumá, nejedná-li se o oběžník, který je též adresován této stanici. Pokud ano, pak se kopie IP datagramu též předá do vstupní fronty (2).
9. Jedná-li se o oběžník, pak se přejde k bodu 11. V opačném případě se přejde k bodu 10.
10. Nyní se zkoumá, není-li cílová adresa adresou této stanice. Pokud ano, pak se IP datagramu též předá do vstupní fronty (2).
11. Jelikož se jedná o oběžník, proběhne jeho přímé mapování na linkovou adresu a odeslání do sítě.
12. Jelikož se jedná o jednoznačnou IP adresu příjemce (*unicast*), tak se pomocí ARP zjistí adresa linkového příjemce, IP datagram se zabalí do linkového rámce a odešle do sítě.

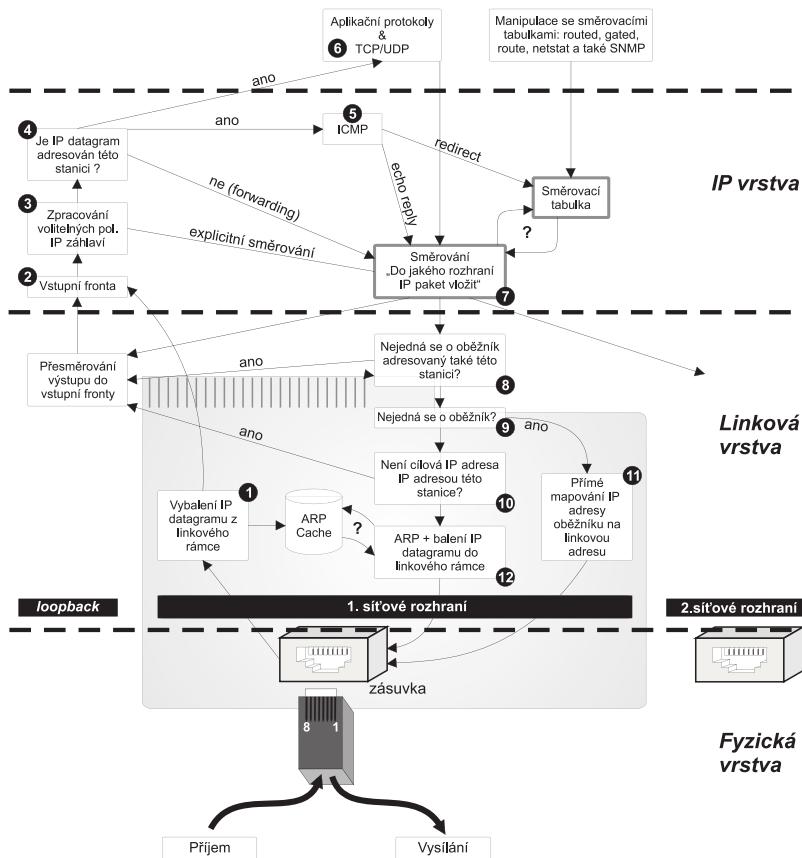


**Poznámka:** Prohlédnete-li si obrázek 7.1 podrobně, pak naleznete i odpověď na otázku: „A proč musí konfigurovat programovou smyčku (loopback)?“ Odpověď je jasná: „Vždy když nějaké rozhraní zjistí, že by se něco mělo předat zpět na vstup, tak se to předá na programovou smyčku, která to zařídí.“

Z obrázku 7.1 je také patrné, že při zpracování vstupů v některých případech operační systém informace automaticky předává na výstup (subsystému Směrování), tj. aplikaci programy do tohoto předávání nezasahují. Jedná se zejména o:

- ◆ explicitní směrování (*source routing*),
- ◆ předávání (*forwarding*),
- ◆ požadavek o ICMP echo (*echo request*),
- ◆ přesměrování (*redirect*).

Operační systémy mají v jádře vždy nějaké parametry, kterými lze takováto automatická zpracování IP datagramů zakázat. Velice častý je např. zákaz explicitního směrování, naopak zpracování požadavku o echo se zakazuje méně často.



Obrázek 7.1: Směrování

## Předávání (*forwarding*) a filtrace (*filtering*)

Předávání umožňuje stanici pracovat jako směrovač. Pokud stanice zjistí, že IP datagram není adresován pro ni, pak se jej pokouší předat dále, tj. odeslat jako odesílá své IP datagramy.

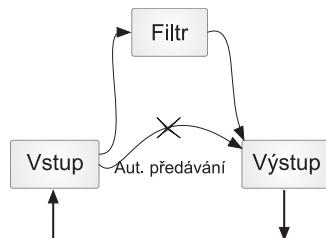
Předávání lze i zakázat – to bývá volbou jádra operačního systému. U starších systémů bylo nutné pro takový zákaz znova sestavit jádro operačního systému. U dnešních systémů je to možné provádět dynamicky. Někdy je však nutné systém po takové změně restartovat.

Zajímavá je situace od Windows 2000 výše (tentokrát nemíme Windows servery). Ve Windows totiž nastavujeme předávání vložením hodnoty 1 do klíče `IpEnableRouter`, který je uložen v registru `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters`.

Zajímavou vlastností mnohých operačních systémů je, že IP datagramy nepředávají mechanicky, ale provádějí filtraci (*filtering* nebo též *screening*), tj. nepředávají všechny pakety, ale jen některé – pro lustrované. Většinou filtrace pracuje tak, že před tím, než je IP datagram předán, tak se celý proces předávání pozastaví a rozhodnutí, zdali IP datagram předat, se ponechá na procesu (službě) běžícím na pozadí.

Předávaný IP datagram se předá filtračnímu procesu (obr. 7.2), který buď předání schválí, nebo zamítne. Filtrační proces se rozhoduje na základě informací v:

1. IP záhlaví, např. není-li adresát nebo příjemce na černé listině,
2. TCP záhlaví, např. podle čísel portu a nastavených příznaků ACK či SYN,
3. aplikačním protokolem, což používají firewally atp.

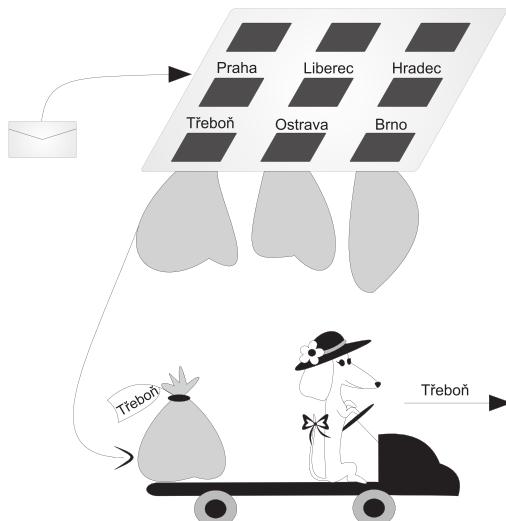


Obrázek 7.2: Filtrace

## Směrování (*routing*)

Směrování IP datagramů je velice podobné třídění dopisů na poště. Na poště mají třídicí stůl s vyřezanými otvory. Pod každým otvorem je přivázán poštovní pytel. Nad otvorem jsou napsány názvy měst, kam je z místní pošty přímé poštovní spojení.

Třídění probíhá tak, že poštovní úředník bere dopis za dopisem. Na každém dopisu si prohlédne adresu. Je-li adresát z Brna, pak dopis vhodí do otvoru Brno. Je-li adresát z Roztok u Prahy, pak dopis vhodí do otvoru Praha (protože do Roztok není přímé poštovní spojení, to je nejbližší Roztokům do Prahy). Až poštovní úředník vytřídí všechny dopisy, pak pytel po pytli odváže z třídicího stolu. Každý pytel zaváže a přiváže k němu visačku, na kterou napiše název města, kam se má pytel odeslat. Poté se pytel naloží...

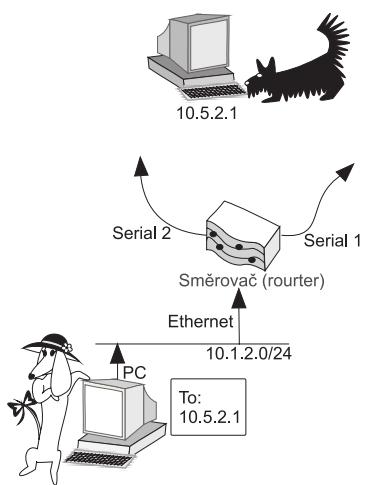


**Obrázek 7.3:** Třídění na poště

Směrovač netřídí dopisy, ale IP datagramy. Tento proces se nazývá směrováním. Směrovač obdrží IP datagram a musí rozhodnout, do kterého svého rozhraní jej má vhodit, kterému svému sousedovi (*next hop*) jej má poslat.

Zjednodušeně řečeno: směrovač je zařízení, které předává IP datagramy z jednoho svého rozhraní do jiného rozhraní. Směrovač umí předat IP datagram i do téhož rozhraní, ze kterého IP datagram přišel. Považuje to však ze výstřednosti, takže o tom odesilatele IP datagramu upozorní ICMP paketem „*redirect*“.

Na obrázku 7.4 směrovač obdržel IP datagram adresovaný stanici 10.5.2.1 a musí rozhodnout, zdali jej vložit do rozhraní Serial1, Serial2 nebo snad zpět do rozhraní Ethernet.



**Obrázek 7.4:** Dilema směrovače: „Do kterého rozhraní IP datagram vložit?“

Směrovač k rozhodování slouží směrovací tabulka (obdoba třídicího stolu na poště). Nás směrovač z obr. 7.4 má následující obsah směrovací tabulky:

Síť	Maska	Next Hop	Síťové rozhraní	Metrika
192.168.1.0	255.255.255.0	192.168.254.5	Seriál 1	4
10.1.2.0	255.255.255.0	Lokální rozhraní	Ethernet	0
10.5.1.0	255.255.255.0	10.10.10.2	Seriál 2	3
10.5.0.0	255.255.0.0	10.5.5.5	Seriál 1	2
...				
0.0.0.0	0.0.0.0	10.10.10.2	Seriál 2	1

Směrovací tabulka má v prvním sloupci IP adresu cílové sítě. Představme si pro jednoduchost, že směrovací tabulka je podle prvního sloupce tříděna sestupně. To nám umožní snadno aplikovat základní pravidlo směrování: **Více specifická adresa cílové sítě má přednost před méně specifickou.** Více specifickou adresou sítě se rozumí adresa, která má v síťové masce více jedniček. V případě, že by se ve směrovací tabulce našly dvě či více cest k cíli, pak se zvolí více specifická cesta. V případě, že se najdou dvě stejně specifické cesty, pak se zvolí cesta s nejnižší metrikou (cenou).

## Zpracování

V případě, že jsou rádky směrovací tabulky tříděny sestupně, pak stačí směrovací tabulku procházet od shora dolů. Na každém rádku se vezme síťová maska, kterou se bit po bitu vynásobí IP adresa příjemce IP datagramu. Výsledek se porovná s prvním sloupcem. Pokud se výsledek nerovná IP adrese sítě v prvním sloupci, pak se přejde na zpracování následujícího rádku. Pokud se výsledek shoduje s IP adresou v prvním sloupci, pak se ještě otestuje následující rádek, zdali ve směrovací tabulce neexistuje k cíli ještě jiná cesta, (pak by vstoupila do hry metrika).

Vraťme se k příkladu z obr. 7.4. Směrovač je postaven před rozhodnutí, kterým svým síťovým rozhraním IP datagram o adrese 10.5.2.1 odeslat. Postupně shora dolů prochází směrovací tabulkou:

1. rádek:

192.168.1.0	255.255.255.0	192.168.254.5	Seriál 1	4
-------------	---------------	---------------	----------	---

Vynásobíme-li bit po bitu cílovou adresu 10.5.2.1 s maskou 255.255.255.0, obdržíme 10.5.2.0, což se nerovná IP adrese sítě v prvním sloupci (ta je 192.168.1.0). Přecházíme na vyhodnocení následujícího rádku.

2. rádek:

10.1.2.0	255.255.255.0	Lokální rozhraní	Ethernet	0
----------	---------------	------------------	----------	---

Vynásobením bit po bitu cílové adresy 10.5.2.1 s maskou 255.255.255.0 obdržíme 10.5.2.0, což se nerovná IP adrese sítě v prvním sloupci (ta je 10.1.2.0). Přecházíme na vyhodnocení následujícího rádku.

3. rádek:

10.5.1.0	255.255.255.0	10.10.10.2	Seriál 2	3
----------	---------------	------------	----------	---

Vynásobením bit po bitu cílové adresy 10.5.2.1 s maskou 255.255.255.0 obdržíme 10.5.2.0, což se nerovná IP adrese sítě v prvním sloupci (ta je 10.5.1.0). Přecházíme na vyhodnocení následujícího řádku.

4. řádek:

10.5.0.0	255.255.0.0	10.5.5.5	Seriál 1	2
----------	-------------	----------	----------	---

Vynásobením bit po bitu cílové adresy 10.5.2.1 s maskou 255.255.0.0 obdržíme 10.5.0.0, což **se rovná** IP adrese sítě v prvním sloupci (ta je 10.5.0.0). Budeme proto náš IP datagram vkládat do rozhraní Serial 1 a předávat jej dalšímu směrovači o IP adrese 10.5.5.5. Pokud by se nejednalo o sériovou linku, ale např. o Ethernet, pak by bylo třeba zjistit linkovou adresu směrovače o IP adrese 10.5.5.5 protokolem ARP.

Poslední řádek obsahující v prvním sloupci 0.0.0.0 s maskou 0.0.0.0 se nazývá **default**. Tímto implicitním směrem jsou pak odesílány všechny IP datagramy, pro které nevyhovoval žádny řádek směrovací tabulky (všimněte si, že vyhovuje každé IP adresy: nula krát nula je nula). Implicitní směr ve směrovací tabulce může a nemusí být – závisí to na správci, jak tabulku naplnil. Implicitní směr používají např. firmy pro cestu do Internetu.

S implicitním směrem se setkáváme i na silnici. Když jedu z Budějovic do Prahy, tak implicitní směr je do Prahy, ale na mnohých křížovatkách je značeno jen odbočení. Je tam šipka do Třeboně či do Bechyně, ale mnohdy chybí přímý směr do Prahy. Implicitně každý ví, že tahle silnice vede do Prahy (tj. *default* je do Prahy), tak to přece není třeba stále na každé mezi opakovat.

## Manipulace se směrovacími tabulkami

Směrovací tabulku je třeba jednotlivými položkami naplnit. Položky jsou pak v tabulce trvale, dokud je někdo nezruší nebo nevypne systém. Pokud je plní směrovací aplikační protokoly, pak je sledována doba jejich života, po které jsou z tabulky vypuštěny.

V příkazech se anglicky často nepoužívá slovo *router*, ale *gateway*. Setkáváme se s tím zejména ve starší literatuře. Ve směrovací tabulce se tím rozumí následující směrovač (*next hop*).

### Výpis obsahu směrovací tabulky

Ve většině operačních systémů se obsah směrovací tabulky vypisuje příkazem:

```
netstat -r
```

Totéž lze zpravidla vypsat i příkazem:

```
route print
```

### Výpis obsahu směrovací tabulky v UNIXu/Linuxu

Položky směrovací tabulky jsou zde příkazem „`netstat -r`“ vypisovány sestupně:

```
# netstat -rn
Kernel IP routing table
```

Destination	Gateway	Genmask	Flags	Iface
192.168.1.0	192.168.254.5	255.255.255.0	UG	ttyS1
10.1.2.0	*	255.255.255.0	U	eth0
10.5.1.0	10.10.10.2	255.255.255.0	UG	ttyS2
10.5.0.0	10.5.5.5	255.255.0.0	UG	ttyS1
127.0.0.0	*	255.0.0.0	U	lo
0.0.0.0	10.10.10.2	0.0.0.0	UG	ttyS2

Sloupec Iface specifikuje síťové rozhraní, do kterého se budou IP datagramy předávat. Zajímavý je ale sloupec s příznaky (*Flags*), které mohou nabývat mj. následujících hodnot:

- ◆ U (*up*). Směr je dostupný.
- ◆ G (*gateway*). Příznak G určuje, že cesta k cílové síti vede přes směrovač, tj. next hop je směrovač. Linková vrstva bude hledat linkovou adresu uvedeného směrovače, nikoliv přímo adresáta (ten není přímo dostupný).
- ◆ H (*host*). Příznak H určuje, že se jedná o adresu počítače, nikoliv o adresu sítě, tj. maska je 255.255.255.255.
- ◆ D (*dynamically*). Položka byla vytvořena dynamicky směrovacím procesem (démonem) nebo na základě ICMP zprávy redirect.
- ◆ M (*modified*). Položka byla modifikována směrovacím procesem (démonem) nebo na základě ICMP zprávy redirect.
- ◆ ! (*reject*). Zakázaný směr.

## Výpis v operačních systémech Microsoft

Příkaz „netstat -r“ zde vypisuje obsah směrovací tabulky setříděný vzestupně, takže pokud chcete vyhodnocovat tabulku, jak jsem popsali, pak ji musíte procházet zdola nahoru. Trochu nezvyklé je, že síťová rozhraní (*interface*) se jmenují svou IP adresou, ale po chvíli si na to zvyknete. Výpis směrovacích tabulek v MS Vista se skládá až ze tří částí:

1. Seznam síťových rozhraní. Každé rozhraní má své číslo, které je pak možné použít v některých příkazech operačního systému.
2. Směrovací tabulku pro IP protokol verze 4.
3. V případě instalovaného IPv6 vypíše i směrovací tabulku pro IP protokol verze 6.

```
C:\ > route print
```

```
=====
Seznam rozhraní
11 ...00 19 d2 29 51 53 ..... Intel(R) PRO/Wireless 3945ABG Network Connection
 8 ...00 16 36 fe f3 b6 ..... Intel(R) PRO/1000 PM Network Connection
 1 ..... Software Loopback Interface 1
20 ...00 00 00 00 00 00 e0 Microsoft ISATAP Adapter
12 ...00 00 00 00 00 00 e0 isatap.{A4A09537-5ED6-4A75-8CB7-74401D56DEA1}
10 ...02 00 54 55 4e 01 ..... Teredo Tunneling Pseudo-Interface
=====
```

## IPv4 Směrovací tabulka

Aktivní směrování:					
Cíl v síti	Síťová maska	Brána	Rozhraní	Metrika	
0.0.0.0	0.0.0.0	10.0.0.138	10.0.0.2	25	
10.0.0.0	255.255.255.0	Propojené	10.0.0.2	281	
10.0.0.2	255.255.255.255	Propojené	10.0.0.2	281	
10.0.0.255	255.255.255.255	Propojené	10.0.0.2	281	
127.0.0.0	255.0.0.0	Propojené	127.0.0.1	306	
127.0.0.1	255.255.255.255	Propojené	127.0.0.1	306	
127.255.255.255	255.255.255.255	Propojené	127.0.0.1	306	
224.0.0.0	240.0.0.0	Propojené	127.0.0.1	306	
224.0.0.0	240.0.0.0	Propojené	10.0.0.2	281	
255.255.255.255	255.255.255.255	Propojené	127.0.0.1	306	
255.255.255.255	255.255.255.255	Propojené	10.0.0.2	281	

## Trvalé trasy:

Síťová adresa	Maska	Adresa brány	Metrika
172.17.0.0	255.255.0.0	10.0.0.138	1

## IPv6 Směrovací tabulka

Aktivní směrování:			
Rozhraní	Metrika	Cíl v síti	Brána
1	306	::1/128	Propojené
11	281	fe80::/64	Propojené
12	286	fe80::5efe:10.0.0.2/128	Propojené
11	281	fe80::594d:e8e9:987e:3c80/128	Propojené
1	306	ff00::/8	Propojené
11	281	ff00::/8	Propojené

## Trvalé trasy:

Žádné

Síť 224.0.0.0 s maskou 224.0.0.0 označuje všechny adresné oběžníky (včetně rezervy IP adres, tj. IP -adresy tříd D a E).

**Výpis obsahu směrovací tabulky na směrovačích CISCO**

Na směrovačích CISCO je možné vypsat obsah směrovací tabulky i s neprivilegovaným uživatelem:

```
Router>sho ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
      D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
      N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
      E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
      i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
      U - per-user static route, o - ODR
```

```

Gateway of last resort is 195.47.37.192 to network 0.0.0.0

      195.47.37.0/27 is subnetted, 1 subnets
C        195.47.37.192 is directly connected, Ethernet0
          10.0.0.0/24 is subnetted, 3 subnets
S          10.4.6.0 [1/0] via 195.47.37.211
S          10.4.4.0 [1/0] via 195.47.37.210
S          10.4.5.0 [1/0] via 195.47.37.210
S*        0.0.0.0/0 [1/0] via 195.47.37.192

```

Dolní část výpisu obsahuje jednotlivé položky směrovací tabulky. Výpis položek směrovací tabulky je rozdelen do sekcí podle jednotlivých sítí. Např. nadpis „10.0.0.0/24 is subnetted, 3 subnets“ nám uvozuje desítkové síť a navíc nám sděluje, že desítková síť je rozdělena na tři subsítě. Velice důležitou informací je první sloupec ve výpisu položky směrovací tabulky. Ten sděluje, jakým způsobem se položka do směrovací tabulky dostala. V našem výpisu máme pouze dva případy:

- ◆ C – položka se do směrovací tabulky dostala z konfigurace rozhraní směrovače.
- ◆ S – položka byla staticky konfigurována atd. Všechny možnosti jsou mj. uvedeny v první části výpisu.

## Naplnění směrovací tabulky a rušení položek

Směrovací tabulka se plní:

- ◆ Při konfiguraci síťového rozhraní, kdy říkáme, jakou má síťové rozhraní adresu a masku. V operačním systému UNIX se jedná o příkaz *ifconfig*.
- ◆ Staticky (ručně) příkazem *route*.
- ◆ Dynamicky z ICMP zpráv *redirect*.
- ◆ Dynamicky směrovacími protokoly.

Staticky se směrovací tabulka plní pomocí příkazu route. Ve Windows má příkaz route následující syntaxi:

```
ROUTE [-f/-p] [command [destination] [MASK netmask] [gateway] [METRIC metric]]
```

-f	Vymaže nejprve obsah směrovací tabulky.
-p	U příkazu ADD zajistí, aby takto přidaná položka zůstala ve směrovací tabulce i po restartu PC, tj. stala se trvalou položkou. U příkazu PRINT způsobí, že se vypíší trvalé položky.
command	Určuje příkaz pro manipulaci se směrovací tabulkou, nabývá následujících hodnot:
PRINT	Vypíš obsah směrovací tabulky.
ADD	Přidej položku do směrovací tabulky.
DELETE	Zruš položku ve směrovací tabulce.
CHANGE	Změň položku.
destination	Specifikuje cílovou síť.
netmask	Specifikuje síťovou masku.
gateway	Specifikuje next hop.
METRIC	Specifikuje metriku.

### Příklad Windows

```
route -p add 10.0.0.32 mask 255.255.255.240 192.168.1.2
```

### Příklad Unix

V operačním systému UNIX je příkaz route podstatně bohatší. Nejsou zde trvalé položky směrovací tabulky – po restartu se statické položky do směrovací tabulky vždy plní příkazem route (byť automaticky nějakou procedurou).

V operačním systému UNIX je i jiný repertoár příkazů pro manipulaci se směrovací tabulkou. Nebývá zde příkaz PRINT, ale naopak bývá příkaz flush (vymazání směrovací tabulky) a někdy též příkaz monitor, který způsobí živé vypisování změn ve směrovací tabulce na standardní výstup (ukončuje se  $\text{^C}$ ).

```
route add -net 10.0.0.32/28 192.168.1.2
```

### Příklad CISCO

Ve směrovačích CISCO staticky přidáváme položky do směrovací tabulky tak, že položku přidáme do konfigurace směrovače. Např.:

```
ip route 10.1.1.0 255.255.255.0 192.169.1.1
```

vloží položku, která bude síť 10.1.1.0/24 směrovat na směrovač (gateway) 192.169.1.1. Obdobně přidáme i default směrující vše ostatní na směrovač 192.168.1.2:

```
ip route 0.0.0.0 0.0.0.0 192.168.1.2
```

## Směrovací protokoly

Směrovací protokoly slouží směrovačům, aby si vzájemnou komunikaci mezi sebou automaticky naplnily a udržovaly směrovací tabulky.

Máme dvojí, na sobě nezávislé, dělení směrovacích protokolů:

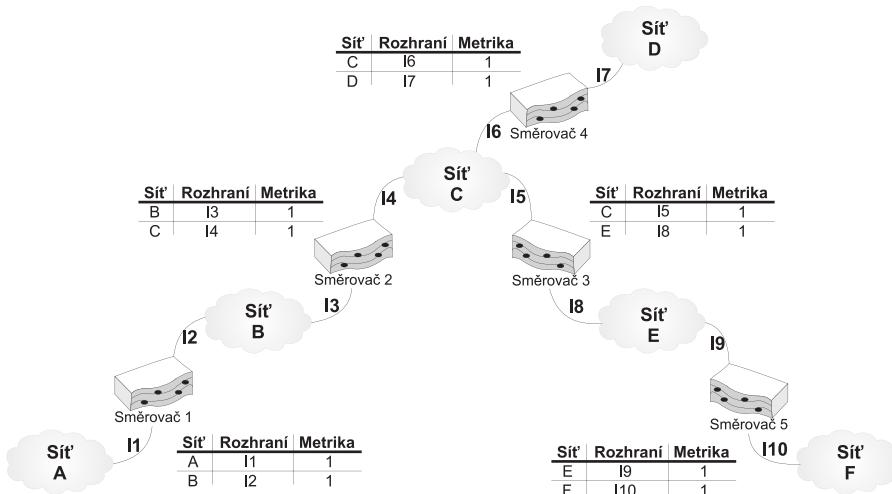
- ◆ Na *Link State Protocols (LSP)* a na *Routing Vector Protocols (RVP)* z hlediska použitého algoritmu.
- ◆ Na *Interior Gateway Protocols (IGP)* a *Exterior Gateway Protocols (EGP)* z hlediska organizačního. IGP jsou určeny pro řešení směrování v rámci autonomního systému (AS). EGP pak pro výměnu směrovacích informací mezi AS.

Většina směrovacích protokolů je určena buď pro IPv4, nebo pro IPv6. Existují však i směrovací protokoly, které umí současně podporovat více síťových protokolů. Jelikož mají protokoly IPv4 a IPv6 každý jinou délku síťové adresy, tak zpravidla každý z nich má i samostatné směrovací tabulky. Na druhou stranu princip dvojice IP adresa + síťová maska je jak v IPv4, tak i v IPv6 stejný. Takže základní principy zůstávají stejné pro oba dva protokoly. Pokud tedy pochopíte princip směrování pro IPv4, pak získáte i základ pro pochopení principu směrování pro IPv6.

## Princip Routing Vector Protocols (RVP)

Princip RVP je jednoduchý. Každý směrovač pravidelně odesílá oběžníkem obsah svých směrovacích tabulek. Sousední směrovače si tuto informaci vzájemně přečtou a promítou ji do svých směrovacích tabulek.

Nejlépe se to demonstruje na příkladu. Představme si, že právě byly zapnuty všechny směrovače na obr. 7.5. Po zapnutí si směrovače naplní své směrovací tabulky ze své startovací konfigurace.



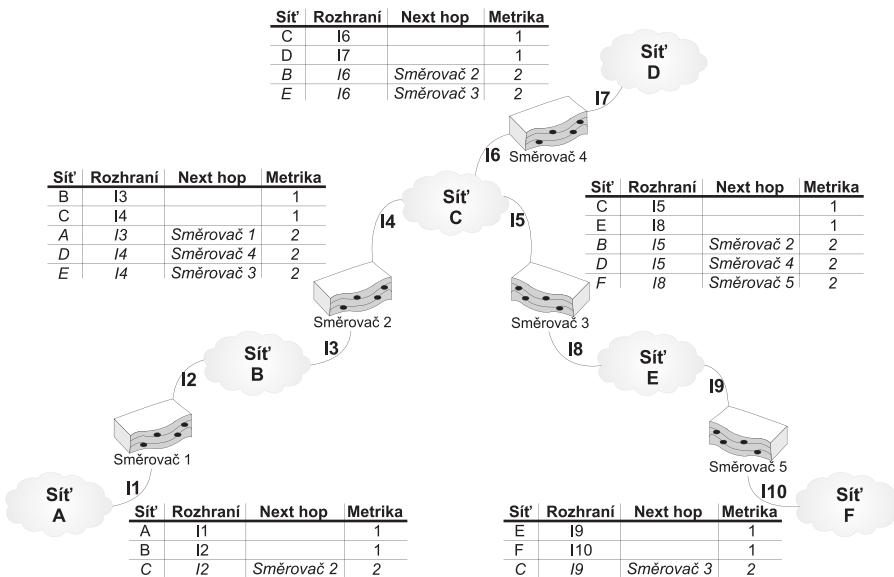
Obrázek 7.5: První krok plnění směrovacích tabulek (zapnutí směrovačů a načtení jejich konfigurace)

Ve druhém kroku (obr. 7.6) si směrovače vzájemně vymění své směrovací tabulky. Sousední směrovací tabulku pak zpracovávají v následujících krocích:

- ◆ Vyhledávají cesty, které buď ve vlastní směrovací tabulce nemají, nebo je mají s menší metrikou.
- ◆ Takto vyhledanou cestu si uloží do své směrovací tabulky, ale s vyšší metrikou. Např. protokoly RIP a RIP2 přiřazují k metrice jedničku.
- ◆ V protokolech RVP je vždy stanovena nějaká horní hranice pro metriku. Pokud by položka dosáhla této horní hranice, pak se cesta považuje za nedosažitelnou a položka se do směrovací tabulky nezapiše. Pro mnohé je překvapivé, že tato horní mez není příliš velká. Např. u protokolů RIP a RIP2 bývá 16. Důvodem je skutečně, že při vyšší horní hranici se v praxi ukázalo, že směrovací tabulky začnou oscilovat.

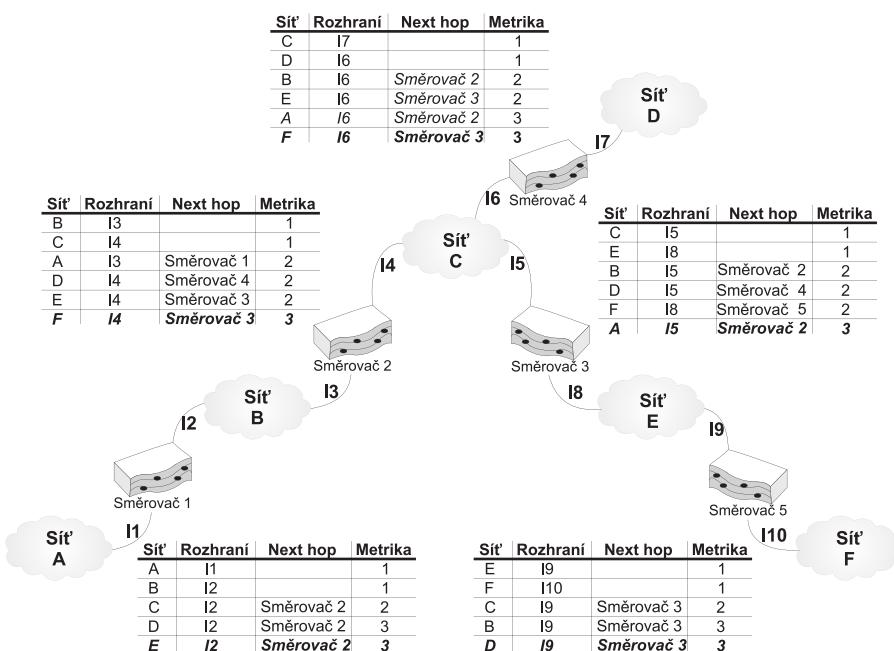
Vezměme si např. směrovací tabulku směrovače 2 z obr. 7.6. V ní jsou:

- ◆ Sítě B a C, které se tam dostaly z konfigurace.
- ◆ Sít' A, kterou obdržel ze směrovací tabulky směrovače 1 (zvýšil metriku na 2).
- ◆ Sít' D, kterou obdržel ze směrovací tabulky směrovače 4 (zvýšil metriku na 2).
- ◆ Sít' E, kterou obdržel ze směrovací tabulky směrovače 3 (zvýšil metriku na 2).



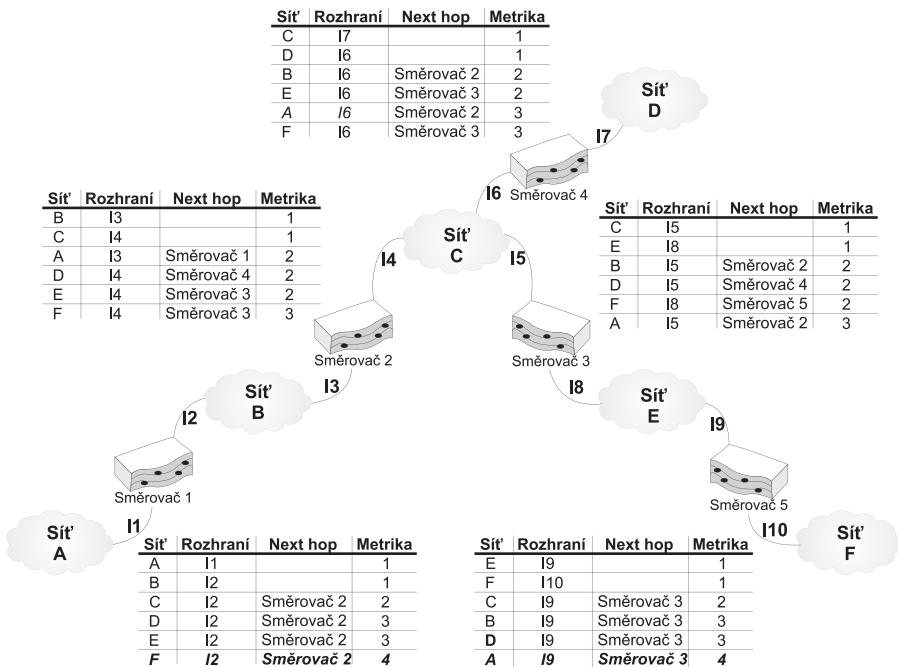
Obrázek 7.6: Druhý krok: doplnění směrovacích tabulek informacemi od sousedů

Ve třetím kroku (obr. 7.7) si směrovače vyměňují směrovací tabulky, které obsahují informace, které získaly ve druhém kroku. Na obr. 7.8 jsou to ty, které mají metriku 3.



Obrázek 7.7: Třetí krok

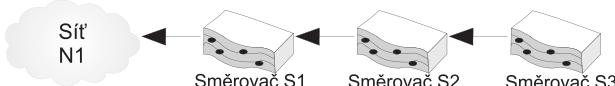
Čtvrtý krok je opět stejná písnička (obr. 7.8). Po čtvrtém kroku ale už všechny směrovače mají informace o všech sítích. Jelikož topologie sítě se může měnit, směrovače takto dynamicky získané informace ve svých směrovacích tabulkách neudržují trvale, ale zpravidla jen 2–5 minut v závislosti na konkrétním směrovacím protokolu.



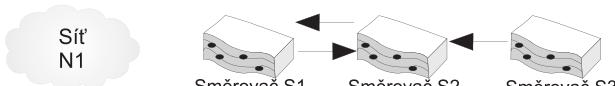
Obrázek 7.8: Čtvrtý krok

Uvedený algoritmus se někdy také označuje jako **Bellman-Fordův algoritmus**. Tento algoritmus má bohužel i některé ne příliš pěkné vlastnosti. Jednou z nich je tzv. „*pomalá konvergencie*“: Směrovač S1 (obr. 7.9) je přímo připojen do sítě N1 s metrikou 1. Periodicky tuto informaci šíří svým sousedům. Nejprve se dozví tuto informaci směrovač S2 a vloží si ji do své směrovací tabulky s metrikou 2. Posléze si tuto informaci vloží do svých směrovacích tabulek i směrovač S3 s metrikou 3.

Směrovače S1, S2 a S3 mají spojení do sítě N1:



Spojení S1 do sítě N1 vypadne:



Obrázek 7.9: Pomalá konvergencie

Nyní vypadne spojení směrovače S1 do sítě N1. Směrovač S1 si proto ve směrovací tabulce zruší cestu do sítě N1. Jenomže směrovač S2 má ve svých směrovacích tabulkách tuto cestu s metrikou 2 a v dalším kroku je nabídne směrovači S1. Směrovač S1 v dobré víře, že směrovač S2 zná cestu do sítě N1, si uloží cestu do sítě N1 přes směrovač S2, ale s metrikou 3. Nyní směrovači S2 expiruje tato položka. Jenže směrovač S1 nabízí cestu do N1 s metrikou 3, tak si ji S2 uloží s metrikou 4. Atd. Tento jev nazýváme oscilací, kterou řešíme dvěma postupy:

- ◆ Již zmíněnou horní mezí pro metriku, kdy např. protokoly RIP a RIP2 už cesty s metrikou větší než 15 považují za nedostupné.
- ◆ Technikou „odříznutí horizontu“, kdy se směrovače snaží předcházet zpětnému použití informací, které původně do sítě samy zaslaly.

Další slabinou těchto algoritmů je, že nepodporují rozdělování výkonu (*load balancing*) mezi paralelní linky.

Závěr k RVP je asi takový, že jsou určeny pro méně rozsáhlé WAN. Avšak na menších sítích je vždy otázkou, jestli je opravdu třeba dynamicky plnit směrovací tabulky, jestli nevystačíme se statickými položkami směrovacích tabulek (tj. s ručním naplněním směrovacích tabulek).

## RIP, RIP2 a RIPng

Protokoly RIP, RIP2 a RIPng jsou příkladem protokolů RVP. Pro mnohé bude překvapením, že se jedná o aplikační protokoly – své pakety totiž balí do UDP datagramů. Pakety pak obsahují celé nebo části směrovacích tabulek. V Linuxu se tyto protokoly aktivují pomocí směrovacího manažera `zebra`. Na serverech Windows je lze aktivovat pomocí *Routing and Remote Access*.

Na počátku byl (dnes již historický) protokol RIP (*Routing Information Protocol*). Ten měl však nevýhodu v tom, že nešířil s cílovými sítěmi i jejich masku – předpokládal standardní síťové masky. Proto se do dnešní doby již nehodí. Protokol RIP2 odstranil tento nedostatek. Mezi RIP a RIP2 je tak rozdíl v tom, že RIP šířil své pakety pomocí všeobecného oběžníku (*broadcast*), kdežto RIP2 používá adresný oběžník (*multicast*) o IP adresu 224.0.0.9.

Takto jsou pakety šířeny každých 30 vteřin. Ve směrovacích tabulkách pak směry po 180 vteřinách expirují, tj. pokud se do 180 vteřin neobjeví znova oběžník s konkrétním směrem, pak je tento směr vymazán ze směrovací tabulky. Metrika 16 je pokládána za nedostupnou síť.

RIPng je modifikací pro IPv6.

Protokoly RIP a RIP2 využívají dobře známý port 520/udp a protokol RIPng využívá dobře známý port 521/udp.

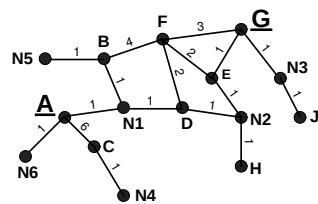
## Princip Link State Protocols (LSP)

Princip LSP je naprostě odlišný. K pochopení si nejprve musíme udělat krátkou exkurzi do teorie grafů.

Grafem rozumíme množinu vrcholů spojených hranami. Dva vrcholy nazýváme sousedními, existuje-li mezi nimi hrana. Na obr. 7.11 jsou vrcholy A a C sousední, ale vrcholy A a B nikoliv. Pro každý vrchol je možné jednoznačně určit množinu všech jeho sousedů. Např. vrcholu N3 na obr. 7.10 přísluší množina sousedů, která obsahuje uzly G a J.

Dále jako cestu v grafu chápeme sekvenci vrcholů, kde následující uzel je vždy sousedem předchozího uzlu.

Každá hrana je ohodnocena číslem, které nazýváme metrika („délka hrany“). Např. hrana z uzlu A do uzlu C má metriku 6. Můžeme si představit, že délka cesty z uzlu A do uzlu C je 6 jednotek. Délkou celé cesty pak myslíme součet délek všech hran, kterými cesta prochází.



Obrázek 7.10: Graf

Nyní si představme, že směrovač A potřebuje odeslat IP datagram do uzlu G (obr. 7.10). Z hlediska grafu hledáme nejkratší cestu z uzlu A do uzlu G. Nejkratší cesta v grafu se řeší pomocí algoritmu, který nese jméno holandského informatika jménem Edsger Wybe Dijkstrou (1930-2002). Tento algoritmus najde z daného uzlu nejkratší cestu ke všem uzlům grafu. Algoritmus však může být zastaven, je-li nalezena nejkratší cesta do požadovaného uzlu grafu.

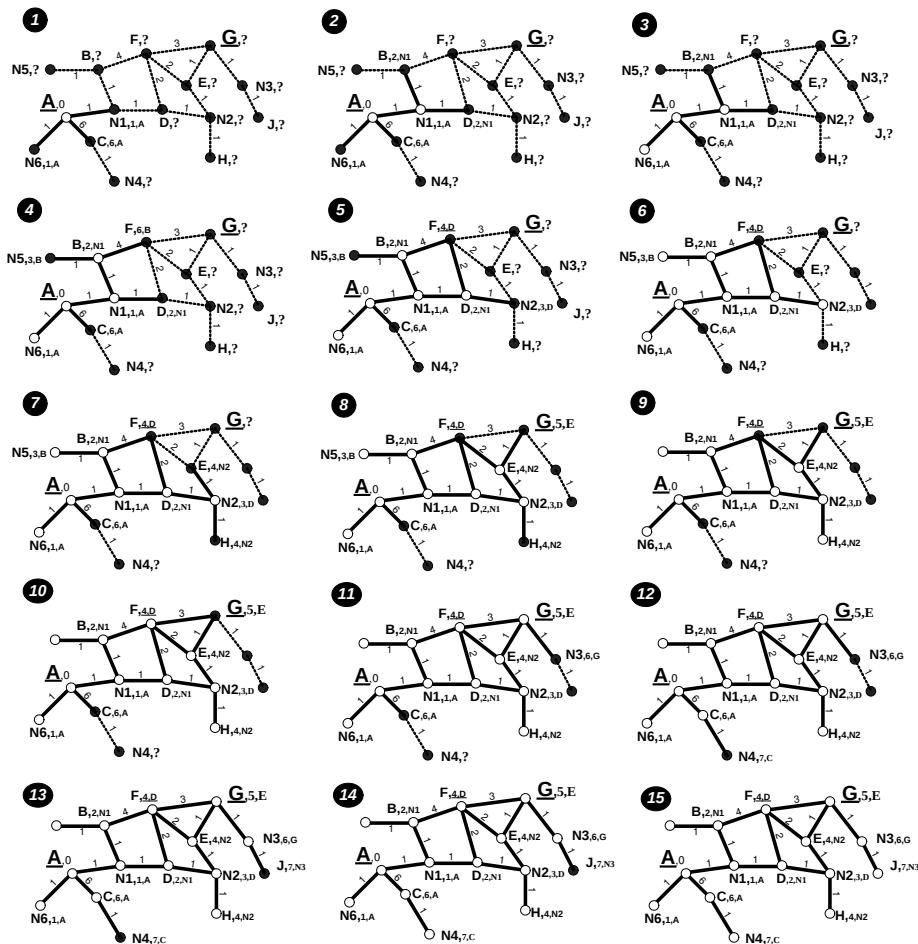
### Algoritmus Nejkratší cesty

1. Vytvoříme seznam vzdáleností, seznam předchozích vrcholů, seznam navštívených vrcholů a aktuální vrchol.
2. Všechny hodnoty v seznamu vzdáleností nastavíme na nekonečno, pouze vzdálenost do výchozího (startovacího) uzlu nastavíme na 0.
3. Všechny hodnoty v seznamu navštívených uzlů nastavíme na FALSE.
4. Všechny hodnoty v seznamu předchozích vrcholů nastavíme na -1.
5. Výchozí vrchol nastavíme jako aktuální vrchol.
6. Aktuální vrchol označíme jako navštívený vrchol.
7. Pro všechny uzly, které mohou být dosaženy z aktuálního vrcholu, nastavíme hodnoty v seznamu vzdáleností a v seznamu předchozích vrcholů (pokud je kratší cesta přes vrchol, ve kterém jsme, tak se aktualizuje, jinak se hodnota ponechá).
8. Jako aktuální vrchol nastavíme ještě nenavštívený vrchol, který má nejmenší vzdálenost od startovacího vrcholu.
9. Opakujme body 6 až 8, dokud nejsou všechny vrcholy označeny jako navštívené.



**Tip:** Dijkstrův algoritmus můžete použít i pro plánování nejkratší cesty na dovolené!

Nejlepší je praktická demonstrace: budeme hledat nejkratší cestu z vrcholu A do všech vrcholů grafu z obr. 7.10. Jednotlivé kroky hledání nejkratší cesty jsou rozfázovány na obr. 7.11. (sledujte též obr. 7.12, kde postupně vzniká strom nejkratší cesty):



**Obrázek 7.11:** Jednotlivé fáze hledání nejkratších cest z výchozího vrcholu A do všech ostatních vrcholů grafu (bílé kolečko = navštívený vrchol)

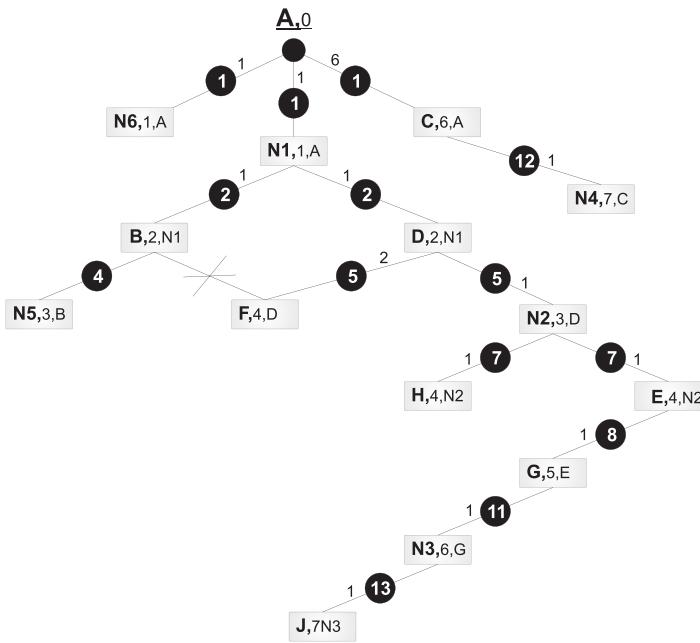
1. První krok obsahuje dvě fáze:

- ◆ Inicializační fáze:
  - Vytvoří se a inicializují se seznamy.
  - Vzdálenost z/do vrcholu A se nastaví na 0.
  - Vrchol A je zvolen jako výchozí, aktuální a navštívený.
- ◆ První krok hledání nejkratší cesty, kdy hledáme všechny uzly, které mohou být dosaženy z vrcholu A. Jedná se o uzly:
  - N6 se vzdáleností 1 od předchozího uzlu A (délka cesty od uzlu A je 1).
  - C se vzdáleností 6 od předchozího uzlu A (délka cesty od uzlu A je 6).
  - N1 se vzdáleností 1 od předchozího uzlu A (délka cesty od uzlu A je 1).
  - Jako aktuální vrchol označíme vrchol N1 (délka cesty od uzlu A je 1).

2. Ve druhém kroku:
  - ◆ Vrchol N1 označíme jako navštívený.
  - ◆ Hledáme všechny uzly, které mohou být dosaženy z vrcholu N1. Jedná se o uzly:
    - D s délkou hrany 1 od předchozího uzlu N1 (délka cesty od uzlu A je 2).
    - B s délkou hrany 1 od předchozího uzlu N1 (délka cesty od uzlu A je 2).
  - ◆ Jako aktuální vrchol označíme vrchol N6 (délka cesty od uzlu A je 1).
3. Třetí krok nám pouze označí vrchol N6 jako navštívený a jako aktuální nastaví např. vrchol B (délka cesty od uzlu A je 2).
4. Čtvrtý krok:
  - ◆ Vrchol B označíme jako navštívený.
  - ◆ Hledáme všechny uzly, které mohou být dosaženy z vrcholu B. Jedná se o uzly:
    - N5 s délkou hrany 1 od předchozího uzlu B (délka cesty od uzlu A je 3).
    - F s délkou hrany 4 od předchozího uzlu B (délka cesty od uzlu A je 6).
  - ◆ Jako aktuální vrchol označíme D (délka cesty od uzlu A je 2).
5. Pátý krok:
  - ◆ Vrchol D označíme jako navštívený.
  - ◆ Hledáme všechny uzly, které mohou být dosaženy z vrcholu D. Jedná se o uzly:
    - F s délkou hrany 2 od předchozího uzlu D (délka cesty od uzlu A je 4) – opravíme hodnotu získanou ve 4. kroku.
    - N2 s délkou hrany 4 od předchozího uzlu D (délka cesty od uzlu A je 6).
  - ◆ Jako aktuální vrchol označíme N5 (délka cesty od uzlu A je 3).
6. Šestý krok nám pouze označí vrchol N5 jako navštívený a jako aktuální nastaví např. vrchol N2 (délka cesty od uzlu A je 3).
7. Sedmý krok :
  - ◆ Označí vrchol N2 jako navštívený.
  - ◆ Hledáme všechny uzly, které mohou být dosaženy z vrcholu N2. Jedná se o uzly:
    - H s délkou hrany 1 od předchozího uzlu N2 (délka cesty od uzlu A je 4).
    - E s délkou hrany 1 od předchozího uzlu N2 (délka cesty od uzlu A je 4).
  - ◆ Jako aktuální vrchol označíme E (délka cesty od uzlu A je 3).
8. Osmý krok:
  - ◆ Označí vrchol E jako navštívený.
  - ◆ Hledáme všechny uzly, které mohou být dosaženy z vrcholu E. Jedná se o uzly:
    - F s délkou hrany 2, ale s horší cestou od vrcholu A, takže nic neopravujeme.
    - G s délkou hrany 1 od předchozího uzlu E (délka cesty od uzlu A je 5).
  - ◆ Jako aktuální vrchol označíme E (délka cesty od uzlu A je 3).
9. Jako navštívený vrchol označí H.
10. Jako navštívený vrchol označí F.
11. Jako navštívený vrchol označí G a dosaženo bude vrcholu N3.
12. Jako navštívený vrchol označí C a dosaženo bude vrcholu N4.
13. Jako navštívený vrchol označí N3 a dosaženo bude vrcholu J.

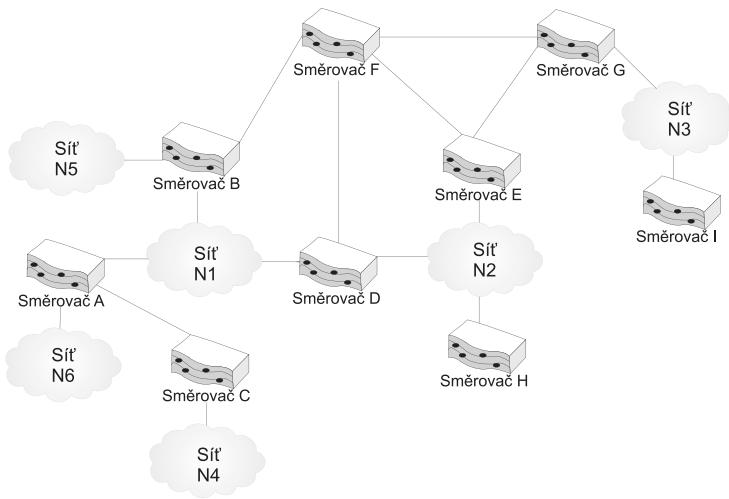
14. Jako navštívený vrchol označí N4.

15. Jako navštívený vrchol označí J.



**Obrázek 7.12:** Strom nejkratší cesty z vrcholu A do všech ostatních vrcholů grafu

Dosud jsme uvažovali jen obecný graf z obr. 7.10. Nyní si pod ním představíme rozlehlou síť z obr. 7.13.



**Obrázek 7.13:** Topologie sítě

Na základě stromu nejkratší cesty pak již snadno vytvoříme směrovací tabulkou směrovače A:

Sít	Rozhraní	Next hop	Metrika
N1	I3	lokální	1
N2	I3	D	3
N3	I3	D	6
N4	I2	C	7
N5	I3	B	3
N6	I1	1	lokální

Obdobně si všechny směrovače naší sítě vytvoří své směrovací tabulky. Co k tomu potřebují? Nepotřebují nic víc, ani nic méně, než znát topologii sítě, tj. znát seznam všech vrcholů a všech hran (včetně metrik) našeho grafu.

Jak směrovač získá kompletní topologii sítě? Topologii získá ve dvou krocích:

1. V prvním kroku každý směrovač odesílá dotazy všem svým sousedům, jsou-li připojeny („pingne“ na sousedy). Jestliže směrovač v zadáném časovém intervalu obdrží alespoň k z n odpověď od souseda ( $1 \leq k \leq n$ ), pak považuje souseda za připojeného (*is up*). Cena (metrika) spoje k sousedovi může být buď nastavena ručně, nebo spočtena (např. z podílu k/n, z doby odezvy souseda pod.).
2. Nyní již každý směrovač ví o svých připojených sousedech, a tak v druhém kroku každý směrovač odešle informaci o svých sousedech ostatním směrovačům – směrovače „zaplavují“ síť těmito informacemi.

Všechny směrovače nyní znají všechny vrcholy a hrany grafu, takže každý z nich si může spustit algoritmus nejkratší cesty a vytvořit své směrovací tabulky.

Jednotlivé implementace směrovacích protokolů LSP mají různé modifikace uvedeného algoritmu. Zmiňme se o dvou:

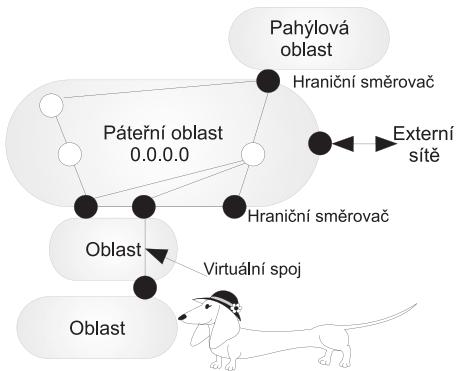
1. Není dobré zaplavovat směrovacími informacemi příliš velké sítě. Proto síť zpravidla rozdělujeme na oblasti (směrovací domény). Směrování pak řešíme v rámci těchto oblastí. Směrovače pak vyměňují směrovací informace mezi oblastmi.
2. Jestliže je na jedné společné síti více směrovačů (např. na LAN), pak se vzájemně dohodnou na jednom z nich (*designated router*), který řeší směrování na této síti jménem ostatních.

LSP-protokoly jsou podstatně stabilnější než protokoly RVP. A hlavně jsou použitelné i na velkých rozlehlych sítích. Nevýhodou je, že je nestačí na směrovacích pouze aktivovat, ale musíme mít experta, který je nakonfiguruje. Jinak je možné, že nám síť nebude chodit tak, jak očekáváme.

## OSPF

Asi nejrozšířenějším příkladem protokolu LSP je protokol OSPF (*Open Shortest Path First*). OSPF implementuje řadu nových vlastností: vzájemnou autentizaci komunikujících směrovačů, load balancing na paralelních linkách o stejné metrice, podporu QoS, redistribuci směrovacích informací získaných mimo AS atd. Existuje i varianta pro protokol IPv6.

OSPF vyžaduje, aby síť byla zvláštním způsobem rozdělena do oblastí. Jádrem sítě je tzv. páteřní oblast, jejíž směrovače mají kompletní směrovací informaci. Ostatní oblasti musí být buď přímo, nebo virtuálně připojeny k páteřní oblasti.



**Obrázek 7.14:** Topologie sítě pro OSPF

Zvláštním případem oblasti je pahýlová oblast. Do této oblasti se nešíří směrovací informace ostatních sítí, protože cesta z této oblasti do ostatních sítí může být snadno udělána pomocí položky *default* směřující do páteřní oblasti. Aby to ale správně chodilo, tak pahýlová oblast může být k páteřní oblasti připojena jen jednou linkou v jednom bodě.

OSPF nepodporuje příčné spoje mezi oblastmi. Výjimkou je spojení oblasti skrze jinou tzv. tranzitní oblast virtuálním spojem.

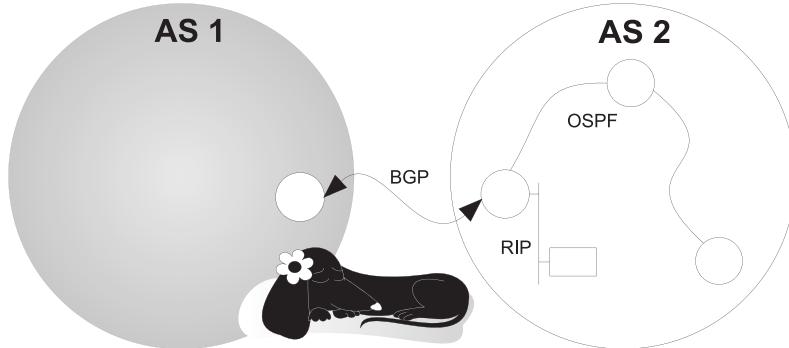
Každá oblast OSPF má svou jedinečnou čtyřbajtovou identifikaci, která se napohled podobá IP adresě (nezaměňovat!). Páteřní oblast má identifikaci 0.0.0.0. Zajímavé mj. jsou hraniční směrovače, tj. směrovače, které mají síťová rozhraní do dvou oblastí. Pak vlastně pro každou oblast jedou zvlášť algoritmus nejkratší cesty.

## Redistribuce

Na obrázku 7.15 je otazníkem označen směrovač, který si vyměňuje současně směrovací informace protokoly BGP, OSPF, RIP a k tomu má možná ve směrovací tabulce několik statických položek.

Otzák je, zda se mají položky směrovací tabulky získané jedním směrovacím protokolem propagovat do ostatních směrovacích protokolů, tj. má-li se provést redistribuce. Redistribucí tak rozumíme přenos směrovacích informací získaných jedním směrovacím protokolem jinému směrovacímu protokolu.

Položka ve směrovací tabulce musí v sobě nést tedy také informaci, jakým protokolem byla vytvořena (statická a konfigurační položky tvoří samostatnou skupinu). To je ovšem pohled administrátora. Ve skutečnosti, pokud na jednom směrovači běží více směrovacích protokolů, tak každý má svou část směrovací tabulky, tj. jako bychom měli více samostatných směrovacích tabulek. Redistribuce pak je přenos informací mezi nimi.



Obrázek 7.15: Redistribuce

## Domácí cvičení

Na svém počítači/serveru:

- ◆ Vypište obsah směrovacích tabulek.
- ◆ Do směrovacích tabulek vložte cestu do sítě 169.254.0.0/8 přes váš nejbližší směrovač.
- ◆ Zrušte cestu do sítě 169.254.0.0/8 vloženou v předchozím bodu.



**Upozornění:** Neodborné vložení/zrušení položky ve směrovacích tabulkách je jednou z nejzá- ludnějších chyb! Po každé manipulaci se směrovacími tabulkami se nezapomeňte přesvědčit, že máte spojení s nějakým vzdáleným serverem.



## Kapitola 8

# IPv6

IP protokol verze 6 se často označuje jako IP protokol nové generace (*IP Next Generation*) – odtud je i někdy používaná zkratka IPng. My budeme dávat přednost zkratce IPv6.

IP protokol verze 4 byl poprvé specifikován v roce 1980 v RFC-760 (dodnes aktuálním standardem IPv4 je RFC-791). Po dalších 14 letech (v roce 1995) vyšel standard IP protokolu verze 6 (dnes aktuálním standardem IPv6 je RFC-2460). Od první specifikace IPv6 uběhlo již 13 let a nutno říci, že je téměř všemi systémy podporován, ale dosud není příliš používán.

IPv6 přináší nejen zvětšení IP adresy ze čtyř na šestnáct bajtů, ale i filozoficky zcela nový pohled na stavbu IP datagramu. V záhlaví IP datagramu chybí kontrolní součet záhlaví a mnohá další málo využívaná pole jsou přesunuta ze základního záhlaví do nepovinných dalších hlaviček.

IP datagram verze 6 se skládá ze čtyřiceti bajtů dlouhého základního záhlaví následovaného rozšířenimi. Čtyřicet bajtů základního záhlaví se může zdát hodně, ale je třeba si uvědomit, že jen IP adresy odesilatele a příjemce zaberou 32 bajtů.

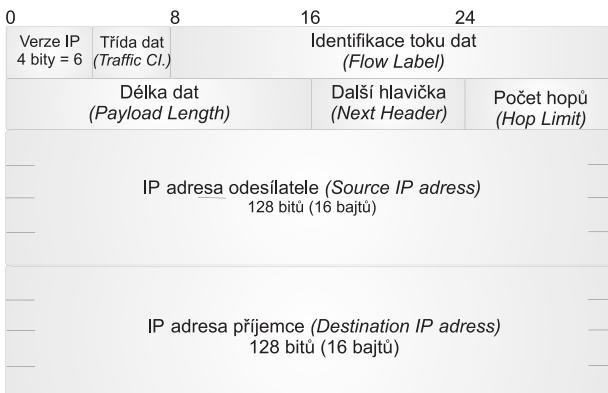
Struktura základního záhlaví je zobrazena na obr. 8.1. Nyní se zastavíme u významu jednotlivých polí základního záhlaví.

Pole **verze IP** obsahuje hodnotu 6 pro IP protokol verze 6 (IP verze 4 zde má hodnotu 4).

Pole **třída dat** se skládá ze čtyř bitů, tj. nabývá hodnoty 0 až 15. Specifikuje přenášená data pro případ rozhodování v okamžiku zahlcení sítě. V okamžiku zahlcení musí směrovač IP datagramy zahazovat. V tomto poli se specifikuje, které IP datagramy je možné zahodit dříve než jiné.

Interval 0–15 je rozdělen na dvě části:

- ◆ 0–7 je určeno pro klasický provoz:
  - 0 – nespecifikovaná data
  - 1 – provoz na pozadí (např. news)
  - 2 – automatický provoz (např. mail)
  - 4 – uživatelem (člověkem) prováděné velké přenosy dat (např. ftp)
  - 6 – interaktivní provoz (např. telnet, X-windows)
  - 7 – řízení sítě (např. směrovací protokoly, SNMP)
- ◆ 8–15 je určeno pro přenosy v reálném čase (např. audio). Datagramy s nižší hodnotou si uživatel přeje zahodit dříve než datagramy s vyšší hodnotou. To však platí pouze v intervalu 8–15, protože intervaly 0–7 a interval 8–15 se zpracovávají odděleně.



**Obrázek 8.1:** Základní záhlaví IP datagramu verze 6

Pole **Identifikace toku dat** je naprostě neotřelá myšlenka. Spolu s adresou odesilatele jednoznačně identifikuje jeden dílčí tok dat v Internetu. Dospod se směrování provádělo výhradně na základě adresy příjemce. Nevýhodou směrování v Internetu je, že jednotlivé IP datagramy se dopravují samostatně, tj. pokud Internetem prochází tok IP datagramů mezi dvěma aplikacemi, pak směrovače na cestě řeší směrování pro každý procházející datagram samostatně. Např. pokud přenášíte několik MB dlouhý soubor, tečou Internetem tisíce datagramů. Každý směrovač na cestě se musí zabývat každým z těchto datagramů samostatně. Pokud nedojde ke změně v topologii sítě, pak pro tisíce datagramů směrovač řeší stejnou úlohu se stejným výsledkem.

Myšlenka spočívá v tom, že datagramy jednoho toku dostanou svou identifikaci. Pak stačí, aby směrovač vyřešil úlohu (do kterého rozhraní datagram předat) pro první datagram toku a do paměti si poznamenal výsledek. Pro další datagram nejprve prohlédne paměť, a pokud by tam nenašel poznamenaný tok, řeší úlohu směrování. Další datagramy stejněho toku pak bude předávat do stejného rozhraní, aniž by řešil úlohu směrování – pouze na základě údajů v paměti.

Tok je určen adresou odesilatele, adresou příjemce, polem identifikace toku dat a třídou dat.

Položka v paměti směrovače nesmí zůstat déle než 6 sekund. Nebezpečí čihá totiž v tom, že odesilatel resetuje svůj počítač. Zavede znova operační systém a shodou okolností vygeneruje stejnou identifikaci pro jiný tok. Nepředpoládá se, že by uživatel stihl provést reset počítače do 6 sekund.

Jinou možností je využít identifikaci toku dat k zajištění šířky přenášeného pásma. Směrovače na cestě od odesilatele k příjemci se nakonfigurují tak, aby pro datový tok o jisté identifikaci zajišťovaly určitou šířku pásma. Datagramy docházejí na směrovač, kde se umisťují do vyrovnávací paměti. Za normálních okolností se jedná o frontu datagramů, která z jednoho konce přibývá a z druhého konce se datagramy odebírají (odesílají). Směrovač však nemusí frontu zpracovávat sekvenčně, ale může přednostně vybírat datagramy tak, aby zajistil dohodnutou šíři pásma. V tomto případě pochopitelně nelze stanovit šestisekundový limit.

Pole **délka dat** specifikuje délku IP datagramu bez základního záhlaví. Jelikož je toto pole dlouhé 2 bajty, může být největší délka přenášeného IP datagramu 65 535 bajtů. Je však možné použít volbu ohromný datagram v další hlavičce informace pro směrovače, která pak umožňuje odesílat ještě větší datagramy.

Pole **další hlavička** specifikuje typ následující hlavičky. Jako vnořená hlavička mohou být např. typy uvedené v tab. 8.1. Mechanismu dalších hlaviček je věnována samostatná část (str. 210).

**Tabulka 8.1:** Typy dalších hlaviček v IP datagramu verze 6

<b>0</b>	<b>Informace pro směrovače (Hop-by-Hop Header)</b>
4	IP protokol
6	Protokol TCP
17	Protokol UDP
<b>43</b>	<b>Směrovací informace (Routing Header)</b>
<b>44</b>	<b>Záhlaví fragmentu (Fragment Header)</b>
45	Protokol IRP
46	Protokol RRP
<b>50</b>	<b>Bezpečnostní hlavička (Encapsulating Security Payload)</b>
<b>51</b>	<b>Autentizační hlavička (Authentication Header)</b>
<b>58</b>	<b>Protokol ICMP</b>
<b>59</b>	<b>Další hlavička již nenásleduje</b>
<b>60</b>	<b>Jiná volba (Destination Options)</b>

Pouze typy, které jsou v tab. 8.1 uvedeny tučně, jsou součástí IP protokolu. Ostatní typy jsou typy protokolů vyšších vrstev.

Pole **počet hopů** v podstatě odpovídá položce TTL (doba života datagramu) v IP verze 4. Využít jej lze především jedním z následujících způsobů:

1. K zahazování zatoulaných datagramů. IP datagramu je položka počet hopů snižována při průchodu každým směrovačem. Po dosažení nuly je datagram považován za zatoulaný a je zahozen.
2. K nalezení nejkratší cesty v Internetu (obdoba příkazu *traceroute*). Cílem je vyhledat nejbližšího člena určitého adresného oběžníku. Nejprve se odešle datagram, jehož adresátem je adresný oběžník (*multicast*) s nastaveným polem počet hopů na jedničku. Pokud se žádný člen neozve, pak se odešle datagram s polem počet hopů nastaveným na dvojku atd.

Programem Wireshark si vypíšeme IPv6 datagram:

```
Ethernet II, Src: 00:19:d2:29:51:53 Dst: 00:4f:62:0f:fc:89
Internet Protocol Version 6
 0110 .... = Version: 6
 .... 0000 0000 .... .... .... .... .... = Traffic class: 0x00000000
 .... .... .... 0000 0000 0000 0000 0000 = Flowlabel: 0x00000000
  Payload length: 40
  Next header: ICMPv6 (0x3a)
  Hop limit: 128
  Source: fe80::594d:e8e9:987e:3c80 (fe80::594d:e8e9:987e:3c80)
  Destination: fe80::213:2ff:fe91:2e8b (fe80::213:2ff:fe91:2e8b)
  Internet Control Message Protocol v6
```



**Poznámka:** Pokud jsme v předešlých kapitolách chtěli něco demonstrovat v prostředí Microsoft Windows, tak jsme často použili fázi „od Windows 2000 výše“, protože z hlediska implementace sítí byly Windows 2000 rozhodujícím zlomem. V případě IPv6 takovým zlomem jsou až Windows Vista (ve Windows XP byla pouze experimentální implementace IPv6). Pokud je tedy dále v této kapitole uvedeno „ve Windows“, tak tím míníme „od Windows Vista výše“.

## Další hlavičky v IP datagramu

Za základním záhlavím IP datagramu mohou následovat další hlavičky.

Pole další hlavička v základním záhlaví ukazuje, jaký typ dat („jaká hlavička“) následuje za základním záhlavím. Teoreticky může ukazovat již na TCP segment nebo jiný protokol vyšší vrstvy. Avšak může také ukazovat na další rozšiřující hlavičky IP protokolu.

Pokud ukazuje na další rozšiřující hlavičku, pak i tato hlavička má pole „další hlavička“, která ukazuje na další hlavičku. Hlavičky tak tvoří řetězec. Řetězec obsahuje jen ty hlavičky, které jsou nutné. Je to rozdílné od IP protokolu verze 4, který ve svém záhlaví často přenáší nadbytečné informace.

Za polem další hlavička je pole délka hlavičky. Pole délka hlavičky specifikuje posunutí, jaké je třeba udělat k další hlavičce. Základní záhlaví délku nemá, protože je dlouhé vždy 40 bajtů. Rovněž u záhlaví fragmentů se délka nepoužívá, protože tato hlavička je vždy dlouhá 8 bajtů.

## Informace pro směrovače

Tato hlavička obsahuje jednotlivé informace (volby), které jsou určeny pro směrovače přepravující datagram. Každý směrovač, který datagram předává, se musí těmito volbami zabývat.

Volba se skládá z pole typ dlouhého 1 B, pole délka dlouhého 1 B a pole obsahujícího vlastní volbu. Pole typ se skládá z osmi bitů:

aabxxxxx

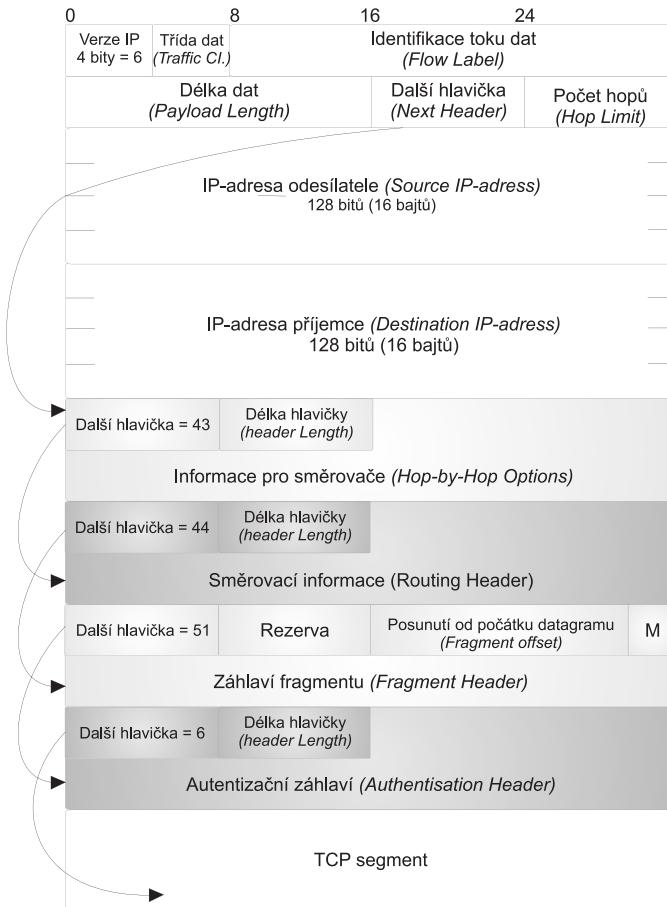
bit y aa sdělují, co má směrovač s datagramem udělat, když nerozezná tuto volbu. Možnosti jsou:

- 00 Pokud volbu nerozpoznáš, pak ji ignoruj a datagram zpracovávej dále.
- 01 Pokud volbu nerozeznáš, pak datagram zahoď a neprováděj žádné další akce.
- 10 Pokud volbu nerozeznáš, pak datagram zahoď a informuj o tom odesilatele protokolem ICMP.
- 11 Pokud volbu nerozeznáš, pak datagram zahoď a v případě, že datagram není adresován adresnému oběžníku, informuj o tom protokolem ICMP odesilatele.

Bit b sděluje, zda může směrovač tuto volbu změnit:

- 0 – Volba se nesmí změnit.
- 1 – Volba se může změnit.

Tabulka 8.2 uvádí některé volby. Volba výplň je dlouhá jeden bajt a skládá se pouze z typu (nula), pole délka a volba má prázdné.



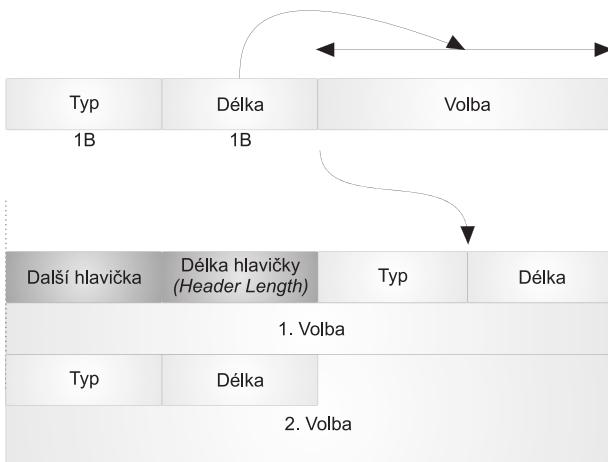
**Obrázek 8.2:** Struktura IP datagramu verze 6, šipky vyjadřují návaznost jednotlivých hlaviček

**Tabulka 8.2:** Některé volby z hlavičky informace pro směrovače

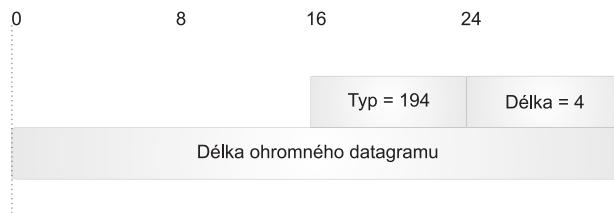
Typ desítkově	Typ dvojkově	Název	Délka	Hodnota v poli délka
0	00000000	Výplň dlouhá 1 bajt	1	není
1	00000001	Výplň dlouhá n bajtů	2 + n	n
194	11000010	Ohromný datagram	2 + 4	4

Výplně jsou určeny pro zarovnání hlavičky na určitou délku. Volba rozsáhlý datagram využívá právě zarovnání. Tato volba musí totiž končit na hranici čtyřbajtu (viz obr. 8.4).

Pokud je volba délka rozsáhlého datagramu použita, je délka datagramu v základním záhlaví nastavena na nulu a používá se délka uvedená v této volbě. Zatímco v základním záhlaví jsou pro délku určeny 2 bajty (tj. datagram může být dlouhý až 64 KB), 4 bajty volby rozsáhlý datagram umožňují délku až 4 GB.



Obrázek 8.3: Struktura jedné volby z hlavičky informace pro směrovače



Obrázek 8.4: Volba délka rozsáhlého datagramu končící na hranici čtyřbajtu

## Směrovací informace

Hlavička směrovací informace používá tč. jedinou volbu (Typ=0), kterou je explicitní směrování. To znamená, že odesíatel specifikuje IP adresy směrovačů, přes které má být datagram dopravován (viz obr. 8.5).

Ve spodní části hlavičky jsou uvedeny IP adresy směrovačů, přes které si odesíatel přeje, aby byl datagram směrován.

Pole **Maska striktního směrování** obsahuje 24 bitů (bity 0 až 23 číslované zleva doprava). Každý bit odpovídá jednomu „hopu“ a říká, zdali v hlavičce uvedený následující směrovač musí být sousední (tzv. striktní směrování) nebo zdali mezi ním mohou být další směrovače. Pakliže je bit nastaven na 1, jedná se o explicitní striktní směrování na následující „hop“.

Pole také určuje, přes kolik vyjmenovaných směrovačů má být datagram ještě směrován. Každý směrovač, který je v hlavičce vyjmenován, snižuje hodnotu tohoto pole o 1.



**Obrázek 8.5:** Volba explicitní směrování

Problém je ale v tom, že pokud má být datagram přes nějaký směrovač směrován, musí být uvedena IP adresa tohoto směrovače v adrese příjemce. Proto je v adrese příjemce vždy vložena IP adresa následujícího směrovače, přes který má být datagram dopravován. Adresa skutečného příjemce se pak uloží do hlavičky směrovací informace mezi ostatní vyjmenované směrovače.

Příklad koloběhu IP adres v hlavičce směrovací informace je uveden na obr. 8.6.

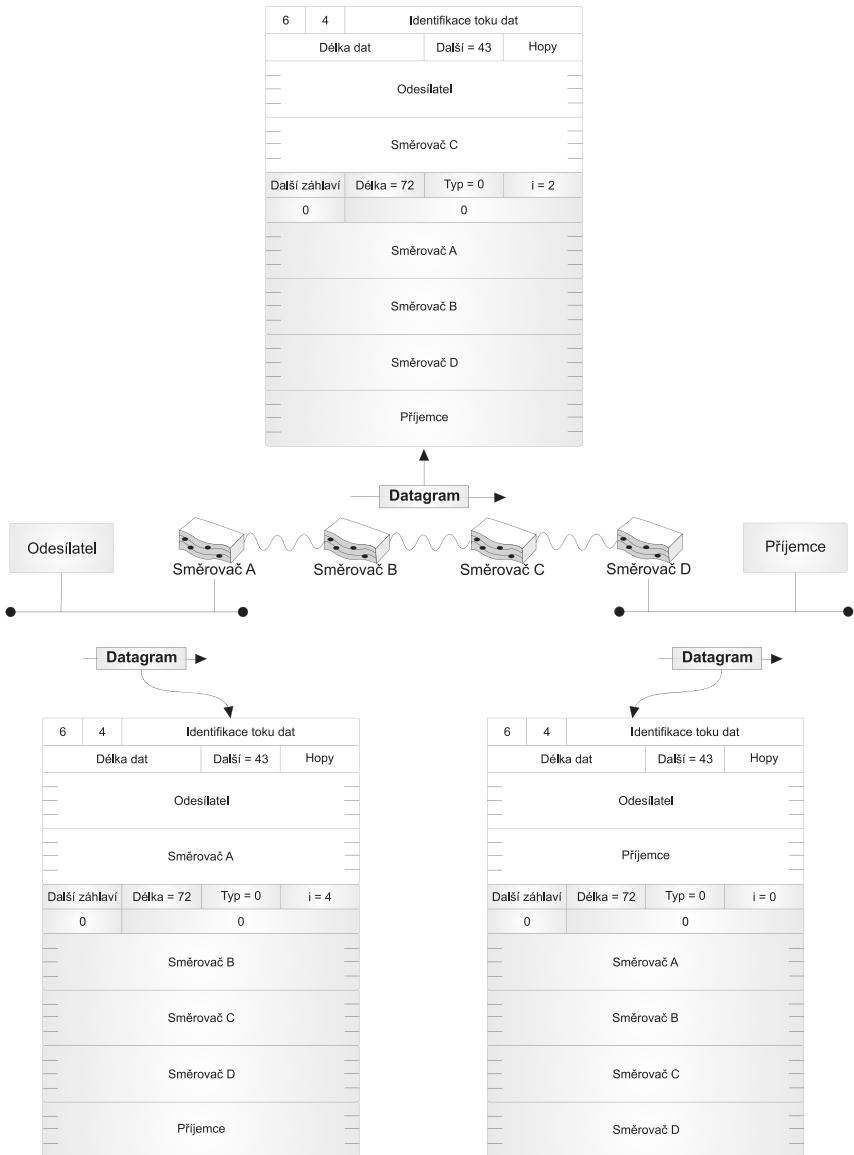
## Záhlaví fragmentu

Fragmentovat IP datagramy verze 6 může pouze operační systém na straně odesilatele. Směrovače, kterými datagram prochází na své cestě od odesilatele k příjemci, fragmentovat na rozdíl od IPv4 již nesmí.

Na rozdíl od IP protokolu verze 4 neobsahuje každý IP datagram (např. v základním záhlaví) identifikaci IP datagramu. Identifikace IP datagramu je nutná pro fragmentaci, aby příjemce zjistil, které fragmenty patří do stejného datagramu. V IP verze 6 je identifikace datagramu pouze v dalším záhlaví, tj. není součástí každého IP datagramu.

Pole posunutí od počátku datagramu slouží k sestavení datagramů. Pomocí tohoto pole příjemce zjistí pořadí, v jakém má fragmenty skládat za sebe. Pole posunutí od počátku datagramu neudává posunutí v bajtech, ale v násobcích osmi bajtů (v „osmabajtech“), proto se při fragmentaci musí zajistit, aby fragmenty byly odřezávány v délce, která je dělitelná osmi.

A konečné pole M (More Fragment) indikuje poslední fragment. Jednobitové pole M je nastaveno na jedničku, pouze v případě posledního fragmentu na nulu.



**Obrázek 8.6:** Příklad koloběhu IP adres při explicitním směrování



**Obrázek 8.7:** Záhlaví fragmentu

### Autentizační hlavička (protokol AH)

Pomocí autentizační hlavičky se zajišťuje integrita dat a odesilatel autentizuje data, tj. prokazuje, že data odeslal on. Datagram je tak zabezpečen proti změně IP datagramu útočníkem i proti záměně celých IP datagramů odesilatele za IP datagramy vložené útočníkem.

Další hlavička	Délka	Rezerva
Index bezpečnostních parametrů ( <i>Security parameters Index</i> )		
Autentizační data		

Obrázek 8.8: Autentizační hlavička

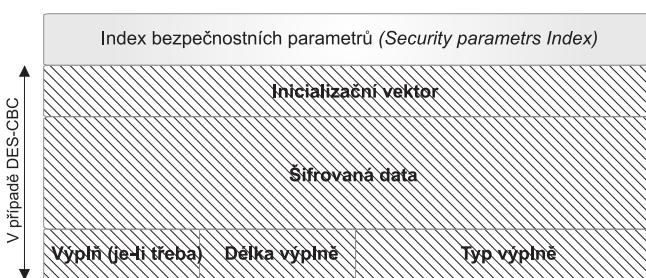
Základní autentizace využívá algoritmus MD-5 (RFC-1321) pro výpočet kontrolního součtu. Autentizační data jsou kontrolním součtem počítaným z tajemství zřetězeného s IP datagramem. IP datagram, který vstupuje do výpočtu, má však vynulována všechna pole v záhlaví, která se mohou měnit (např. pole počet hopů).

Tajemství je 128 bitů dlouhý řetězec (je-li menší, doplní se nulami). Tento řetězec si předem musí vyměnit odesilatel s příjemcem. Těchto řetězců – tajemství může mít jak odesilatel, tak i příjemce více. Má je uloženy v tabulce. Pole Index bezpečnostních parametrů je indexem do této tabulky řetězců – tajemství.

### Bezpečnostní hlavička (protokol ESP)

Bezpečnostní hlavička umožňuje nejen zajišťovat integritu a autentizaci dat, ale přenášená data i šifrovat. Musí to být poslední hlavička IP datagramu, protože data jsou již dále šifrována, tj. nedostupná ke zpracování směrovačům dopravujícím IP datagram.

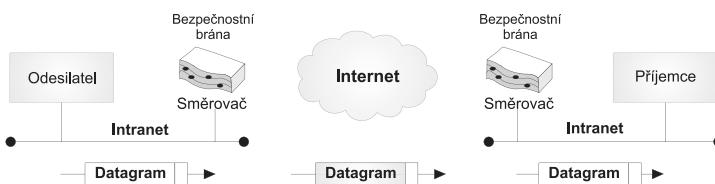
První čtyři bajty jsou obdobně jako u autentizační hlavičky indexem do tabulky udržující informace nutné pro šifrování (typ šifry, mód šifry, šifrovací klíče atd.). Na obr. 8.9 je příklad struktury šifrovaných dat pro případ, že je použita šifra DES v módu CBC.



Obrázek 8.9: Bezpečnostní hlavička

Možnosti využití bezpečnostní hlavičky jsou překvapivě dvě (a jejich kombinace):

1. Šifrování provádí odesilatel, dešifrování příjemce.
2. Odesilatel ani příjemce se šifrováním nezabývají, ale směrovače přepravující datagram jej šifrují/dešifrují při průchodu méně bezpečnou sítí. Tyto směrovače se označují jako bezpečnostní brány (*Security Gateway*).



**Obrázek 8.10:** Bezpečnostní brána

Bezpečnostní brány je praktické použít pro oddělení intranetu od Internetu. Provoz v intranetu je možný nešifrovaný, kdežto provoz mezi dvěma částmi firmy přes Internet se zabezpečí šifrováním. O protokolech AH a ESP se lze blíže dočíst v publikaci „Velký průvodce protokoly TCP/IP, bezpečnost“.

## ICMPv6

Podobně jako v případě IP protokolu verze 4 se pro signalizaci chybových stavů a pro diagnostiku používá protokol ICMP, tak i v IPv6 se používá protokol ICMP. Protokol ICMP pro IP protokol verze 6 je specifikován RFC-2463.

Protokol ICMP verze 6 v mnoha případech přináší zcela odlišné funkčnosti oproti protokolu ICMPv4. Řeší např. překlad IP adres na linkové adresy. (Na tento problém jsou v protokolu IP verze 4 určeny samostatné protokoly ARP a RARP.)

Z hlediska struktury paketu se ICMP paket jeví jako protokol vyšší vrstvy, tj. je předcházen základním záhlavím IP protokolu i případnými dalšími hlavičkami (viz obr. 8.11).

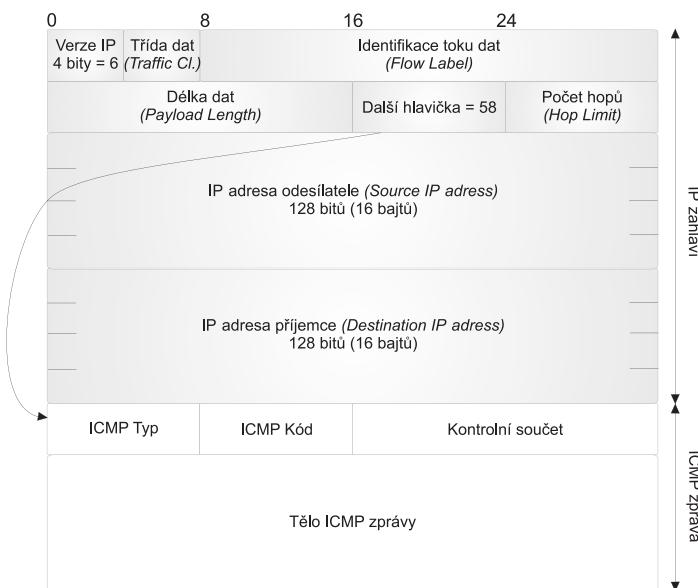
Pole **ICMP typ** obsahuje typ zprávy (hrubší dělení zpráv) a pole **ICMP kód** obsahuje jemnější dělení zpráv.

RFC-2461 a RFC-2463 specifikují typy a kódy ICMP zpráv, které jsou uvedeny v tab. 8.3.

**Tabulka 8.3:** Typy a kódy ICMP zpráv

Typ	Kód	Popis
1	0	Nedoručitelný IP datagram ( <i>Destination Unreachable</i> )
	1	Ve směrovací tabulce neexistuje směr pro tuto adresu ( <i>no route to destination</i> )
	3	Spojení s adresátem je administrativně uzavřeno ( <i>communication with destination administratively prohibited</i> )
	4	Nedosažitelná adresa ( <i>address unreachable</i> )
	2	Nedosažitelný port ( <i>port unreachable</i> )
2	0	Příliš velký datagram ( <i>Packet Too Big</i> )

Typ	Kód	Popis
3	0	Čas vypršel ( <i>Time Exceeded</i> )
	1	Dosažen počet hopů ( <i>hop limit exceeded in transit</i> )
		Vypršel čas na sestavení IP datagramu z jeho fragmentů ( <i>fragment reassembly time exceeded</i> )
4	0	Chybný parametr ( <i>Parameter Problem</i> )
	1	Chybné pole v záhlaví ( <i>erroneous header field encountered</i> )
	2	Nepodporovaný typ v poli další hlavička ( <i>unrecognized Next Header type encountered</i> )
128	0	Žádost o echo ( <i>Echo Request</i> )
129	0	Echo ( <i>Echo Reply</i> )
133	0	Žádost o směrování ( <i>Router Solicitation</i> )
134	0	Oznámení o směrování ( <i>Router Advertisement</i> )
135	0	Žádost o linkovou adresu ( <i>Neighbor Solicitation</i> )
136	0	Oznámení o linkové adrese ( <i>Neighbor Advertisement</i> )
137	0	Změň směrování ( <i>Redirect Message</i> )



**Obrázek 8.11:** Struktura ICMP paketu

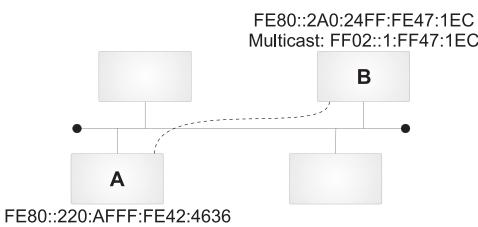
Typy ICMP zpráv se dělí na dva intervaly:

- ◆ Interval 0 až 127, což jsou chybové zprávy.
- ◆ Interval 128 až 255, což jsou informativní zprávy.

Smysl ICMP zpráv typů 1 až 129 uvedených v tab. 8.3 je obdobný ICMP zprávám v protokolech IP verze 4. Proto se zastavíme u zbývajících typů zpráv.

## Překlad IP adres na linkové adresy

Jak jsme se již zmínili, jedná se o funkčnost, kterou v IPv4 zajišťuje protokol ARP. Na obrázku 8.12 chce stanice A o IP adresě FE80::220:AFFF:FE42:4636 odeslat stanici B IP datagram. Jenže IP datagram musí být vložen do linkového rámce. Stanice A sice zná IP adresu stanice B (ta je FE80::2A0:24FF:FE47:1EC), avšak pro vytvoření linkového rámce potřebuje také linkovou adresu stanice B.



**Obrázek 8.12:** Stanice A chce odeslat IP datagram stanici B

Proto odesílá zprávu „žádost o linkovou adresu“.

Pro zjištění linkové adresy svého souseda na lokální síti slouží dvě ICMP zprávy:

1. **Žádost o linkovou adresu**, kterou stanice A žádá oběžníkem stanici B, aby stanici A poskytla svou linkovou adresu.
2. **Oznámení o linkové adrese**, kterou stanice B poskytuje stanici A svou linkovou adresu.

Žádost o linkovou adresu (viz obrázek 8.13) má v poli maximální počet hopů uvedeno číslo 255. Pokud by tato žádost prošla přes směrovač (tj. přišla z jiné sítě), pak by bylo toto pole sníženo na hodnotu menší než 255 a příjemce by tak zjistil, že se jedná o zatoulanou žádost z jiné sítě, což mu znemožňuje na ni odpovědět. (Linkové adresy jsou jednoznačné pouze v rámci LAN).

Základní záhlaví má na obr. 8.13 v poli příjemce uvedenu zvláštní adresu FF02::1:FF47:1EC. Jedná se: oběžník (Multicast) speciálně konstruovaný pro tuto ICMP zprávu. Je tvořen prefiksem FF02:0:0:0:0:1:FF/108 a ve zbývajících 24 bitech má nejnižších 24 bitů z adresy IP verze 6. Jak si popíšeme dále, na těchto 24 bitech budou zpravidla nejnižší tři bajty z linkové adresy (přidělené výrobcem karty).

Pole typ v ICMP zprávě má hodnotu 135. Pole kód má hodnotu 0, ale směrovače mohou upozorňovat, že se jedná o směrovač nastavením pole kód na jedničku.

Tělo ICMP zprávy obsahuje IP adresu stanice, o jejíž linkovou adresu se žádá, a dále obsahuje volbu, ve které je uvedena linková adresa žadatele (odesilatele). Volby se v ICMP zprávách zásadně skládají ze tří polí:

- ◆ jednobajtové pole volba obsahujícího identifikaci volby (např. odesilatelova linková adresa má identifikaci volby 1 v dotazu a volbu 2 v odpovědi),
- ◆ jednobajtové pole délka obsahujícího délku volby,
- ◆ vlastní volby.

Zajímavý je i výpis z programu Wireshark:

```
Ethernet II
Internet Protocol Version 6
```

```

Internet Control Message Protocol v6
Type: 135 (Neighbor solicitation)
Code: 0
Checksum: 0xe085 [correct]
Target: fe80::2a0:24ff:fe47:1ec
ICMPv6 Option (Source link-layer address)
    Type: Source link-layer address (1)
    Length: 8
    Link-layer address: 00:20:af:42:46:36

```

6	15	0
Délka dat = 48	Další = 58	Hopy = 255
<hr/> <b>Odesílatel:FE80::220:AFFF:FE42:4636</b> <hr/>		
<hr/> <hr/>		
<hr/> <b>Příjemce: FF02::1:FF47:1EC</b> <hr/>		
Typ = 135	Kód = 0	Kontrolní součet
Nevyužito = 0		
<hr/> Hledá se linková adresa k IP adrese: FE80::2A0:24FF:FE47:1EC <hr/>		
Volba = 1	Délka volby	
<b>Odesílatelova linková adresa (tj. stanice A)</b> 00-20-AF-42-46-36		

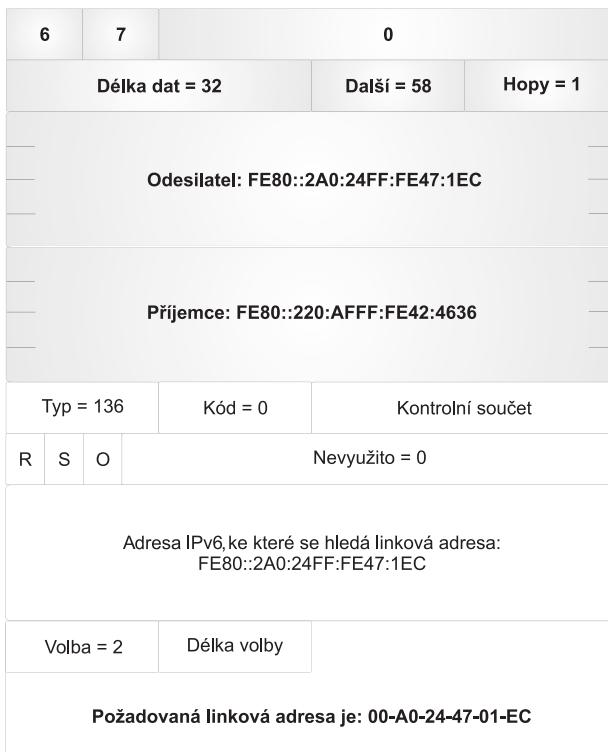
**Obrázek 8.13:** Žádost o linkovou adresu

Odpověď na žádost je ICMP zpráva **oznámení o linkové adrese** (viz obr. 8.14). Popišme si podrobněji tuto zprávu.

Základní záhlaví IP datagramu obsahuje v poli počet hopů hodnotu 1, aby se zamezilo zatoulání odpovědi na jinou síť, a v poli pro IP adresu příjemce již není oběžník, ale IP adresa žadatele.

Tělo ICMP zprávy obsahuje tři tajuplné byty: R, S a O. Dále je v těle zprávy zopakována IP adresa stanice, o jejíž linkovou adresu se žádalo, a konečně volba o identifikaci 2 obsahuje požadovanou linkovou adresu.

Bit R je nastaven na jedničku, když odesílatel je směrovač, bit S je nastaven na jedničku, jedná-li se o odpověď na žádost (tj. předcházela-li ICMP zpráva žádost o linkovou adresu), a konečně bit O je nastaven na jedničku, když odesílatel chce zdůraznit, že příjemce si má přepsat hodnoty uložené v paměti.



**Obrázek 8.14:** Oznámení o linkové adrese

Příjemce si odpověď „oznámení o linkové adrese“ uloží do paměti, aby nemusel s dalším odesílaným IP datagramem znova podstupovat martyrium žádosti o linkovou adresu svého souseda.

Zajímavý je i výpis z programu Wireshark:

```

Ethernet II
Internet Protocol Version 6
Internet Control Message Protocol v6
Type: 136 (Neighbor advertisement)
Code: 0
Checksum: 0x6d5d [correct]
Flags: 0x60000000
    0... .... .... .... .... .... .... = Not router
    .1.. .... .... .... .... .... .... = Solicited
    ..1. .... .... .... .... .... .... = Override
Target: fe80::2a0:24ff:fe47:1ec
ICMPv6 Option (Target link-layer address)
    Type: Target link-layer address (2)
    Length: 8
    Link-layer address: 00:a0:24:47:01:ec

```



**Poznámka:** Pokud si ve Windows chceme vypsat obsah paměti s uloženými odpověďmi, pak použijeme příkaz:

```
C:\> netsh interface ipv6 show neighbors
```

Rozhraní 10: Wireless Network Connection 2

Internetová adresa	Fyzická adresa	Typ
fe80::594d:e8e9:987e:3c80	00-19-d2-29-51-53	Zastaralé
fe80::213:2ff:fe91:2e8b	00-13-02-91-2e-8b	Trvalé

Rozhraní 1: Loopback Pseudo-Interface

Internetová adresa	Fyzická adresa	Typ
::1		Trvalé
fe80::213:2ff:fe91:2e8b		Trvalé
fe80::594d:e8e9:987e:3c80		Trvalé
fe80::1		Trvalé

Z uvedeného příkladu je pěkně vidět, že obsah této paměti je obdobný jako obsah ARP-cache – opět jsou zde položky trvalé a dočasné. Dočasné vznikly právě na základě přijetí ICMPv6 zprávy Oznámení o linkové adrese. Obsah paměti jsem vypsal po zadání příkazu

```
C:\> ping -6 fe80::594d:e8e9:987e:3c80
```

Jenže než jsem si vypsal obsah paměti, tak mně položka fe80::594d:e8e9:987e:3c80 v paměti expirovala (je „zastaralá“).



**Poznámka:** V IPv6 si musíme zvyknout, že se podstatně častěji používají indexy (čísla) jednotlivých síťových rozhraní.

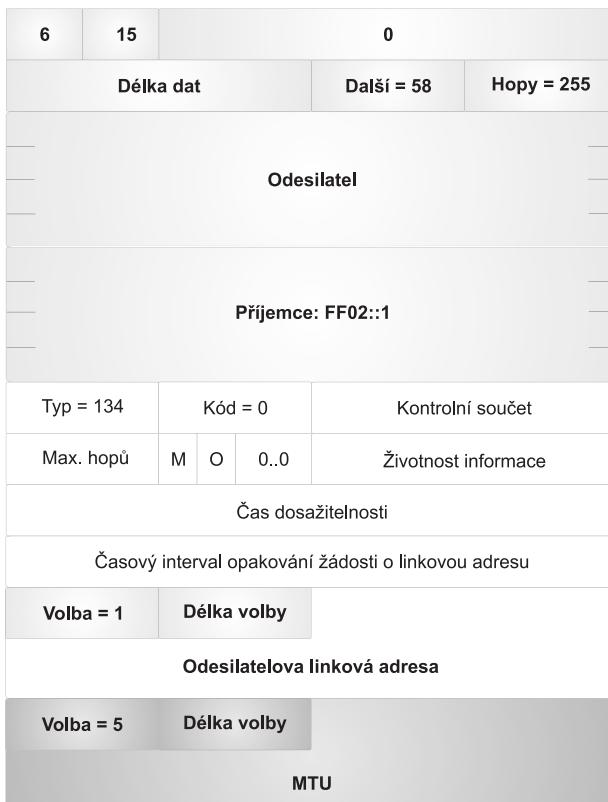
## Zjištění adresy směrovače na LAN

Pokud stanici nestačí komunikace pouze na LAN, musí pro komunikaci mimo LAN znát adresu směrovače. Řečí směrovačů: stanice potřebuje získat položku *default* do své směrovací tabulky. Směrovače v pravidelných intervalech rozšiřují tuto informaci oběžníkem pro všechny počítače na LAN (FF02::1) pomocí ICMP zprávy oznamení o směrování – viz obr. 8.15.

Po přijetí zprávy „oznámení o směrování“ si stanice vytvoří položku *default* do své směrovací tabulky jednoduše tak, že směr *default* bude ukazovat na odesilatele této zprávy.

ICMP zpráva oznamení o směrování obsahuje následující pole:

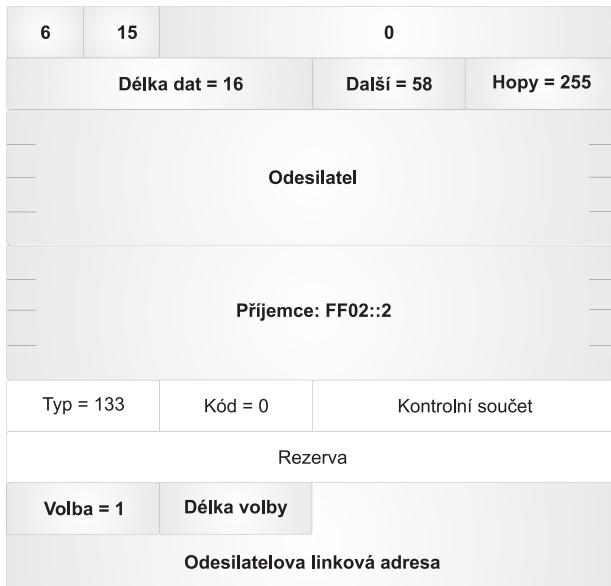
- ◆ Pole typ ICMP zprávy, které má hodnotu 134, a pole kód o hodnotě nula.
- ◆ Pomocí pole max. hopů je stanicím doporučována hodnota, kterou mají vyplňovat do pole počet hopů v základním záhlaví IP datagramu.



**Obrázek 8.15:** Oznámení o směrování

- ◆ Bity M a O slouží pro protokoly vyšších vrstev zabývajících se automatickou konfigurací stanice (např. protokol DHCP).
- ◆ Pole životnost informace (*Router Lifetime*) obsahuje čas v sekundách, po který by měla stanice položku *default* vytvořenou na základě této zprávy udržovat ve svých směrovacích tabulkách. Hodnota nula specifikuje, že se žádná položka *default* ukazující na odesílatele tohoto datagramu nemá ve směrovacích tabulkách dále udržovat.
- ◆ Pole čas dosažitelnosti a pole časový interval opakování žádosti o linkovou adresu se týkají žádostí/odpovědí o linkovou adresu souseda (tj. předchozí kapitoly). Oba údaje se uvádějí v milisekundách. Čas dosažitelnosti (*Reachability Timeout*) specifikuje čas, po který lze předpokládat, že soused je dosažitelný. Časový interval opakování žádosti o linkovou adresu je interval, po který se má udržovat v paměti položka získaná ze zprávy oznámení o linkové adrese.

Dále může ICMP zpráva oznámení o směrování také obsahovat některé volby. Na obr. 8.15 jsou uvedeny volby odesíatelova linková adresa a MTU. S volbou odesíatelova linková adresa jsme se setkali již u zprávy oznámení o linkové adrese. Zde má napomoci k tomu, aby stanice nemusela další zprávu ICMP požádat směrovač o jeho linkovou adresu. Volba MTU specifikuje maximální podporovanou velikost linkového rámce.



**Obrázek 8.16:** ICMP zpráva Žádost o směrování

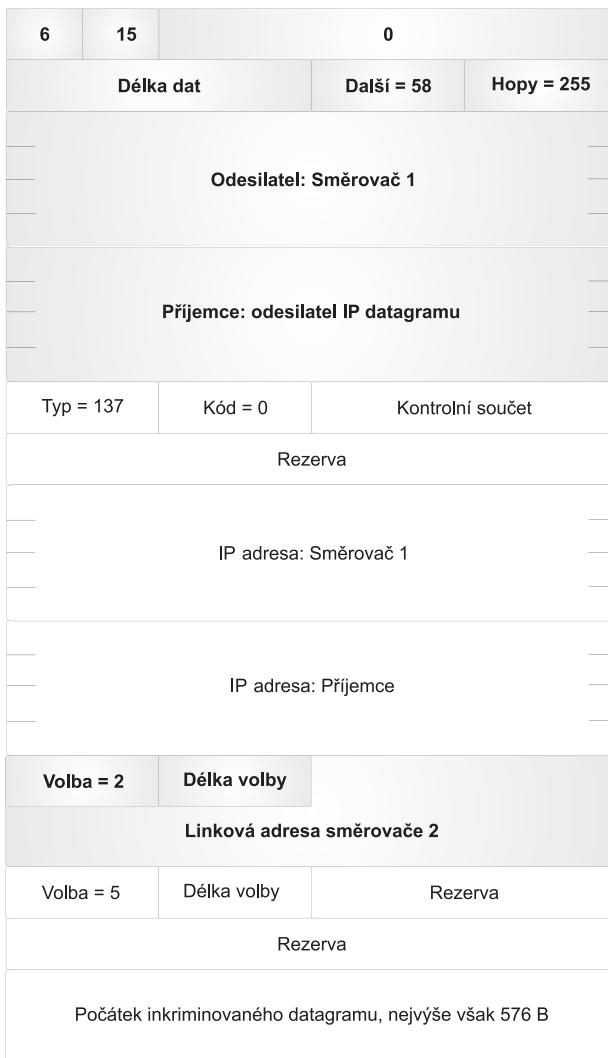
Stanice může také sama požádat směrovač o zaslání zprávy o směrování. Protokol ICMP má pro tento případ k dispozici ICMP zprávu žádost o směrování. Tuto zprávu (viz obr. 8.16) odesílá stanice oběžníkem pro všechny směrovače (FF02::2). Směrovač pak již neodpovídá oběžníkem pro všechny počítače, ale přímo jednoznačnou adresou tazatele pomoci zprávy oznámení o směrování.

## Změň směrování

Situace je znázorněna na obr. 5.11. Na LAN je více směrovačů. Odesíatel posílá IP datagram příjemci. Jelikož ve směrovací tabulce nemá přímý směr přes směrovač 2, využije položku *default*, kterou získal např. z ICMP zprávy oznámení o směrování a která ukazuje na směrovač 1. Směrovač 1 přijme datagram, ale je nucen jej stejným rozhraním předat směrovači 2. Směrovač 1 to sice provede, ale neodpustí si o tom ICMP zprávou změň směrování informovat odesilatele. Odesíatel přijme zprávu změny směrování, ze které si vytvoří novou položku do své směrovací tabulky, která ukazuje příjemce přímo přes směrovač 2.

ICMP zpráva změny směrování je znázorněna na obr. 8.17.

Zajímavé je též pole kód, které je 0 v případě, že adresátem ICMP zprávy je počítač, a 1 v případě, že adresátem je směrovač.

**Obrázek 8.17:** Změň směrování

## IPv6 – adresa

IP adresa je v protokolu IPv6 šestnáctibajtová (128 bitů). Rozeznáváme tři typy adres:

- ◆ Jednoznačná adresa síťového rozhraní (**Unicast**).
- ◆ **Anycast** – adresa skupiny síťových rozhraní, IP datagram adresovaný adresou typu *anycast* bude doručen jednomu z těchto rozhraní („nejbližšímu“ z hlediska topologie sítě). Tyto adresy jsou přidělovány z prostoru jednoznačných adres (Unicast). Příkladem tohoto typu adresy je adresa označovaná jako „subnet-router anycast“. Ta má na místě pro adresu rozhraní samé nuly – jedná

se tedy z pohledu IP verze 4 o „adresu sítě“, ale ozvat by se měl jen jeden počítač. Její využití je praktické např. pro adresaci přístupového bodu z mobilního zařízení: „Chci nejbližší přístupový bod“.

- ◆ Oběžník (*Multicast*).

Neexistuje zde všeobecný oběžník (*Broadcast*).

## Zápis adresy

Používají se tři zápisy IP adresy:

- ◆  $hhhh:hhhh:hhhh:hhhh:hhhh:hhhh$ , kde h je jedna šestnáctková číslice (0 až F) reprezentující 4 byty adresy.

*Příklad: ABCE:3:89AD:134:FEDC:E4D1:34:4321 (vedoucí nuly se nemusí uvádět)*

- ◆ Zkrácený zápis pomocí zdvojené dvojtečky. Zdvojená dvojtečka se může v adrese vyskytnout pouze jednou. Zdvojená dvojtečka nahrazuje libovolné množství čtevěřic nul.

*Příklad: Adresu 12A1:0:0:0:5:15:500C:44 je možné zkráceně zapsat jako 12A1::5:15:500C:44*

*Adresu 1234:0:0:0:0:0:14 je možné zkráceně zapsat jako 1234::14*

*Smyčku, tj. adresu 0:0:0:0:0:0:1, je možné zkráceně zapsat jako ::1*

- ◆  $hhhh:hhhh:hhhh:hhhh:hhhh:d.d.d.d$ , kde poslední čtevěřice je vyjádřena obdobně jako u IP adresy verze 4, tj. každý bajt je vyjádřen desítkovou číslicí. Tato forma zápisu je vhodná v prostředí, kde se budou společně používat adresy IP verze 4 a IP verze 6.

*Příklady:*

$::195.47.103.12$

$12::A54:147.123.25.4$

Adresy sítí se zapisují ve tvaru prefixu (obdobně jako u adres IPv4). Prefix je vždy následován lomítkem a počtem bitů tvořících adresu sítě. Např.: 80:1::1/64.

**Tabulka 8.4:** Některé části adresního prostoru IPv6

$::/128$	Nespecifikovaná adresa (nepoužívá se pro síťovou komunikaci mezi počítači)
$::1/128$	Lokální smyčka ( <i>Loopback</i> )
...	
$2000::/3$	Jednoznačné adresy ( <i>Global Unicast</i> ). Tyto adresy jsou veřejně přidělovány. Viz kap. 13.
...	
$FC00::/3$	Lokálně jednoznačné adresy ( <i>Unique Local Unicast</i> )
...	
$FE80::/10$	Linkově jednoznačné adresy ( <i>Link Local Unicast</i> )
...	
$FF00::/8$	Oběžníky ( <i>Multicasts</i> )

## Oběžníky (Multicasts)

Oběžníky mají prefix obsahující v prvním bajtu samé binární jedničky, tj. šestnáctkově FF (viz obr. 8.18).



**Obrázek 8.18:** Adresa oběžníku

Druhý bajt je rozdělen na poloviny. První polovina má významný pouze bit T; pokud má hodnotu 0, je oběžník přiřazen trvale, pokud má hodnotu 1, jedná se o dočasné přiřazení.

Druhá polovina druhého bajtu specifikuje rozsah skupiny tvořící oběžník. Nabývá např. hodnot:

- 1 – oběžník v rámci lokálního uzlu,
- 2 – oběžník v rámci LAN,
- 5 – oběžník v rámci firmy,
- 8 – oběžník v rámci vyššího organizačního celku,
- E – globální oběžník.

Existují vyhrazené oběžníky, např. (za x si dosaďte hodnoty rozsahu skupiny – viz předchozí seznam):

- FF0x::1 – oběžník pro všechny stanice (počítače i směrovače),
- FF0x::2 – oběžník pro všechny směrovače,
- FF0x::9 – oběžník pro všechny směrovače provozující protokol RIP atd.

Konkrétně FF02::2 je oběžník určený všem směrovačům na LAN.

## Jednoznačné adresy

Jednoznačné adresy (*Unicasts*) se skládají z:

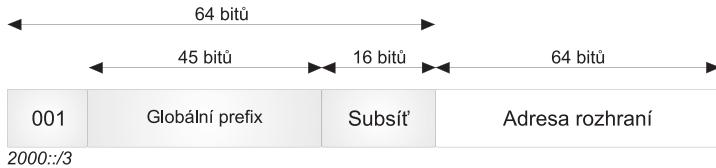
- ◆ adresy sítě – prvních 64 bitů,
- ◆ adresy síťového rozhraní – zbylých 64 bitů.

V zásadě rozlišujeme dva typy jednoznačných adres (obr. 8.19):

- ◆ Globálně jednoznačné adresy, které mohu být přidělovány veřejně skrze poskytovatele Internetu.
- ◆ Lokálně jednoznačné adresy, které jsou jednoznačné mezi sousedy – např. v rámci LAN.

Adresa rozhraní (resp. identifikace rozhraní) má 64 bitů. Máme několik možností nastavení adresy síťového rozhraní:

- ◆ Ruční nastavení.
- ◆ Vygenerování náhodného čísla a jeho nastavení jako adresy síťového rozhraní (viz RFC-4941).

**Jednoznačné adresy (Global Unicast):****Jednoznačné adresy v rámci LAN (Link-Local Addresses):****Obrázek 8.19:** Jednoznačné adresy

- ◆ Microsoft sice vygeneruje náhodnou adresu síťového rozhraní, ale tu už následně používá trvale, což lze odstranit příkazem `netsh interface ipv6 set global randomizeidentifiers=disabled`.
- ◆ Preferovaný způsob adresace je odvozen od adresace rozhraní podle IEEE EUI-64, která má 8 bajtů (64 bitů). Skládá se ze tří bajtů identifikujících výrobce a pěti bajtů přidělených výrobcem. Převod adresy EUI-64 na adresu rozhraní v adresy IPv6 je jednoduchý (obr. 8.20 – spodní část). Transformuje se jediný bit. Jedná se o druhý nejnižší bit v prvním bajtu. Na str. 113 jsme si uvedli, že tento bit specifikuje adresu jako celosvětově jednoznačnou, nebo lokálně administrovanou. Obsah tohoto bitu se přehodí.

My však na linkové vrstvě zpravidla používáme šestibajtovou adresu podle IEEE 802 (tři bajty identifikující výrobce a tři bajty přidělené výrobcem). Musíme si tedy říci, jak se z šestibajtové adresy IEEE 802 udělá osmibajtová adresa IEEE EUI-64. Konverze je jednoduchá: mezi původně třetí a čtvrtý bajt se vloží dva bajty o šestnáctkové hodnotě FFFE. Viz obr. 8.20 – horní část.

**Obrázek 8.20:** Konverze adresy IEEE 802 na adresu rozhraní používanou v IPv6

### Příklad

Moje síťová karta obdržela od výrobce linkovou adresu 00-A0-24-47-01-EC. Po konverzi do EUI-64 obdržím: 00-A0-24-FF-FE-47-01-EC. V protokolu IPv6 se pak použije adresa rozhraní: 2A0:24FF:FE47:1EC. Jelikož na mého počítače běží Windows, tak si mohu příkazem `ipconfig /all` vypsat jednotlivá síťová rozhraní a ejhle, vidím tam rozhraní: FE80::2A0:24FF:FE47:1EC (prefix FE80:: viz obr 8.19 – spodní část).

### Identifikátor (index) síťového rozhraní

Jednoznačné adresy v rámci LAN (*Local Link Unicasts*) mohou být odesány libovolným rozhraním. Ale kterým? Odpověď je jednoduchá. Operační systémy Microsoft přidělují jednotlivým rozhraním čísla (od jedničky výše). Jiné operační systémy zase používají název síťového rozhraní jako index.

Chci-li pak v nějakém příkazu specifikovat, do kterého rozhraní se mají pakety odesílat, zapíšu za IPv6 adresu znak procento a index síťového rozhraní. Např. (parametr -6 indikuje IPv6 adresu):

```
C:\> ping -6 fe80::594d:e8e9:987e:3c80%11
```

A jak zjistím, jaké indexy mají jaká rozhraní? Ve Windows na to máme příkaz:

```
C:\> netsh interface ipv6 show interface
Idx Met MTU     Stav      Název
--- --- ----- -----
 11   0 1500 Připojeno Wireless Network Connection 2
    6   2 1280 Odpojeno  Teredo Tunneling Pseudo-Interface
    5   0 1500 Odpojeno Local Area Connection
    3   1 1280 Připojeno 6to4 Pseudo-Interface
    2   1 1280 Připojeno Automatic Tunneling Pseudo-Interface
    1   0 1500 Připojeno Loopback Pseudo-Interface
```

## Domácí cvičení

Na PC si aktivujte IPv6. Ve Windows Vista je to zcela analogické aktivaci IPv4. Ve Windows XP použijete příkaz `ipv6 install`.

Nyní si:

- ◆ Prohlédněte seznam svých síťových rozhraní a uvidíte, že síťová rozhraní mají IPv6 adresy. Najděte mezi nimi „lokálně jednoznačnou adresu“.
- ◆ Spusťte program Wireshark a sledujte IPv6 provoz vaší stanice.
- ◆ Pokud pracujete na systému Microsoft, pak si procvičte příkaz `netsh interface ipv6`.

## Kapitola 9

# Protokol TCP (Transmission Control Protocol)

Protokol TCP je proti protokolu IP protokolem vyšší vrstvy. První otázkou každého začátečníka vždy je: „Proč jsou třeba dva protokoly?“

Zatímco protokol IP přepravuje data mezi libovolnými počítači v Internetu, protokol TCP dopravuje data mezi dvěma konkrétními aplikacemi běžícími na téhoto počítačích. Protokol IP totiž adresuje IP adresou pouze síťové rozhraní počítače. Abychom data dopravili konkrétní aplikaci, pak nám nestačí identifikace pouze IP adresou, ale musíme rozlišit mezi aplikacemi – přidáváme tzv. port, který je adresou v protokolu TCP.

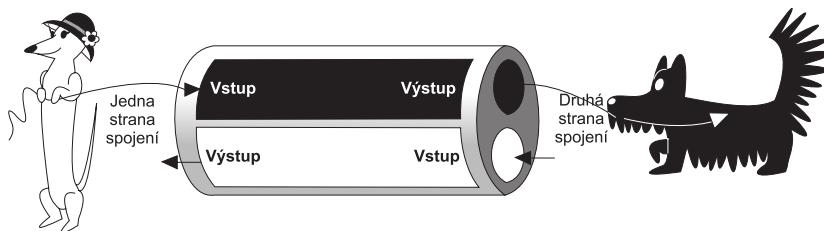


**Poznámka:** Pokud bychom použili přirovnání k běžnému poštovnímu styku, pak IP adresa odpovídá adrese domu a port pak odpovídá jménu konkrétního obyvatele domu.

Protokol TCP je spojovanou službou (*connection oriented*), tj. službou, která mezi dvěma aplikacemi naváže spojení – vytvoří na dobu spojení virtuální okruh. Tento okruh je plně duplexní (data se přenášejí současně na sobě nezávisle oběma směry). Přenášené bajty jsou číslovány. Ztracená nebo poškozená data jsou znova vyzádána. Integrita přenášených dat je zabezpečena kontrolním součtem.

Jinými slovy: aplikace používající protokol TCP si nemusí dělat starosti s tím, zdali náhodou nebyla nějaká data během přenosu ztracena nebo díky chybě přenosu modifikována. Toto zabezpečení je účinné pouze proti poruchám technických prostředků. Neklade si za cíl zabezpečit data proti intelligentním útočníkům, kteří mohou data modifikovat a současně také přepočítat kontrolní součet. Ochrannou přenášených dat proti takovýmto cíleným útokům se v rodině protokolů TCP/IP zabývají např. protokoly SSL, S/MIME.

Konce spojení („odesilatel“ a „adresát“) jsou určeny tzv. číslem portu. Toto číslo je dvoubajtové, takže může nabývat hodnot 0 až 65535. U čísel portů se často vyjadřuje okolnost, že se jedná o porty protokolu TCP tím, že se za číslo napíše lomítko a název protokolu (tcp). V další kapitole se dozvímeme, že protokol UDP má svou (jinou) sada portů (též 0 až 65535), tj. např. port 53/tcp nemá nic společného s portem 53/udp.



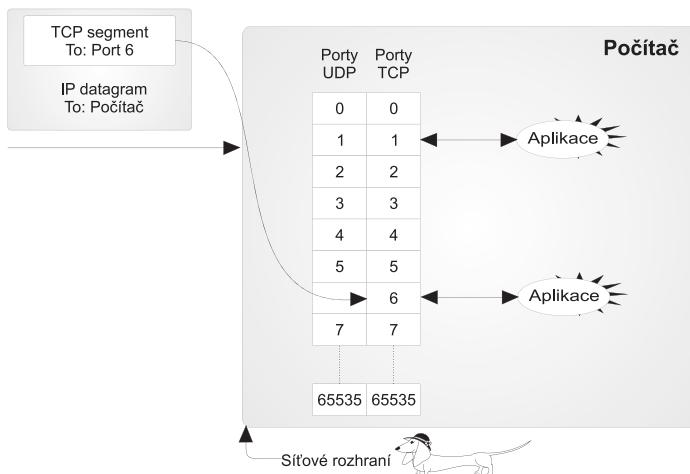
**Obrázek 9.1:** Protokol TCP vytváří plně duplexní spoj mezi oběma konci spojení

Jak se přidělují čísla portů? To závisí na tom, o jaké porty jde. Rozlišujeme totiž dva typy portů:

- ◆ Serverové porty, na kterých servery naslouchají požadavkům klientů. Tyto porty jsou dobře známy klientům (*well known ports*). Pokud by jim totiž známy nebyly, tak by klienti na ně nemohli navázat spojení (nevěděli by, jaký port vyplnit do záhlaví TCP segmentu). Seznam těchto portů najdete na počítači v souboru /etc/hosts, resp. v C:\WINDOWS\system32\drivers\etc\hosts. Aktuální výčerpávající seznam registrovaných dobře známých portů pak naleznete na <http://www.iana.org>.
- ◆ Klientské porty. Klienti si alokují nepoužívané porty větší než 1023. K této alokaci slouží část operačního systému, která se nazývá Správa portů. Klienti používají porty větší než 1023, jelikož porty nižších čísel jsou tzv. privilegované porty určené především pro servery. Název „privilegované“ znamená, že o tyto porty nemůže Správa portů žádat běžný uživatel operačního systému – pouze privilegováný uživatel.

Dobré je si uvědomit, že cílová aplikace je v Internetu adresována (jednoznačně určena) IP adresou, číslem portu a použitým protokolem (TCP nebo UDP). Protokol IP dopraví IP datagram na konkrétní počítač. Na tomto počítači běží jednotlivé aplikace. Podle čísla cílového portu operační systém pozná, které aplikaci má TCP segment doručit.

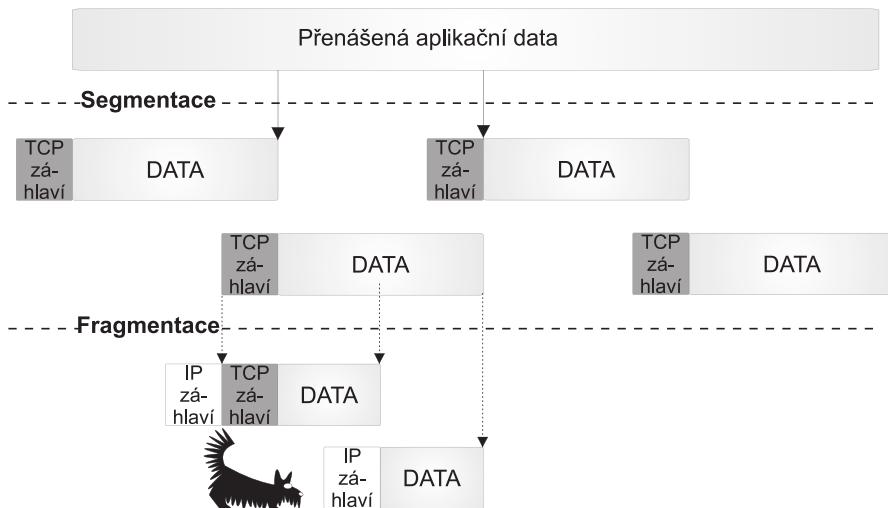
Porty připomínají poštovní schránky v paneláku, jak je znázorněno na obr. 9.2.



**Obrázek 9.2:** Porty TCP a UDP

Základní jednotkou přenosu v protokolu TCP je segment TCP. Někdy se také říká paket TCP. Proč segment? Aplikace běžící na jednom počítači posílá protokolem TCP data aplikaci běžící na jiném počítači. Aplikace potřebuje přenést např. soubor velký 2 GB. Jelikož segmenty TCP jsou baleny do IP datagramů, které mají pole délka dlouhé 16 bitů, tak TCP segment může být dlouhý maximálně 65535 minus délka TCP záhlaví. Přenášené 2 GB dat musí být rozděleny na segmenty, které se vkládají do paketu TCP – přeneseně se pak místo paket TCP říká segment TCP.

TCP segment se vkládá do IP datagramu. IP datagram se vkládá do linkového rámce. Použije-li se příliš velký TCP segment, který se celý vloží do velkého IP datagramu, jenž je větší než maximální velikost přenášeného linkového rámce (MTU), pak IP protokol musí provést fragmentaci IP datagramu (viz obr 9.3).



Obrázek 9.3: Segmentace a fragmentace

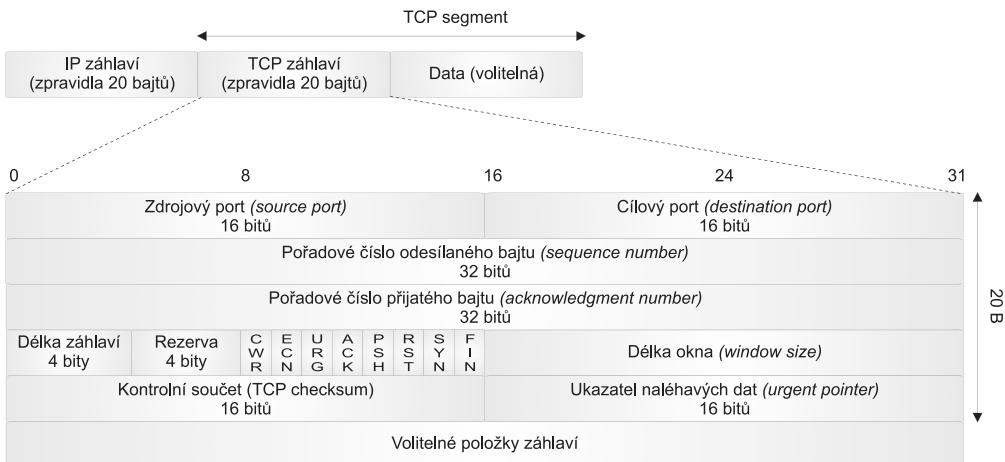
Fragmentace zvyšuje režii, proto je cílem vytvářet segmenty takové velikosti, aby fragmentace nebyla nutná.

## TCP segment

TCP segment má strukturu uvedenou na obr. 9.4.

**Zdrojový port (source port)** je port odesilatele TCP segmentu, **cílový port (destination port)** je portem adresáta TCP segmentu. Pětice: zdrojový port, cílový port, zdrojová IP adresa, cílová IP adresa a protokol (TCP) jednoznačně identifikuje v daném okamžiku spojení v Internetu.

TCP segment je část z toku dat tekoucích od odesilatele k příjemci. **Pořadové číslo odesílaného bajtu** je pořadové číslo prvního bajtu TCP segmentu v toku dat od odesilatele k příjemci (TCP segment nese bajty od **pořadového čísla odesílaného bajtu** až do délky segmentu). Tok dat v opačném směru má samostatné (jiné) číslování svých dat. Jelikož pořadové číslo odesílaného bajtu je 32 bitů dlouhé, tak po dosažení hodnoty  $2^{32}-1$  nabude cyklicky opět hodnoty 0. Číslování obecně nezačíná od nuly (ani od nějaké určené konstanty), ale číslování by mělo začínat od náhodně zvoleného čísla. Vždy

**Obrázek 9.4:** TCP segment

když je nastaven příznak SYN, tak operační systém odesilatele začíná znova číslovat, tj. vygeneruje startovací pořadové číslo odesílaného bajtu, tzv. ISN (*Initial Sequence Number*).

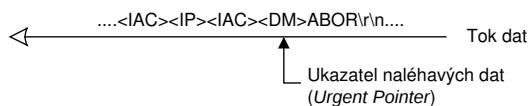
Naopak **pořadové číslo přijatého bajtu** vyjadřuje číslo následujícího bajtu, který je příjemce připraven přijmout, tj. příjemce potvrzuje, že správně přijal vše až do pořadového čísla přijatého bajtu minus jedna.

**Délka záhlaví** vyjadřuje délku záhlaví TCP segmentu v násobcích 32 bitů (4 bajtů) – podobně jako u IP záhlaví.

**Délka okna** vyjadřuje přírůstek pořadového čísla přijatého bajtu, který bude příjemcem ještě akceptován (viz str. 245).

**Ukazatel naléhavých dat** může být nastaven pouze v případě, že je nastaven příznak URG. Přičte-li se tento ukazatel k pořadovému číslu odesílaného bajtu, pak ukazuje na konec úseku naléhavých dat. V některých aplikacích ale ukazuje těsně před úsekem naléhavých dat.

Ukažme si využití ukazatele naléhavých dat na příkladu. Aplikační protokol Telnet přenáší data mezi klientem a serverem. Kromě běžných aplikačních dat může tok dat obsahovat i příkaz IAC, tj. „následující data interpretuj jako příkaz protokolu Telnet“ (*Interpret As Command*). Příkaz IAC začíná znakem IAC (desítkově 255) následovaným příslušným příkazem. Jako příkaz může být např. IP (přeruš proces, desítkově 244).

**Obrázek 9.5:** Naléhavá data

Řídicí kanál protokolu FTP využívá rovněž příkazy protokolu Telnet. Pomocí těchto příkazů může např. klient signalizovat serveru abnormální ukončení přenosu v datovém kanálu. Uživatel takovou signalizaci může provést velice jednoduše např. stlačením ^C, což vygeneruje v řídicím kanálu sekvenci znázorněnou na obr. 9.5.

První tři bajty urgentních dat jsou <IAC><IP><DM>. Ukazatel naléhavých dat pak ukazuje za tato data na příkaz <DM>, tj. *Data Mark*. Právě příkaz <DM> je prázdným příkazem sloužícím k takovému účelu (aby se na něj ukazovalo). Za těmito Telnetovými příkazy pak následuje příkaz protokolu FTP, kterým je příkaz ABOR.

V poli příznaků mohou být nastaveny následující příznaky:

- ◆ **CWR** – Potvrzení přijetí segmentu s nastaveným příznakem *ECN-Echo*. Po přijetí tohoto segmentu příjemce již dále nenastavuje u odesílaných segmentů příznak *ECN-Echo*.
- ◆ **ECE (ECN-Echo)** – Potvrzování přijetí IP datagramu s nastaveným příznakem ECN (viz str. 164). Tento příznak je nastaven u všech odesílaných TCP segmentů do doby, než je přijat segment s nastaveným příznakem CWR.
- ◆ **URG** – TCP segment nese naléhavá data.
- ◆ **ACK** – TCP segment má platné pole „Pořadové číslo přijatého bajtu“ (nastaven ve všech segmentech kromě prvního segmentu, kterým klient navazuje spojení).
- ◆ **PSH** – Zpravidla se používá k signalizaci, že TCP segment nese aplikační data, která má příjemce předávat aplikaci. Použití tohoto příznaku není ustáleno.
- ◆ **RST** – Odmítnutí TCP spojení.
- ◆ **SYN** – Odesílatel začíná s novou sekvencí číslování, tj. TCP segment nese pořadové číslo prvního odesílaného bajtu (ISN).
- ◆ **FIN** – Odesílatel ukončil odesílání dat. Pokud bychom použili přirovnání k práci se souborem, pak příznak FIN odpovídá konci souboru (EOF). Přijetí TCP segmentu s příznakem FIN neznamená, že v opačném směru není dále možný přenos dat. Jelikož protokol TCP vytváří plně duplexní spojení, příznak FIN způsobí jen uzavření přenosu dat v jednom směru. V tomto směru už dále nebudou odesílány TCP segmenty obsahující příznak PSH (nepočítaje v to případné opakování přenosu).

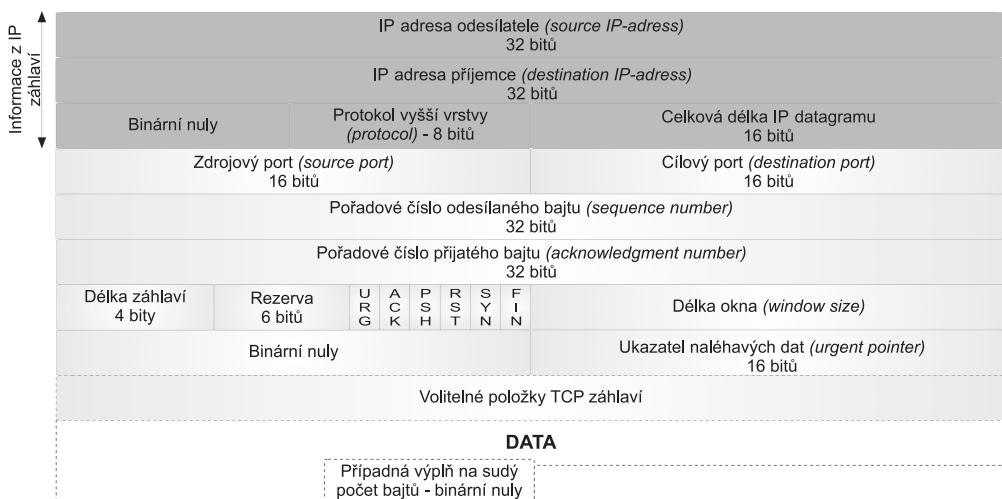


**Poznámka:** Příznaky CWR a ECE jsou novější méně používané příznaky, kterými se protokoly IP a TCP snaží řídit tok dat a bránit tím zahlcení sítě.

V dalším textu budeme kombinaci nastavených příznaků zapisovat podle prvních písmen z názvu příznaku. Pokud není příznak nastaven, pak místo něj napíšeme tečku. Např. skutečnost, že TCP segment má nastaveny příznaky ACK a FIN a ostatní příznaky nenastaveny, zapišeme: .A...F .

**Kontrolní součet** IP záhlaví se počítá pouze ze samotného IP záhlaví. Z hlediska zabezpečení integrity přenášených dat je důležitý kontrolní součet v záhlaví TCP segmentu počítaný i z přenášených dat. Tento kontrolní součet se počítá nejen ze samotného TCP segmentu, ale i z některých položek IP záhlaví. Kontrolní součet vyžaduje sudý počet bajtů, proto v případě lichého počtu se data fiktivně doplní jedním bajtem na konci.

Kontrolní součet se počítá z polí znázorněných na obrázku 9.6. Tato struktura slouží pouze pro výpočet kontrolního součtu – v každém případě se žádná taková struktura nepřenáší Internetem! Někdy se tato struktura označuje jako pseudozáhlaví.



Obrázek 9.6: Pole, ze kterých se počítá kontrolní součet TCP záhlaví

## Volitelné položky TCP záhlaví

Povinné položky TCP záhlaví tvoří 20 B. Za povinnými položkami následují volitelné položky.

Volitelná položka se skládá z typu volitelné položky, délky volitelné položky a hodnoty. Délka záhlaví segmentu TCP musí být dělitelná čtyřmi. V případě, že délka záhlaví by nebyla dělitelná čtyřmi, pak se záhlaví doplňuje prázdnou volitelnou položkou – NOP.

Jelikož pole délka záhlaví je pouze 4 byty dlouhé, tak toto pole může nabývat maximálně hodnoty  $1111_2 = 15_{10}$ . Délka záhlaví se udává v násobcích čtyř, tudíž záhlaví může být dlouhé maximálně  $15 \times 4 = 60$  bajtů. Povinné položky zabere 20 bajtů, takže na volitelné zbývá nejvýše 40 bajtů.

Některé volby záhlaví-TCP včetně jejich struktury jsou uvedeny na obrázku 9.7. Nás bude zajímat zejména volitelná položka Maximum Segment Size (MSS). Pomocí této položky se na počátku spojení obě strany dohodnou na maximální délce segmentu.

## Příklad výpisu TCP segmentu

Následující TCP segment vypsáný programem Wiresherk má nastaveny příznaky ACK, PSH a FIN:

```
Ethernet II, Src: 00:1a:92:ca:e6:b9 (00:1a:92:ca:e6:b9), Dst: 00:13:02:91:2e:8b
(00:13:02:91:2e:8b)
```

```
Internet Protocol, Src: 213.29.7.61 (213.29.7.61), Dst: 10.0.0.1 (10.0.0.1)
```

```
Version: 4
```

```
Header length: 20 bytes
```

```
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
```

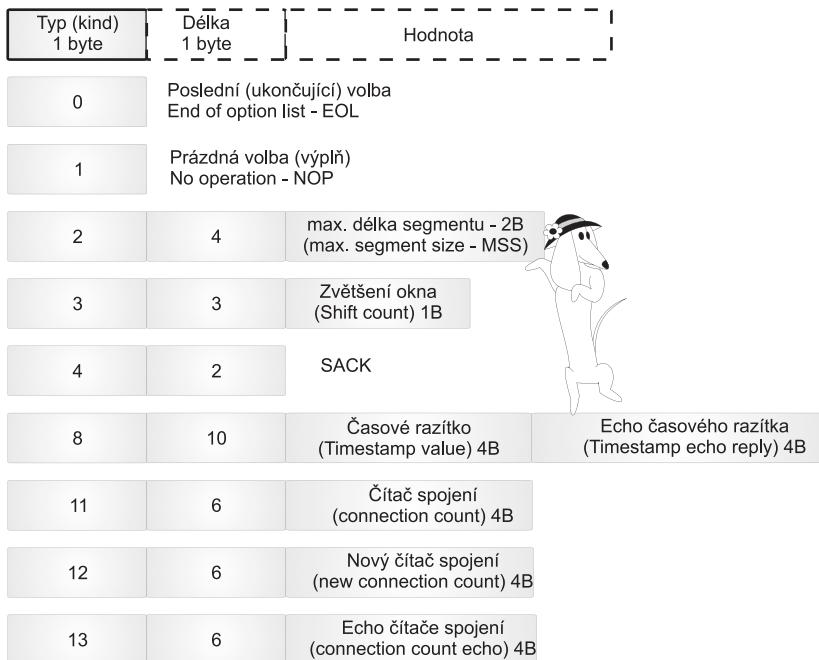
```
0000 00... = Differentiated Services Codepoint: Default (0x00)
```

```
.... ..0. = ECN-Capable Transport (ECT): 0
```

```
.... ...0 = ECN-CE: 0
```

```
Total Length: 1168
```

(Výpis pokračuje na další straně.)



Obrázek 9.7: Některé volitelné položky TCP záhlaví

```

Identification: 0xf248 (62024)
Flags: 0x04 (Don't Fragment)
    0... = Reserved bit: Not set
    .1.. = Don't fragment: Set
    ..0. = More fragments: Not set
Fragment offset: 0
Time to live: 246
Protocol: TCP (0x06)
Header checksum: 0xa7c3 [correct]
Source: 213.29.7.61 (213.29.7.61)
Destination: 10.0.0.1 (10.0.0.1)
Transmission Control Protocol, Src Port: 80 (80), Dst Port: 1384 (1384), Seq: 1474, Ack: 382, Len: 1128
Source port: 80 (80)
Destination port: 1384 (1384)
Sequence number: 1474 (relative sequence number)
[Next sequence number: 2602 (relative sequence number)]
Acknowledgement number: 382 (relative ack number)
Header length: 20 bytes
Flags: 0x19 (FIN, PSH, ACK)
    0... .... = Congestion Window Reduced (CWR): Not set
    .0... .... = ECN-Echo: Not set
    ..0. .... = Urgent: Not set
    ...1 .... = Acknowledgment: Set
  
```

```
.... 1... = Push: Set
.... 0... = Reset: Not set
.... .0. = Syn: Not set
.... ..1 = Fin: Set
Window size: 4161
Checksum: 0xc8cf [correct]
[SEQ/ACK analysis]
Hypertext Transfer Protocol
```



**Poznámka:** Všimněte si položky Sequence numer (pořadové číslo odesílaného bajtu). Za ní je v závorce uvedeno, že se jedná o relativní číslo. Ve skutečnosti v TCP segmentu je na tomto místě ISN (Initial Sequence Number) + pořadové číslo odesílaného bajtu. To rozhodně nebude takto malé číslo (=1474). To jenom program Wireshark pro uživatela pohodlí odečetl od zde uvedené hodnoty právě ISN!

## Navázání a ukončení spojení protokolem TCP

Jádrem protokolu IP byl zejména popis IP datagramu. Jelikož je protokol IP datagramovou (nespojovanou službou), tak se protokol IP příliš nepotřeboval zatěžovat případy, kdy IP datagram není doručen. IP protokol takové stavy může nanejvýš signalizovat protokolem ICMP. Signalizace protokolem ICMP je pouze dobrou vůlí protokolu IP. V praxi se setkáváme s mnoha případy, kdy signalizace pomocí protokolu ICMP je potlačována, protože je nežádoucí – např. z bezpečnostních důvodů.

Protokol TCP využívá k transportu dat Internetem protokol IP, avšak nad tímto protokolem zřizuje spojovanou službu. Musí řešit problémy navázání a ukončení spojení, potvrzování přijatých dat, vyžádání ztracených dat, ale také problémy průchodnosti přenosové cesty. Popis TCP segmentu je tedy jen malou částí TCP protokolu, podstatně rozsáhlejší částí je popis výměny TCP segmentů (*handshaking*) mezi oběma konci TCP spojení.

### Navazování spojení

Protokol TCP umožňuje jedné straně navazovat spojení. Druhá strana spojení buď akceptuje, nebo odmítne. Z hlediska aplikační vrstvy bude stranou navazující spojení klient a server ta strana, která spojení očekává. Existují i jiné aplikační modely než klient-server, kterým protokol TCP zabezpečuje přenos dat, přesto budeme v zájmu usnadnění výkladu pojmem klient používat pro stranu iniciující spojení a termín server pro stranu očekávající spojení.

Zajímavostí je, že protokol TCP umožňuje, aby obě strany navazovaly spojení i současně. V praxi jsme se však s využitím této možnosti nesetkali, proto nadále budeme popisovat jen případ, kdy jedna strana navazuje spojení, které druhá strana očekává.

Představme si, že náš klient si přeje navázat spojení např. se serverem běžícím na počítači „Server“ na portu 23 (v praxi bychom napsali, že klient si přeje navázat spojení se serverem – „Server:23“). Klient pro spojení použije klientský port 18475. Zatímco port serveru musí být předem všeobecně znám (aby o něm klienti věděli), tak s portem klienta je to zpravidla jinak. Je sice také možné, aby klient vždy používal pevný port (v našem případě 18475), běžnější však je, že klient na konkrétním čísle portu netrvá. Požádá správu volných portů svého operačního systému o přidělení nějakého

volného portu na dobu spojení. Předpokládejme, že operační systém klienta klientovi přidělil port 18475. Operační systém takto přiděluje porty větší než 1023. Tyto porty se označují jako klientské či neprivilegované, na rozdíl od portů menších než 1024, o jejichž použití může žádat pouze privilegovaný uživatel (např. v operačním systému UNIX uživatel root). Pro nás výklad je však jedno, zdali klient trval na přidělení konkrétního portu 18475 nebo chtěl nějaký port a operační systém mu přidělil zrovna port 18475.

Windows 2000 standardně přiděluje porty pouze do 5000. Pokud chceme toto číslo změnit, pak jej musíme uvést v klíči MaxUserPort registru HKEY\_LOCAL\_MACHINE\ SYSTÉM\ CurrentControlSet\ Services\ Tcpip\ Parameters. Tento klíč je typu REG\_DWORD. Tento klíč platí i pro porty protokolu UDP.

Porty menší než 1024 se označují jako neprivilegované. Tyto porty používají nejčastěji servery, protože servery zpravidla spouští privilegovaný uživatel nebo jsou spouštěny při startu operačního systému, jako by je spouštěl privilegovaný uživatel. Nás server používá privilegový port 23 (23<1024). Takový server může spustit pouze privilegovaný uživatel operačního systému. (Pochopitelně pokud je příslušný port volný, tj. nemá jej alokován jiný proces.)

Nyní se vraťme k našemu klientovi. Klient začíná navazovat spojení odesláním prvního TCP segmentu ① (viz obr. 9.8), kde je port odesilatele 18475 a port příjemce 23. Tento TCP segment klient vloží do IP datagramu, jehož IP adresa odesilatele bude „Klient“ a IP adresa příjemce „Server“. To je vcelku jasné. Jak však budou vypadat ostatní pole TCP segmentu?

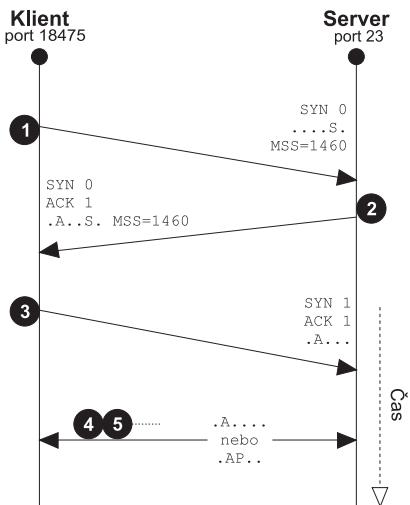
Klient vygeneruje náhodné číslo v intervalu 0 až  $2^{32}-1$ , které použije jako startovací pořadové číslo odesílaného bajtu (tzv. ISN). V našem případě bylo vytvořeno ISN=145165778. Na obr. 9.8 však není uvedeno přímo ISN, ale programem Wireshark vypisované jen relativní číslo (tj. 0).

Skutečnost, že klient právě vytvořil startovací pořadové číslo odesílaného bajtu, vyznačí v TCP segmentu nastavením příznaku SYN (...S.). Během spojení již pořadové číslo odesílaného bajtu bude vždy vyjadřovat číslo odesílaného bajtu, tj. nemůže být znova vygenerováno. To klient nemůže v žádném dalším TCP segmentu během tohoto spojení nastavit příznak SYN. Segment ① je prvním segmentem v TCP komunikaci, proto nemůže potvrzovat žádná přijatá data. Pole pořadové číslo přijatého bajtu nemá platný význam (bývá vyplňeno binárními nulami), a nemůže být tedy ani nastaven příznak ACK (příznak ACK je nastaven ve všech dalších TCP segmentech až do ukončení spojení). TCP segment s nastaveným příznakem SYN a nenastaveným příznakem ACK je velice zvláštním segmentem. Tato kombinace nastaveného příznaku SYN a nenastaveného příznaku ACK je specifická pro první TCP segment spojení. Pokud se chce klientům zamezit v navázání spojení nějakým směrem, pak stačí v tomto směru odfiltrovat všechny TCP segmenty s nastaveným příznakem SYN a klient (útočník) nemá šanci. Tento mechanismus se též často využívá pro ochranu intranetů od Internetu.



**Upozornění:** Program Wireshark nevypisuje ISN, ale jen relativní číslo! Proto i na obr. 9.8 začínáme jakoby od 0.

Součástí segmentů ① a ② byla volitelná položka záhlaví TCP segmentu MSS (*Maximum segment size*). Tato položka oznamuje druhé straně maximální délku datové části TCP segmentu, jakou si přeje přijímat (aby se pokud možno zamezilo fragmentaci IP datagramu). Tato volba se může vyskytovat pouze v TCP segmentech s nastaveným příznakem SYN (tj. v prvních dvou segmentech).



Obrázek 9.8: Navazování spojení

Implicitně se MSS používá 536 bajtů. Tato hodnota je používána pro spojení mimo lokální síť (přes WAN). Pro nás příklad jsme použili spojení v rámci lokální sítě protokolem Ethernet II. Pro linkový protokol Ethernet II je maximální délka datové části rámce rovna 1500. Pokud od 1500 odečteme 20 pro IP záhlaví a dalších 20 pro TCP záhlaví, pak dojdeme k hodnotě z našeho příkladu (1460).

Pokud bychom na linkové vrstvě použili Ethernet ISO 8802-3, pak bychom museli ještě odečíst další 3 bajty na záhlaví podle ISO 8802-2 a dalších 5 bajtů na záhlaví protokolu SNAP, takže bychom mohli nabízet pouze MSS=1452.

Na obr. 9.8 si všimněte, že segment ② sice žádná data nemůže potvrzovat, protože segment ① žádná data nenesl, ale ve skutečnosti potvrzuje přijetí jednoho bajtu. Je to taková zvláštnost. Jako kdyby příznak SYN měl jeden bajt. Podobně se potvrzuje i příznak FIN.

Výpis navazování spojení v programu Wireshark (vypisujeme pouze sumární řádek protokolu TCP):

#### ① segment Klient → Server

TCP 18475 > 23 TCP [SYN] Seq=0 len=0 MSS=1460

#### ② segment Server → Klient

TCP 23 > 18475 [SYN, ACK] Seq=0 Ack=1 len=0 MSS=1460

Jak jsme se již zmínili, druhý segment potvrzuje jakoby jeden znak – potvrzuje příznak SYN.

Od druhého segmentu budou všechny další segmenty potvrzovat přijatá data, tj. budou mít nastaven příznak ACK!



**Poznámka:** Skutečnost, že pouze první segment celého TCP spojení nemá nastaven příznak ACK a má nastaven příznak SYN, se využívá při ochraně sítě pomocí filtrace (viz kap. 18).

**③ segment Klient → Server**

TCP 18475 > 23 TCP [SYN] Seq=1 Ack=1 len=0 MSS=1460

Třetí segment rovněž potvrzuje příznak SYN, takže potvrzuje jakoby jeden bajt. Třetí a další segment již nemůže nést volitelnou položku záhlaví MSS (maximální délka segmentu).

Třetím segmentem končí navazování spojení. Někdy se proto také „česky“ říká, že protokol TCP potřebuje pro navázání spojení „trifázový handshaking“ (pro ukončení spojení pak „čtyřfázový“).

**④ segment Klient → Server atd.**

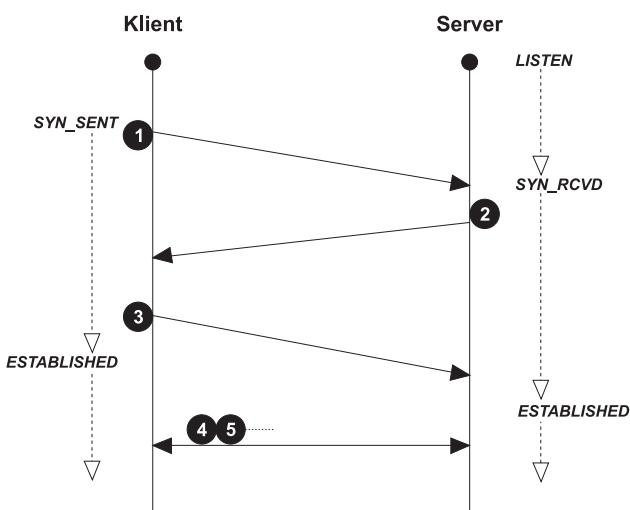
Až jste překvapeni, že čtvrtý segment posílá klient serveru – určitě jste očekávali, že čtvrtý segment pošle server klientovi. Ono je to jedno, jakmile kterákoli strana obdrží první ACK, může začít s vysíláním. Vždyť TCP je plně duplexní spoj. A náš klient byl už netrpělivý z toho, kolik má dat k odeslání. Skutečnost, že TCP segment nese aplikační data, je vyjádřena nastavením příznaku PSH.

Ne všechny TCP segmenty musí nutně nést aplikační data, tj. mít nastaven příznak PSH (.AP...). Může se stát, že jeden konec spojení odesílá data, avšak druhý konec nemá momentálně žádná data k odeslání. I když druhý konec nemá co posílat, musí potvrzovat přijatá data. Takové potvrzování provádí TCP segmenty s nenastaveným příznakem PSH (.A...), tj. segmenty bez dat.

V každém okamžiku spojení je spojení v tzv. stavu. Při navazování spojení může být:

- ◆ Server ve stavu:
  - LISTEN – server je připraven na spojení s klienty.
  - SYN\_RCVD – server přijal od klienta první TCP segment, tj. segment s příznakem SYN.
- ◆ Klient ve stavu:
  - SYN\_SENT – klient odesílá první TCP segment, tj. segment s příznakem SYN.

Pokud se spojení naváže, pak klient i server přecházejí do stavu ESTABLISHED, tj. spojení navázáno. V tomto stavu si mohou oba konec současně předávat data.



**Obrázek 9.9:** Stavy při navazování spojení

Spojení a jejich stavy snadno vypíšete na počítači (ať se jedná o UNIX či Windows) příkazem:

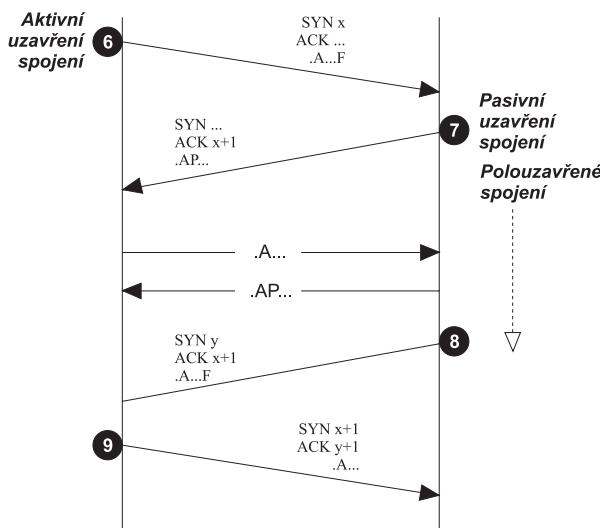
```
netstat -a
```

## Ukončování spojení

Zatímco spojení navazoval v architektuře klient-server zpravidla klient, ukončit spojení může libovolná strana. Strana, která první odešle TCP segment s příznakem FIN (ukončení spojení), provádí tzv. aktivní ukončení spojení (*active close*), druhé straně nezbývá než provést pasivní ukončení spojení (*passive close*). Teoreticky je možné i současné ukončení spojení.

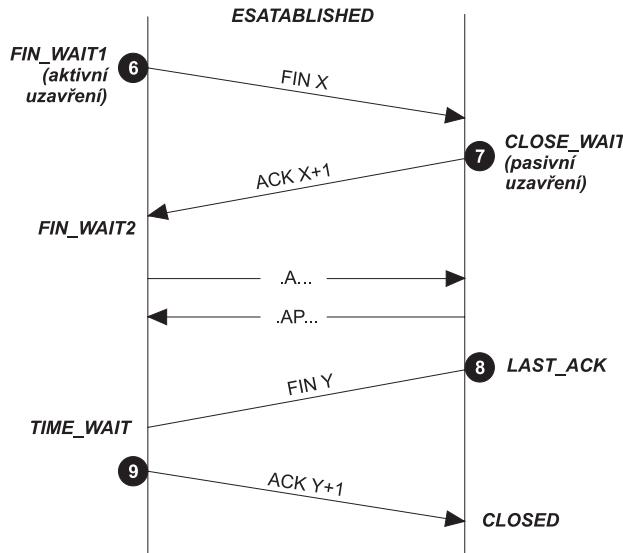
Provede-li jedna strana aktivní ukončení spojení, pak již nemůže odesílat data (nemůže odeslat segment TCP s příznakem PSH). Druhá strana však může v odesílání dat pokračovat až do té doby, dokud neprovede sama ukončení spojení. Mezidobí od aktivního ukončení spojení do ukončení spojení nazýváme polouzavřeným spojením (*half close*). TCP segment s příznakem FIN je obdobou konce souboru (EOF).

Pro řádné uzavření spojení jsou nutné čtyři TCP segmenty. Příznak FIN se opět jako příznak SYN při navazování spojení potvrzuje, jako by zabíral 1 bajt dat.



Obrázek 9.10: Ukončování spojení

Segment 6 startuje aktivní uzavření spojení nastavením příznaku FIN. Segment 7 potvrzuje uzavření spojení druhou stranou, tj. provádí pasivní uzavření spojení. Většinou segment 7 obsahuje též příznak FIN a dochází tím ke startu uzavření celého spojení. Nás obrázek však znázorňuje obecný případ (využívaný např. programem rsh), kdy segment 7 příznak FIN neobsahuje, protože tato strana chce pokračovat ve spojení, tj. chce použít tzv. **polouzavřené spojení** pro přenos aplikačních dat. Strana, která spojení uzavřela, již nemůže odesílat žádná data (jí odesílané segmenty nemohou obsahovat příznak PSH).



Obrázek 9.11: Stavy při uzavírání spojení

Stavy při ukončování spojení:

- ◆ **FIN\_WAIT1** – strana zjistila, že již všechna data odeslala (a potřebuje signalizovat konec souboru – EOF), tak v TCP segmentu nastaví příznak FIN, čímž signalizuje aktivní uzavření spojení segmentem ⑥.
- ◆ **CLOSE\_WAIT** – druhá strana obdržela aktivní uzavření spojení a nezbývá jí nic jiného než potvrdit segmentem ⑦ přechod do pasivního uzavření spojení, kterému odpovídá stav **CLOSE\_WAIT**.
- ◆ **FIN\_WAIT2** je stav poté, co strana obdrží segmentem ⑦ potvrzení aktivního uzavření spojení od protějšku. Ve stavu **FIN\_WAIT2** strana zůstává do té doby, dokud protějšek nezašle TCP segment s příznakem FIN, tj. do přechodu do stavu **TIME\_WAIT**. Pokud aplikace nepočítá s přenosem dat v polouzavřeném spojení a spojení je nečinné po 11,25 minuty, pak operační systém automaticky změní stav spojení na **CLOSED**.
- ◆ **LAST\_ACK** – druhá strana již odeslala všechna data a signalizuje úplné ukončení spojení segmentem ⑧.
- ◆ **TIME\_WAIT** – všechna data oběma směry již byla přenesena. Je nutné pouze potvrdit úplné uzavření spojení. Odesláním TCP segmentu ⑨ je potvrzeno úplné ukončení spojení. Tento segment již není potvrzován, proto strana zůstává ve stavu **TIME\_WAIT** po dobu 2 minut (některé implementace TCP/IP zkracují tuto dobu až na 30 s). Tato doba by totiž měla přibližně odpovídat dvojnásobku doby života TCP segmentu v síti. Důvod je prostý: segment ⑨ se může v síti ztratit a druhá strana si může vyžádat jeho opakování. Pokud by však spojení bylo uzavřeno, pak by opakování již nebylo možné.
- ◆ **CLOSED** – druhá strana obdržela potvrzení úplného uzavření spojení a přechází do stavu **CLOSED**. Strana, která odeslala segment ⑨, přechází do stavu **CLOSED** až po uplynutí zmíněného intervalu.

## Odmítnutí spojení

Spojení se odmítá nastavením příznaku RST (*Reset*) v záhlaví TCP segmentu.

Spojení je odmítáno v zásadě ve dvou případech:

- ◆ Klient požaduje spojení se serverem na portu, na kterém žádný server neběží. To je rozdíl oproti protokolu UDP. Pokud je zaslán UDP datagram na port, kde neběží žádný server, pak systém odpoví ICMP zprávou nedosažitelný port.
- ◆ Druhým případem je situace, kdy je odmítnuto dále pokračovat v již navázaném spojení. Zde lze rozlišit také dva případy:
  - Jedna z komunikujících stran zjistí, že protějšek je nedůvěryhodný, pak okamžitě ukončuje spojení. To je případ například protokolu SSL. Protokol SSL zabezpečuje bezpečnou komunikaci (šifrovanou a autentizovanou). Pokud se nedokáže jedna strana autentizovat nebo použít tak silné kryptografické algoritmy, jak vyžaduje druhá strana, je spojení okamžitě ukončeno nastavením příznaku RST.
  - Řádné ukončení spojení je poměrně dlouhou záležitostí (např. aplikace je nucena posečkat ve stavu TIME\_WAIT). Aplikace si po odeslání všech dat přeje ukončit spojení rychleji – použije odmítnutí spojení. V praxi se setkáváme s tím, že buď místo segmentu ❸ (obr. 9.10) je odeslán segment s nastaveným příznakem RST, nebo po segmentu ❸ následuje ještě potvrzení segmentu ❸ pomocí TCP segmentu s nastaveným příznakem RST. Takovýmto způsobem je však možné ukončit spojení, až si obě strany vymění všechna data.

## Zjištění stavu spojení

Výpis všech spojení protokoly TCP a UDP lze získat pomocí příkazu netstat s parametrem -a.

```
$ netstat -an
```

```
Active Internet connections (including servers)
```

Proto	Recv-Q	Send-Q	Local Address	Foreign Address	(state)
tcp	0	0	194.149.105.18.22	194.149.103.204.24695	TIME_WAIT
tcp	0	0	194.149.105.18.3099	194.108.145.128.25	SYN_SENT
tcp	0	34472	194.149.105.18.3079	195.47.32.245.25	ESTABLISHED
tcp	0	0	*.22	*.*	LISTEN
tcp	0	0	*.25	*.*	LISTEN
tcp	0	0	*.53	*.*	LISTEN
udp	0	0	*.53	*.*	
udp	0	0	127.0.0.1.53	*.*	

První dva řádky tvoří záhlaví výpisu. Význam jednotlivých sloupců:

- ◆ Sloupec Proto obsahuje název použitého protokolu (TCP nebo UDP).
- ◆ Sloupec Recv-Q vyjadřuje počet bajtů ve vstupní frontě spojení (čekajících na zpracování aplikací).
- ◆ Sloupec Send-Q vyjadřuje počet bajtů ve výstupní frontě (čekajících na odeslání).
- ◆ Sloupec Local Address obsahuje adresu lokálního síťového rozhraní tečkou odděleného od čísla lokálního portu. Servery čekající na spojení mohou mít na místo IP adresy uvedenu hvězdičku. Hvězdička označuje, že server očekává spojení na všech svých síťových rozhraních.

- ◆ Sloupec `Foreign Address` obsahuje IP adresu a port vzdáleného konce spojení. Hvězdičky vyznačují, že server očekává spojení z libovolné IP adresy a libovolného portu.
- ◆ Sloupec `state` obsahuje stav spojení.

Pro servery mohou nastat následující kombinace:

V předchozím příkladu rádek

```
tcp      0      0  194.149.105.18.22      194.149.103.204.24695  TIME_WAIT
```

vyjadřuje, že server na lokálním počítači běžící na portu 22 (tj. program sshd) potvrdil úplné ukončení spojení (TIME\_WAIT) s počítačem o IP adrese 194.149.103.204. Klientovi byl pro toto spojení přiřazen port 24695.

Řádek:

```
tcp      0      0  *.53                  *.*                  LISTEN
```

vyjadřuje, že na lokálním počítači na jeho všech síťových rozhraních čeká server na portu 53 (tj. program named) na spojení s libovolným klientem.

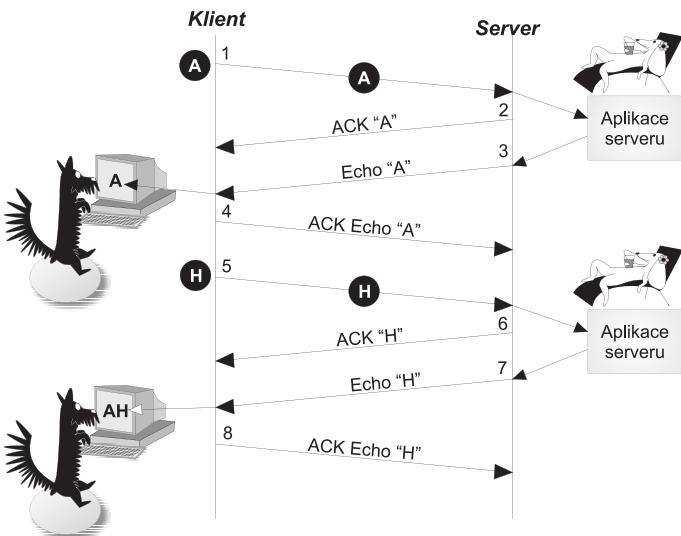
## Technika zpoždění odpovědi

Interaktivní aplikace jako Telnet či příkazový kanál FTP jsou komplikované tím, že jsou interaktivní. Tzn., že zmáčknete-li na klávesnici klávesu A, znak A se zabalí do TCP segmentu ( $20+1=21$  B), TCP segment se vloží do IP datagramu ( $20+21=41$  B) a tento IP datagram pak putuje Internetem jako segment (1) na obr. 9.12, až doputuje na server (vše se pochopitelně ještě vkládá do linkových rámů od směrovače ke směrovači). Server:

- ◆ Jednak potvrdí příjem znaku, tj. pokud nemá žádná data k odeslání, tak vyšle nedatový segment (40 B) – segment (2).
- ◆ Jednak předá znak A ke zpracování serveru, server musí vyslat znak A zpět – segment (3), aby klientův software zobrazil znak A na obrazovce klienta (provedl echo). Echo je nutné právě pro subjektivní pocit interaktivnosti aplikace. Klient přijme echo (znak A) a zobrazí jej na obrazovce. Pak potvrdí příjem echa serveru segmentem (4). Pokud nemá žádná další data k odeslání, pak potvrzení provede nedatovým TCP segmentem.

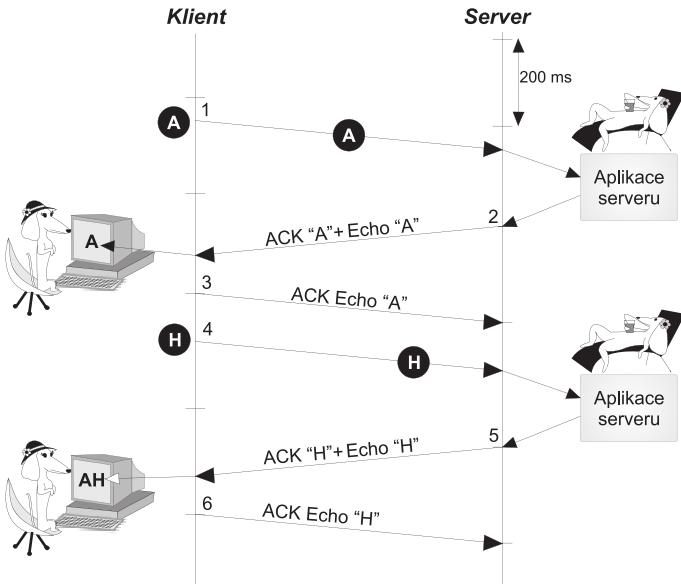
Pokud by takto aplikace pracovala, pak zmáčknutí jedné klávesy znamená přenést 81 B v každém směru (nezapočítávaje režii linkové vrstvy). 81 B se skládá z 41 B při odeslání znaku a 40 B při jejich potvrzení. Na následujícím obrázku je tato situace vyznačena pro stisk kláves A a H (začátek řetězce AHOJ).

Je vcelku pochopitelné, že je snaha objem přenášených dat zmenšit, čímž se snažíme zabránit ucpání přenosových cest. Myšlenka spočívá v tom, že potvrzování přijatých dat nebude probíhat okamžitě, ale se zpožděním. Během tohoto zpoždění se pak mohou objevit i další data k přenosu.

**Obrázek 9.12:** Stisk kláves A a H

Princip spočívá v tom, že operační systém spustí pro tento účel hodiny zpravidla s tikem 200 ms (délka tiku musí být pod 500 ms). Po každém tiku systém zkontroluje, zdali není něco k odeslání (potvrzení přijatých dat či odeslání dat). Pokud je třeba něco odeslat, pak vše odešle najednou.

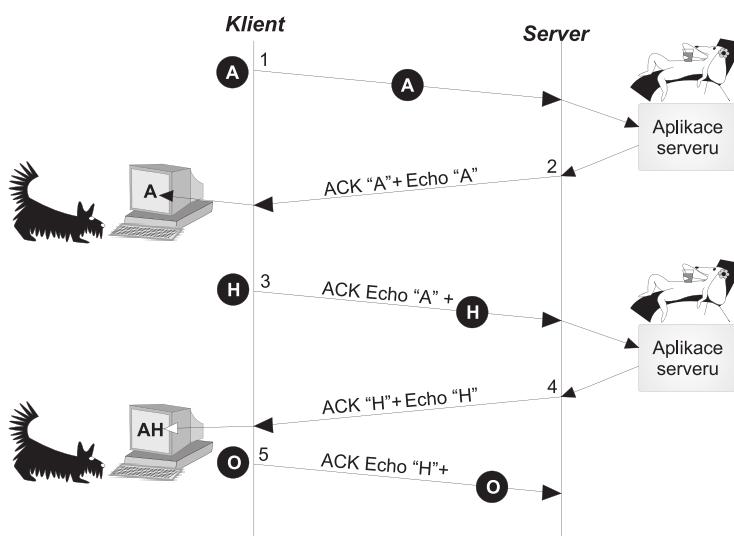
Na obrázku 9.13 server přijal znak A – segment (1), ale bude odpovídat až s dalším 200ms tikem. Do té doby aplikace stihne i připravit data k odeslání (echo A). Operační systém jedním TCP segmentem (segment (2)) provede potvrzení znaku A a zároveň odesle echo A.

**Obrázek 9.13:** Zpoždění 200 ms

Jenže je již málo pravděpodobné, že klient stiskne další klávesu H tak, aby software klienta byl schopen znak H odeslat společně s potvrzením přijetí echa klávesy A. Proto, jak je z následujícího obrázku 9.13 patrné, klient provede potvrzení echa klávesy A pomocí segmentu (3) a stisk klávesy H způsobí vyslání dalšího TCP segmentu (4).

V ideálním případě dojde k redukci přenášených TCP segmentů ze čtyř na tři. Ještě větší úspory přinese použití Nagleova algoritmu znázorněného na obrázku 9.14.

V tomto případě software klienta nečeká na další tik (200 ms), ale čeká, až dojdou nějaká data od protější strany. Tento algoritmus vyrovnává dobu odezvy vůči kapacitě přenosové linky (řídí tok dat). To znamená, že je-li linka více zatížena, pak je na ní delší odezva a odpověď se také pozdrží.



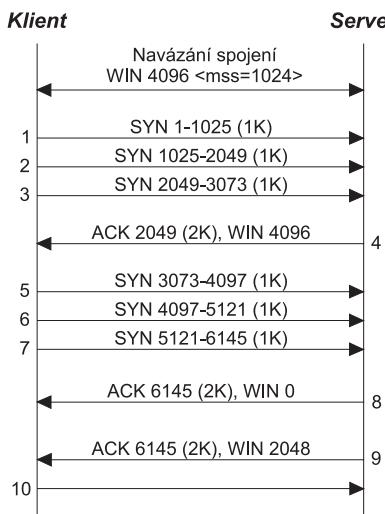
**Obrázek 9.14:** Nagleův algoritmus

Technika zpozdění odpovědi je výhodná pro aplikace typu telnet, ale např. pro X-server je naopak nežádoucí. Pokud bychom ji použili pro X-server, pak pohyb myši by se nám na obrazovce mohl jevit trhaný. Pro aplikace přenášející velké objemy dat (např. HTTP, datový kanál FTP atd.) technika pozdržení odpovědi také ztrácí smysl.

## Technika okna

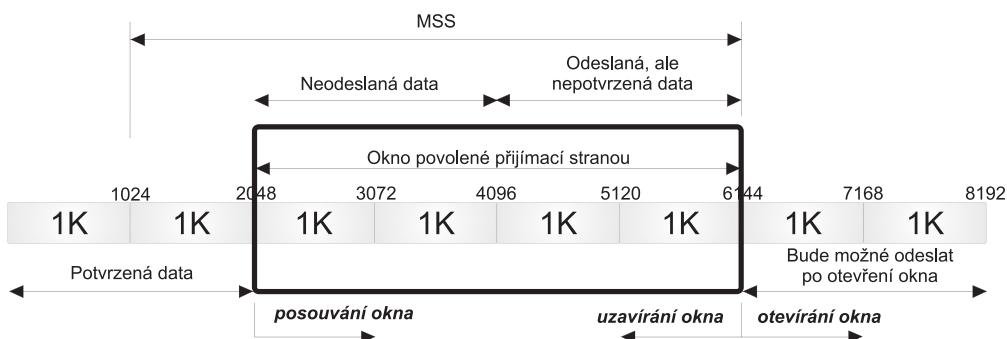
Nyní je naším problémem případ, kdy klient potřebuje odeslat velké množství dat. Klient (resp. Server) může odesílat data druhé straně, aniž by jejich příjem měl potvrzen až do tzv. okna (*Window* – zkratkou WIN).

Představme si (obr. 9.15), že klient se serverem navázal spojení a vzájemně se dohodli na maximální velikosti segmentu (MSS) o velikosti 1 K (tj. 1 024 B) a vzájemně velikosti okna 4 K (tj. 4 096 B).

**Obrázek 9.15:** Technika okna

Klient začne s odesíláním dat a odešle segmenty 1, 2 a 3. Poté obdrží od serveru potvrzení (4), které potvrzuje segmenty 1 a 2. Klient vzápětí odesílá segmenty 5, 6 a 7. Jenže server data mezičím nedokázal zpracovat a data mu zaplnila vyrovnávací paměť, proto segmentem 8 sice potvrdí příjem segmentů 3, 5, 6 a 7, ale zároveň klientovi uzavře okno, tj. klient nemůže s odesíláním dat pokračovat. Poté co server zpracuje část dat (2 KB), umožní klientovi pokračovat v odesílání, ale neotevře mu segmentem 9 okno celé – pouze 2 KB, protože všechna data ve vyrovnávací paměti ještě nezpracoval a pro více dat nemá místo.

Prozkoumejme na obr. 9.16, jak vidí klient okno po odeslání segmentu 4 (krok 5 na obr. 9.15).

**Obrázek 9.16:** Okno

První 2 KB jsou již potvrzeny, okno je tedy posunuto za bajt 2048. Tato potvrzená data již klient nemusí udržovat v paměti. Odeslaná, ale nepotvrzená data (segmenty 3 a 4) tvoří 2 KB. Klient může tedy odeslat bez dalšího potvrzení 3 KB dat.

Postupně jak jsou data odesílána, tak se okno pohybuje po odesílaných datech.

## Zahlcení sítě

Okno je množství dat, které je příjemce schopen přijmout. Okno je také příjemcem inzerováno. Problém je však také na straně odesilatele. Pokud je odesilatel na rychlé síti a příjemce na pomalé síti, tak by odesilatel mohl doslova nacpat do sítě data až do velikosti okna (předpokládáme, že velikost okna by byla shodou okolností značně velká). Jelikož síť by nebyla schopna toto velké množství dat přenést, došlo by k zahlcení sítě a data, která síť není schopna přenést, zahodí. Směrovače ukládají IP datagramy do vyrovnavací paměti, ale i ta je omezená.



Obrázek 9.17: Zahlcení

Ztráta dat je vždy nepříjemná. Snahou je se ztrátě dat vyhnout. Na straně odesilatele se proto také zavádí okno. Toto okno se snaží specifikovat, kolik může odesilatel odeslat nepotvrzených dat, aniž by došlo k zahlcení sítě. Toto okno na straně odesilatele se anglicky nazývá *Congestion Window* (zkratkou CWND). Odesilatel postupně zvyšuje CWND. CWND však nelze zvyšovat neomezeně. Hranice, za kterou už je větší pravděpodobnost zahlcení, se označuje STHRESH. Internet však budeme chtít využívat maximálně, tj. budeme chtít najít největší použitelné CWND (to bude někde nad STHRESH). SSHTRESH má smysl udržovat pouze v násobcích velikosti odesílaného segmentu (MSS).

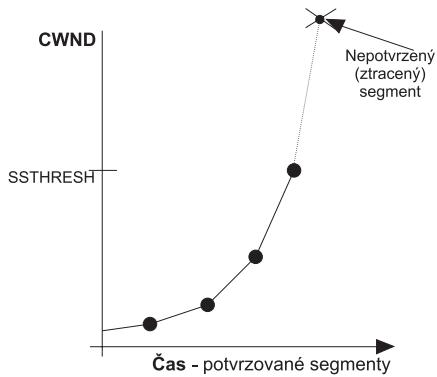
Odesilatel odesílá vždy jen takové množství nepotvrzených dat, které nepřevyšuje okno inzerované příjemcem (WINDOW) ani nepřevyšuje CWND, tj. odesílá jen nejvýše  $\min(\text{WINDOW}, \text{CWND})$  nepotvrzených dat.

## Pomalý start

Otázkou je, jak určit maximální možné CWND. To odesilatel určuje dynamicky. Nejprve odešle jeden segment a výčká na jeho potvrzení. Pokud potvrzení obdrží, pak vyšle dva segmenty. Pokud potvrzení obdrží, pak odešle čtyři atd. Jedná se o řadu  $2^n$  (která je exponenciální).

Je pochopitelné že po několika krocích se odesilatel dostane do situace, kdy síť zahltí a potvrzení nedostane, tj. musí opakovat odesílání segmentů, protože se segment ztratil. V takovém okamžiku se CWND zmenší na polovinu a tato hodnota se také nastaví do veličiny STHRESH (pokud by SSTRESH mělo být menší než dva segmenty, pak se nastaví na velikost dvou segmentů).

Je třeba rozlišovat, jakým způsobem byl zjištěn stav, že segment nebyl potvrzen. Nyní jsme předpokládali, že se segment po cestě k příjemci ztratil. Příjemce segment nedostal, a tak stále potvrzuje předchozí (přijatý) segment. Po třech zopakovaných potvrzeních předchozího přijatého segmentu se segment považuje za ztracený a odesilatel jej opakuje. Může však nastat situace, že odesilatel ve stanoveném časovém intervalu nedostane vůbec žádné potvrzení (ani žádného předchozího segmentu). V takovém případě se CWND nastaví na velikost jednoho segmentu (MSS) a STHRESH na dvojnásobek velikosti segmentu (2x MSS) a začne se s pomalým startem od počátku.



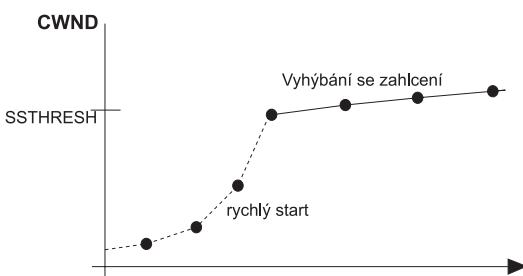
Obrázek 9.18: Pomalý start

### Vyhýbání se zahlcení

Odesílatel udržuje pro každé spojení aktuální hodnoty proměnných MSS, WINDOW, CWND i SSTHRESH. MSS je určeno příjemcem v okamžiku navazování spojení (segment s touto volbou má příznak SYN). WINDOW určuje příjemce dynamicky během spojení – specifikuje množství dat, která se mu vejdu do vyrovnávací paměti.

Odesílatel v okamžiku odesílání segmentu nemůže meditovat nad tím, co má udělat, ale musí se rozhodnout rychle na základě udržovaných proměnných:

1. Když je CWND menší nebo rovno SSTHRESH, pak se jedná o pomalý start. Je tedy možné se pokusit o odeslání dvojnásobku dat.
2. V případě, že CWND je již větší než STHRESH, pak odeslání dvojnásobku by již pravděpodobně způsobilo zahlcení. V tomto případě se CWND zvyšuje pouze o  $(MSS \times MSS / CWND + MSS / 8)$  počítáno v celočíselné aritmetice. Toto „drobné zvětšování“ CWND se nazývá algoritmus vyhýbání se zahlcení (*Congestion Avoidance Algorithm*).



Obrázek 9.19: Vyhýbání se zahlcení

### Ztráta segmentu

Ztrátě TCP segmentu nezamezí ani uvedený algoritmus. Ke ztrátě může dojít i změnou situace na přenosových cestách, poruchou přenosové cesty atd.

V případě, že CWND je značně velké (může být řádově i v MB, či dokonce GB), pak opakování celého nepotvrzeného okna v případě ztráty jednoho segmentu by bylo velice nepřijemné, protože by enormně zvyšovalo režii. Je proto snaha využít tzv. algoritmus rychlého zopakování.

Jak však odesilatel zjistí, že došlo ke ztrátě TCP segmentu? (Nyní se již nebudeme zmiňovat o případu, že odesilatel nedostává od příjemce potvrzováno vůbec nic.) Příjemce zjistí, že došlo ke ztrátě segmentu tím, že dostává další segmenty (a ztracený segment stále nedochází) – tj. chybí mu data v posloupnosti přijímaných dat. Přichází mu tedy segmenty mimo pořadí. Příjemce je povinen v případě přijetí segmentu mimo pořadí zopakovat (duplikovat) potvrzení posledního správně přijatého segmentu.

Jenž TCP segmenty jsou zabaleny do IP datagramu. Každý IP datagram putuje Internetem samostatně a teoreticky jinou cestou. Některé cesty jsou rychlejší a jiné pomalejší. Může se stát, že segment putoval pomalejší cestou a přirozeně dorazil později než následující segment. Mezitím však příjemce již provedl odeslání duplikovaného potvrzení.

Přijetí jedné duplikované odpovědi je proto bráno vcelku za běžnou záležitost. Jiná je situace, když se segment opravdu ztratil. Příjemce pak segment nedostal a dostał následující segment. provedl duplikaci potvrzení posledního správně přijatého segmentu. Poté však dostane ještě následující segment, ten je však také mimo pořadí, protože příjemce ještě neobdržel ztracený segment. Opět provede duplikaci. Příjemce obdrží opět další segment, který je rovněž mimo pořadí – zopakuje duplikaci atd.

Odesilatel pak od příjemce postupně dostává první duplikát, druhý duplikát a stále si říká, že to je vcelku normální. Po obdržení třetího duplikátu si však řekne: „To už se opravdu muselo něco stát, zopakuji mu segment, o kterém se příjemce domnívá, že je ztracen“ a provede zopakování ztraceného segmentu. Příjemce obdrží ztracený segment. Avšak jelikož příjemce nezahodil žádný z následujících segmentů (původně přijatých mimo pořadí), tak potvrdí přijetí všech přijatých segmentů.

Tento algoritmus rychlého zopakování umožňuje zopakovat pouze ztracený segment, a nikoliv nutnost opakování všech nepotvrzených dat. Opakování všech nepotvrzených dat je nutné pouze v případě, že se algoritmus rychlého opakování nestihne v zadaném časovém intervalu.

Jak je to v případě rychlého zopakování s proměnnými CWND a SSTHRESH?

1. Když příjemce obdrží třetí duplikát, pak:
  - a) nastaví SSHTRESH na CWND/2,
  - b) zopakuje odeslání TCP segmentu,
  - c) nastaví CWND na SSHTRESH+3xMSS.
2. Po přijetí čtvrtého a dalšího duplikátu odesilatel vždy zvětší CWND o velikost segmentu (MSS).
3. V případě, že ztracený segment je konečně potvrzen příjemcem, odesilatel nastaví CWND na hodnotu SSTHRESH.

## Volba zvětšení okna

Okno inzerované příjemcem má v TCP záhlaví vyhrazeny 2 B. Příjemce proto může inzerovat okna pouze v rozmezí 0 až 65535. Taková okna jsou u gigabitových sítí příliš malá. Řešením je použití volitelné položky „zvětšení okna“ v záhlaví TCP segmentu. Tato volba může být použita pouze v segmentech inicializujících spojení, tj. v segmentech s příznakem SYN.

Pomocí volitelné položky „zvětšení okna“ se oba konce spojení dohodnou na zvětšení okna v intervalu 0 až 14. Označme toto zvětšení jako  $n$ . V každém směru spojení může být dohoda jiná.

Hodnota zvětšení okna se použije zajímavým způsobem. V případě, že odesilatel nabízí okno o velikosti  $q$  a nabídí zvětšení okna  $n$ , pak příjemce chápe, že odesílatel nabízí okno  $ox2^n$  (tj. posune šířku okna o  $n$  bitů).

Největší inzerovatelné okno je  $65535 \times 2^{14}$  ( $=1073725440=1G-16384$ ). Oknem lze tedy maximálně inzerovat téměř 1GB.

Proč nemůže být okno ještě větší? Je to jednoduché. Přenášená data se číslují od 0 do  $2^{32}$  (=4GB). V případě přetečení  $2^{32}$  se opět začíná od nuly. Při zvětšení okna např. na 8 GB by odesilatel mohl odeslat 8 GB nepotvrzených dat. V případě, že by si příjemce přál nějaký segment z těchto 8 GB opakovat (např. segment začínající na 2 GB), pak by odesílatel nevěděl, zdali mu opakovat segment začínající od 2 GB nebo od 6 GB, protože oba budou mít stejná pořadová čísla odeslaného bajtu (po 4 GB se začalo opět počítat od nuly).

I při použití okna velkého stovky MB (což je přípustné) se může stát, že Internetem bude putovat segment z již potvrzeného okna, avšak s číslem používaným v aktuálním okně. Tento problém se řeší použitím další volitelné položky v záhlaví TCP segmentu – „časového razítka“ (tato položka může být v každém segmentu). Odesílatel do této volby vkládá jednoznačnou rostoucí posloupnost (čas). Příjemce pak do odpovědi vloží své časové razítko a zopakuje poslední přijaté časové razítko. Lze tak poznat, které segmenty jsou staré a které aktuální. Lze tak detektovat starý zatoulaný segment.

## Domácí cvičení

Programem Wireshark si odchyťte TCP spojení a prozkoumejte:

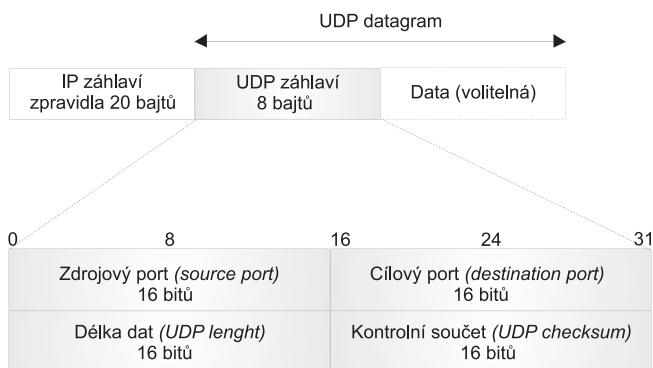
- ◆ jestli tvar vámi odchycených TCP segmentů odpovídá obr. 9.3,
- ◆ navazování a ukončování spojení.

## Kapitola 10

# Protokol UDP (User Datagram Protocol)

Protokol UDP je jednoduchou alternativou protokolu TCP. Protokol UDP je nespojovaná služba (na rozdíl od protokolu TCP), tj. nenavazuje spojení. Odesilatel odešle UDP datagram příjemci a už se nestará o to, zdali se datagram náhodou neztratil (o to se musí postarat aplikační protokol).

UDP datagramy jsou baleny do IP datagramu, jak je znázorněno na obrázku 10.1.



Obrázek 10.1: Záhlaví UDP datagramu

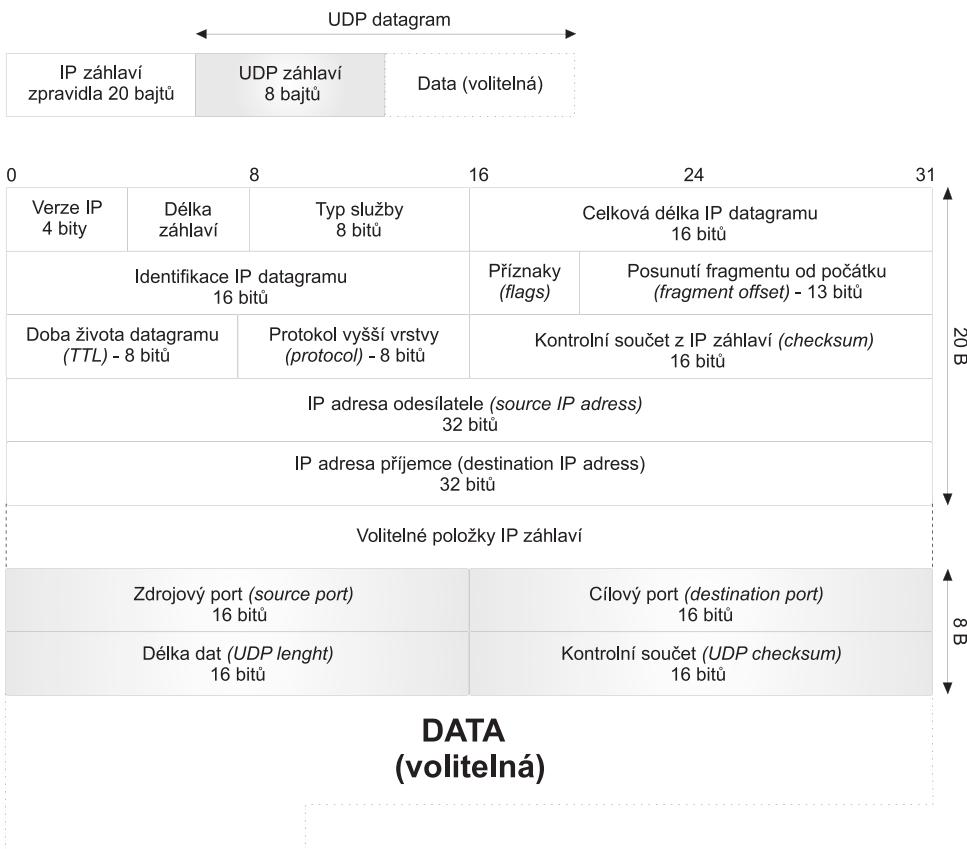
Celková podoba UDP datagramu včetně jeho IP záhlaví je znázorněna na obrázku 10.2.

Z předchozího obrázku je patrné, že záhlaví UDP protokolu je velice jednoduché. Obsahuje čísla zdrojového a cílového portu – což je zcela analogické protokolu TCP. Opět je třeba dodat, že čísla portů protokolu UDP nesouvisí s čísly portů protokolu TCP. Protokol UDP má svou nezávislou sadu čísel portů.

Pole délka dat obsahuje délku UDP datagramu (délku záhlaví + délku dat). Minimální délka je tedy 8, tj. UDP datagram obsahující pouze záhlaví a žádná data.

Zajímavé je, že pole kontrolní součet nemusí být povinně vyplňené. Výpočet kontrolního součtu je tak v protokolu UDP nepovinný.

V minulosti bylo u některých počítačů zvykem výpočet kontrolního součtu vypínat – zejména se jednalo o počítače s instalovaným systémem NFS (*Network File System*). Důvodem bylo zrychlení odezvy počítače.



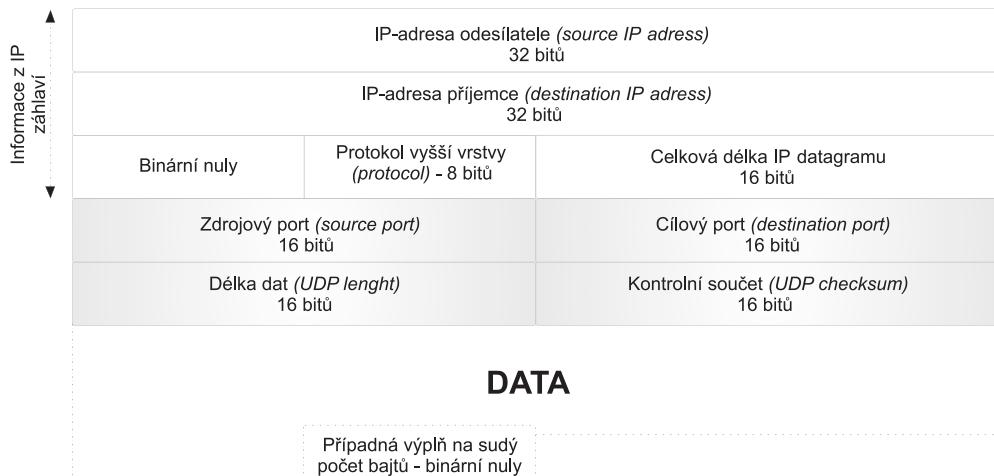
**Obrázek 10.2:** UDP datagram včetně záhlaví IP datagramu

Zejména u důležitých serverů je třeba vždy zkонтrolovat, zdali je opravdu výpočet kontrolního součtu zapnut. Nejnebezpečnější je to v případě DNS serveru, protože kontrolní součet je pak počítán jen na linkové vrstvě, ale např. linkový protokol SLIP výpočet kontrolního součtu také nepočítá, takže i technická porucha může způsobit poškození aplikačních dat, aniž by měl příjemce šanci to zjistit. Pakliže se kontrolní součet počítá, pak se podobně jako pro protokol TCP počítá ze struktury (pseudozáhlaví) znázorněné na obrázku 10.3.

## Fragmentace

I u UDP datagramů je možná fragmentace v IP protokolu. Avšak u UDP protokolu se zásadně snažíme fragmentaci vyhýbat.

Typickým případem je DNS. Klient DNS položí dotaz protokolem UDP. Pakliže by odpověď serveru přesáhla 512 B, server odešle jen tolik informací, aby nepřekročil hranici 512 B, a navíc v aplikačních datech nastaví příznak TC (Truncation) specifikující, že odpověď byla zkrácena. Pokud klientovi taková odpověď nestačí, zopakuje ji protokolem TCP, kterým mu server vrátí kompletní odpověď.



**Obrázek 10.3:** Pseudozáhlaví pro výpočet kontrolního součtu v UDP datagramu

## Příklad UDP datagramu

```

Ethernet II, Src: 00:1a:92:ca:e6:b9 (00:1a:92:ca:e6:b9), Dst: 00:13:02:91:2e:8b
          (00:13:02:91:2e:8b)
Internet Protocol, Src: 10.0.0.138 (10.0.0.138), Dst: 10.0.0.1 (10.0.0.1)
User Datagram Protocol, Src Port: 53 (53), Dst Port: 4454 (4454)
  Source port: 53 (53)
  Destination port: 4454 (4454)
  Length: 128
  Checksum: 0x2c1c [correct]
Domain Name System (response)

```

## Oběžníky

Na první pohled by se zdálo, že protokol UDP je chudým příbuzným protokolu TCP. Může však existovat něco, co umí protokol UDP, a nelze to udělat protokolem TCP? Právě zvláštností protokolu UDP je skutečnost, že adresátem UDP datagramu nemusí být pouze jednoznačná IP adresa, tj. síťové rozhraní konkrétního počítače. Adresátem může být skupina stanic – adresovat lze i oběžník.

Adresovat lze všeobecně oběžníky (*broadcast*), ale podstatně zajímavějším případem je adresování adresních oběžníků (*multicast*). Např. u aplikací typu RealAudio navazuje každý klient spojení se serverem, kdežto u ProgresiveRealAudio se šíří data pomocí adresních oběžníků, tj. dochází k ohromné úspoře kapacity přenosových cest. A právě to je přiležitost pro UDP.

## Domácí cvičení

Programem Wireshark odchyťte UDP datagram a porovnejte jej s obr. 10.2.



## Kapitola 11

# DNS

Všechny aplikace, které v Internetu zajišťují komunikaci mezi počítači, používají k identifikaci komunikujících uzlů IP adresu. Pro člověka jako uživatele jsou však IP adresy těžko zapamatovatelné. A adresy IPv6 už jsou naprostě nezapamatovatelné. Proto se používá místo IP adresy síťového rozhraní název síťového rozhraní. Pro každou IP adresu máme zavedeno jméno síťového rozhraní (počítače), přesněji řečeno doménové jméno. Toto doménové jméno můžeme používat ve všech příkazech, kde je možné použít IP adresu. (Výjimkou, kdy se musí jedině použít IP adresa, je specifikace samotného jmenného serveru.)

Jedna IP adresa může mít přiřazeno žádné, jedno nebo i několik doménových jmen.

Vazba mezi jménem počítače a IP adresou je definována v databázi DNS. DNS (*Domain Name System*) je celosvětově distribuovaná databáze. Databáze DNS obsahuje jednotlivé záznamy, které se také nazývají „DNS věty“ (*Resource Records – RR*). Jednotlivé části této databáze jsou umístěny na tzv. **jmenných serverech** (*Name Servers*). Jmenný server vyřizuje dotazy na překlad jména na IP -adresu a opačně.

Chci-li se přihlásit programem telnet na počítač info.pvt.net (obr. 11.1), použiji příkaz:

```
telnet info.pvt.net.
```

Ještě předtím, než se tento příkaz provede, musí se za pomocí jmenného serveru DNS přeložit jméno info.pvt.net na IP adresu 194.149.104.203. Teprve poté se může navazovat spojení (obr. 11.1).



**Poznámka:** Ve Windows Vista je nutné program telnet nejprve povolit v menu „Zapnout nebo vypnout funkce systému Windows“.

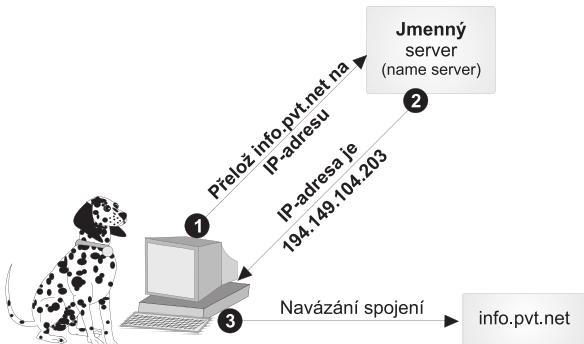
Použití IP adres místo doménových jmen je praktické vždy, když máme podezření, že nám na počítači DNS nepracuje korektně. Pak, ač to vypadá nezvykle, můžeme napsat např.:

```
ping 194.149.104.203  
http://194.149.104.203
```

či odeslat e-mail na: [dostalek@\[194.149.104.203\]](mailto:dostalek@[194.149.104.203])



**Poznámka:** Pokud chceme v poštovní adrese uvést IP adresu (na místo jména DNS), pak ji nesmíme zapomenou napsat v hranatých závorkách [ ].



**Obrázek 11.1:** Před navázáním spojení je nutné přeložit jméno na IP adresu

Reakce však může být neočekávaná zejména pro poštu, protokol HTTP a protokol HTTPS. Poštovní server totiž nemusí podporovat dopravu na servery uvedené v hranatých závorkách. Protokol HTTP nám vrátí primární webovou stránku a protokol HTTPS se bude rozčilovat, že jméno serveru neodpovídá příslušné položce certifikátu nesoucí jméno serveru.

## Domény a subdomény

Celý Internet je rozdělen do tzv. domén, tj. skupin jmen, která k sobě logicky patří. **Doména** specifikuje, patří-li jména jedné firmě, jedné zemi apod. V rámci domény je možné vytvářet podskupiny, tzv. subdomény, např. v doméně firmy lze vytvořit subdomény pro oddělení. Doménové jméno odráží příslušnost síťového rozhraní do skupiny a podskupiny. Každá skupina má přiřazeno jméno. Z jednotlivých jmen skupin je pak složeno doménové jméno síťového rozhraní. Např. síťové rozhraní se jménem `jakub.firma.cz` je rozhrani se jménem `jakub` v subdoméně `firma` domény `cz`.

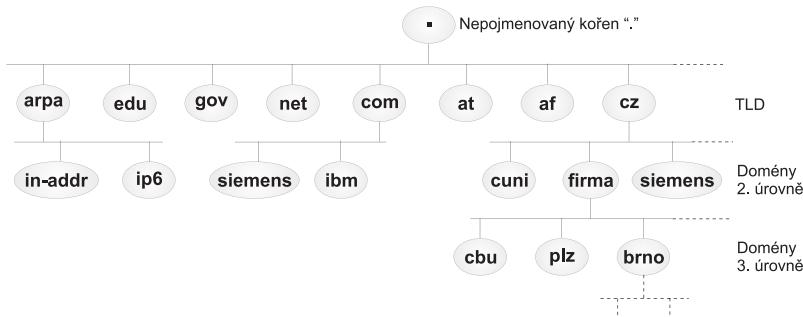
Doménové jméno se skládá z řetězců vzájemně oddělených tečkou. Jméno se zkoumá zprava doleva. Nejvyšší instancí je tzv. kořenová (*root*) doména, která se vyjadřuje tečkou zcela vpravo (tato tečka bývá často vypouštěna). V kořenové doméně jsou tzv. **Top Level Domains (TLD)**, které jsou:

- ◆ **Generické TLD (gTDL)**, které mají tři a více znaků: `aero`, `asia`, `biz`, `cat`, `com`, `coop`, `info`, `jobs`, `mobi`, `museum`, `name`, `net`, `org`, `pro`, `tel`, `travel`, `gov`, `edu`, `mil`, `int` a `arpa`.
- ◆ **Národní TLD**, které jsou dvouznamkové. Tyto dva znaky jsou totožné s identifikací země podle normy ISO-3166. Anglicky se označují **ccTLD** (*cc = Country Code*). Pro Českou republiku tak máme doménu `.cz`, pro Evropskou unii máme doménu `.eu` atd.

Např. TLD doména `cz` se dělí na subdomény pro jednotlivé organizace: `siemens.cz` (pro Siemens), `cas.cz` (pro Akademii věd), `cvut.cz` (pro ČVUT) atd. Subdomény se mohou dělit na subdomény nižší úrovně. Např. `entu.cas.cz` (Entomologický ústav AV ČR) atd.

Jména tvoří stromovou strukturu – viz obr. 11.2.

Představme si, že doména `cz` obsahuje doménu `firma`. Doména `firma.cz` obsahuje např. „krajské“ subdomény: `cbu`, `plz` a `brno`. Teoreticky by mohly existovat prvky těchto subdomén i subdomény ještě nižší úrovně atd.



Obrázek 11.2: Jména v systému DNS tvoří stromovou strukturu

## Syntaxe jména

Jméno je uváděno v tečkové notaci. Např. abc.cbu.pipex.cz. Jméno má obecně syntaxi:

řetězec.řetězec.řetězec....řetězec.

kde první řetězec je nejčastěji jméno síťového rozhraní počítače, další jméno nejhlohouběji vnořené domény, další jméno je jménem vyšší domény atd. Pro jednoznačnost se na konci uvádí také tečka, vyjadřující kořenovou doménu.

Celé jméno může mít maximálně 255 znaků, jeden řetězec pak maximálně 63 znaky. Řetězec se může skládat z písmen, číslic a pomlčky. Pomlčka nesmí být na začátku ani na konci řetězce. Existují i rozšíření specifikující bohatší repertoár znaků použitelných pro tvorbu jmen. Pokud je to možné, pak se těmto dalším znakům vyhýbáme, protože toto rozšíření podporují jen některé aplikace.

Mohou se použít velká i malá písmena, ale není to zase tak jednoduché. Z hlediska uložení a zpracování v databázi jmen (databázi DNS) se velká a malá písmena nerozlišují, tj. jméno newyork.com bude uloženo v databázi na stejném místě jako NewYork.com nebo NEWYORK.com atp. Tedy při překladu jména na IP adresu je jedno, kde uživatel zadá velká a kde malá písmena. Avšak v databázi je jméno uloženo i s velkými a malými písmeny, tj. bylo-li tam uloženo např. NewYork.com, pak při dotazu databáze vrátí NewYork.com. Poslední tečka je součástí jména.

V některých případech se může část jména zprava vynechat. Téměř vždy můžeme koncovou část doménového jména vynechat v aplikačních programech. V databázích popisujících domény je však situace složitější.

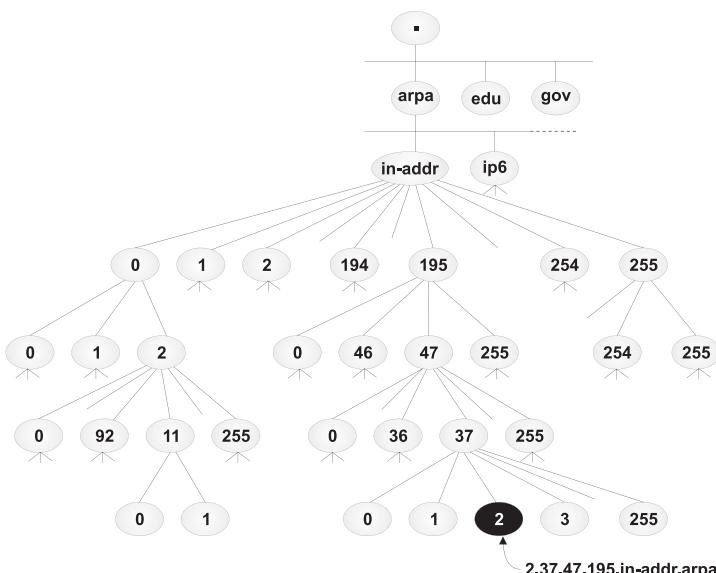
Je možné vynechat:

- ◆ Poslední tečku téměř vždy.
- ◆ Na počítačích uvnitř domény se zpravidla může vynechat konec jména, který je shodný s názvem domény. Např. uvnitř domény pipex.cz, je možné psát místo počítač.abc.pipex.cz jen počítač.abc (nesmí se ale uvést tečka na konci!). Do kterých domén počítač patří, se definuje příkazy domain a search v konfiguračním souboru resolveru (viz. obr. 11.10).

## Reverzní domény

Uvedli jsme, že komunikace mezi počítači probíhá na základě IP adres, nikoli doménových jmen. Některé aplikace naopak potřebují k IP adrese nalézt jméno, tj. nalézt tzv. reverzní záznam. Jedná se tedy o překlad IP adresy na doménové jméno. Tento překlad se často nazývá zpětným (reverzním) překladem.

Podobně jako domény tvoří i IP adresy stromovou strukturu (viz obr. 11.3). Domény tvořené IP -adresami se často nazývají reverzními domény. Pro účely reverzního překladu adres IPv4 byla definována pseudodoména „in-addr.arpa“, pro IPv6 pak „IP6.arpa“.



Obrázek 11.3: Reverzní doména k IP adrese 195.47.37.2



**Poznámka:** Jméno domény in-addr.arpa má historický původ – jde o zkratku „inverse addresses in the Arpanet“.

Pod doménou `in-addr.arpa` jsou domény jmenující se jako první číslo z IP adresy sítě, tj. doména `in-addr.arpa` má subdomény 0 až 255. Každá z těchto subdomén obsahuje nižší subdomény opět 0 až 255 atd. Např. síť 195.47.37.0/24 patří do domény 37.47.195.in-addr.arpa. Ta sama patří do domény 47.195.in-addr.arpa atd. Všimněte si, že domény jsou zde tvořeny jakoby IP adresami sítí, ale psanými pozpátku.

Celý tento mechanismus pracuje, pokud jsou přidělovány IP adresy třídy A, B nebo C. Jenže jak to udělat, pokud mám přidělenou pouze subsítě třídy C? Mohu i pak provozovat vlastní jmenný server pro reverzní překlad? Ano, přestože IP adresa má pouze 4 bajty, a klasická reverzní doména má tedy maximálně 3 čísla (4. číslem jsou už prvky domény – IP adresy), jsou reverzní domény pro subsítě sítí třídy C tvořeny čtyřmi čísly. To znamená, že např. pro subsítě 194.149.150.16/28 použijeme doménu 11.150.149.194.in-addr.arpa. Jakoby IP adresa měla najednou 5 bajtů! Původně to byla chyba

v implementaci DNS, ale později se tato chyba ukázala velice praktickou, takže se standardizovala jako RFC.

O reverzních doménách pro IP verze 6 se více dozvíte na str. 283.

## Doména 0.0.127.in-addr.arpa

Jistou komplikací (zvláštností) je IP adresa 127.0.0.1. Síť 127 je totiž určena pro *loopback*, tj. softwarovou smyčku na každém počítači.

Zatímco ostatní IP adresy jsou v Internetu jednoznačné, adresa 127.0.0.1 se vyskytuje na každém počítači.

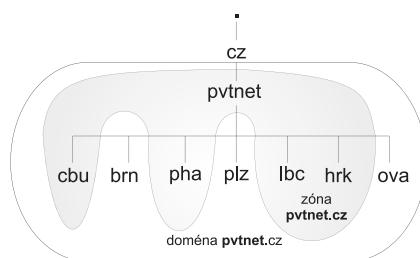
Každý jmenný server je tak autoritou nejen „obyčejných“ domén, ale ještě autoritou (jmenným serverem) k doméně 0.0.127.in-addr.arpa. V dalším textu budeme tento fakt považovat za samozřejmost a v tabulkách jej pro přehlednost nebudeme uvádět, ale nikdy na něj nesmíte zapomenout. Tj. i caching only server bude primárním serverem pro tuto doménu.

## Zóna

Často se setkáváme s otázkami: „Co je to zóna?“ „Jaký je vztah mezi doménou a zónou?“ Vysvětleme si tedy vztah těchto pojmu na doméně pvt.net.cz.

Jak jsme již uvedli, doména je skupina počítačů, které mají společnou pravou část svého doménového jména. Doména je např. skupina počítačů, jejichž jméno končí pvt.net.cz. Doména pvt.net.cz je však velká. Dělí se dále na subdomény cbu.pvt.net.cz, brn.pvt.net.cz atd. Celou doménu pvt.net.cz můžeme spravovat na jednom jmenném serveru nebo pro některé subdomény zřídíme samostatné jmenné servery (na obr. 11.4 jsme konkrétně zřídili samostatné jmenné servry pro subdomény brn.pvt.net.cz, plz.pvt.net.cz a ova.pvt.net.cz). Původní jmenný server tak bude spravovat doménu pvt.net.cz až na subdomény brn.pvt.net.cz a plz.pvt.net.cz, tj. původní jmenný server bude spravovat zónu pvt.net.cz. Zóna je část prostoru jmen domény, kterou obhospodařuje konkrétní jmenný server.

Zóna obsahující data domény nižší úrovně bývá označována jako „podřízená zóna“.



Obrázek 11.4: Zóna pvt.net.cz

## Speciální zóny

Vedle klasických zón, které obsahují data o části domény nebo subdomény, se při implementaci DNS používají ještě speciální zóny. Jedná se zejména o následující zóny:

### Zóna typu stub

Zóna stub je v podstatě podřízená zóna, která však obsahuje pouze informace o tom, které jmenné servery danou subdoménu obsluhují (obsahuje jen tzv. věty NS pro zónu). Zóna stub tedy neobsahuje celou zónu.

### Zóna typu cache/hint

V zóně hint je uveden seznam kořenových jmenných serverů (neautoritativní data načítaná do paměti při startu jmenného serveru). Název hint pro tento typ zóny zavedl až BIND 8, v předchozích verzích se používalo označení zóna „cache“. Připomeňme, že kořenové jmenné servery jsou autoritou pro kořenovou doménu označenou „.“.

## Rezervované domény a pseudodomény

Podobně jako máme rezervované IP adresy pro interní síť (např. 10.0.0.0/8), máme i rezervované TLD:

- ◆ doména .localhost pro softwarovou smyčku
- ◆ doména .local pro intranety
- ◆ doména .test pro testování
- ◆ doména .example pro vytváření dokumentace a příkladů
- ◆ doména .invalid pro navozování chybových stavů

## Dotazy (překlady)

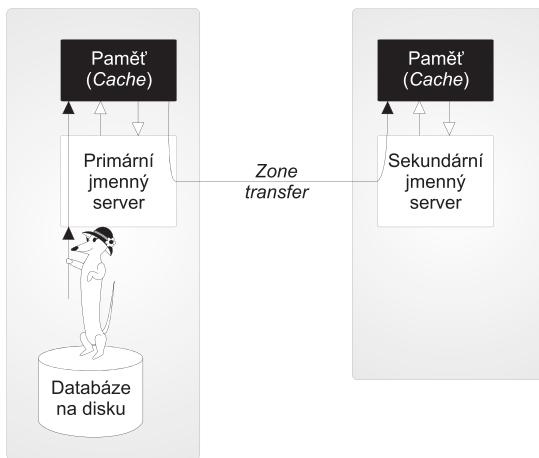
Přeložení jména na IP adresu zprostředkovává tzv. resolver. Resolver je klient, který se dotazuje jmenného serveru. Jelikož je databáze DNS celosvětově distribuována, nemusí nejbližší jmenný server znát konečnou odpověď a může požádat o pomoc další jmenné servery. Získaný překlad pak jmenný server vrátí jako odpověď resolveru. Veškerá komunikace se skládá z dotazů a odpovědí.



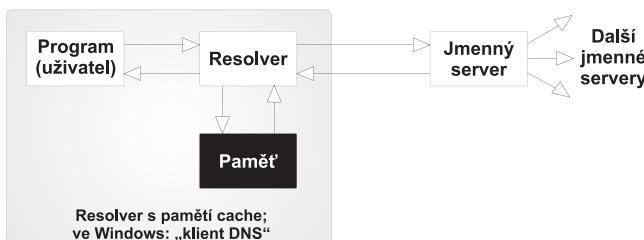
**Poznámka:** V terminologii programu BIND se od verze 8 místo „Primární jmenný server“ používá „Primární master server“ a místo „Sekundární jmenný server“ se používá „Slave server“.

Jmenný server si během svého startu do paměti načte data pro zónu, kterou spravuje. Primární jmenný server načte data z lokálního disku, sekundární jmenný server data získá dotazem zone transfer data z primárního jmenného serveru a rovněž je uloží do paměti. Tato data primárního a sekundárního jmenného serveru se označují jako autoritativní (nezvratná). Dále jmenný server načte z lokálního disku do paměti data zóny cache/hint, která nejsou součástí dat jeho spravované zóny, ale umožní mu spojení s kořenovými (root) jmennými servery. Tato data se označují jako neautoritativní.

Jmenné servery během své další práce ukládají do paměti kladné (někdy i záporné) odpovědi na dotazy, o které musejí požádat další jmenné servery. Z hlediska našeho jmenného serveru jsou tato další data získaná z jiných jmenných serverů rovněž neautoritativní – pouze šetří čas při opětovných dotazech.



**Obrázek 11.5:** Primární jmenný server načítá data z disku; sekundární server získává data dotazem zone transfer protokolu DNS



**Obrázek 11.6:** Pahýlový resolver a resolver s pamětí

Požadavky na překlady vznikají v aplikacích (programech), které o překlady žádají komponentu operačního systému – resolver. Resolver překlady předává jmennému serveru. Na klientských systémech zpravidla běží pouze resolver. Resolver v takovém případě předává požadavky protokolem DNS jmennému serveru běžícímu na jiném počítači (viz obr. 11.6). Resolver bez paměti nazýváme pahýlovým resolverem.

Ačkoliv i resolver může mít paměť. To je případ i Windows (od Windows 2000 výše). Windows takový resolver spouští jako službu. Tato služba se ve Windows označuje jako „Klient DNS“.



**Poznámka:** Připadá mi to trochu divné – pahýlový revolver asi není pro Windows tím „pravým ořechovým“ klientem DNS!

Na některých počítačích poběží jen resolver (ať již pahýlový nebo s pamětí), na jiných poběží jak resolver, tak i jmenný server. Nyní je možná celá řada různých kombinací (viz obr. 11.7). Princip zůstává stejný:

1. Uživatel zadá příkaz, který např. vyžaduje přeložit jméno info.pvt.net na IP adresu (na obr. 11.7 číslo 1).
2. Pokud revolver využívá paměť, tak se pokusí najít výsledek přímo v ní.
3. Pokud se odpověď v paměti resolveru nenajde, pak resolver předá požadavek jmennému serveru (3).
4. Jmenný server hledá odpověď ve své paměti.
5. Pokud jmenný server nenaleze odpověď ve své paměti, pak hledá pomoc u jiných jmenných serverů.
6. Jmenný server může kontaktovat více jmenných serverů procesem, který se nazývá iterace. Iterací se server může dostat až na jmenný server, který je autoritou pro zadaný dotaz. Autorelativní jmenný server pak s konečnou platností odpoví (např. i záporně, že k zadanému jménu v DNS žádné informace nejsou).
7. Jenže pokud proces popsán v předešlých bodech dostatečně rychle nevrátí výsledek, tak resolver opakuje svůj dotaz (viz obr. 11.8). Pokud je v konfiguraci resolveru uvedeno více jmenných serverů, pak další dotaz pošle na další jmenný server uvedený v seznamu (tj. jiný jmenný sever). Seznam jmenných serverů se prochází cyklicky. Cyklus se pro daný dotaz startuje od jmenného serveru, který je v seznamu uveden jako první.

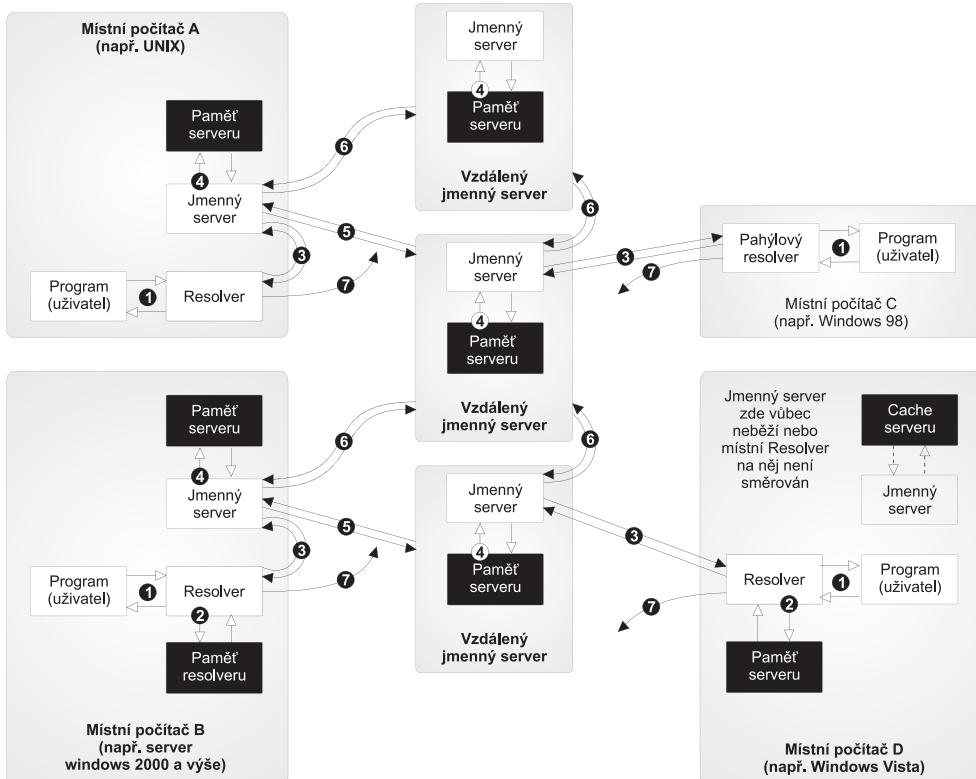
DNS používá pro transport svých dotazů/odpovědí jak protokol UDP, tak i protokol TCP. Oba protokoly používají port 53 (tj. porty 53/udp a 53/tcp). Běžné dotazy, jako je překlad jména na IP adresu a naopak, se provádějí protokolem UDP. Délka přenášených dat protokolem UDP je implicitně omezena na 512 B (příznakem *truncation* může být signalizováno, že se odpověď nevešla do 512 B a pro případnou kompletní odpověď je nutné dotaz zopakovat protokolem TCP). Délka UDP paketu je omezena na 512 B, protože u větších IP datagramů by mohlo dojít k fragmentaci. Fragmentaci UDP datagramu DNS nepovažuje za rozumnou.

Dotazy, kterými se přenáší data o zóně (*zone transfer*), např. mezi primárním a sekundárním jméným serverem, se přenáší protokolem TCP.

Běžné dotazy (např. překlad jména na IP adresu a naopak) se tak provádí pomocí datagramů protokolu UDP. Překlad požaduje klient (resolver) na jmenném serveru. Neví-li si jmenný server rady, může požádat o překlad (o pomoc) jiný jmenný server. Jmenné servery řeší dotazy mezi sebou pomocí iterace, která vždy začíná od kořenového jmenného serveru.

V Internetu platí pravidlo, že databáze s daty nutnými pro překlad jsou vždy uloženy alespoň na dvou nezávislých počítačích (nezávislých jmenných serverech). Je-li jeden nedostupný, pak se překlad může provést na druhém počítači.

Obecně se nepředpokládá, že by byly všechny jmenné servery dostupné. V případě, že by se pro překlad použil protokol TCP, pak by navazování spojení na nedostupný jmenný server znamenalo přetrvávat časové intervaly protokolu TCP pro navázání spojení, a teprve poté by bylo možno se pokusit navázat spojení s dalším jmenným serverem.

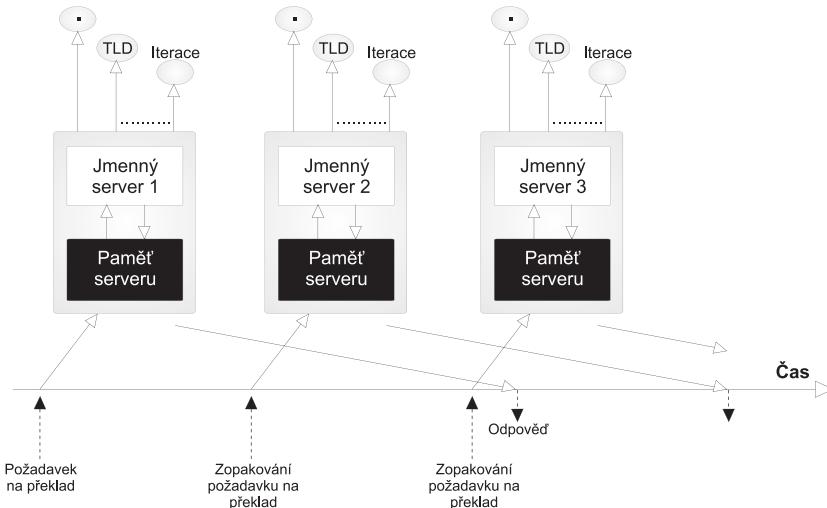


Obrázek 11.7: Jmenný server a resolver

Řešení pomocí protokolu UDP je elegantnější (obr. 11.8): Vyšle se datagram obsahující žádost o překlad prvnímu serveru. Nepřijde-li odpověď do krátkého časového intervalu, pak se pošle datagram s žádostí dalšímu jmennému serveru, nepřijde-li opět odpověď, pak se pošle dalšímu atd. V případě, že se vyčerpají všechny možné jmenné servery, pak se opět začne prvním a celý kolotoč se zopakuje, dokud nepřijde odpověď nebo nevyprší stanovený časový interval.

## Round Robin

Jedná se o techniku, pomocí které můžeme rovnoměrně zatěžovat více strojů (*Load Balancing*). U většiny jmenných serverů (včetně Windows) lze tuto techniku zapnout. Jedná se o situaci, kdy k jednomu jménu máme v DNS více IP adres. Např. provozujeme exponovaný webový server a jeli-kolž nám nestačí výkon stroje, tak si koupíme např. další dva servery. Na všech třech spustíme webový server (např. [www.firma.cz](http://www.firma.cz)). První bude mít IP adresu 195.47.40.5, druhý 195.47.37.200 a třetí 194.149.105.103. V DNS budou tedy tři záznamy pro [www.firma.cz](http://www.firma.cz) a každý bude mít jinou IP adresu. Technika *Round Robin* zajistí, že odpověď na první dotaz (dotaz prvního uživatele) bude, že webový server má adresy 194.47.40.5, 195.47.37.200 a 194.149.105.103. Odpověď na další dotaz (druhému uživateli) bude, že server má IP adresy 195.47.37.200, 194.149.105.103 a 194.47.40.5 atd – tj. IP adresy se točí dokola.



**Obrázek 11.8:** Řešení požadavku na překlad

## Konfigurace resolveru

Resolver je komponenta systému zabývající se překladem IP adresy. Resolver je klient. Resolver není konkrétní program. Je to soustava knihovních funkcí, která se sestavuje s aplikačními programy požadujícími tyto služby (např. telnet, ftp, WWW prohlížeč atd.). To znamená, že potřebuje-li např. telnet převést jméno počítače na jeho IP adresu, pak zavolá příslušné knihovní funkce.

Klient (např. zmíněný telnet) zavolá knihovní funkce, které zformulují dotaz a vyšlou jej na server. Naopak jmenný server je realizován samostatným programem.

Do hry o překlad ještě vstupují časová omezení. Může se totiž stát, že na položený dotaz nedostane resolver odpověď, ale další stejný dotaz již bude korektně zodpovězen (serveru se mezitím podařilo získat odpověď a první dotaz nebyl zodpovězen proto, že odpověď z jiného jmenného serveru dlouho nepřicházela). Z hlediska uživatele se to jeví tak, že napoprve se překlad nepovede a při dalším zadání téhož příkazu už ano.

Podobný efekt způsobuje i použití protokolu UDP. Může se totiž také stát, že server vůbec žádost o překlad neobdrží, protože je síť přetížená a UDP datagram se prostě někde ztratil.

## Konfigurace resolveru v Unixu

Konfigurační soubor pro resolver se v operačním systému UNIX jmenuje `/etc/resolv.conf`. Zpravidla obsahuje dva typy řádků (druhý příkaz se může i několikrát opakovat):

```
domain      jméno_místní_domény
nameserver  IP-adresa_jmenného_serveru
```

V případě, že uživatel zadal jméno bez tečky na konci, pak resolver za zadané jméno přidá jméno domény z příkazu `domain` a pokusí se jméno předat jmennému serveru k přeložení. V případě, že se

překlad neprovede (negativní odpověď jmenného serveru), resolver se pokusí ještě přeložit jméno samotné, tj. bez přípony z příkazu domain.

Některé resolversky umožňují zadat příkazem search více jmen místních domén.

Příkazem nameserver se specifikuje IP adresa jmenného serveru, který má resolver kontaktovat. Je možné uvést i další příkazy nameserver pro případ, že některé jmenné servery jsou nedostupné.

V konfiguračním souboru resolveru se musí vždy uvést IP adresa jmenného serveru – nikoliv doménové jméno jmenného serveru!

V případě konfigurace resolveru na jmenném serveru může příkaz nameserver ukazovat na místní jmenný server 127.0.0.1 (nemusí to však být pravidlem, jak uvidíme u firewallů).

Další parametry resolveru (např. maximální počet příkazů nameserver) lze nastavit v konfiguračním souboru jádra operačního systému. Tento soubor se často jmenuje /usr/include/resolv.h. Musí pak pochopitelně následovat sestavení jádra operačního systému.

Obecně je možné konfigurovat všechny počítače též bez použití DNS. Pak se veškeré dotazy na překlady adres provádějí lokálně pomocí souboru /etc/hosts. Je možné obě metody i kombinovat (nejčastější případ), pak však je třeba být opatrný na obsah databáze /etc/hosts. Většinou je možné i nastavit, v jakém pořadí se mají databáze prohlížet. Zpravidla se prohlíží nejprve soubor /etc/hosts a posléze DNS. Např. v Linuxu slouží pro konfiguraci pořadí prohledávání soubor /etc/nsswitch.conf.

## Konfigurace resolveru ve Windows

Zajímavá situace je u operačních systémů Microsoft od Windows 2000 a výše. Zde máme navíc zmíněnou službu „Klient DNS“. Jedná se o implementaci revolveru s pamětí. Tato služba je implicitně spuštěna. V dokumentaci je striktně nedoporučováno tuto službu zastavovat, ale podle mých testů po zastavení této služby se Windows chovají jako stanice s pahýlovým resolverem.

Obsah paměti resolveru lze dokonce i vypsat, a to příkazem IPCConfig /displayDNS, nebo vymazat příkazem IPCConfig /flushDNS.

Součástí paměti resolveru je i obsah souboru %SystemRoot%\ System32\ Drivers\ etc\ hosts, jehož obsah příkaz IPCConfig /flushDNS nemění.

Paměť resolveru lze parametrizovat vložením nebo změnou klíčů v registru Windows: HKEY\_LOCAL\_MACHINE\ SYSTEM\ CurrentControlSet\ Services\ DnsCache\ Parameters. Např. klíčem NegativeCacheTime lze měnit dobu, po kterou jsou v paměti resolveru udržovány negativní odpovědi.

Na operační systémy Microsoft je třeba se dívat z historického pohledu. Předchůdcem sítě Windows byl systém označovaný jako LAN Manager (zkratkou nědy též LM), který je postaven na protokolu NetBIOS. Protokol NetBIOS rovněž používá jména počítačů, která musí na síťové vrstvě překládat na síťové adresy. V případě, že Windows použijí jako síťový protokol TCP/IP, pak potřebují překládat jména počítačů na IP adresy a naopak.

LAN Manager si zavedl vlastní systém jmen. Jména s příslušnými IP adresami byla ukládána lokálně v souboru %SystemRoot%\ System32\ Drivers\ etc\ lmhosts. Windows později zavedly i obdobu DNS – databázi označovanou jako WINS (*Windows Internet Name Service*).

Zajímavým problémem ve Windows je překlad jmen. Když se překlad nenajde ani v souboru lmhosts, ani na WINS serveru, pak je ještě odeslán všeobecný oběžník s paketem protokolu NetBIOS s žádostí, jestli se náhodou hledaný počítač nenachází přímo na LAN. Po zavedení DNS do Windows se celý mechanismus rozšířil ještě o vyhledávání v DNS, takže programy ve Windows 2000, které mají kořeny v systému LAN Manager (např. příkaz net), hledají překlad:

1. V paměti LAN Manageru lokálního počítače (vypisuje se příkazem nbtstat -c). Jedná se o paměť pro protokol NetBIOS! Do této paměti se při startu načítají i ty řádky ze souboru lmhosts, které mají jako poslední parametr uveden řetězec #PRE. Po úpravě souboru lmhosts lze vynutit načtení těchto řádků příkazem nbtstat -R.
2. Na serverech WINS.
3. Všeobecným a případně adresným oběžníkem na LAN.
4. V souboru lmhosts.
5. V paměti resolveru (je do ní načten i obsah souboru hosts).
6. V systému DNS.

Naopak programy, které jsou internetově orientovány, vyhledávají překlad (např. příkaz ping):

1. V paměti resolveru lokálního počítače (je do ní načten i obsah souboru hosts).
2. V DNS.
3. Na serverech WINS.
4. Všeobecným oběžníkem s paketem protokolu NetBIOS.
5. V souboru lmhosts.

Pokud se tedy spletete ve jméně počítače v příkazu ping, pak při sledování sítě oprogramem Wireshark uvidíte i pakety protokolu NetBIOS, a dokonce i vyhledávání všeobecným oběžníkem.

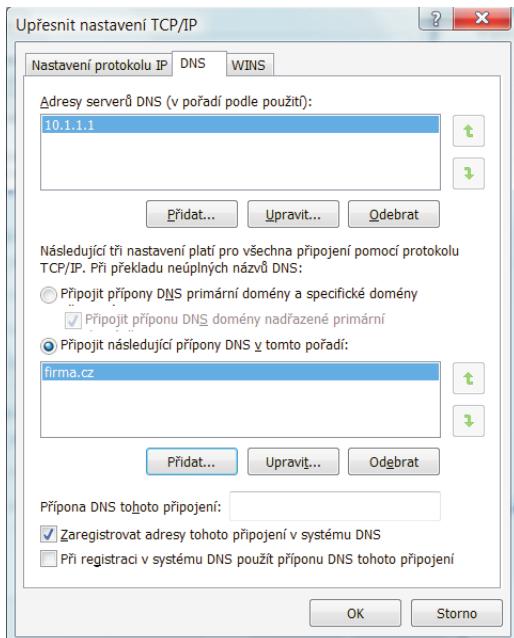
Zastavme se u konfigurace resolveru ve Windows Vista (obr. 11.9). První věc, která nás překvapí, je, že konfigurace je poměrně komplikovaná. Je to dánno dvěma faktory:

1. Stanice může být součástí domény Windows (*Active Directory*). Pak se tato doména Windows chápe i jako doména DNS a nazývá se „**Primární doménou DNS**“.
2. DNS se ve Windows konfiguruje jakoby pro každé síťové rozhraní zvlášť. Avšak když počítač překládá DNS jméno na IP adresu, ještě neví, skrze jaké rozhraní bude komunikovat, takže se zpravidla uplatní DNS přípony a IP adresy jmenných serverů od všech nakonfigurovaných síťových rozhraní.

Nyní se vraťme k obr. 11.9. Do horního okna zadáváme IP adresy jmenných serverů (v terminologii Windows „serverů DNS“). Ty není nutné zadat, pokud je získáme při startu počítače např. z DHCP serveru nebo při navázání telefonického spojení pomocí protokolu PPP.

Dále jsou zde dvě možnosti:

1. Volbou „Připojit přípony DNS primární domény a specifické domény“ (tato volba není na obr. 11.9 zvolena). Tím se jen sděluje, že při konfiguraci tohoto rozhraní se nebudou specifikovat žádné nové přípony. Budou se postupně zkoušet přípona primární domény a následně případně přípony specifikované u jiných rozhraní.
2. Volbou „Připojit následující přípony DNS v tomto pořadí“ (tato volba je zvolena na obr. 11.9.) se sděluje, že budou vyjmenovány specifické přípony (na obr. 11.9 je vyjmenována pouze jedna přípona `firma.cz`).



**Obrázek 11.9:** Konfigurace resolveru ve Windows Vista

Velice zajímavá je též volba „Zaregistrovat adresy toho připojení v systému DNS“, která způsobí, že se počítač pokusí o dynamickou registraci adresy IP tohoto připojení v systému DNS (viz DNS Update - str. 299).

## Jmenné servery

Jmenný server udržuje informace pro překlad jmen počítačů na IP adresy (resp. pro reverzní překlad). Jmenný server obhospodařuje vždy nějakou část z prostoru jmen všech počítačů. Tato část se nazývá zóna. Každý jmenný server spravuje minimálně zónu 0.0.127.in-addr.arpa. Zóna je tvořena doménou nebo její částí. Jmenný server totiž může pomocí věty typu NS (ve své konfiguraci) delegovat spravování subdomény na jmenný server nižší úrovně, tj. vyjmout je ze své zóny a delegovat je na jmenný server „nižší úrovne“.

Jmenný server je program, který provádí překlad na žádost resolveru nebo jiného jmenného serveru. Např. v Unixu je jmenný server realizován programem named. Nejznámější implementací jmenného serveru v Unixu/Linuxu je Bind (*Berkeley Internet Name Domain*).

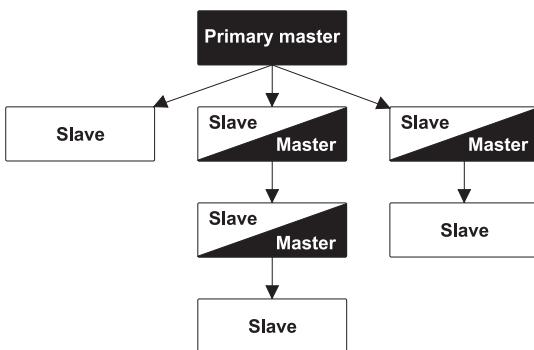
Pořadí uložení dat rozlišujeme následující typy jmenných serverů:

- ◆ **Primární jmenný server / primární master server** je hlavním zdrojem dat pro zónu. Je autoritativním serverem pro zónu. Tento server získává data o své zóně z databází uložených na lokálním disku. Označení tohoto typu serveru závisí na používané verzi Bindu. Zatímco u verze 4.x se výhradně používalo označení primární jmenný server, od verze 8 se používá označení primary master server. Databáze tohoto serveru jsou manuálně vytvářené správcem. Primární server musí

být zveřejněn jako autoritativní jmenný server pro doménu v tzv. NS-záznamu, zatímco primary master server zveřejněn být nemusí. Tento typ serveru je pro každou zónu pouze jeden.

- ◆ **Master server** je autoritativní server pro zónu. Master server je vždy zveřejněn jako autoritativní server pro doménu v tzv. NS-záznamech. Je zdrojem dat o zóně pro podřízené servery (slave/sekundární servery). Master serverů může existovat několik. Tento typ serveru se používá od verze Bind 8.
- ◆ **Sekundární jmenný server / slave server** získává data o doméně z primárního jmenného serveru, respektive z master serveru domény tak, že si je v pravidelných časových intervalech kopíruje z jeho databáze. Tyto databáze, přestože jsou uložené na lokálním disku serveru, nemá smysl na sekundárním jmenném serveru editovat, neboť budou při dalším kopírování přepsány. Tento typ jmenných serverů je tzv. autoritou pro své zóny, tj. jeho data pro příslušnou zónu se považují za nezvratná (autoritativní). Označení tohoto typu serveru opět závisí na používané verzi Bindu. U verze 4 se používalo výhradně označení sekundární a termín slave server byl použit pro zcela jiný typ serveru. U verze 8 se můžete setkat v dokumentaci s oběma označeními.
- ◆ Zóna udržovaná na podřízeném serveru se někdy označuje jako podřízená zóna.
- ◆ **Caching only server** není pro žádnou doménu ani primárním, ani sekundárním jmenným serverem (není žádnou autoritou). Avšak využívá obecné vlastnosti jmenných serverů, tj. data, která jím prochází, ukládá ve své paměti. Tato data se označují jako neautoritativní. Každý server je caching server, ale slovy caching only zdůrazňujeme, že pro žádnou zónu není ani primárním, ani sekundárním jmenným serverem. (Pochopitelně i caching only server je primárním jmenným serverem pro zónu 0.0.127.in-addr.arpa, ale to se nepočítá.)
- ◆ **Stealth server** je tajný server. Tento typ jmenného serveru není nikde zveřejňován. Je znám pouze těm serverům, které mají staticky uvedenu jeho IP adresu v konfiguraci. Je to autoritativní server. Získává data pro zónu pomocí zónového přenosu. Může být hlavním serverem pro zónu. Stealth server může být použit jako lokální záloha, pokud oficiální servery jsou nedostupné.
- ◆ **Kořenový jmenný server (Root name server)** je jmenný server obsluhující doménu root. Každý root name server je primárním serverem, což jej odlišuje od ostatních jmenných serverů – nemáme žádné sekundární root servery nebo něco podobného!

Na obrázku 11.10 je zobrazena architektura master/slave.



Obrázek 11.10: Architektura master/slave

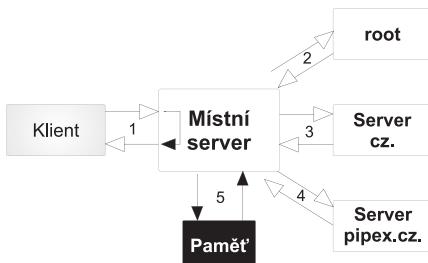
Jeden jmenný server může být pro nějakou zónu master (primárním) serverem, pro jiné slave (sekundárním) serverem.

Z hlediska klienta není žádný rozdíl mezi master (primárním) a slave (sekundárním) jmenným serverem. Oba mají data stejné důležitosti – oba jsou pro danou zónu autoritami. Klient nemusí ani vědět, který server pro zónu je master (primární server) a který slave (sekundární). Naproti tomu caching server není autoritou, tj. nedokáže-li provést překlad, pak kontaktuje autoritativní server pro danou zónu.

Takže přidá-li správce zóny (*hostmaster*) do databáze na master serveru další počítač, pak se databáze na všech slave serverech automaticky opraví po době stanovené parametrem ve větě SOA (pokud by opravil ručně jen databázi na sekundárním jmenném serveru, pak by po stejně době oprava zmizela!). Problém nastane v případě, že uživatel v době, kdy ještě není slave server aktualizován, dostane první odpověď od slave serveru. Ta je negativní, tj. takový počítač v databázi není.

Ještě horší je následující případ: master server pracuje korektně, ale na slave serveru z nějakého důvodu nejsou data pro zónu. Klienti náhodně dostávají autoritativní odpovědi z master serveru nebo ze slave serveru. Odpovědi z master serveru správně překládají, kdežto odpovědi ze slave serveru jsou negativní (uživatelé pak říkají: „Jednou to jde a podruhé ne“).

Autoritativní data pocházejí z databází na disku. Je zde pouze jedna výjimka. Pro správnou činnost jmenného serveru musí jmenný server znát kořenové jmenné servery. Pro ty však není autoritou, přesto každý jmenný server má na disku databázi informací o root serverech, kterou ale zavádí příkazem cache do sekundární paměti (není k nim autorita) – viz zóna cache/hint.



**Obrázek 11.11:** Překlad jména `abc.pipex.cz` na IP adresu (nejdříve se o forwarder nebo slave server)

Na obrázku 11.11 je zobrazen proces iterace na překladu jména `abc.pipex.cz` na IP adresu:

1. Resolver zformuluje požadavek na jmenný server a očekává jednoznačnou odpověď. Umí-li jmenný server odpovědět, pak obratem zaše odpověď. Odpověď hledá ve své paměti (5). Tam jsou jak autoritativní data z databází na disku, tak i neautoritativní data získaná při předešlých překladech. Nenajde-li server odpověď ve své paměti, pak kontaktuje další servery. Vždy začíná kořenovým jmenným serverem.

Nezná-li jmenný server odpověď přímo, pak kontaktuje kořenový jmenný server – proto každý jmenný server musí znát IP adresy kořenových jmenných serverů. Není-li však žádný kořenový server dostupný (to je např. případ všech uzavřených sítí), pak po několika neúspěšných pokusech celý proces překladu zkolaže.

Kořenový jmenný server zjistí, že informace o doméně `cz` delegoval větou typu NS jmennému serveru nižší úrovně, a zašle našemu jmennému serveru IP adresy serverů spravujících doménu `cz`.

2. Jmenný server se obrátí na server pro doménu `cz`, který však zjistí, že informace o doméně delegoval větou typu NS jmennému serveru nižší úrovně, a zašle našemu jmennému serveru IP adresy serverů spravujících doménu `pipex.cz`.
3. Náš server se tedy obrátí na server spravující doménu `pipex.cz`, který mu požadavek vyřeší (nebo ne). Výsledek předá klientovi.
4. Informace, které server postupně získal, si též uloží do své paměti.



**Poznámka:** Nepřipomíná vám to pohádku O kohoutkově a slepičce?

Jmenný server ukládá do své paměti i odpovědi popsané v předešlých čtyřech bodech (při překladu `abc.pipex.cz`). Při následných překladech již pak může využít odpovědi z paměti a tím si nejenom ušetřit čas, ale ulehčí tím i kořenovým jmenným serverům. Avšak pokud požadujete např. překlad jména z kořenové domény (TLD), která není v paměti, pak se opravdu kontaktuje kořenový jmenný server. Z toho je vidět, že kořenové servery budou v Internetu velice zatěžované a jejich nedostupnost by zhroutila komunikaci v celém Internetu.

Jmenný server nepožaduje úplnou (rekurzivní) odpověď. Důležité jmenné servery (např. kořenové jmenné servery nebo jmenné servery pro TLD) dokonce nesmí podporovat rekurzivní odpovědi, aby se nepřetížily, a nestaly se tak nedostupnými. Nelze na ně tedy např. nasměrovat resolver vašeho počítače.

Program `nslookup` je užitečný program nejenom pro správce jmenného serveru. Chcete-li programem `nslookup` provádět dotazy jakoby jmenným serverem, pak zakažte iterace (rekurzivní dotazy) a přidávání doménových jmen z konfiguračního souboru resolveru příkazy:

```
$ nslookup  
set norecurse  
set nosearch
```

## Předávání dotazů DNS

Krom dělení jmenných serverů podle uložení dat ještě můžeme rozlišovat jmenné servery podle způsobu iterování (způsobu překladu). Zavádíme tak termín **forwarder server**. Tato vlastnost serveru nesouvisí s tím, zda jsme primárním nebo sekundárním serverem pro nějakou zónu, ale výhradně souvisí se způsobem, jakým jmenný server provádí překlad dotazů DNS.

Zatím jsme si řekli, že resolver předává požadavek na překlad jmennému serveru, tj. pošle jmennému serveru dotaz a čeká na konečnou odpověď (klient posílá rekurzivní dotaz). Když jmenný server neumí odpovědět sám, tak sám zajistí rekurzivní překlad. Nejprve kontaktuje kořenový jmenný server, který mu poradí, pak podle jeho rady kontaktuje další jmenný server atd. – Jmenný server posílá do Internetu mnoho paketů.

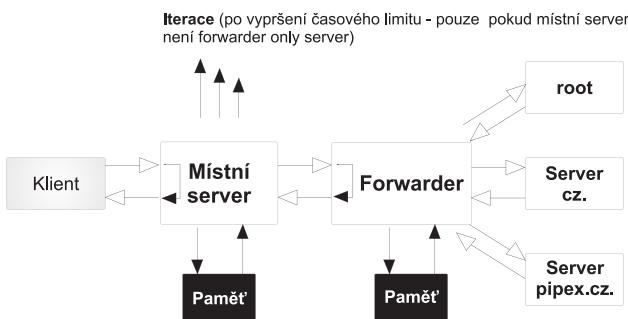
Je-li podniková síť připojena k Internetu pomalou linkou, pak místní jmenný server zatěžuje linku svými překlady. V takovém případě je výhodné si některý z jmenných serverů v Internetu nakonfigurovat jako **forwarder server** (viz obr. 11.12). Místní server předá požadavek na překlad forwarder serveru, avšak tváří se jako resolver, tj. požaduje od forwarder serveru až konečnou odpověď (dotaz

opatří příznakem, že se jedná o rekurzivní dotaz). Forwarder server vezme požadavek od místního jmenného serveru a provede sám jednotlivé dotazy v Internetu. Našemu jmennému serveru pak předá až konečný výsledek.

Místní jmenný server čeká na konečný výsledek od forwarder serveru. Nedočká-li se místní jmenný server odpovědi v daném časovém intervalu, pak sám kontaktuje kořenové jmenné servery a pokouší se sám vyřešit překlad.

Nemá-li místní jmenný server kontaktovat kořenové jmenné servery, ale pouze čekat na odpověď, pak je nutné označit takový server v jeho konfiguraci jako „**forwarder only**“ (dříve se takový server označoval jako „**slave**“). Forwarder only servery se používají v uzavřených podnikových sítích (za firewalllem), kde není možný kontakt s kořenovými jmennými servery. Forwarder only server pak kontaktuje jmenný server, který je součástí firewallu.

Forwarder server v obou variantách může být jak caching only server, tak i primární nebo sekundární jmenný server pro nějakou zónu.



**Obrázek 11.12:** Komunikace místního jmenného serveru s forwarder serverem

S forwarder servery se setkáme ještě u firewallů v kap. 19.

## Věty RR

Informace o doménových jménech a jim příslušejících IP adresách, stejně tak jako všechny ostatní informace distribuované pomocí DNS, jsou uloženy v paměti jmenných serverů ve tvaru zdrojových vět (*Resource Records – RR*). Jmenný server (označovaný též DNS server) naplňuje svou paměť několika způsoby. Autoritativní data načte ze souborů na disku nebo je získá pomocí dotazu *zone transfer* z paměti jiného serveru. Neautoritativní data jmenný server získává postupně z paměti jiných serverů, tak jak vyřizuje jednotlivé DNS dotazy.

V případě, že DNS klient potřebuje získat informace z DNS, pak požaduje po jmenném serveru věty RR podle zadaných požadavků. Klient může např. požadovat po jmenném serveru věty RR typu A s IP adresami zadанého doménového jména apod. Klientem může být buď resolver, nebo jmenný server, který sám neumí dotaz zodpovědět.

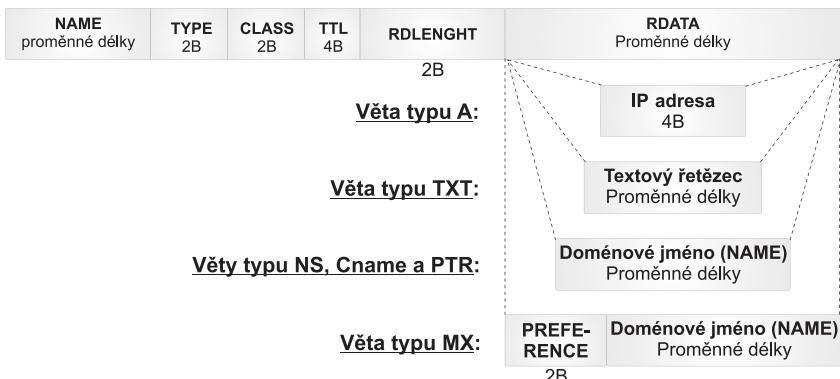
Všechny věty RR mají v protokolu DNS stejnou strukturu. Struktura věty RR je znázorněna na obr. 11.13.

Jednotlivá pole vět RR obsahují:

- ◆ **NAME** – Doménové jméno.
- ◆ **TYPE** – Typ věty (viz tab. 11.1).
- ◆ **CLASS** – Třída věty (vždy „IN“ = Internet).
- ◆ **TTL** – *Time to live*. 32bitové číslo udávající dobu, po kterou může být tento RR udržován v paměti serveru jako platný. Po vypršení této doby musí být věta považována za neplatnou. Hodnota 0 zabraňuje neautoritativním serverům uložit větu RR do paměti.
- ◆ **RDLENGTH** – 16bitové číslo specifikující délku pole RDATA.
- ◆ **RDATA** – Vlastní data ve tvaru řetězce proměnné délky. Formát tohoto pole závisí na typu a třídě věty RR.

Všimněte si, že formát věty RR je v protokolu DNS binární, tj. pro člověka nečitelný. Takto se přenáší věty RR počítacovou sítí protokolem DNS a my se s nimi setkáme v kap. 12 o samotném protokolu DNS. Uživatelé ale budou chtít zadávat věty RR textově a rovněž zónové soubory jsou textové. Převod z binárního do textového tvaru je jednoduchá konverze, kde se jednotlivá pole převedou do textového tvaru a oddělí se mezerou, tabelátem nebo kombinací těchto znaků. Jednotlivé řetězce v doménových jmenech se navíc oddělí tečkou. S tímto textovým formátem budeme pracovat dále v této kapitole.

#### Obecný formát



Obrázek 11.13: Struktura věty RR

Tabulka 11.1: Nejčastější typy vět RR

Typ	Anglický název	Význam pole RDATA
A	A host address	32-bitová IP adresa.
NS	Authoritative name server	Doménové jméno jmenného serveru, který je autoritou pro danou doménu.
CNAME	Canonical name for an alias	Doménové jméno specifikující synonymum k NAME.
SOA	Start Of Authority.	Právě jedna věta SOA uvozuje každou zónu. Obsahuje 7 polí. Přesný popis viz DNS databáze.

Typ	Anglický název	Význam pole RDATA
PTR	Domain name pointer	Doménové jméno. Věta se používá pro reverzní překlad.
HINFO	Host information	Obsahuje dva znakové řetězce. První obsahuje popis HW a druhý popis SW, které jsou používané na počítači NAME.
MX	Mail exchange	Obsahuje dvě pole. První 16bitové pole bez znaménka obsahuje preferenci a druhé obsahuje doménové jméno e-maillového serveru.
TXT	Text string	Textový řetězec s popisem.
AAAA	IP6 address	128bitová IP adresa (IP verze 6).
WKS	Well known service description	Popisuje obvyklých služeb serveru v protokolech TCP a UDP. Obsahuje 3 části: 32bitovou adresu, číslo protokolu, porty služeb.
SIG	Security signature	Podpisová věta, používaná při autentizaci v Secure DNS.
KEY	Security key	Veřejný klíč zóny používaný pro podepisování při autentizaci.
NXT	Next domain	Další doménové jméno. Autentizace neexistence doménového jména a typu.
A6	A6 host address	Může obsahovat až tři pole: délku prefixu, koncovou část IPv6 adresy a jméno prefixu.

## Databáze DNS

Databáze DNS jsou uloženy na primárním jmenném serveru v souborech. Správci DNS (hostmástr) udržují databáze DNS pro jednotlivé zóny zpravidla v textových souborech. Jednotlivé věty RR jsou v těchto textových souborech uvedeny tak, jak bylo popsáno v předešlém odstavci.

Při startu primárního jmenného serveru se jejich obsah databáze DNS nahraje do paměti jmenného serveru. Při startu sekundárního jmenného serveru se databáze DNS zkopiuje dotazem *zone transfer* z primárního jmenného serveru na sekundární jmenný server.

Databáze DNS na primárním jmenném serveru se skládá z jednotlivých souborů, které obsahují jednotlivé věty RR. Databáze na disku může obsahovat následující typy dat:

- ◆ Autoritativní data k obhospodařované zóně. Musí začínat větou typu SOA. Mohou být udržována pouze na primárním jmenném serveru. Sekundární jmenný server je obdrží pomocí dotazu *zone transfer* z primárního nebo jiného sekundárního jmenného serveru.
- ◆ Data umožňující přístup ke kořenovým jmenným serverům (zónu Cache/hint). Nezačínají větou SOA, proto se explicitně ve všech větách tohoto souboru musí uvést pole TTL. Jsou to neautoritativní data pro místní jmenný server. Musí být na každém jmenném serveru s výjimkou *forwarder only* serverů a kořenových serverů.
- ◆ Data, pomocí nichž deleguje jmenný server autoritu k subdoménám na jiné jmenné servery. K delegování autority se používají věty typu NS. Tato data jsou součástí nadřízené zóny, ke které je místní server autoritou. V případě, že se deleguje autorita na jmenný server, jehož doménové jméno je součástí delegované subdomény, je třeba k záznamu typu NS ještě přidat záznam typu A specifikující IP adresu tohoto jmenného serveru. Tyto přidávané věty typu A se anglicky nazývají *glue records*.

Obecná syntaxe jednotlivých řádků databáze (tzv. DNS vět) je:

```
[name] [TTL] třída typ Data_závislá_na_typu_věty
```

Pole v [ ] jsou nepovinná – jejich hodnoty se přejímají z předchozího záznamu (např. ze záznamu SOA). Komentáře jsou uvozeny středníky.

Význam jednotlivých polí:

- ◆ Pole name obsahuje doménové jméno. Může nastat několik situací:
  - Pole není vyplněné – pak se jeho hodnota bere z pole name předchozího záznamu.
  - V záznamu typu SOA může mít pole name hodnotu @. Takováto hodnota znamená, že se do pole name má dosadit jméno domény uvedené v konfiguračním souboru jmenného serveru.
  - Doménové jméno je uvedeno v poli name bez tečky na konci – pak se automaticky za toto jméno dodá jméno domény uvedené v předchozí větě SOA. V případě, že před větou (bez tečky na konci) je uveden příkaz \$ORIGIN, dodává se jméno domény uvedené v příkazu \$ORIGIN.
  - Doménové jméno je uvedeno v poli name s tečkou na konci – pak se jedná o tzv. absolutní jméno, které se bere tak, jak je napsáno.
- ◆ Pole TTL obsahuje v sekundách dobu života záznamu v paměti. Neautoritativní jmenné servery tuto hodnotu automaticky snižují. Dosáhne-li hodnota nuly, pak se záznam prohlásí za neplatný. Implicitně má pole hodnotu nula, avšak pokud předchází záznamu záznam typu SOA, pak se implicitní hodnota bere z pole TTL záznamu typu SOA. Záznam typu SOA je uveden vždy na počátku souboru, tj. nemusí nás záznam předcházet zcela bezprostředně.
- ◆ Třída je IN (Internet), HS (Hesiod) či CH (Chaos). V našem případě se budeme zabývat výhradně záznamy typu IN.
- ◆ Typ je jeden z typů uvedených v tab. 11.1.
- ◆ Poslední pole obsahuje data závislá na typu záznamu. Pokud se zde použije doménové jméno, pak nesmíme za ním zapomenout napsat tečku, protože v opačném případě by se za jméno automaticky dodalo jméno domény. Naopak pokud je zde IP adresa, tak za čtvrtou číslicí v IP adresě tečka být nesmí.

Jména v databázi musí začínat na první pozici. Pokud je prvním znakem řádku mezera, pak se použije jméno z předchozího řádku.

## SOA

Záznam typu **SOA** (*Start Of Authority*) – určuje jmenný server, který je autoritativním zdrojem informací pro danou doménu. Věta SOA je vždy právě jedna, a to na počátku konfiguračního souboru zóny.

Jako příklad uvedeme záznam pro server domény *pvt.cz*.

```
@ IN SOA cbu.pvt.cz. bindmaster.cbu.pvt.cz. (
    1 ;Serial
    86400 ;Refresh after 24 hours
    600 ;Retry after 5 min.
    120960 ;Expire after 2 weeks
    86400) ;Minimum TTL of 1 day
```

- ◆ Jméno @ musí začínat od první pozice na řádku a musí končit tečkou. Označuje název zóny. Zpravidla se zde místo jména zóny napíše @, který říká, aby se název zóny vzal z konfiguračního souboru jmenného serveru.
- ◆ IN označuje typ adresy (IN Internet).
- ◆ První jméno za SOA (*cbu.pvt.cz.*) je jméno počítače, na kterém jsou data uložena (jméno primárního jmenného serveru), a druhé jméno (*bindmaster.cbu.pvt.cz.*) definuje poštovní adresu osoby zodpovědné za data. Jelikož znak @ má v záznamu SOA zvláštní význam, tak se místo znaku @ v poštovní adrese píše tečka, tj. místo *bindmaster@cbu.pvt.cz* zde píšeme *bindmaster.cbu.pvt.cz.*
- ◆ Závorky umožňují pokračování záznamu na dalších řádcích (pro přehlednost).
- ◆ Serial označuje sériové číslo verze databázového souboru. Pokud soubor změníme, musíme zvětšit i toto sériové číslo. Doporučujeme zásadně používat číslo tvaru rrrrmmddč (rok, měsíc, den, číslo aktualizace v rámci dne).

Sekundární jmenný server se dotazuje primárního jmenného serveru pouze na záznam typu SOA. Porovná hodnotu pole serial v záznamu typu SOA a pouze v případě, že primární jmenný server má v záznamu typu SOA hodnotu pole serial větší, než má sekundární jmenný server, pak se provede přenos zóny z primárního jmenného serveru na sekundární jmenný server.

Takže pokud správce opraví databázi DNS na primárním jmenném serveru a opomene zvýšit hodnotu pole serial, pak se žádná sekundární data nepřenesou a změny se na sekundární jmenný server vůbec nedostanou. Pokud takovouto chybu správce primárního jmenného serveru objevíte, pak máte jedinou možnost: zrušit na sekundárním jmenném serveru soubor pro příslušnou zónu, program named na sekundárním jmenném serveru ukončit a znova jej nastartovat.

Hodnota pole serial neovlivňuje chování primárního jmenného serveru, tj. pokud pole opomenete na primárním jmenném serveru zvýšit, tak po restartu jmenného serveru se na primárním jmenném serveru změny provedou.

- ◆ Následující hodnoty specifikují různé časové údaje (v sekundách):

**Refresh** – udává, jak často mají sekundární servery ověřovat svá data. Zjistí-li při ověřování, že mají data s nižším serial, pak provedou protokolem TCP transfer zóny.

**Retry** – jestliže sekundární server nemůže kontaktovat primární po uplynutí intervalu refresh, bude to dále zkoušet každých retry sekund.

**Expire** – jestliže se sekundárnímu jmennému serveru nepodaří kontaktovat primární do expire sekund, přestane poskytovat informace (data jsou příliš stará). Musí platit: **Expire > Refresh**.

**TTL** – tato hodnota se vztahuje ke všem záznamům v databázovém souboru (jako výchozí hodnota) a je poskytována jmenným serverem v každé jeho odpovědi. Říká, jak dlouho mohou ostatní servery (tj. neautoritativní servery) udržovat daný záznam ve své paměti (nula znemožňuje ukládání vět do paměti).

Nevíte-li, co zadat do věty SOA, pak RFC-1537 doporučuje pro top level domény:

```
86400 ; Refresh 24 hours
7200 ; Retry 2 hours
2592000 ; Expire 30 days
345600 ; Minimum TTL 4 days
```

Pro ostatní domény:

```
28800 ; Refresh 8 hours
7200 ; Retry 2 hours
604800 ; Expire 7 days
86400 ; Minimum TTL 1 day
```

## Záznamy typu A

Záznamy typu A (*Address*) přiřazují doménovým jménům počítačů IP adresy. Za IP adresou nesmí být tečka.

### Příklad

Příklad 1:

pvt.cz.	IN	SOA	...
...			
www	IN	A	172.17.14.1
www.cbu	IN	A	172.17.18.1
muj.cbu.pvt.cz.	IN	A	172.17.14.2
tvuj	IN	A	10.1.1.3
...			

V příkladě 1 záznamu typu A přiřazují IP adresy počítačům: *www.pvt.cz*, *www.cbu.pvt.cz*, *muj.cbu.pvt.cz* a *tvuj.pvt.cz*.

## CNAME

Větami typu CNAME vytváříme synonyma k doménovým jménům. Často se říká, že vytváříme „aliasy k jménům počítačů“.

### Příklad

Příklad 2:

firma.cz.	IN	SOA	...
...			
mail	IN	A	192.1.1.2
www	IN	CNAME	mail.firma.cz.
...			

Příklad 2 popisuje situaci, kdy firma má jeden počítač *mail.firma.cz*, který chce také používat jako WWW server.

Na pravé straně příkazu CNAME musí být doménové jméno, kterému je přiřazena IP adresa záznamem typu A. Na pravé straně nesmí být synonymum, tj. CNAME nesmí ukazovat na CNAME. Příklad 3 ukazuje chybnou delegaci jmen.

### Příklad

Příklad 3 (chybný!):

firma.cz.	IN	SOA	...
...			
mail	IN	A	192.1.1.2
www	IN	CNAME	mail.firma.cz.
server	IN	CNAME	www.firma.cz.
...			

V záznamech typu CNAME se zásadně snažíme na pravé straně zapisovat úplné doménové jméno s tečkou na konci. V případě, že tečku neuvedeme, dodá se ještě jméno domény. Toho se sice v malých databázích dá využít, ale jak velikost databáze roste, tak se stává nepřehlednou a případné chyby tohoto druhu se někdy i těžko dohledávají.

### HINFO a TXT

Záznamy typu HINFO a TXT jsou pouze informativní záznamy.

Záznam typu HINFO má ve své datové části dva údaje. Prvním údajem je informace o hardwaru a druhým údajem informace o softwaru.

Záznam typu TXT obsahuje ve své datové části obecný textový řetězec.

### Příklad

Příklad 4:

firma.cz.	IN	SOA	...
...			
mail	IN	A	192.1.1.2
	IN	HINFO	AlphaServer UNIX
	IN	TXT	Muj server
...			

### NS

Záznamy typu NS definují autoritativní jmenné servery pro doménu. Na pravé straně musí být doménové jméno, kterému je přiřazena IP adresa větou typu A. Na pravé straně nesmí být synonymum, tj. záznam typu NS nesmí ukazovat na záznam typu CNAME.

Stejné záznamy typu NS jsou vždy ve dvou databázích:

1. V databázi zóny vyšší úrovně. Těmito záznamy typu NS je delegována pravomoc na jmenný server nižší úrovně. V případě, že jméno jmenného serveru nižší úrovně samo leží v doméně nižší úrovně, musí být za tento záznam typu NS přidán ještě záznam typu A s IP adresou jmenného serveru. To je nutné proto, že jmenný server vyšší úrovně musí IP adresu jmenného serveru nižší úrovně uvádět v dodatečných informacích v DNS odpovědi – aby bylo možné se na jmenný server nižší úrovně vůbec dostat.

2. Na autoritativním jmenném serveru pro zónu (tj. podle terminologie předchozího bodu na jmenném serveru „nižší úrovně“).

### Příklad

Příklad 5:

Autoritativní jmenný server zóny vyšší úrovně *firma.cz* deleguje pravomoc pro doménu *cbu.firma.cz* na server *ns.cbu.firma.cz*. Jelikož je podřízený jmenný server prvkem podřízené domény, je nutné do nadřízené zóny přidat neautoritativní záznam typu A pro počítač *ns.cbu.firma.cz*:

<i>firma.cz.</i>	IN	SOA	...
	IN	NS	<i>ns.provider.cz.</i>
	IN	NS	<i>ns.firma.cz</i>
<i>ns</i>	IN	A	11.1.1.1
<i>cbu</i>	IN	NS	<i>ns.firma.cz.</i>
<i>ns.cbu</i>	IN	NS	<i>ns.cbu.firma.cz.</i>
	IN	A	11.2.2.2
	...		

Jmenný server domény *cbu.firma.cz*, tj. autoritativní jmenný server domény nižší úrovně, má k dispozici databázi:

<i>cbu.firma.cz.</i>	IN	SOA	...
	IN	NS	<i>ns.firma.cz.</i>
	IN	NS	<i>ns.cbu.firma.cz.</i>
<i>ns</i>	IN	A	11.2.2.2
	...		

Opět musíme zdůraznit, že na pravé straně vět typu NS je dobré psát úplná doménová jména s tečkou na konci.

### MX

Záznamy typu MX specifikují poštovní server domény. V drtivé většině případů totiž nechceme mít e-mailovou adresu tvaru:

*uživatel@počítač.firma.cz*,

ale pouze tvaru:

*uživatel@firma.cz*,

tj. přejeme si skrýt jméno poštovního serveru.

Záznam typu MX ukazuje, na jaký počítač má být pro doménu dopravena pošta. Navíc je v záznamu typu MX číselná priorita, pomocí které lze určit několik počítačů, na něž může být pošta pro doménu posílána. Nejprve se zkouší odeslat pošta na počítač s nejvyšší prioritou (nejnižším číslem), když se to nepodaří, pak na další počítač (s vyšším číslem) atd. Příklad 6 popisuje situaci, kdy pošta pro doménu *firma.cz* má být doručována na počítač *mail.firma.cz*. Pokud tento počítač není dostupný, pak se pošta odešle na počítač *mail1.provider.cz*, kde vyčká do dostupnosti počítače *mail.firma.cz*. Pokud ani počítač *mail1.provider.cz* není dostupný, pak se pošta odešle na počítač *mail2.provider.cz*.

### Příklad

Příklad 6:

```

firma.cz.           IN      SOA     ...
                   IN      MX      30 mail2.provider.cz.
                   IN      MX      20 mail1.provider.cz.
                   IN      MX      10 mail.firma.cz.
mail               IN      A       11.1.1.8
...

```

### PTR

Záznam typu PTR slouží k překladu IP adresy na doménové jméno, tj. k překladu prvků domény in-addr.arpa na jméno počítače.

### Příklad

Příklad 7:

Záznamy typu PTR pro počítač *ns.firma.cz* o IP adrese 195.47.200.1 a pro počítač *www.firma.cz* o IP adrese 195.47.200.201:

```

1.200.47.195.in-addr.arpa.    IN      PTR      ns.firma.cz.
201.200.47.195.in-addr.arpa.   IN      PTR      www.firma.cz.

```

Jenže takovýto příklad je zcela vytržený z kontextu. Ve skutečnosti je třeba si uvědomit celý mechanismus delegací. Nás příklad je podrobně popsán v příkladu 8.

### Příklad

Příklad 8:

Předpokládejme, že naší firmě (*firma.cz*) jsou přiděleny IP adresy v rozsahu 195.47.200.0/24, tj. celá síť třídy C. Pak pro reverzní překlad musí být:

1. **Na kořenových serverech Internetu** je provedena delegace zóny 195.in-addr.arpa, na jmenovité servery pro Evropu – ns.ripe.net. Pokud by kořenové servery používaly program named verze 4, pak by delegace byla provedena:

- A. V souboru named.boot by byl uveden řádek:

```
primary          .          root.db
```

- B. V souboru root.db by mj. bylo uvedeno:

195.in-addr.arpa	IN	NS	ns.ripe.net
ns.ripe.net	IN	A	193.0.0.193
195.in-addr.arpa	IN	NS	ns.apnic.net
ns.apnic.net	IN	A	203.37.255.97
195.in-addr.arpa	IN	NS	munnari.oz.au
munnari.oz.au	IN	A	128.250.22.2
	IN	A	128-.250.1.21
...			

(Zóna 195.in-addr.arpa je tak důležitá, že tč. má 7 autoritativních serverů rozložených po celé Zemi.)

**2. Na jmenném serveru ns.ripe.net, tj. jmenný server vyšší úrovně pro Evropu (Amsterdam):**

- A. V souboru named.boot bude uveden např. řádek:

```
primary      195.in-addr.arpa      195.rev
```

- B. V souboru 195.rev je pak např. uvedeno:

195.in-addr.arpa.	IN	SOA	...
...			
200.47	IN	NS	ns.firma.cz.
	IN	NS	ns.provider.cz.
...			

**3. Na jmenném serveru ns.firma.cz (primární jmenný server):**

- A. V souboru named.boot bude uveden např. řádek:

```
primary      200.47.195.in-addr.arpa 200.47.195.rev
```

- B. V souboru 200.47.195.rev je pak např. uvedeno:

200.47.195.in-addr.arpa.	IN	SOA	...
	IN	NS	ns.firma.cz.
	IN	NS	ns.provider.cz.
1	IN	PTR	ns.firma.cz.
201	IN	PTR	www.firma.cz.
...			

**4. Na jmenném serveru ns.provider.cz (sekundární jmenný server):**

V souboru named.boot bude uveden např. řádek:

```
secondary 200.47.195.in-addr.arpa 195.47.200.1 200.47.195.rev
```

Je třeba opět připomenout, že se nesmí zapomínat psát tečky za jménem počítače (na pravé straně), protože v případě opomíjení tečky se dodá doména končící in-addr.arpa, což je opravdu nepoužitelné.

Asi čekáte, že také zdůrazním, že na pravé straně nesmí být synonymum (CNAME), tj. že záznam typu PTR nesmí ukazovat na větu typu CNAME. Avšak není tomu tak. Dlouhá léta to bylo považováno za chybu v systému BIND, až se to stalo velice užitečnou pomůckou, a posléze dokonce normou RFC-2317. Využití tohoto mechanismu se věnuje kap. 16.

## Věta typu SRV

Věta typu SRV byla experimentálně zavedena v RFC-2052 a v RFC-2782 pak byla zavedena definativně. Zásadní rozdíl mezi oběma normami spočívá zejména v tom, že RFC-2782 předřazuje znak podtržitko ( \_ ) před název služby a název protokolu. Věty typu SRV využívají i systémy Microsoft.

Cílem věty SRV je v databázi DNS udržovat nejen jména počítačů, ale i jména služeb. Již jsme se s takovým případem setkali – šlo o věty MX, kde službou byla elektronická pošta. Věty MX specifikují poštovní servery, na které se má pošta zasílat. Pomocí priority je vyjádřeno, který poštovní server má být kontaktován jako první a které poštovní servery následně.

Zastavme se u případu WWW serveru. V praxi, pokud chceme získat nějaké informace o české firmě, tak do prohlížeče napišeme:

`http://www.firma.cz`

tj. chceme protokolem HTTP kontaktovat WWW server z domény. Jenže to je jen nepsané pravidlo, že webové servery se jmennují WWW.

Věta typu SRV systematicky zavádí do DNS informace k nalezení příslušného webového serveru. V našem případě budeme v DNS hledat jméno (protokol HTTP ke svému transportu využívá protokol TCP):

`http.tcp.www.firma.cz.`

Syntaxe věty SRV je (kód v protokolu DNS má pro větu typu SRV hodnotu 33):

`_Služba._Protokol.doménové-jméno [TTL] IN SRV Priorita Váha Port Počítač.`

Kde:

**Služba** specifikuje symbolické jméno služby (serveru). Např. ldap, http, smtp atd.

**Protokol** specifikuje protokol, např. tcp či udp.

**Priorita** určuje prioritu. Firma může provozovat několik WWW serverů, aby v případě výpadku byl vždy nějaký server dostupný. Firma bude chtít zavést prioritu, který server se má klient pokoušet kontaktovat jako první a který jako další.

**Váha** – firma může mít webový server extrémně zatížen, proto zřídí několik paralelně běžících webových serverů o stejné prioritě. Každý bude spuštěn na počítači o jiném výkonu. Zavádí se proto váha (závaží ve smyslu váženého průměru). V DNS jsou např. dva záznamy:

`_http._tcp.www.firma.cz. IN SRV 10 1 80 server1.firma.cz.`

`IN SRV 10 3 88 server2.firma.cz.`

Jelikož oba záznamy mají stejnou prioritu (10), tak v případě, že oba servery jsou dostupné, může klient náhodně kontaktovat libovolný z nich. Avšak `server2.firma.cz` je výkonnější než `server1.firma.cz`. Váha proto sděluje, že servery mají být kontaktovány sice náhodně, ale při velkém počtu navazovaných spojení má být 25 % spojení navázáno se `server1.firma.cz` a 75 % spojení se `server2.firma.cz`, tj. `server2` je třikrát výkonnější než `server1`.

Váha nula je určena pro administrátory, tj. neprovádí se žádné vyrovnávání výkonu mezi počítači.

**Port** specifikuje port, na kterém server běží.

**Počítač** specifikuje jméno počítače (odkaz na větu typu A), na kterém je služba poskytována (na kterém běží server). Pokud je jako počítač uvedena pouze tečka, pak služba není poskytována.

## Příklad

Příklad 9:

```
$ORIGIN      firma.cz.  
@           IN      SOA ...  
             IN      NS  ...
```

...  
; následující řádky specifikují, že protokolem Telnet má být kontaktován bu server1  
; nebo server2. Server2 je třikrát výkonnější.

```

_telnet._tcp      IN      SRV      0      1      23      server1.firma.cz.
                  IN      SRV      0      3      23      server2.firma.cz.
; v případě, že není dostupný ani server1 ani server2, pak administrátor má kontaktovat
; server3:
                  IN      SRV      10     0      23      server3.firma.cz.
; Jsou provozovány dva WWW-servery. Klient má kontaktovat server1 a v případě
; nedostupnosti počítače server1 má kontaktovat server2, ale na portu 88:
_http._tcp        IN      SRV      0      0      80      server1.firma.cz.
                  IN      SRV      5      0      88      server2.firma.cz.
; Jelikož je zvykem v případě protokolu HTTP psát www před jméno domény,
; tak přidáme ještě:
_http._tcp.www   IN      SRV      0      0      80      server1.firma.cz.
                  IN      SRV      5      0      88      server2.firma.cz.
; Nesmíme zapomenout na věty typu A. Raději znakem @ specifikujeme aktuální
; doménu (aby se jméno nevzalo z předchozí věty):
@                 IN      A       10.1.1.2
                  IN      A       10.1.1.2
; Pochopitelně uvedeme i věty typu A jednotlivých serverů:
server1          IN      A       10.1.1.1
server2          IN      A       10.1.1.2
server3          IN      A       10.1.1.3
; Ostatní služby nejsou podporovány:
*.tcp            IN      SRV      0      0      0      .
*.tcp            IN      SRV      0      0      0      .

```



**Poznámka:** V celé publikaci jsme se důsledně snažili vyhnout používání hvězdiček v doménovém jméně. V praxi jsme si totiž mnohokrát ověřili, že hvězdičky jsou ve jméně zdrojem neočekávaných chyb. Používají se proto pouze u vět typu MX a snad v budoucnu u vět typu SRV.

Jak taková hvězdička pracuje? Představme si případ věty typu A:

```
*.firma.cz      IN      A       10.1.1.10
```

Pak na jakýkoliv dotaz na prvek domény *firma.cz* explicitně neuvedený v DNS bude DNS odpovídat, že má adresu 10.1.1.10, tj. *pocitac1.firma.cz* má adresu 10.1.1.10, *pocitac2.firma.cz* má také adresu 10.1.1.10 atd. I kdybychom si přáli, aby tomu tak bylo, pak v případě, že se splèteme a místo *pocitac1.firma.cz* napíšeme *pocitac2.firma.cz*, nám DNS nevrátí, že jsme se spletli, ale vrátí adresu, ale s největší pravděpodobností jinou, než čekáme.

## §ORIGIN

V parametru name databázového záznamu se uvádí doménové jméno buď absolutně (s tečkou na konci), nebo relativně (bez tečky na konci). Právě za relativní doménové jméno se automaticky přidává implicitní doména. Příkaz \$ORIGIN slouží pro změnu implicitní domény. Databáze DNS může obsahovat příkaz:

```
$ORIGIN implicitní_doména
```

Od tohoto okamžiku je implicitní doména změněna na hodnotu uvedenou jako první parametr příkazu \$ORIGIN.

V případě, že se uvede relativní jméno, pak se doplňuje na úplné jméno přidáním domény definované v záznamu typu SOA nebo parametrem příkazu \$ORIGIN, který předchází databázový záznam. Příkaz \$ORIGIN tak mění implicitně nastavenou doménu.

Není-li změněna implicitní doména příkazem \$ORIGIN, pak se bere doména z příkazu SOA. Je-li v příkazu SOA místo domény znak @, pak se bere první parametr příkazu primary, resp. secondary ze souboru /etc/named.boot.

## \$INCLUDE

Do zdrojového souboru na disku je možné vložit další soubor příkazem:

```
$INCLUDE soubor
```

Soubor se vloží na místo příkazu. Je možné uvést ještě:

```
$INCLUDE soubor implicitní_doména
```

Tím se nejenom vloží soubor, ale změní se i implicitní doména. Změna implicitní domény platí jen pro řádky vloženého souboru.

# Rozšíření DNS pro IP verze 6

Rozšířením DNS pro IP verze 6 se věnuje RFC-1886, které je později doplněno a částečně nahrazeno RFC-2874.

## Záznam typu AAAA

IP verze 4 používá pro překlad jména na IP adresu záznam typu A. Pro IP verze 6 byl nejprve zaveden obdobný záznam typu AAAA. Rozdíl spočívá v tom, že záznam typu AAAA má v poli IP adresa nikoliv čtyřbajtovou, ale 32bajtovou adresu IPv6.

## Záznam typu A6

RFC-2874 nahrazuje větu AAAA větou A6.

Věta A6 je používána pro uzly, které používají pro svou adresaci IP verze 6. Pole RDATA věty A6 má tam, kde je u věty A a AAAA uvedena IP adresa, například takovýto tvar:

```
64      ::1244:67E3:589A:9ABC    subneta.isp.cz
```

kde:

číslo 64	je tzv. délka prefixu (počet jedniček v síťové masce)
::1244:67E3:589A:9ABC	je tzv. koncová část adresy (suffix)
isp.cz	je tzv. jméno prefixu

celá věta A6 pak může vypadat takto:

```
www      IN      A6      64      ::1244:67E3:589A:9ABC    subneta.isp.cz
```

Vysvětlete si nyní význam nových částí věty A6.

**Koncová část (suffix) adresy** je 1 až 8 dvojic bajtů z IP adresy zprava.

**Jméno prefixu** je textové označení (DNS jméno) definované pro zbylou část adresy zleva. Toto jméno prefixu je definováno jinou větou A6 v příslušné zóně. V našem případě v zóně pro doménu *isp.cz*.

```
subneta      IN      A6      0      36AB:12:90A4:56::
```

**Délka prefixu** je číslo 0 až 128 označující počet bitů prefixu adresy, která odpovídá jménu prefixu.

Pokud je délka prefixu 0, není jméno prefixu ve větě uvedeno a věta A6 pak má tento tvar:

```
www      IN      A6      0      36AB:12:90A4:56:1244:67E3:589A:9ABC
```

Jak je vidět z uvedeného příkladu, jedna adresa IP verze 6 je v DNS uložena pomocí několika vět A6, přitom každá z vět obsahuje část IP adresy. Pokud chce resolver přeložit jméno dns na adresu IP verze 6, musí získat nikoli jednu větu, jak tomu je u typu A a AAAA, ale několik vět typu A6. Informace z těchto vět pak musí resolver poskládat dohromady tak, aby výsledkem byla platná IP adresa. Mechanismus, který resolver používá k sestavení IP adresy ze získaných vět A6, se označuje **A6 record chains**, tj. řetězení informací z vět A6.

## Příklad

Příklad 10:

Pro osvětlení mechanismu A6 record chains si uvedeme ještě jeden příklad. Předpokládejme, že firma se jménem FIRMA a.s. je k Internetu připojena prostřednictvím providera ISPA, který má přidělenu subsít 2435:00A1:BA00::/40. Firmě FIRMA a.s. provider přidělil subsít 2435:00A1:BA01::/48.

FIRMA a.s. má svůj jmenný server s adresou 2435:00A1:BA01:1:1234:5678:1 a WWW server s adresou 2435:00A1:BA01:1:1234:5678:2. FIRMA a.s. používá doménu *firma.cz*.

V DNS budou uvedeny tyto záznamy:

```
$ORIGIN FIRMA.CZ
NS      IN      A6      48      ::1:1:1234:5678:1      FIRMA-NET.ISPA.CZ
WWW     IN      A6      48      ::1:1:1234:5678:2      FIRMA-NET.ISPA.CZ
$ORIGIN ISPA.CZ
FIRMA-NET IN      A6      40      0:0:0001::          PHA.ISPA.CZ
PHA      IN      A6      0       2435:00A1:BA00::
```

Připomeňme, že takovýto tvar bude mít i věta glue uváděná v nadřízené doméně. Ve větách A6 pro jmenné servery je doporučováno uvádět plnou IP adresu bez použití A6 record chains. Pokud uzel používá adresu IP verze 4, pak není vhodné tuto adresu mapovat na IP verze 6, ale je naopak doporučováno uvést v DNS přímo adresu IP verze 4 ve větě typu A.

## Příklad

Příklad 11:

```
NS1      IN      A6      0      4EE8::55:6:78E:1234:6578
NS2      IN      A      195.168.16.1
```

## Reverzní domény

Pro reverzní překlad byla nejprve zavedena nová doména IP6.INT. V listopadu 2001 pak IANA pro reverzní překlad zaregistrovala doménu IP6.ARPA. Doména je obdobou domény IN-ADDR.ARPA pro IP verze 4.

### IP6.INT

V doméně IP6.INT se prvky zapisují v tzv. „nibble“ formátu. Jednotlivé bajty IP adresy se opět zapisují „pozpátku“, ale nikoliv celé bajty, ale poloviny bajtů. Polovina bajtu je reprezentovaná jednou šestnáctkovou číslicí. Jednotlivé šestnáctkové číslice se oddělují tečkou (tj. oddělovačem v doménovém jméně).

#### Příklad

Příklad 12: IP adresa 4321::1:2:3:4:567:89AB bude jako prvek domény IP6.INT zapsána:

B.A.9.8.7.6.5.0.4.0.0.0.3.0.0.0.2.0.0.0.1.0.0.0.0.0.0.1.2.3.4.IP6.INT.

(IP adresa 4321::1:2:3:4:567:89AB je jen zkráceným zápisem adresy 4321:0000:0001:0002:0003:0004:0567:89AB.)

### IP6.ARPA

V doméně IP6.ARPA se prvky zapisují v tzv. „bitstring“ formátu.

#### Příklad

Příklad 13: IP adresa 4321::1:2:3:4:567:89AB bude jako prvek domény IP6.ARPA zapsána:

\ [x43210000001000200030004056789AB/128].IP6.ARPA.

Všimněte si, že zápis prvku v doméně IP6.ARPA začíná zpětným lomítkem, sekvence číslic z IP adresy je uzavřena v hranatých závorkách a uvozena znakem „x“. Pořadí číslic v prvku je stejně jako v IP adrese. Za lomítkem v hranatých závorkách je uveden počet bitů IP adresy.

Následující zápis označuje také prvek reverzní domény z předchozího příkladu.

\ [x43210000/32].\ [x0001/16].\ [000200030004056789AB/80].IP6.ARPA.

## Záznam typu DNAME

Věta DNAME je analogií věty CNAME. Věta DNAME umožňuje pojmenovat podstrom ve stromové struktuře doménových jmen.

Věty DNAME mohou mít takovouto podobu:

pha.firma.ispa.cz	IN	DNAME	firma.cz
\ [x43210000/32]	IN	DNAME	plz-rev.ispb.com

Pro delegaci reverzních domén se již nepoužívají NS-věty, ale na místo klasické delegace se používá právě posloupnosti vět DNAME. Použití vět DNAME zmenšuje počet zónových souborů používaných pro reverzní delegace.

Mechanismus použití vět DNAME pro delegaci reverzních domén si vysvětlíme na situaci firmy FIRMA a.s. z části 13.5.2.

Pro reverzní překlad musí v DNS být tyto záznamy:

```
$ORIGIN IP6.ARPA
\ [x243500A1BA /40] IN DNAME IP6-REV.ISPA.CZ
$ORIGIN IP6-REV.ISPA.CZ
\ [01/8] IN DNAME FIRMA-REV.FIRMA.CZ
$ORIGIN FIRMA-REV.FIRMA.CZ
\ [00010001123456780001/80] IN PTR NS.FIRMA.CZ
\ [00010001123456780002/80] IN PTR WWW.FIRMA.CZ
```

Kroky resolveru při pokusu o překlad IP adresy 2435:00A1:BA01:1:1234:5678:2 na jméno:

Dotaz:

```
\ [x243500A1BA0100010001123456780002/128].IP6.ARPA
```

na server pro doménu IP6.ARPA

Odpověď:

```
\ [x243500A1BA /40].IP6.ARPA DNAME IP6-REV.ISPA.CZ
```

Dotaz:

```
\ [x0100010001123456780002/88].IP6-REV.ISPA.CZ
```

na server pro doménu IP6-REV.ISPA.CZ

Odpověď:

```
\ [01/8] DNAME FIRMA-REV.FIRMA.CZ
```

Dotaz:

```
\ [x00010001123456780002/80]. FIRMA-REV.FIRMA.CZ
```

na server pro doménu FIRMA-REV.FIRMA.CZ

Odpověď:

```
\ [x00010001123456780001/128]. FIRMA-REV.FIRMA.CZ PTR NS.FIRMA.CZ
```

## Nástroje pro sledování DNS

Pokud chceme pochopit, jak pracuje DNS, tak nejlepší metodou je naučit se DNS sledovat alespoň pomocí programu nslookup nebo dig.

### Program nslookup

Nejčastěji používaným programem pro kontrolu DNS je program nslookup. Tento program má jednu velkou výhodu. Je dnes totiž součástí balíku TCP/IP na Unixu i Windows, a nemusíme jej tedy nikde shánět a kompilovat.

Programem nslookup posíláme dotazy DNS na server DNS a kontrolujeme, zda server DNS odpovídá správně. Pomocí programu nslookup můžeme vystupovat v roli resolveru a požadovat po jmeném serveru konečnou odpověď na náš dotaz. Nebo můžeme pomocí programu nslookup simulovat

chování jmenného serveru, který coby klient komunikuje s jiným jmenným serverem (tj. požadovat jen částečné odpovědi). Záleží na tom, co chceme pomocí programu nslookup testovat.

Program nslookup posílá dotazy DNS implicitně na jmenný server, který je na systému nastavený jako resolver. V UNIXu tedy posílá dotazy na jmenný server uvedený v souboru /etc/resolv.conf.

Program nslookup spustíme v interaktivním režimu příkazem nslookup, výsledkem spuštění programu je např. prompt:

>

Na výzvu (*prompt*) > zadáme svůj dotaz. Ptát se můžeme např. na IP adresu nebo jméno nějakého uzlu.

Zadám-li na prompt jméno uzlu, např. www.prf.jcu.cz, pokusí se jmenný server zjistit příslušnou IP adresu.

*Dotaz:*

```
> www.prf.jcu.cz  
Server: 160.217.1.10  
Address: 160.217.1.10#53
```

*Odpověď:*

```
www.prf.jcu.cz canonical name = botanika.bf.jcu.cz.  
Name: botanika.bf.jcu.cz  
Address: 160.217.208.33
```

(V prvním rádku jsme se dozvěděli, že www.prf.jcu.cz je CNAME počítače botanica.bf.jcu.cz. Dále je pak uvedena IP adresa počítače botanica.bf.jcu.cz.)

Zadám-li na prompt IP adresu např. 160.217.208.33, pokusí se default server zjistit doménové jméno uzlu s touto IP adresou.

*Dotaz:*

```
> 160.217.208.33  
Server: 160.217.1.10  
Address: 160.217.1.10#53
```

*Odpověď:*

```
Name: www.pvt.cz  
33.208.217.160.in-addr.arpa name = botanika.bf.jcu.cz.  
>
```

Jak je zřejmé z uvedených příkladů, program nslookup implicitně hledá v DNS odpovídající záznam A nebo záznam PTR. Programem nslookup se však můžeme zeptat jmenného serveru na libovolný záznam RR. Typ záznamu, který nás zajímá, definujeme v programu nslookup pomocí příkazu set querytype=typ\_záznamu, který je možné zkrátit na set q=typ\_záznamu.

Použití si opět ukážeme na příkladu. Tentokrát nás bude zajímat seznam serverů, na které je směrována pošta pro doménu jcu.cz. Již víme, že směrování pošty je definováno větami MX v zónovém souboru příslušné domény. Zajímají nás tedy všechny věty typu MX. Nastavíme tedy požadovaný typ vět MX takto:

*Dotaz:*

```
> jcu.cz  
Server: 160.217.1.10  
Address: 160.217.1.10#53
```

*Odpověď:*

```
jcu.cz mail exchanger = 12849 dx.spamfree.cz.  
jcu.cz mail exchanger = 12801 ax.virusfree.cz.  
jcu.cz mail exchanger = 12817 bx.virusfree.cz.  
jcu.cz mail exchanger = 12833 cx.spamfree.cz.  
>
```

Pro doménu `jcu.cz` je pošta směrována k poskytovateli ochrany proti škodlivému obsahu (viz str. 469).

Další případ, kdy se nslookup často používá, je zjištění autoritativních serverů pro doménu. Zajímají nás tentokrát jména jmenných serverů, které se o danou doménu starají. Požadovanou informaci jednoduše získáme, když nastavíme typ věty na NS:



**Tip:** Pokud příkazem nslookup chceme zjistit veškeré údaje k danému jménu, nastavíme `q=any`.

*Dotaz:*

```
> jcu.cz  
Server: 160.217.1.10  
Address: 160.217.1.10#53
```

*Odpověď:*

```
jcu.cz nameserver = eros.zcu.cz.  
jcu.cz nameserver = virgo.jcu.cz.  
jcu.cz nameserver = taurus.zf.jcu.cz.
```

Doména `jcu.cz` využívá 3 autoritativní jmenné servery.

## Domácí cvičení:

1. Zjistěte autoritativní jmenné servery pro vaši doménu.
2. Zjistěte autoritativní jmenné servary pro doménu `cz`.

Podobně si můžeme zjistit i všechny kořenové servery Internetu. Stačí si uvědomit, že ty jsou autoritativními servery pro doménu tečka:

*Dotaz:*

```
> set q=ns  
> .  
Server: 160.217.1.10  
Address: 160.217.1.10#53
```

*Odpověď:*

*Non-authoritative answer:*

```
. nameserver = H.ROOT-SERVERS.NET.  
. nameserver = I.ROOT-SERVERS.NET.  
. nameserver = J.ROOT-SERVERS.NET.  
. nameserver = K.ROOT-SERVERS.NET.  
. nameserver = L.ROOT-SERVERS.NET.  
. nameserver = M.ROOT-SERVERS.NET.  
. nameserver = A.ROOT-SERVERS.NET.  
. nameserver = B.ROOT-SERVERS.NET.  
. nameserver = C.ROOT-SERVERS.NET.  
. nameserver = D.ROOT-SERVERS.NET.  
. nameserver = E.ROOT-SERVERS.NET.  
. nameserver = F.ROOT-SERVERS.NET.  
. nameserver = G.ROOT-SERVERS.NET.
```

*Authoritative answers can be found from:*

```
B.ROOT-SERVERS.NET internet address = 192.228.79.201  
C.ROOT-SERVERS.NET internet address = 192.33.4.12  
D.ROOT-SERVERS.NET internet address = 128.8.10.90  
E.ROOT-SERVERS.NET internet address = 192.203.230.10  
F.ROOT-SERVERS.NET internet address = 192.5.5.241  
F.ROOT-SERVERS.NET has AAAA address 2001:500:2f::f  
G.ROOT-SERVERS.NET internet address = 192.112.36.4  
H.ROOT-SERVERS.NET internet address = 128.63.2.53  
H.ROOT-SERVERS.NET has AAAA address 2001:500:1::803f:235  
I.ROOT-SERVERS.NET internet address = 192.36.148.17  
J.ROOT-SERVERS.NET internet address = 192.58.128.30  
J.ROOT-SERVERS.NET has AAAA address 2001:503:c27::2:30  
K.ROOT-SERVERS.NET internet address = 193.0.14.129  
K.ROOT-SERVERS.NET has AAAA address 2001:7fd::1
```

Všiměte si, že některé servery mají i adresy IPv6.

## Ladicí režim

Při hledání chyby v konfiguraci nám často nestačí informace běžně vypisované programem nslookup a chceme vědět více. Použijeme tedy ladicí režim programu. U programu nslookup je možné nastavit dvě úrovně ladicího režimu: režim debug a režim d2. Ladící úrovně se nastavují příkazem set.

## Ladicí úroveň debug

Ladicí úroveň debug vypisuje detailní informace z přicházejících paketů DNS.

Nastavení ladicí úrovni debug:

```
> set debug
```

Vezměte-li si k ruce kapitolu 12 (Protokol DNS), pak je výstup ladicího režimu docela dobře čitelný. Ve výpisu jsou jednotlivé sekce uvozeny nadpisem. Do výpisu je pro lepší orientaci autorem doplněn český komentář.

Použití si ukážeme na příkladu. Zajímá nás IP adresa serveru www.ripe.net:

*Dotaz:*

```
> set debug  
> www.ripe.net  
Server: 160.217.1.10  
Address: 160.217.1.10#53
```

*Odpověď (skládá se ze sekcí Questions, Answers, Authority a Additional – viz odstavec 12.1):*

```
QUESTIONS:  
    www.ripe.net, type = A, class = IN  
ANSWERS:  
-> www.ripe.net  
    canonical name = kite-www.ripe.net.  
-> kite-www.ripe.net  
    internet address = 193.0.0.214  
AUTHORITY RECORDS:  
-> ripe.net  
    nameserver = ns-ext.isc.org.  
-> ripe.net  
    nameserver = ns-pri.ripe.net.  
-> ripe.net  
    nameserver = ns3.nic.fr.  
-> ripe.net  
    nameserver = sunic.sunet.se.  
ADDITIONAL RECORDS:  
-> ns3.nic.fr  
    internet address = 192.134.0.49  
-> ns3.nic.fr  
    has AAAA address 2001:660:3006:1::1:1  
-> sunic.sunet.se  
    internet address = 192.36.125.2  
-> sunic.sunet.se  
    has AAAA address 2001:6b0:7::2  
-> ns-ext.isc.org  
    internet address = 204.152.184.64  
-> ns-ext.isc.org  
    has AAAA address 2001:4f8:0:2::13  
-> ns-pri.ripe.net  
    internet address = 193.0.0.195  
-> ns-pri.ripe.net  
    has AAAA address 2001:610:240:0:53::3  
-----  
Non-authoritative answer:  
www.ripe.net canonical name = kite-www.ripe.net.  
Name: kite-www.ripe.net  
Address: 193.0.0.214
```

## Ladicí úroveň d2

Ladicí úroveň d2 detailně vypisuje obsah odcházejících paketů (dotazů) i příchozích paketů (odpovědí). Použitím ladicí úrovně d2 získáme úplný přehled o komunikaci resolveru se jmenným serverem, jehož vypovídací schopnost je téměř shodná s výstupy programu Ethereal.

Nastavení ladící úrovně d2:

```
> set d2
```

## Změna implicitního jmenného serveru

DNS paket s dotazem můžeme pomocí programu nslookup poslat na libovolný jmenný server. Jméno testovaného serveru nastavíme příkazem `server`.

```
> server ns.internic.cz
```

Po použití tohoto příkazu bude všechny následně zadané DNS dotazy řešit nově nastavený server, tedy v našem případě server ns.internic.net.

Toto nastavení je velice praktické, protože většinou se nám náš místní jmenný server z naší LAN jeví jako korektně pracující. Takže jistotu získáme, až si náš jmenný server ověříme z pohledu jiného jmenného serveru.

## Simulace dotazů od jmenného serveru

Chceme-li simulovat komunikaci mezi jmennými servery, musíme potlačit dvě implicitní nastavení programu nslookup.

Program nslookup implicitně používá searchlist, tedy podobně jako resolver přidává implicitní doménu za doménové jméno, které není ukončeno tečkou. Toto chování zakážeme příkazem:

```
> set nosearch
```

Nslookup implicitně požaduje po jmenném serveru rekurzivní, tedy konečnou odpověď. Víme, že jmenné servery si mezi sebou posílají nerekurzivní odpovědi, a proto opět toto chování musíme potlačit, tentokrát příkazem:

```
> set norecurse
```

## Dig

Dalším ze známějších používaných programů pro kontrolu DNS je program `dig`. Tento program nenajdeme na Windows, ale zato je součástí Linuxu.

Program `dig` posílá na jmenný server paket dotazy DNS a zobrazuje uživateli odpovědi jmenného serveru. Uživatel může určit, který server má dotaz zodpovědět, jaké informace jej zajímají a specifikovat dodatečné podmínky dotazu. Výhodou tohoto programu je standardní formát odpovědi. Odpověď tedy můžeme dále zpracovávat svým programem. Zatímco nslookup se používá nejčastěji interaktivně, dig se spouští často z dávek.

Nejčastěji používaná syntaxe:

```
dig @server domena query-type
```

Za znakem `@` uvedeme jméno serveru, kterého se chceme dotazovat (uvedení serveru je nepovinné). Druhý parametr je jméno domény, kterou chceme kontrolovat, `query-type` je požadovaný typ záznamu. Na místo řetězce `query-type` můžeme uvést libovolný typ záznamu RR nebo řetězec axfr, kterým požadujeme zónový přenos, nebo řetězec any, který je požadavkem na libovolný typ záznamu.

V následujícím příkladu požadujeme zobrazení vět typu mx pro doménu jcu.cz (explicitně nespecifikujeme, který jmenný server má být dotazován):

```
dig jcu.cz mx

# dig jcu.cz mx

; <>> DiG 9.3.2 <>> jcu.cz mx
;; global options: printcmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 24394
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 4, AUTHORITY: 3, ADDITIONAL: 7

;; QUESTION SECTION:
;jcu.cz.           IN      MX

;; ANSWER SECTION:
jcu.cz.          3600    IN      MX    12801 ax.virusfree.cz.
jcu.cz.          3600    IN      MX    12817 bx.virusfree.cz.
jcu.cz.          3600    IN      MX    12833 cx.spamfree.cz.
jcu.cz.          3600    IN      MX    12849 dx.spamfree.cz.

;; AUTHORITY SECTION:
jcu.cz.          3600    IN      NS     virgo.jcu.cz.
jcu.cz.          3600    IN      NS     taurus.zf.jcu.cz.
jcu.cz.          3600    IN      NS     eros.zcu.cz.

;; ADDITIONAL SECTION:
ax.virusfree.cz. 2105    IN      A      212.80.67.162
bx.virusfree.cz. 2174    IN      A      195.113.87.34
cx.spamfree.cz.  2174    IN      A      87.106.24.115
dx.spamfree.cz.  2174    IN      A      212.80.95.156
eros.zcu.cz.     86400   IN      A      147.228.1.10
virgo.jcu.cz.    3600    IN      A      160.217.1.10
taurus.zf.jcu.cz. 3600    IN      A      160.217.161.1

;; Query time: 0 msec
;; SERVER: 160.217.1.10#53(160.217.1.10)
;; WHEN: Mon Feb 11 10:22:11 2008
;; MSG SIZE  rcvd: 298
```

## Domácí cvičení

- ◆ Ve Windows 2000, XP či Vista příkazem `ipconfig /displaydns` vypište obsah paměti DNS.
- ◆ Vypište seznam jmenných serverů domén: doména vaší firmy, doména .cz a kořenová doména (RR-věty NS).
- ◆ Vypište seznam e-mailových serverů vaší firmy (RR-věty MX).
- ◆ Zjistěte, jaký jmenný server využívá v Internetu vaše firma pro reverzní záznamy (RR-věty PTR např. pro webový nebo poštovní server).

## Kapitola 12

# Protokol DNS

Zatímco v minulé kapitole jsme si vysvětlili, k čemu DNS slouží, tak nyní se budeme věnovat samotnému protokolu DNS. DNS protokol používá několik typů operací. Nejčastěji používanou operací je operace DNS QUERY. Jedná se o dotaz vedoucí k získání jedné nebo více vět RR z databáze DNS. Operace DNS QUERY byla dlouhá léta jedinou operací poskytovanou systémem DNS. S novými rozšířeními protokolu DNS přibývají i nové typy operací, např. DNS NOTIFY nebo DNS UPDATE.

Protokol DNS pracuje způsobem dotaz – odpověď. Klient pošle dotaz serveru a server na dotaz odpoví. Jistou komplikací je komprese jmen, která se provádí proto, aby DNS pakety byly co nejúspornější.

Protokol DNS je protokol aplikační vrstvy, neřeší tedy otázku vlastního přenosu paketů. Přenos svých paketů svěřuje transportnímu protokolu. Na rozdíl od drtivé většiny ostatních aplikačních protokolů využívá DNS jako transportní protokoly UDP i TCP. Dotaz i odpověď jsou přenášeny vždy stejným transportním protokolem.

U dotazů na překlad (tj. žádosti o RR record) je dávána přednost protokolu UDP. V případě, že je odpověď DNS delší než 512 B, vloží se do odpovědi pouze část informací nepřesahující 512 B a v záhlaví se nastaví bit TC, specifikující, že se jedná o neúplnou odpověď. Klient si může kompletní odpověď vyžádat protokolem TCP.

U přenosu zón např. mezi primárním a sekundárním jmenným serverem se používá protokol TCP. Jmenný server standardně očekává dotazy jak na portu 53/udp, tak na portu 53/tcp.



**Poznámka:** U protokolu UDP je třeba upozornit, že některé implementace protokolu UDP využívají možnosti nevyplňovat pole pro kontrolní součet v záhlaví UDP paketu. Tato vlastnost může být užitečná např. pro NFS, ale pro DNS je nebezpečná. Porucha sítě tak může způsobit nesmyslnou odpověď, zejména je-li na cestě mezi serverem a klientem použit např. linkový protokol SLIP. Zkontrolujte si proto před instalací jmenného serveru, zda váš systém vyplňuje kontrolní součet v UDP paketu.

## DNS QUERY

Operace DNS QUERY se skládá z dotazu a odpovědi. Dotaz obsahuje požadavek na získání věty RR (resp. více vět RR) z databáze DNS. Odpověď pak obsahuje příslušné věty RR nebo je záporná. Věty RR obsažené v odpovědi mohou obsahovat konečnou odpověď nebo mohou klientovi pomoci k formulaci další operace DNS QUERY napomáhající k dosažení kýzeného cíle, tj. k formulaci další iterace.

## Formát paketu DNS query

DNS *query* používá stejný formát paketu pro dotaz i odpověď (viz obr. 12.1). Paket se může skládat až z pěti sekcí. Každý paket musí obsahovat sekci záhlaví (HEADER).

Zajimavým je zde slovo „dotaz (*query*)“, které se zde používá ve dvou významech:

1. Operace DNS QUERY. Jedná se o základní operaci protokolu DNS, pomocí které se v databázích DNS vyhledávají záznamy (tzv. věty RR).
2. Sama operace DNS QUERY se vždy skládá z dotazu (zasílá jej klient) a odpovědi na dotaz, kterou vrací jmenný server klientovi. Klientem může být buď resolver, nebo jmenný server, který sám neumí dotaz zodpovědět.

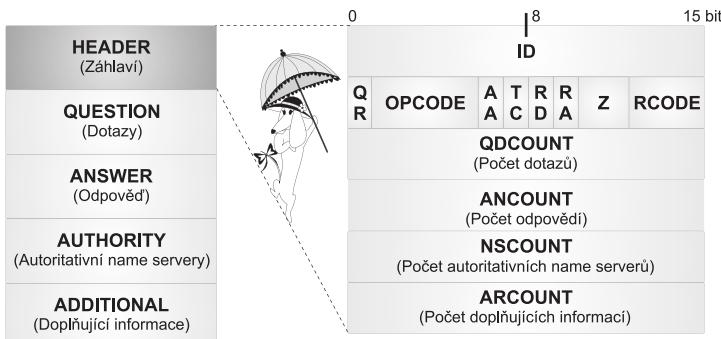
Resolver zpravidla svůj dotaz opatří příznakem, že se jedná o rekurzivní dotaz – žádá jmenný server, aby mu zjistil konečný výsledek. Naopak pokud dotaz posílá jmenný server, tak jej zpravidla označí příznakem, že se jedná o interaktivní dotaz – žádá jiný jmenný server o pomoc s překladem, ale nechce jej obtěžovat rekurzivním dotazem – je sám schopen se iteracemi dobrat ke kýženému výsledku.

## Záhlaví paketu DNS query

Záhlaví paketu je povinné a je obsaženo v dotazu i v odpovědi.

První dva bajty (16 bitů) záhlaví obsahují identifikátor zprávy (ID). Identifikaci zprávy generuje klient a server ji kopíruje do odpovědi. Identifikátor slouží k párování dotazu a odpovědi. Jednoznačně určuje, ke kterému dotazu patří která odpověď. Identifikátor umožňuje klientovi posílat více dotazů současně, aniž by musel čekat na odpověď.

Další dva bajty záhlaví obsahují řídicí byty. Tabulka 12.1 uvádí význam jednotlivých hodnot těchto řídicích bitů.



Obrázek 12.1: Formát paketu DNS Query

Další čtyři dvoubajtová pole v záhlaví paketu obsahují počet vět obsažených v jednotlivých sekcích, které následují za záhlavím.

- ◆ **QDCOUNT** – číslo určující, z kolika vět se skládá dotaz.
- ◆ **ANCOUNT** – číslo určující, z kolika vět se skládá odpověď.
- ◆ **NSCOUNT** – číslo určující, z kolika vět se skládá sekce obsahující odkazy na autoritativní jmenovité servery.
- ◆ **ARCOUNT** – číslo určující, z kolika vět se skládá sekce doplňující informace.

**Tabulka 12.1:** Význam jednotlivých řídicích bitů ze záhlaví DNS paketu

Pole	Počet bitů	Hodnota
QR	1	0 pokud je zpráva dotazem 1 pokud je zpráva odpovědí
Opcode	4	<b>Typ otázky je stejný v dotazu i odpovědi:</b> 0 – standardní otázka (QUERY) 1 – inverzní otázka (IQUERY) 2 – otázka na status (STATUS) 4 – notify otázka (NOTIFY) 5 – update otázka (UPDATE)
AA	1	0 – odpověď není autoritatívní 1 – odpověď je autoritatívní
TC	1	1 – odpověď byla zkrácena na 512 bajtů. Pokud má klient zájem o celou odpověď, musí dotaz zopakovat pomocí protokolu TCP.
RD	1	1 – pokud klient požaduje rekurzivní překlad (důležité pro dotaz)
RA	1	1 – pokud server umožňuje rekurzivní překlad (důležité pro odpověď)
Z	3	rezervované pro budoucí použití
Rcode	4	<b>Výsledkový kód odpovědi</b> 0 – Bez chyby ( <i>Noerror</i> ) 1 – Chyba ve formátu dotazu, server jej neumí interpretovat ( <i>FormErr</i> ) 2 – Server neumí odpovědět ( <i>ServFail</i> ) 3 – Jméno z dotazu neexistuje (tj. negativní odpověď), tuto odpověď mohou vydat pouze autoritatívní jmenné servery ( <i>NXDomain</i> ) 4 – Server nepodporuje tento typ dotazu ( <i>NotImp</i> ) 5 – Server odmítá odpovědět, např. z bezpečnostních důvodů ( <i>Refused</i> )

## Příklad

Příklad DNS paketu odchyceného na síti programem Wireshark:

Ethernet II, Src: 00:13:02:91:2e:8b (00:13:02:91:2e:8b), Dst: 00:1a:92:ca:e6:b9 (00:1a:92:ca:e6:b9)  
Internet Protocol, Src: 10.0.0.1 (10.0.0.1), Dst: 10.0.0.138 (10.0.0.138)  
User Datagram Protocol, Src Port: 2879 (2879), Dst Port: 53 (53)  
    Source port: 2879 (2879)  
    Destination port: 53 (53)  
    Length: 41  
    Checksum: 0xf78c [correct]  
Domain Name System (query)  
    Transaction ID: 0x7fa3  
    Flags: 0x0100 (Standard query)  
        0... .... .... .... = Response: Message is a query  
        .000 0... .... .... = Opcode: Standard query (0)  
        .... .0 .... .... = Truncated: Message is not truncated  
        .... ..1 .... .... = Recursion desired: Do query recursively  
        .... .... 0... .... = Z: reserved (0)  
        .... .... ...0 .... = Non-authenticated data OK: Non-authenticated data is unacceptable

```

Answer RRs: 0
Authority RRs: 0
Additional RRs: 0
Queries
www.playboy.com: type AAAA, class IN
Name: www.playboy.com
Type: AAAA (IPv6 address)
Class: IN (0x0001)

```

## Sekce dotaz (Question section)

Pakety DNS dotazů obsahují většinou pouze jednu sekci, a to sekci dotazu s jedním dotazem (QDCOUNT=1). Sekce dotazu obsahuje tři pole:

**QNAME** – obsahuje doménové jméno. Protokol DNS nepoužívá pro vyjádření doménového jména tečkovou notaci. Každá část doménového jména (v běžném zápisu mezi tečkami) je uvozena bajtem obsahujícím délku řetězce. Na konci doménového jména je nula označující konec doménového jména (nulová délka řetězce). Příklad obsahu tohoto pole v dotazu na překlad doménového jména info.pvt.net: 0416info0316pvt0316net0016. Délky řetězce jsou v binárním tvaru.

**QTYPE** – specifikuje typ dotazu, tj. požadovaný typ věty v odpovědi.

Nejčastější typy dotazu uvádí v tabulce 12.2.

**Tabulka 12.2:** Hodnoty typů dotazů

Typ	Hodnota (desítkově)	Význam
A	1	Požadavek na získání IP adresy verze 4
NS	2	Požadavek na získání autoritativních jmenných serverů
CNAME	5	Požadavek na získání věty CNAME
SOA	6	Požadavek na získání věty SOA
WKS	11	Požadavek na získání věty WKS
PTR	12	Požadavek na získání PTR věty
HINFO	13	Požadavek na získání HINFO věty
MX	15	Požadavek na získání věty MX
TXT	16	Požadavek na získání TXT věty
SIG	24	Požadavek na získání věty SIG
KEY	25	Požadavek na získání věty KEY
NXT	30	Požadavek na získání věty NXT
AAAA	28	Požadavek na získání IP adresy verze 6
AXFR	252	Požadavek na získání transferu celé zóny, tj. přenesení všech vět RR z primárního serveru
IXFR		Požadavek na získání inkrementálního (přírustkového) zone transferu
*	255	Požadavek na získání všech vět



**Poznámka:** Dotazy AXFR a IXFR se zásadně odlišují od ostatních dotazů tím, že jimi nežádáme o konkrétní věty RR, ale o přenos dat zóny – tzv. zone transfer.

**QCLASS** – třída dotazu (viz tab. 12.3).

**Tabulka 12.3:** Jednotlivé třídy

Číselná hodnota (desítkově)	Význam
1	IN – Internet
3	CH – Chaos
4	HS – Hesiod
255	* – všechny třídy (pouze jako QCLASS)

### Příklad

Příklad DNS paketu odchyceného na síti je uveden v tab. 12.5. Sekce dotazu je v příkladu zvýrazněna tučně.

Příklad paketu obsahujícího všechny sekce (odchyceno programem Wireshark):

```

Ethernet II, Src: 00:1a:92:ca:e6:b9 (00:1a:92:ca:e6:b9), Dst: 00:13:02:91:2e:8b
(00:13:02:91:2e:8b)
Internet Protocol, Src: 10.0.0.138 (10.0.0.138), Dst: 10.0.0.1 (10.0.0.1)
User Datagram Protocol, Src Port: 53 (53), Dst Port: 3518 (3518)
Domain Name System (response)
[Request In: 248]
[Time: 0.218028000 seconds]
Transaction ID: 0xd09b
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
    .... .... .... 0000 = Reply code: No error (0)
Questions: 1
Answer RRs: 1
Authority RRs: 5
Additional RRs: 5
Queries
    www.playboy.com: type A, class IN
        Name: www.playboy.com
        Type: A (Host address)
        Class: IN (0x0001)
Answers
    www.playboy.com: type A, class IN, addr 216.163.137.3

```

```

Authoritative nameservers
playboy.com: type NS, class IN, ns ns15.customer.level3.net
playboy.com: type NS, class IN, ns ns1-chi.playboy.com
playboy.com: type NS, class IN, ns ns2-chi.playboy.com
playboy.com: type NS, class IN, ns ns21.customer.level3.net
playboy.com: type NS, class IN, ns ns29.customer.level3.net
Additional records
ns15.customer.level3.net: type A, class IN, addr 209.244.4.244
ns21.customer.level3.net: type A, class IN, addr 209.244.5.84
ns29.customer.level3.net: type A, class IN, addr 209.244.5.212
ns1-chi.playboy.com: type A, class IN, addr 216.163.136.24
ns2-chi.playboy.com: type A, class IN, addr 64.202.105.36

```

## Sekce odpověď, autoritativní servery a doplňující informace

Pakety odpovědi obsahují obvykle vedle sekce záhlaví a zopakované sekce dotazu ještě tři sekce: sekci odpovědi, sekci autoritativních serverů a sekci doplňujících informací. Sekce odpověď nese vlastní odpověď. Sekce autoritativní jmenné servery obsahuje jména jmenných serverů uvedených ve větách NS. Sekce doplňkové údaje obsahuje obvykle IP adresy autoritativních jmenných serverů. Věty v těchto sekcích jsou běžné resource recordy (RR) – obdobné větám v paměti jmenného serveru – a mají společný formát:

**NAME** – doménové jméno, stejný formát jako v sekci dotazu QNAME.

**TYPE** – typ věty, stejný formát jako v sekci dotazu QTYPE.

**CLASS** – třída věty, stejný formát jako v sekci dotazu QCLASS.

**TTL** – doba platnosti RR, tj. jak dlouho může být odpověď udržována jako platná v paměti.

**RDLENGTH** – délka části RDATA.

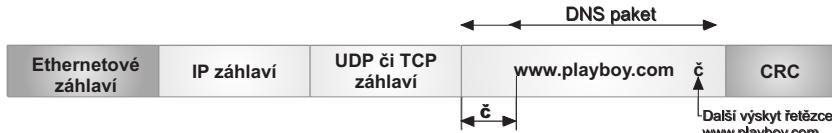
**RDATA** – pravá strana zdrojové věty (IP adresa nebo doménové jméno).

## Komprese

Komprese slouží k redukci velikosti DNS paketu. V DNS paketu se může stejně doménové jméno nebo jeho část vyskytovat opakováně. Komprese spočívá v tom, že je toto opakující se jméno uvedeno pouze jednou a každý další výskyt tohoto jména je nahrazen ukazovátkem na první výskyt.

Doménové jméno není v DNS paketu uvedeno v tečkové notaci, ale jako oddělovač jednotlivých částí doménového jména je použito číslo určující délku následující části. Tento oddělovač je uložen v jednom bajtu. Každá část doménového jména může být dlouhá maximálně 63 znaků, tedy oddělovací bajt bude mít maximální hodnotu 63 vyjádřenou dvojkově  $0011111_2 = 63_{10}$ . Pokud je v tomto bajtu číslo 192 nebo větší, pak je to příznak toho, že nebude uvedeno doménové jméno, ale pouze odkaz na jeho předcházející výskyt. Ukazatel je dlouhý 16 bitů. V prvních dvou bitech obsahuje jedničku, čímž se odlišuje od běžného oddělovače. V dalších bitech pak obsahuje pořadové číslo bajtu od začátku DNS paketu, kde začíná doménové jméno, na které má odkaz ukazovat.

Posun 0 by ukazoval na první bajt, tj. pole ID v sekci záhlaví.



Obrázek 12.2: Komprese DNS paketu

## Domácí cvičení

Spusťte program Wireshark a v odchyceném paketu protokolu DNS nalezněte komprimovaný řetězec. Dobré je komprimované řetězce vyhledávat v DNS paketech, které nesou odpověď (nikoliv jen dotaz).

Domácí cvičení: Porovnejte výpis DNS paketu v programu Wireshark s obr. 12.1.:

1. programu nslookup v ladícím režimu d2,
2. programu dig.

## Inverzní dotaz

Inverzní dotaz se nesmí zaměňovat s reverzním dotazem. Při inverzním dotazu se také např. překládá IP adresa opět na jméno, ale k vyhledání se použijí věty typu A. Při reverzním překladu se používají věty typu PTR.

Inverzní dotazy nemusí být jmenným serverem podporovány. Jejich specifikace je uvedena v RFC 1035.

## DNS UPDATE

Operace DNS UPDATE se rovněž skládá z dotazu a odpovědi. Mechanismus DNS UPDATE je popsán v RFC-2136.

Operace DNS UPDATE umožňuje dynamicky opravovat záznamy v databázi DNS. Proto se pro tento mechanismus používá i označení „dynamický update“. DNS UPDATE umožňuje přidat jednu nebo několik vět do zónového souboru nebo jednu, případně několik vět ze zónového souboru zrušit. Záznamy v databázi DNS tedy nemusí již staticky opravovat správce serveru, ale je možné opravit dynamicky pomocí příslušného klienta na dálku použitím protokolu DNS.

Data v zóně lze pomocí DNS UPDATE opravovat pouze na primary master serveru. Pokud DNS Update dotaz obdrží slave server, forwarduje tento dotaz primary master serveru.

DNS UPDATE neumožňuje vytváření nových zón, umožňuje pouze opravovat zóny již existující. Pomocí DNS UPDATE tedy není možné přidávat novou větu SOA nebo větu SOA rušit. Větu SOA je možné pouze opravit.

Jedním dotazem DNS UPDATE je možné opravit jednu nebo několik vět v jedné zóně.

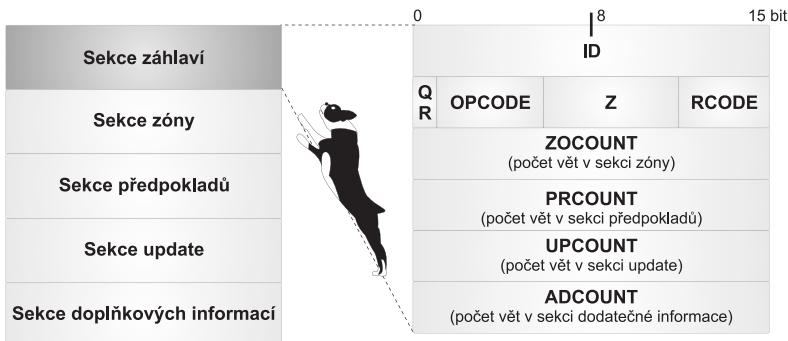
Opravu zóny pomocí DNS UPDATE je možné provést za specifikovaných podmínek. Podmínkou je existence nebo neexistence daných vět RR v zóně před opravou. Např. požaduji-li zrušení věty v zóně, musí tato věta v zóně před opravou existovat. Podmínek tohoto typu může být pro provedení

opravy specifikováno několik. Z hlediska splnění jsou podmínky posuzovány jako celek. Tedy není -li splněna jedna z uvedených podmínek, nejsou splněny podmínky jako celek a žádná z požadovaných oprav se neprovede.

V balíku operace DNS UPDATE jsou odděleně specifikovány podmínky pro provedení opravy a odděleně jsou uvedeny věty RR, které se mají do zónového souboru přidat nebo z něho zrušit.

Formát DNS paketu zůstává stejný i pro DNS UPDATE, skládá se opět z pěti částí. Jednotlivé části však mají v tomto případě specifický obsah i pojmenování. Paket DNS UPDATE se skládá ze sekcí (viz obr. 12.3):

- ◆ **sekce záhlaví**
- ◆ **sekce zóny** – definuje zónu, ke které se vztahují opravy
- ◆ **sekce předpokladů** – sada vět RR, které musí v zóně existovat
- ◆ **sekce update** – sada vět RR, které se mají opravit nebo zrušit
- ◆ **sekce doplňkových informací** – informace, jež nejsou součástí update, ale jsou třeba k provedení update



Obrázek 12.3: Formát paketu DNS UPDATE

## Sekce záhlaví

Sekce záhlaví podobně jako u DNS QUERY obsahuje v prvních dvou bajtech identifikaci (pole ID), dále následují dva bajty řídicích polí a délky jednotlivých sekcí (každá délka je 2 B):

- ◆ **ZOCOUNT** – počet vět v sekci Zóny
- ◆ **PRCOUNT** – počet vět v sekci Předpokladů
- ◆ **UPCOUNT** – počet vět v sekci Update
- ◆ **ADCOUNT** – počet vět v sekci Doplňkové informace

Tabulka 12.4: Význam jednotlivých řídicích polí

ID	16	Identifikátor zprávy, je kopírován do odpovědi.
QR	1	0 pokud je zpráva dotazem, 1 pokud je zpráva odpovědí. Ostatní kontrolní byty DNS Update nepoužívají. Za QR bezprostředně následuje pole Opcode.

Opcode	4	5(UPDATE), kopíruje se z dotazu do odpovědi
Z	7	Rezerva, musí být naplněna binárními nulami
Rcode	4	V odpovědi výsledkový kód, v dotazu ne definované

**Tabulka 12.5:** Výsledkové kódy odpovědí (pole Rcode)

Kód chyby	Číselná hodnota	Popis chyby
NOERROR	0	Bez chyby
FORMERR	1	Chybný formát zprávy, jmenný server ji neumí interpretovat
SERVFAIL	2	Při zpracování zprávy došlo k interní chybě serveru (např. chybě OS nebo vypršel timeout při forwardingu)
NXDOMAIN	3	Jméno, které by mělo existovat, neexistuje
NOTIMP	4	Jmenný server nepodporuje daný typ zprávy ( <i>Opcode</i> )
REFUSED	5	Jmenný server odmítá provést zprávu, např. z bezpečnostních důvodů
YXDOMAIN	6	Jméno, které by nemělo existovat, existuje
YXRRSET	7	Sada vět RR, která by neměla existovat, existuje
NXRRSET	8	Sada vět RR, která by měla existovat, neexistuje
NOAUTH	9	Server není autoritou pro danou zónu
NOTZONE	10	Jméno uvedené v sekci předpokladů nebo v sekci update není v dané zóně

## Sekce zóny

Sekce zóny určuje zónu, která bude opravována. Jedním dotazem DNS UPDATE je možné opravovat pouze jednu zónu, tj. sekce zóny povoluje pouze jednu větu.

Sekce je tvořena třemi částmi:

- ◆ **ZNAME** – jméno zóny,
- ◆ **ZTYPE** – musí být řetězec SOA,
- ◆ **ZCLASS** – třída zóny – IN (Internet).

## Sekce předpokladů

Sekce předpokladů obsahuje skupinu vět RR, které musí v dané zóně existovat na primary master serveru v okamžiku doručení UPDATE-paketu. V sekci předpokladu je možné použít pět variant:

1. **V zóně musí existovat minimálně jedna věta RR s daným NAME a TYPE (RR set exists, value independent).**

Sekce předpokladu obsahuje jednu větu s daným NAME a TYPE, kterou očekává v zóně. Ostatní položky jsou nevýznamné, proto se použije RDLENGTH=0, RDATA je prázdné, CLASS=ANY, TTL=0.

*Např. požaduji, aby v doméně existovala věta typu A s doménovým jménem aaa.firma.cz s libovolným RDATA.*

- 2. V zóně musí existovat sada vět RR daného NAME a TYPE a jejich pravá strana se musí shodovat s pravou stranou vět v UPDATE-paketu (*RR set exists, value dependent*). Na pořadí vět nezáleží.**

V sekci je skupina vět RR s daným NAME a TYPE a RDATA, TTL=0, CLASS je určena v sekci zone.

*Např. požaduji, aby v zóně existovaly věty:*

mail.firma.cz.	A	195.47.11.11
www.firma.cz.	A	195.47.11.12
firma.cz.MX	10	mail.firma.cz.

- 3. V zóně neexistuje žádná věta RR s daným NAME a TYPE (RR set does not exist).** Sekce obsahuje jednu větu RR s daným NAME a TYPE, RDLEGTH=0, RDATA je prázdné, CLASS=NONE, TTL=0.

*Např. požaduji, aby v zóně neexistovala žádná věta typu A s doménovým jménem mail.firma.cz.*

- 4. V zóně musí existovat alespoň jedna věta s daným NAME a typem definovaným v sekci ZONE (Name is in use).** V sekci je jedna věta RR s daným NAME, RDLENGTH=0, RDATA je prázdné, CLASS=ANY, TYPE=ANY, TTL=0.

*Např. požaduji, aby v zóně existovala alespoň jedna věta, jejíž doménové jméno obsahuje řetězec firma.cz – tj. nejde o prázdnou zónu.*

- 5. V zóně neexistuje žádná věta RR jakéhokoli typu s daným NAME (Name is not in use).** Sekce obsahuje jednu větu RR s daným NAME, RDLENGTH=0, RDATA je prázdné, CLASS=NONE, TYPE=ANY, TTL=0.

*Např. požaduji, aby v zóně neexistovala žádná věta, jejíž doménové jméno obsahuje řetězec firma.cz – tj. jde o prázdnou zónu.*

## Sekce update

Sekce update obsahuje věty RR, které mají být do zóny přidány nebo z ní zrušeny. Je možné provést čtyři druhy změn:

### 1. Přidat věty RR

Sekce update obsahuje několik vět. Do souboru se přidají věty s daným NAME, TYPE, TTL, RDLENGTH a RDATA. CLASS se převezme ze sekce zone. Duplicitní věty v tomto seznamu se ignorují.

*Např. požaduji přidat věty:*

firma.cz.	MX	20	mh.firma.cz.
mh.firma.cz.	A	195.47.12.32	

### 2. Zrušit sadu vět RR daného typu

Sekce obsahuje jednu větu, uvedené NAME a TYPE určuje, které věty mají být zrušeny. TTL=0, CLASS=ANY, RDLENGTH=0, RDATA je prázdné.

*Např. požaduji zrušit všechny věty s doménovým jménem mail.firma.cz.*

### 3. Zrušit všechny věty RR daného jména

Sekce obsahuje jednu větu RR, dané NAME určuje, které věty mají být zrušeny. TYPE=ANY, TTL=0, CLASS=ANY, RDLENGTH=0, RDATA je prázdné.

*Např. požaduji zrušit všechny věty typu MX s doménovým jménem firma.cz.*

### 4. Zrušit jednu větu RR

Sekce obsahuje větu, která má být zrušena (NAME, TYPE, RDLENGTH, RDATA). TTL=0, CLASS=None. *Např. požaduji zrušit větu:*

*firma.cz. IN MX 10 mail.firma.cz.*

## Sekce doplňujících informací

Sekce doplňkových informací obsahuje věty RR, které se vztahují k vlastnímu update nebo k novým větám přidaným pomocí update. Např. může zde být uvedena věta glue pro zónu, pokud se přidává nová věta NS pro zónu.

## Soubor žurnál

Změny provedené pomocí operace DNS UPDATE nejsou promítány přímo do zónových souborů, ale jmenný server ukládá tyto změny do tzv. zónových žurnálových souborů. V pravidelném časovém intervalu je pak obsah žurnálových souborů promítán do zónových souborů. Aktualizace zónových souborů podle žurnálu proběhne i v okamžiku zastavení nebo restartu jmenného serveru.

Každá zóna používá svůj žurnálový soubor, který se automaticky vytváří v okamžiku první operace DNS UPDATE zóny. Tento soubor má jméno totožné se jménem zóny a standardní příponu .jnl. Žurnálové soubory mají binární obsah, z čehož vyplývá, že není možné ani povolené tyto soubory manuálně opravovat. Zákaz manuální úpravy platí i pro ty zónové soubory, které používají operace DNS UPDATE. Důvod tohoto zákazu je zřejmý. Zónové soubory totiž nemusí obsahovat aktuální informace, neboť část posledních informací o zóně může být uložena právě v žurnálovém souboru. Pokud potřebujeme z nějakého důvodu dynamicky opravenou zónu ručně upravit a rozhodneme se zákaz porušit, budeme postupovat takto:

- ◆ provedeme shutdown jmenného serveru (příkazem rndc stop),
- ◆ odstraníme žurnálové soubory; jejich obsah je již promítnut do zónových souborů a po provedení manuálních změn v zóně by byl jejich obsah nekonzistentní,
- ◆ upravíme zónový soubor,
- ◆ restartujeme jmenný server.



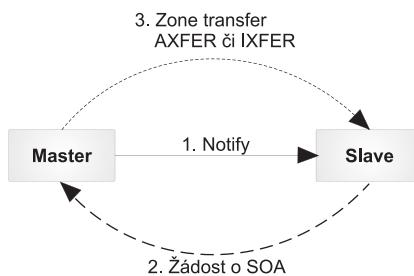
**Poznámka** (k bezpečnosti) Je doporučováno používat operace DNS UPDATE spolu s nějakým systémem zabezpečení (viz RFC-4033 až RFC-4035).

## DNS Notify

Operaci DNS Notify popisuje RFC-1996. Operace DNS Notify umožňuje informovat slave servery o změně dat v zóně. Při používání operace DNS Notify může slave server získávat aktuální data zóny dříve než v okamžiku vypršení intervalu v poli *refresh* uvedeném ve větě typu SOA zóny.

Komunikaci o zóně mezi master a slave serverem při použití operace DNS Notify zahajuje master server. Master server posílá v případě změny zóny zprávu slave serverům: „Požádej mě o zone transfer.“ Slave server po obdržení zprávy Notify může okamžitě požádat o zone transfer.

Zprávu o změně zóny (DNS Notify) obdrží všechny servery, které jsou uvedeny v NS-větách pro zónu. Server uvedený ve věti SOA není informován, protože se předpokládá, že právě tento server zprávy generuje. Některé implementace umožňují správci master serveru sadu jmenných serverů doplnit o další IP adresy dalších jmenných serverů. Množina serverů, pro které se DNS Notify generuje, se nazývá Notify set.



Obrázek 12.4: Notify

## Zpráva Notify

Zpráva Notify používá formát DNS paketu definovaný v RFC-1035. DNS Notify používá pouze podmnožinu polí v paketu, nepoužitá pole musí být vyplněna binárními nulami. Typ zprávy (Opcode) je nastaven na hodnotu 4 (NOTIFY). Master server může jako součást zprávy Notify poslat NAME, CLASS, TYPE i RDATA změněných vět v zóně. Zprávy Notify nevyužívají sekci autoritatitivních serverů ani sekci doplňkových informací.

K přenosu paketu Notify se používá přenosový protokol UDP nebo TCP. Pokud je použit protokol TCP, je zpráva Notify vyslána pouze jednou. Na master serveru je nastaven časový interval, po který master server čeká na odpověď. Při použití TCP slave server ani master server nesmí přerušit poskytování služeb během transakce.

Pokud je použit protokol UDP, master server periodicky posílá zprávy Notify na slave server. Vysílání zpráv Notify master server zastaví po obdržení odpovědi. Pokud master server neobdrží odpověď, zastaví vysílání těchto zpráv po vyčerpání nastaveného počtu opakování zpráv nebo poté, co mu protokol ICMP oznámí, že port je nedostupný. Interval mezi vysíláním jednotlivých zpráv může být specifikován jako parametr v konfiguraci master serveru (zpravidla 60 s). Podobně může být nastaven i počet povolených opakování (zpravidla 5 s).

Jediná událost, která aktivuje vyslání zprávy Notify, je zvětšení sériového čísla (pole *serial number*) ve věti SOA. Po obdržení zprávy Notify by se měl slave server chovat stejně, jako když zóně uvedené v QNAME vyprší interval uvedený v poli *refresh* věty SOA. Slave server by tedy měl požádat master server o větu SOA pro zónu a zkontoval pole *serial number*. Pokud došlo ke změně sériového čísla (pole *serial number*), pak bude iniciovat AXFR nebo IXFR (tj. provede buď úplný, nebo přírůstkový zone transfer).

Platí následující pravidla:

- ◆ Zprávou *zone transfer* by se měl dělat zone transfer z master serveru, který poslal na slave server zprávu Notify.
- ◆ Ve zprávě Notify může master server poslat i změněné věty RR (změněné jméno, třídu, typ a nepovinné i RDATA). V žádném případě však nemohou být tyto informace (změněné věty RR v answer sekci Notify otázky) použity pro opravu dat na slave serveru nebo jako indikace toho, že je třeba provést zone transfer nebo změnit *refresh time* u zóny. Jedná se pouze o informaci, ze které může slave server např. zjistit, že má již aktuální data a nepotřebuje iniciovat zone transfer.
- ◆ Odpověď Notify neobsahuje žádné podstatné informace. Rozhodující je pouze fakt, že master server obdrží odpověď Notify.
- ◆ Pokud slave server obdrží zprávu Notify obsahující QNAME od serveru, který není master pro zónu, měl by tuto zprávu ignorovat a generovat chybovou hlášku do logu.
- ◆ Při startu by server měl poslat Notify pro každou autoritativní zónu. Při restartu serveru je poslána zprávy Notify volitelné.
- ◆ Master server se snaží vyhnout velkému množství současných zone transferů. Může tedy zprávu Notify posílat s určitým zpožděním. Toto zpoždění bude vybíráno náhodně, takže každý slave server začne svůj zone transfer v jiném čase. Toto zpoždění nesmí být větší než pole refresh. Zpoždění může být nastavitelné parametrem master zóny (30-60 s). Slave server, který obdrží Notify, musí nejprve tuto iniciovanou transakci dokončit a teprve potom posílat zprávy Notify níže (serverům, vůči kterým je master).

V Bindu 8.1 a vyšších verzích je mechanismus DNS Notify standardně implementován.

## Inkrementální zone transfer

Inkrementální zone transfer (IXFR) umožňuje přenášet při změně dat v zóně z master serveru na slave server pouze změněná data, tedy pouze část zóny. Naproti tomu úplný zone transfer (AXFR) přenáší při sebemenší změně zóny celou zónu. Inkrementální zone transfer specifikuje RFC-1995.



**Poznámka:** Při úplném zone transferu (AXFR) se přenáší z primárního serveru (primary master serveru) všechny věty RR zóny. Při přírůstkovém (inkrementálním) zone transferu (IXFR) se přenáší jen změněné věty RR.

K tomu, aby mohl master server poskytovat slave serveru pouze změněné věty zóny, musí udržovat historii stavů databáze. Master server tedy musí udržovat nejnovější verzi zóny a rozdíly mezi nejnovější verzí a několika staršími verzemi. Master server posílá slave serveru mechanismem IXFR ty zóny, které jsou na master serveru opravované pomocí DNS UPDATE, nikoli ručně upravované zóny. Verze souborů se liší sériovým číslem v SOA-větě. Pokud slave server zjistí, že potřebuje nová data k zóně a podporuje IXRF, pošle dotaz na master server, kde uvede, jakou verzi zóny má jako poslední např.: 98052001 (*serial*). Master server jako odpověď pošle slave serveru pouze změněné věty, tj. věty, které mají být zrušeny, a věty nové. Alternativně může server poslat v odpovědi celou zónu. Celou zónu posílá server i tehdy, když má klient tak starou větu SOA, že již server není schopen poslat IXFR.

Po opravě zóny v paměti musí slave server uložit změny do souboru, teprve potom může sám poskytovat odpověď na dotaz IXFR. Změny v zónových souborech provedené pomocí IXFR využívají pro zápis žurnálové soubory podobně, jako je využívá mechanismus DNS UPDATE. Pokud master server obdrží dotaz s vyšším číslem SOA, než má sám, vrací jako odpověď pouze svou aktuální větu SOA.

Pro přenos IXFR dotazů je možné použít TCP i UDP. Pokud klient zašle dotaz UDP, měl by server zaslat odpověď UDP. Pokud je odpověď delší než 512 B, pošle server v odpovědi UDP pouze větu SOA a klient musí navázat spojení TCP.

Slave server, který požaduje inkrementální zone transfer, bývá označován jako IXFR-klient. Master nebo slave server, který poskytuje inkrementální zone transfer, bývá označován jako IXFR-server.

IXFR používá formát DNS paketu tak, jak je definovaný v RFC 1035.

## **Formát dotazu**

Jako typ otázky (Opcode) je uvedeno IXFR a sekce autoritativních jmenných serverů obsahuje větu SOA zóny uložené na slave serveru.

## **Formát odpovědi**

Jako typ otázky (Opcode) je v odpovědi opět uvedeno IXFR. První a poslední RR v sekci odpovědi je věta SOA zóny, která se má aktualizovat.

V IXFR je možné poslat jako odpověď jednu nebo několik změn (poslední nebo několik posledních verzí zóny) v rámci jedné zóny. Seznam všech změn v rámci jedné verze je v sekci odpovědi uzavřen z obou stran větami SOA.

Za změnu se považuje přidání nebo zrušení RR. Zrušené věty jsou předcházeny starou SOA a přidané věty jsou předcházeny novou SOA RR. Oprava věty je chápána jako zrušení věty původní a přidání věty nové.

Změny jsou v sekci odpovědi uspořádány od nejstarší po nejnovější.

IXFR-klient může změnit starou verzi souboru na novou pouze po úspěšném provedení všech obdržených změn.

Inkrementální odpověď se liší od neinkrementální odpovědi tím, že začíná dvěma větami SOA.

V IXFR není možné vracet v odpovědi celou zónu. Pokud se zjistí, že změn v zóně je příliš a nevyplatí se použít IXFR, pak musí klient vyslat dotaz znovu a požádat o přenos AXFR.

## **Strategie čištění (purging)**

IXFR-server nemusí udržovat všechny předcházející verze zón, ale staré verze může kdykoli zrušit. U velkých a často se měnících zón můžeme narazit na rozpor mezi velikostí obsazeného diskového prostoru a možností dělat IXFR. Informace ze starších verzí souborů mohou být zahodeny v okamžiku, kdy by byl přenos IXFR delší než eventuální AXFR.

## Negativní caching (DNS NCACHE)

Pod pojmem **negativní caching** chápeme, že do paměti neautoritativního (!) serveru jsou vloženy informace o tom, že v DNS neexistuje požadovaná věta RR nebo doménové jméno. Udržování negativních odpovědí na DNS dotazy specifikují RFC-1034 a RFC-2308.

Dnes používané resolversy negenerují stejné negativní odpovědi na stejný dotaz. Aby bylo možné negativní odpovědi korektně používat, je třeba přesně definovat obsah negativní odpovědi a dobu, po kterou má být negativní odpověď udržována v paměti. RFC-1034 definovalo negativní caching jako nepovinný.

Nyní z jiného soudku. RFC-2308 definuje negativní caching jako povinnou vlastnost resolveru a definuje obsah negativní odpovědi. Windows 2000 a výše provádí dokonce negativní caching „Klienta DNS“. Doba uložení je implicitně 5 minut. Pokud chceme tuto dobu změnit, musíme opravit klíč NegativeCacheTime (je typu REG\_DWORD) v registru HKEY\_LOCAL\_MACHINE\ SYSTEM\ CurrentControlSet\ Services\ Tcpip\ Parameters. V tomto klíči je doba uložení negativní odpovědi v sekundách.

### Příklad

Ověření funkce negativní caching „Klienta DNS“ ve Windows. Nejprve provedeme dotaz na neexistující jméno:

```
C:\WINDOWS\system32> ping xxx.yyy.cz  
Hostitele xxx.yyy.cz se pomocí příkazu Ping nepodařilo najít. Zkontrolujte název hostitele a akci opakujte.
```

Nyní si vypíšeme obsah C:\WINDOWS\system32>ipconfig /displaydns

Konfigurace protokolu IP systému Windows

```
1.0.0.127.in-addr.arpa  
-----  
Název záznamu . . . . : 1.0.0.127.in-addr.arpa.  
Typ záznamu . . . . . : 12  
Hodnota Time To Live. : 328880  
Délka dat . . . . . : 4  
Sekce . . . . . . . : Odpově  
Záznam PTR . . . . . : localhost
```

```
xxx.yyy.cz  
-----  
Název neexistuje.  
... dále vynecháno
```

### Jaké negativní odpovědi ukládat do paměti?

RFC-2308 definuje jako povinné ukládání negativních odpovědí s RCODE nastaveným na NXDOMAIN a NOERROR\_NODATA.

**Tabulka 12.6:** Povinně ukládané negativní odpovědi

Chyba	Popis chyby
Name error (NXDOMAIN)	Doménové jméno uvedené v QNAME-dotazu neexistuje. RCODE je nastaveno na NXDOMAIN. Autoritativní sekce může obsahovat věty SOA a NS.
NOERROR_NODATA	RCODE nastaveno na NOERROR, ale sekce odpovědi neobsahuje žádný RR record. Autoritativní sekce může obsahovat větu SOA a NS-věty.

Zbylé typy negativních odpovědí jsou ponechány jako volitelné. Může se jednat např. o negativní odpovědi způsobené chybou jmenného serveru (viz tab. 12.7).

**Tabulka 12.7:** Volitelně ukládané negativní odpovědi

Server failure	Server neposkytuje data o zóně z důvodu chybné konfigurace zóny. Server neposkytuje data o zóně z důvodu nedostupnosti master serveru pro zónu.
Dead / Unreachable server	Server neexistuje, nefunguje nebo není dostupný.

Pokud server podporuje ukládání odpovědí jiných typů než NXDOMAIN a NOERROR\_NODATA, nesmí tyto odpovědi udržovat v paměti déle než 5 minut. Jako součást ukládání informace musí udržovat i IP adresu serveru z odpovědi.

## Jak dlouho udržovat negativní odpovědi v paměti?

Všechny věty RR uložené v paměti jsou považovány za platné, pokud mají TTL větší než 0. TTL je tedy rozhodující položka vzhledem k paměti. I negativní odpovědi, pokud mají být udržovány v paměti, musí mít definováno TTL. Jak však TTL negativní odpovědi určit, když negativní odpověď obvykle neobsahuje žádnou větu RR v sekci odpověď? A tak se určení TTL pro negativní odpověď řeší vložením umělé SOA, věty pro zónu do autoritativní sekce odpovědi.

## Pole MINIMUM ve větě SOA

Pole MINIMUM ve větě SOA mělo postupně tři interpretace:

1. Minimum TTL pro všechny věty RR v zóně. (Nikdy se v praxi takto nepoužívalo.)
2. Implicitní TTL pro všechny věty RR v zóně, které neobsahují pole TTL. (Pouze na primárním jmenném serveru.) Po provedení zone transferu mají všechny věty RR pole TTL naplněno.
3. TTL negativní odpovědi pro zónu.

Nadále se bude uplatňovat pole MINIMUM ve větě SOA ve významu 3. TTL pro jednotlivé věty RR musí být definována přímo ve větách RR nebo pomocí nového příkazu \$TTL v zóně souboru.

### Syntaxe příkazu:

```
$TTL ttl komentář
```

Všechny věty RR uvedené v souboru za příkazem \$TTL, které nemají explicitně uvedeno TTL, pře- bírají TTL z příkazu \$TTL.

Reálné TTL se určuje jako minimum z pole TTL obsaženého v SOA-větě a z pole MINIMUM. TTL se u negativních odpovědí v paměti snižuje stejným způsobem jako u kladných odpovědí. Pokud je TTL u negativní odpovědi 0, je informace v paměti neplatná.

### **Pravidla ukládání negativních odpovědí**

Pro práci s negativními odpověďmi platí následující pravidla:

- ◆ Ukládání negativních odpovědí je povinné. Pokud resolver ukládá odpovědi do paměti, musí ukládat i negativní odpovědi.
- ◆ Neautorizované negativní odpovědi nemohou být ukládány.
- ◆ V paměti musí být uložena i věty SOA z autoritativní sekce odpovědi.
- ◆ Negativní odpověď bez věty SOA nesmí být ukládána.
- ◆ Věta SOA z paměti musí být přidávána do odpovědi.
- ◆ Odpověď NXDOMAIN musí být ukládána spolu s QNAME a QCLASS.
- ◆ Odpověď NOERROR\_NODATA musí být ukládána spolu s QNAME, QTYPE a QCLASS.
- ◆ Do master souboru musí být vložen příkaz \$TTL.

## **Domácí cvičení**

Spusťte program Wireshark a provedte dotaz na překlad pomocí programu nslookup nebo dig. Následně porovnejte oba výstupy.



## Kapitola 13

# Mezinárodní a národní organizace Internetu

Na Internetu je třeba jednoznačně přidělovat IP adresy, čísla autonomních systémů, jména domén i udržovat nejrůznější číselníky protokolů atd. Žádná centrální autorita, která by řídila Internet, však neexistuje. Máme celou škálu mezinárodních organizací, které se touto problematikou zabývají. Jak to už v mezinárodních vztazích bývá, neexistuje nějaká strikní nadřízenost/podřízenost mezi těmito organizacemi a mnohé je navíc postaveno na dobrovolných vztazích.

V době bouřlivého rozvoje Internetu počátkem 90. let vznikla řada neziskových organizací, která se zabývaly koordinací výzkumu, vzdělávání a standardizací Internetu. Jednalo se zejména o *Internet Society* (ISOC), založenou roku 1992. ISOC podporuje a propaguje organizace pracující na internetových standardech. Zejména se jedná o následující neziskové organizace:

- ◆ *Internet Engineering Task Force* (IETF),
- ◆ *Internet Architecture Board* (IAB),
- ◆ *Internet Engineering Steering Group* (IESG),
- ◆ *Internet Research Task Force* (IRTF).

Z našeho pohledu je nejpřitažlivější IETF, která se mj. zabývá tvorbou internetových standardů – RFC. Technologie tvorby RFC je naprosto odlišná od tvorby ostatních světových standardů. IETF standardizaci Internetu rozdělila do následujících oblastí: *Applications Area, General Area, Internet Area, Operations and Management Area, Real-time Applications and Infrastructure Area, Routing Area, Security Area* a nakonec *Transport Area*. Pro každou oblast pak vytvořila pracovní skupiny. Přitom jmenování shora jsou jen ředitelé oblasti a předsedové jednotlivých pracovních skupin. Členem pracovních skupin naopak může být kdokoliv. A tak v těchto pracovních skupinách najdeme experty z akademické sféry, z jednotlivých firem, ale třeba i studenty.

Pracovní skupiny IETF pracují prostřednictvím e-mailových konferencí. Třikrát ročně je pořádán IETF Meeting, kde se tyto skupiny schází. Pracovní skupiny navrhují internetové standardy (*Internet Drafts*). Navrhované internetové standardy mohou být po připomínkovém řízení následně schváleny jako internetové standardy – RFC.

RFC jsou editovány a distribuovány editorem RFC. Na jeho webu [www.rfc-editor.org](http://www.rfc-editor.org) je pak nalezne široká veřejnost.

## ICANN

Internet vznikl na popud vlády USA, kterou byl i financován. V roce 1998 vznikla nezisková organizace *Internet Corporation for Assigned Names and Numbers* (ICANN), která převzala většinu úloh na sebe. ICANN má tč. tři účelové organizace:

- ◆ *The Address Supporting Organization* (ASO), která má na starosti oblast IP adres. ASO tak zastřešuje Regional Internet Registeries (RIR).
- ◆ *The Country Code Names Supporting Organisation* (ccNSO), která má na starosti oblast národních domén (ccTLD). Mj. je rovněž forem hostmasterů národních domén.
- ◆ *Generic Names Supporting Organization* (GNSO) je obdobou pro generické domény.

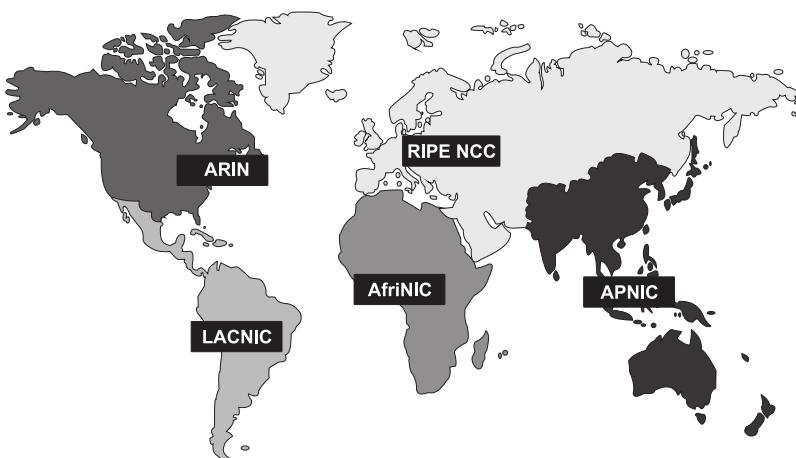
Dnes ICANN zastřešuje i *Internet Assigned Numbers Authority* (IANA), která je zodpovědná za:

- ◆ globální alokaci IP adres (IPv4 i IPv6),
- ◆ správu číselníků rodiny protokolů TCP/IP. Např. číselník čísel protokolů vyšších vrstev v IPv4 záhlaví apod.),
- ◆ správu kořenových jmenných serverů Internetu.

## RIR

Alokace veřejných IP adres je dnes víceúrovňová:

- ◆ Na nejvyšší úrovni je IANA, která rozděluje veřejný adresní prostor mezi tzv. *Regional Internet Registeries* (RIR).
- ◆ RIR mají na starosti alokaci adresního prostoru v rámci své geografické působnosti. Dnes máme následující RIR:
  - RIPE NCC (*Reseaux IP Europeens Network Coordination Centre*) pro oblast Evropy a Blízkého východu. Blíže viz <http://www.ripe.net>.
  - ARIN pro USA a Kanadu. Blíže viz <http://www.arin.net>.



**Obrázek 13.1:** Rozdělení světa na jednotlivé RIR (zdroj RIPE NCC)

- APNIC pro asijsko-pacifickou oblast. Blíže viz <http://www.apnic.net>.
- LACNIC pro Latinskou Ameriku. Blíže viz <http://www.lacnic.net>.
- AfriNIC pro Afriku. <http://www.afrinic.net>.
- ◆ Poskytovatelé Internetu, kteří spolupracují s RIR (např. žádají RIR o alokaci adresního prostoru pro sebe a své zákazníky).

Z hlediska DNS jsou RIR zodpovědné za chod reverzních domén, tj. provozují jmenné servery pro reverzní domény sítí třídy C; např. pro zónu 194.in-addr.arpa. V této zóně pak na požadavek poskytovateli internetu provádějí delegaci na zóny nižší úrovně provozované poskytovateli nebo zákazníky.



**Poznámka:** RIR nejsou v žádném případě zodpovědné za národní nebo generické domény!

## Národní organizace

Jednotlivé země mají přiděleny jednu nebo i více národních domén. Správu těchto domén zajišťuje vždy konkrétní národní organizace – tzv. národní registrátor. Typ organizace je závislý na legislativě konkrétní země. V České republice tuto správu provádí sdružení CZ.NIC, z. s. p. o.

Národní registrátoři se sdružují v mezinárodní organizace:

- ◆ *Council of European National Top-Level Domain Registries* (CENTR),
- ◆ *Organización de ccTLDs Latinoamericanos y del Caribe (Latin American & Caribbean ccTLDs Organization* – LAC TLD),
- ◆ *Asia Pacific Top Level Domain Association* (APTLD).

Tyto mezinárodní organizace však nemají takovou pozici jako RIR. Vysvětlení je prosté. Nerozdělují totiž žádné zdroje (jako např. IP adresy nebo jména domén). Jsou to jen zájmová sdružení. Navíc nejsou ani geograficky omezena. Např. členem CENTR je i Kanada. Silné pozice mají naopak národní registrátoři, protože ti registrují domény druhé úrovně.

## O sdružení CZ.NIC

Vilém Sládek

Zájmové sdružení právnických osob CZ.NIC ([www.nic.cz](http://www.nic.cz)) bylo založené předními poskytovateli internetových služeb v roce 1998 a v březnu 2008 mělo více než 50 členů. Hlavní činností sdružení je provozování registru doménových jmen .CZ a 0.2.4.e164.arpa (ENUM), zabezpečování provozu domény nejvyšší úrovně CZ a osvěta v oblasti doménových jmen.

Sdružení se intenzivně věnuje rozšiřování systému ENUM (<http://enum.nic.cz/>) a rozvoji systému správy domén, zavádění technologie DNSSEC (<http://www.nic.cz/dnssec/>) a podpoře nových technologií a projektů prospěšných pro internetovou infrastrukturu v České republice.

CZ.NIC je členem sdružení EURid spravujícího evropskou doménu EU a dalších obdobně zaměřených mezinárodních společností (CENTR, ccNSO, ITU a další).



**Upozornění:** Nezaměňovat CZ.NIC za NIX.CZ, což je organizace zabývající se výměnou IP datagramů mezi poskytovateli v Česku (tzv. peeringem).

## Protokol whois

Již jsme zmínili termín registrátor. Registrátoři registrují přidělení internetových zdrojů jednotlivým subjektům. Za tímto účelem vedou registry (databáze) registrovaných zdrojů. Těmito zdroji jsou přidělené IP adresy, přidělené domény, přidělená čísla autonomních systémů atd. Navíc v těchto databázích jsou registrovány i subjekty, kterým byly tyto zdroje registrovány. Můžeme proto např. nalézt k šifram (zkratkám) jednotlivých kontaktních osob i jejich plné kontakty.

Registrátorů je celá řada:

- ◆ IANA vede registr TLD.
- ◆ Jednotliví RIR vedou registry k jim registrovaným zdrojům.
- ◆ Jednotliví národní registrátoři vedou registry ke svým národním doménám.
- ◆ InterNic vede registry ke generickým doménám .aero, .arpa, .biz, .cat, .com, .coop, .edu, .info, .int, .jobs, .mobi, .museum, .name, .net, .org, .pro, a .travel.

K prohlížení registrovaných zdrojů v databázích registrátorů slouží jednoduchý aplikační protokol whois. Na UNIXech máme i příslušný příkaz `whois`, který je klientem tohoto protokolu. Na hraní nepotřebujeme ani tento jednoduchý příkaz. Stačí vědět, že whois server běží na dobře známém portu 43/tcp. A potom si vystačíme i s programem `telnet`. Na příkazový řádek zadáme vždy zdroj, který nás zajímá.

### Příklad

Zjistěte informace o doméně `siemens.com`. Řešení je jednoduché. K doméně `com` je vedena databáze na serveru `whois.internic.org`. Pak stačí zadat příkaz:

```
# telnet whois.internic.org 43
siemens.com
Whois Server Version 2.0
Domain names in the .com and .net domains can now be registered
with many different competing registrars. Go to http://www.internic.net
for detailed information.

Domain Name: SIEMENS.COM
Registrar: ASCIO TECHNOLOGIES, INC.
Whois Server: whois.ascio.com
Referral URL: http://publicwhois.ascio.com
Name Server: DAVID.SIEMENS.COM
Name Server: DAVID.SIEMENS.COM.SG
Name Server: DAVID.SIEMENS.DE
Name Server: NS.SBS.DE
Status: ok
Updated Date: 10-feb-2007
Creation Date: 29-sep-1986
Expiration Date: 28-sep-2008
```

Z odpovědi je patrné, že více bychom se mohli dozvědět na *whois.ascio.com*. A tak neváháme a zadáme:

```
# telnet whois. ascio. com 43
siemens. com
Registrant:
Siemens AG (SIEMENSA240)
Wittelsbacherplatz 2
Muenchen, Bayern, 80333
DE
Domain name: siemens. com
Administrative and Technical contact:
Administration, Domain (DA6818)
Siemens AG
Otto-Hahn-Ring 6
Munich, Bavaria, 81739
DE
hostmaster@siemens. de
+49.8963655450 Fax: +49.8963644901

Record created: 2007-02-05 09:46:11
Record last updated: 2007-02-10 10:09:53
Record expires: 2008-09-28 04:00:00

Domain servers in listed order:
DAVID.SIEMENS.DE (DAVIDSIE693)
NS.SBS.DE (NSSBSDE683)
DAVID.SIEMENS.COM (DAVIDSIE989)
DAVID.SIEMENS.COM.SG (DAVIDSIE674)
```

A získáme kontaktní informace o správci domény i o příslušných jmenných serverech.

Jinou cestou, jak získat tyto informace, je webové rozhraní. V podstatě každý registrátor na svém webu má formulář, do kterého můžeme zadat hledaný řetězec.

## TLD

Tč. jsou registrovány následující TLD:

.AC	Ostrov Ascension
.AD	Andorra
.AE	Spojené arabské emiráty
.AERO	gTLD (letecký průmysl)
.AF	Afganistan
.AG	Antigua a Barbuda
.AI	Anguilla (GB)
.AL	Albánie
.AM	Arménie

.AN	Nizozemské Antily (Nizozemsko)
.AO	Angola
.AQ	Antarktida
.AR	Argentina
.ARPA	gTLD (rezervováno pro infrastrukturu Internetu)
.AS	Americká Samoa (USA)
.ASIA	gTLD (Asijsko-pacifická komunita)
.AT	Rakousko
.AU	Austrálie
.AW	Aruba (Nizozemsko)
.AX	Ålandske ostrovy (Finsko)
.AZ	Ázerbájdžán
.BA	Bosna a Hercegovina
.BB	Barbados
.BD	Bangladéš
.BE	Belgie
.BF	Burkina Faso
.BG	Bulharsko
.BH	Bahrajn
.BI	Burundi
.BIZ	gTLD (omezeno pro business)
.BJ	Bénin
.BL	Sv. Bartoloměj
.BM	Bermudy
.BN	Brunei
.BO	Bolívie
.BR	Brazílie
.BS	Bahamy
.BT	Bhútán
.BV	Bouvetův ostrov (Norsko)
.BW	Botswana
.BY	Bělorusko
.BZ	Belize
.CA	Kanada
.CAT	gTLD (Katalánská jazyková a kulturní komunita)
.CC	Kokosové ostrovy (Austrálie)
.CD	Demokratická republika Kongo
.CF	Středoafrická republika

.CG	Kongo
.CH	Švýcarsko
.CI	Pobřeží slonoviny
.CK	Cookovy ostrovy
.CL	Chile
.CM	Kamerun
.CN	Čína
.CO	Kolumbie
.COM	gTLD
.COOP	gTLD (Cooperative associations)
.CR	Kostarika
.CU	Kuba
.CV	Kapverdy
.CX	Vánoční ostrov (Austrálie)
.CY	Kypr
.CZ	Česká republika
.DE	Německo
.DJ	Džibutsko
.DK	Dánsko
.DM	Dominika
.DO	Dominikánská republika
.DZ	Alžírsko
.EC	Ekvádor
.EDU	gTLD (školství USA)
.EE	Estonsko
.EG	Egypt
.EH	Západní Sahara
.ER	Eritrea
.ES	Španělsko
.ET	Etiopie
.EU	Evropská unie
.FI	Finsko
.FJ	Rádži
.FK	Falklandské ostrovy (UK)
.FM	Mikronésie
.FO	Faerské ostrovy (Dánsko)
.FR	Francie
.GA	Gabon

.GB	Velká Británie
.GD	Grenada
.GE	Gruzie
.GF	Francouzská Guayana (Francie)
.GG	Guernsey (GB)
.GH	Ghana
.GI	Gibraltar (GB)
.GL	Grónsko (Dánsko)
.GM	Gambie
.GN	Guinea
.GOV	gTLD (vláda USA)
.GP	Guadeloupe (Francie)
.GQ	Rovníková Guinea
.GR	Řecko
.GS	Jižní Georgie a Jižní Sandwichovy ostrovy (GB)
.GT	Guatemala
.GU	Ostrov Guam (USA)
.GW	Guinea-Bissau
.GY	Guayana
.HK	Hongkong (Čína)
.HM	Heardův ostrov a Macdonaldovy ostrovy (Austrálie)
.HN	Honduras
.HR	Chorvatsko
.HT	Haiti
.HU	Maďarsko
.ID	Indonésie
.IE	Irsko
.IL	Izrael
.IM	Ostrov Man (GB)
.IN	Indie
.INFO	gTLD
.INT	gTLD (mezivládní organizace)
.IO	Britské Indickoocenánské území (GB)
.IQ	Irák
.IR	Irán
.IS	Island
.IT	Itálie
.JE	Jersey (GB)

.JM	Jamajka
.JO	Jordánsko
.JOBS	gTLD (vyhrazeno pro kádrováky)
.JP	Japonsko
.KE	Keňa
.KG	Kyrgyzstán
.KH	Kambodža
.KI	Kiribati
.KM	Komory
.KN	Svatý Kryštof a Nevis
.KP	Korejská lidově demokratická republika
.KR	Korea
.KW	Kuvajt
.KY	Kajmanské ostrovy (GB)
.KZ	Kazachstán
.LA	(nejasné)
.LB	Libanon
.LC	Svatá Lucie
.LI	Lichtenštejnsko
.LK	Srí Lanka
.LR	Libéria
.LS	Lesotho
.LT	Litva
.LU	Lucembursko
.LV	Litva
.LY	Lybie
.MA	Maroko
.MC	Monako
.MD	Moldávie
.ME	Černá Hora
.MF	Svatý Martin (Francouzská část)
.MG	Madagaskar
.MH	Marshallovy ostrovy
.MIL	gTLD (Ministerstvo obrany USA)
.MK	Makedonie
.ML	Mali
.MM	Barma – Myanmar
.MN	Mongolsko

.MO	Macao (Čína)
.MOBI	gTLD (poskytovatelé a zákazníci mobilních služeb)
.MP	Severní Mariany (USA)
.MQ	Martinik
.MR	Mauretánie
.MS	Monserrat (GB)
.MT	Malta
.MU	Mauricius
.MUSEUM	gTLD (muzea)
.MV	Maledivy
.MW	Malawi
.MX	Mexiko
.MY	Malajsie
.MZ	Mozambik
.NA	Namíbie
.NAME	gTLD (rezervováno pro jména)
.NC	Nová Kaledonie (Francie)
.NE	Niger
.NET	gTLD
.NF	Norfolk (Austrálie)
.NG	Nigérie
.NI	Nikaragua
.NL	Nizozemsko
.NO	Norsko
.NP	Nepál
.NR	Nauru
.NU	Niue (Nový Zéland)
.NZ	Nový Zéland
.OM	Oman
.ORG	gTLD
.PA	Panama
.PE	Peru
.PF	Francouzská Polynésie (Francie)
.PG	Papua Nová Guinea
.PH	Filipíny
.PK	Pákistán
.PL	Polsko
.PM	Saint-Pierre a Miquelon (Francie)

.PN	Pitcarinovy ostrovy (GB)
.PR	Portoriko
.PRO	gTLD (certifikovaní profesionálové)
.PS	Palestina
.PT	Portugalsko
.PW	Palauská republika
.PY	Paraguay
.QA	Katar
.RE	Réunion (Francie)
.RO	Rumunsko
.RS	Srbsko
.RU	Rusko
.RW	Rwanda
.SA	Saudská Arábie
.SB	Šalamounovy ostrovy
.SC	Seychelly
.SD	Súdán
.SE	Švédsko
.SG	Singapur
.SH	Svatá Helena (GB)
.SI	Slovinsko
.SJ	Svalbard a Jan Mayen (Norsko)
.SK	Slovensko
.SL	Sierra Leone
.SM	San Marino
.SN	Senegal
.SO	Somálsko
.SR	Surinam
.ST	Svatý Tomáš a Princův ostrov
.SU	Sovětský svaz (zastaralé)
.SV	Salvador
.SY	Sýrie
.SZ	Svazijsko
.TC	Turks a Caicos (GB)
.TD	Čad
.TEL	gTLD (kontaktní informace osob)
.TF	Francouzská jižní teritoria (Francie)
.TG	Togo

.TH	Thajsko
.TJ	Tádžikistán
.TK	Tokelau (Nový Zéland)
.TL	Východní Timor
.TM	Turkmenistán
.TN	Tunisko
.TO	Tonga
.TP	Portugalský Timor (zastaralé)
.TR	Turecko
.TRAVEL	gTLD (cestovní ruch)
.TT	Trinidad a Tobago
.TV	Tuvalu
.TW	Tchaj-wan
.TZ	Tanzanie
.UA	Ukrajina
.UG	Uganda
.UK	Spojení království
.UM	Menší odlehlé ostrovy USA (USA)
.US	USA
.UY	Uruguay
.UZ	Uzbekistán
.VA	Vatikán
.VC	Svatý Vincenc a Grenadiny
.VE	Venezuela
.VG	Britské Panenské ostrovy (GB)
.VI	Americké Panenské ostrovy (USA)
.VN	Vietnam
.VU	Vanuatu
.WF	Wallis a Futuna (Francie)
.WS	Samoa
.YE	Jemen
.YT	Mayotte (Francie)
.YU	Jugoslávie (zastaralé)
.ZA	Jihoafrická republika
.ZM	Zambie
.ZW	Zimbabwe

Opět můžeme hledat informace o registrovaných zdrojích. Pokud chceme vyhledat informace např. o subdoménách domény CZ, pak víme, že informace o samotné doméně CZ udržuje IANA, takže si vypíšeme tyto informace:

```
# telnet whois.iana.org 43

cz

IANA Whois Service
Domain: cz
ID: cz

Sponsoring Organization:
Organization: CZ.NIC, z.s.p.o
Address1: Americka 23
City: Prague 2
Country: Czech Republic
Postal Code: 120 00
Registration Date: 01-January-1985
Last Updated Date: 04-December-2006

Administrative Contact:
Name: Ondrej Filip
Organization: CZ.NIC, z.s.p.o.
...
Technical Contact:
Name: Martin Peterka
Organization: CZ.NIC, z.s.p.o.
Address1: Americka 23
...
URL for registration services: http://www.nic.cz/
Whois Server (port 43): whois.nic.cz
...
```

A už vidíme, kde najít informace o subdoménách domény cz. Můžeme si vypsat informace např. o doméně *siemens.cz*:

```
# telnet whois.nic.cz 43

siemens.cz

domain:      siemens.cz
registrar:   SB:GB642-RIPE_XX
admin-c:     STUDENY_JAROMIR
temp-c:      MACH_MILOS
temp-c:      LEDL_PAVEL
nset:        NSS:VB105-RIPE_XX:1
registrar:   REG-COL
status:      paid and in zone
registered:  15.08.1997 02:00:00
changed:    22.10.2005 07:20:00
```

---

```

expire:      27.10.2008

contact:     LEDL_PAVEL
name:        Pavel Ledl
address:    Evropska 33a
address:    Praha
address:    160 00
address:    CZ
phone:       +420.233031506
fax-no:      +420.266065273
e-mail:      pavel.ledl@siemens.com
registrar:   REG-COL
created:     04.03.2004 13:20:00
...

```

## Adresní prostor IPv4

Historicky byly některé velké intervaly přiděleny koncovým zákazníkům. Snaha je tuto skutečnost odstranit a veškerý prostor IPv4 alokovat RIR. V současné době jsou alokovány následující intervaly IP adres jednotlivým RIR:

<b>Prefix</b>	<b>RIR</b>	<b>Whois</b>
024.0.0.0/8	ARIN	whois.arin.net
041.0.0.0/8	AfriNIC	whois.afrinic.net
058.0.0.0/8	APNIC	whois.apnic.net
059.0.0.0/8	APNIC	whois.apnic.net
060.0.0.0/8	APNIC	whois.apnic.net
061.0.0.0/8	APNIC	whois.apnic.net
062.0.0.0/8	RIPE NCC	whois.ripe.net
063.0.0.0/8	ARIN	whois.arin.net
064.0.0.0/8	ARIN	whois.arin.net
065.0.0.0/8	ARIN	whois.arin.net
066.0.0.0/8	ARIN	whois.arin.net
067.0.0.0/8	ARIN	whois.arin.net
068.0.0.0/8	ARIN	whois.arin.net
069.0.0.0/8	ARIN	whois.arin.net
070.0.0.0/8	ARIN	whois.arin.net
071.0.0.0/8	ARIN	whois.arin.net
072.0.0.0/8	ARIN	whois.arin.net
073.0.0.0/8	ARIN	whois.arin.net
074.0.0.0/8	ARIN	whois.arin.net
075.0.0.0/8	ARIN	whois.arin.net
076.0.0.0/8	ARIN	whois.arin.net

<b>Prefix</b>	<b>RIR</b>	<b>Whois</b>
077.0.0.0/8	RIPE NCC	whois.ripe.net
078.0.0.0/8	RIPE NCC	whois.ripe.net
079.0.0.0/8	RIPE NCC	whois.ripe.net
080.0.0.0/8	RIPE NCC	whois.ripe.net
081.0.0.0/8	RIPE NCC	whois.ripe.net
082.0.0.0/8	RIPE NCC	whois.ripe.net
083.0.0.0/8	RIPE NCC	whois.ripe.net
084.0.0.0/8	RIPE NCC	whois.ripe.net
085.0.0.0/8	RIPE NCC	whois.ripe.net
086.0.0.0/8	RIPE NCC	whois.ripe.net
087.0.0.0/8	RIPE NCC	whois.ripe.net
088.0.0.0/8	RIPE NCC	whois.ripe.net
089.0.0.0/8	RIPE NCC	whois.ripe.net
090.0.0.0/8	RIPE NCC	whois.ripe.net
091.0.0.0/8	RIPE NCC	whois.ripe.net
092.0.0.0/8	RIPE NCC	whois.ripe.net
093.0.0.0/8	RIPE NCC	whois.ripe.net
094.0.0.0/8	RIPE NCC	whois.ripe.net
095.0.0.0/8	RIPE NCC	whois.ripe.net
096.0.0.0/8	ARIN	whois.arin.net
097.0.0.0/8	ARIN	whois.arin.net
098.0.0.0/8	ARIN	whois.arin.net
099.0.0.0/8	ARIN	whois.arin.net
114.0.0.0/8	APNIC	whois.apnic.net
115.0.0.0/8	APNIC	whois.apnic.net
116.0.0.0/8	APNIC	whois.apnic.net
117.0.0.0/8	APNIC	whois.apnic.net
118.0.0.0/8	APNIC	whois.apnic.net
119.0.0.0/8	APNIC	whois.apnic.net
120.0.0.0/8	APNIC	whois.apnic.net
121.0.0.0/8	APNIC	whois.apnic.net
122.0.0.0/8	APNIC	whois.apnic.net
123.0.0.0/8	APNIC	whois.apnic.net
124.0.0.0/8	APNIC	whois.apnic.net
125.0.0.0/8	APNIC	whois.apnic.net
126.0.0.0/8	APNIC	whois.apnic.net
127.0.0.0/8	IANA – Loopback	
128.0.0.0/8	Spravováno RIPE NCC	whois.ripe.net

<b>Prefix</b>	<b>RIR</b>	<b>Whois</b>
129.0.0.0/8	Spravováno AfriNIC	whois.afrinic.net
130.0.0.0/8	Spravováno RIPE NCC	whois.ripe.net
131.0.0.0/8	Spravováno LACNIC	whois.lacnic.net
132.0.0.0/8	Spravováno LACNIC	whois.lacnic.net
133.0.0.0/8	Spravováno APNIC	whois.apnic.net
134.0.0.0/8	Spravováno RIPE NCC	whois.ripe.net
135.0.0.0/8	Spravováno ARIN	whois.arin.net
136.0.0.0/8	Spravováno ARIN	whois.arin.net
137.0.0.0/8	Spravováno AfriNIC	whois.afrinic.net
138.0.0.0/8	Spravováno LACNIC	whois.lacnic.net
139.0.0.0/8	Spravováno APNIC	whois.apnic.net
140.0.0.0/8	Spravováno APNIC	whois.apnic.net
141.0.0.0/8	Spravováno RIPE NCC	whois.ripe.net
142.0.0.0/8	Spravováno ARIN	whois.arin.net
143.0.0.0/8	Spravováno LACNIC	whois.lacnic.net
144.0.0.0/8	Spravováno APNIC	whois.apnic.net
145.0.0.0/8	Spravováno RIPE NCC	whois.ripe.net
146.0.0.0/8	Spravováno RIPE NCC	whois.ripe.net
147.0.0.0/8	Spravováno ARIN	whois.arin.net
148.0.0.0/8	Spravováno LACNIC	whois.lacnic.net
149.0.0.0/8	Spravováno RIPE NCC	whois.ripe.net
150.0.0.0/8	Spravováno APNIC	whois.apnic.net
151.0.0.0/8	Spravováno RIPE NCC	whois.ripe.net
152.0.0.0/8	Spravováno LACNIC	whois.lacnic.net
153.0.0.0/8	Spravováno APNIC	whois.apnic.net
154.0.0.0/8	Spravováno AfriNIC	whois.afrinic.net
155.0.0.0/8	Spravováno AfriNIC	whois.afrinic.net
156.0.0.0/8	Spravováno AfriNIC	whois.afrinic.net
157.0.0.0/8	Spravováno APNIC	whois.apnic.net
158.0.0.0/8	Spravováno RIPE NCC	whois.ripe.net
159.0.0.0/8	Spravováno RIPE NCC	whois.ripe.net
160.0.0.0/8	Spravováno AfriNIC	whois.afrinic.net
161.0.0.0/8	Spravováno LACNIC	whois.lacnic.net
162.0.0.0/8	Spravováno ARIN	whois.arin.net
163.0.0.0/8	Spravováno APNIC	whois.apnic.net
164.0.0.0/8	Spravováno RIPE NCC	whois.ripe.net
165.0.0.0/8	Spravováno AfriNIC	whois.afrinic.net
166.0.0.0/8	Spravováno ARIN	whois.arin.net

<b>Prefix</b>	<b>RIR</b>	<b>Whois</b>
167.0.0.0/8	Spravováno LACNIC	whois.lacnic.net
168.0.0.0/8	Spravováno LACNIC	whois.lacnic.net
169.0.0.0/8	Spravováno AfriNIC	whois.afrinic.net
170.0.0.0/8	Spravováno LACNIC	whois.lacnic.net
171.0.0.0/8	Spravováno APNIC	whois.apnic.net
172.0.0.0/8	Spravováno ARIN	whois.arin.net
173.0.0.0/8	ARIN	whois.arin.net
174.0.0.0/8	ARIN	whois.arin.net
186.0.0.0/8	LACNIC	whois.lacnic.net
187.0.0.0/8	LACNIC	whois.lacnic.net
188.0.0.0/8	Spravováno RIPE NCC	whois.ripe.net
189.0.0.0/8	LACNIC	whois.lacnic.net
190.0.0.0/8	LACNIC	whois.lacnic.net
191.0.0.0/8	Spravováno LACNIC	whois.lacnic.net
192.0.0.0/8	Spravováno ARIN	whois.arin.net
193.0.0.0/8	RIPE NCC	whois.ripe.net
194.0.0.0/8	RIPE NCC	whois.ripe.net
195.0.0.0/8	RIPE NCC	whois.ripe.net
196.0.0.0/8	Spravováno AfriNIC	whois.afrinic.net
198.0.0.0/8	Spravováno ARIN	whois.arin.net
199.0.0.0/8	ARIN	whois.arin.net
200.0.0.0/8	LACNIC	whois.lacnic.net
201.0.0.0/8	LACNIC	whois.lacnic.net
202.0.0.0/8	APNIC	whois.apnic.net
203.0.0.0/8	APNIC	whois.apnic.net
204.0.0.0/8	ARIN	whois.arin.net
205.0.0.0/8	ARIN	whois.arin.net
206.0.0.0/8	ARIN	whois.arin.net
207.0.0.0/8	ARIN	whois.arin.net
208.0.0.0/8	ARIN	whois.arin.net
209.0.0.0/8	ARIN	whois.arin.net
210.0.0.0/8	APNIC	whois.apnic.net
211.0.0.0/8	APNIC	whois.apnic.net
212.0.0.0/8	RIPE NCC	whois.ripe.net
213.0.0.0/8	RIPE NCC	whois.ripe.net
216.0.0.0/8	ARIN	whois.arin.net
217.0.0.0/8	RIPE NCC	whois.ripe.net
218.0.0.0/8	APNIC	whois.apnic.net

<b>Prefix</b>	<b>RIR</b>	<b>Whois</b>
219.0.0.0/8	APNIC	whois.apnic.net
220.0.0.0/8	APNIC	whois.apnic.net
221.0.0.0/8	APNIC	whois.apnic.net
222.0.0.0/8	APNIC	whois.apnic.net

Opět si můžeme zjistit informace o veřejných IP adresách. Např. si můžeme zjistit, jak je to s IP -adresou 77.75.76.3:

```
# telnet whois.ripe.net 43
```

```
77.75.76.3
```

```
inetnum:          77.75.76.0 - 77.75.76.255
netname:         SEZNAM-CZ
descr:          Seznam.cz
country:        CZ
admin-c:        SZN5-RIPE
tech-c:         SZN5-RIPE
status:         ASSIGNED PA
mnt-by:         SEZNAM-MNT
source:         RIPE # Filtered

role:            Seznam.cz IT department
address:        Radlicka 608/2
                 150 00 Prague 5
                 Czech Republic
phone:          +420 602 126 570
abuse-mailbox: abuse@seznam.cz
...
...
```

## Adresní prostor IPv6

IPv6 adresa má 128 bitů, tj. máme  $2^{128}$  IPv6 adres. Pokud se to někomu zdá málo, pak jistě každý atom na Zemi IPv6 adresu obdržet nemůže, protože Země se skládá z přibližně  $2^{178}$  atomů.

To, co v případě IPv4 adres nazýváme veřejnými adresami, se v případě IPv6 adres nazývá *Global Unicaasts*. T.č. jsou *Global Unicaasts* alokovány následovně:

<b>Global Unicast Prefix</b>	<b>RIR</b>
2001:0000::/23	IANA
2001:0200::/23	APNIC
2001:0400::/23	ARIN
2001:0600::/23	RIPE NCC
2001:0800::/23	RIPE NCC
2001:0A00::/23	RIPE NCC

<b>Global Unicast Prefix</b>	<b>RIR</b>
2001:0C00::/23	APNIC
2001:0E00::/23	APNIC
2001:1200::/23	LACNIC
2001:1400::/23	RIPE NCC
2001:1600::/23	RIPE NCC
2001:1800::/23	ARIN
2001:1A00::/23	RIPE NCC
2001:1C00::/22	RIPE NCC
2001:2000::/20	RIPE NCC
2001:3000::/21	RIPE NCC
2001:3800::/22	RIPE NCC
2001:3C00::/22	RESERVED
2001:4000::/23	RIPE NCC
2001:4200::/23	AfriNIC
2001:4400::/23	APNIC
2001:4600::/23	RIPE NCC
2001:4800::/23	ARIN
2001:4A00::/23	RIPE NCC
2001:4C00::/23	RIPE NCC
2001:5000::/20	RIPE NCC
2001:8000::/19	APNIC
2001:A000::/20	APNIC
2001:B000::/20	APNIC
2003:0000::/18	RIPE NCC
2400:0000::/12	APNIC
2600:0000::/12	ARIN
2610:0000::/23	ARIN
2620:0000::/23	ARIN
2800:0000::/12	LACNIC
2A00:0000::/12	RIPE NCC
2C00:0000::/12	AfriNIC

Opět si můžeme zjistit informace o nějaké IPv6 adrese. Tentokrát už ale nepoužijeme příkaz telnet, ale vyzkoušíme si příkaz whois na Linuxu. Bude nás zajímat adresa 2001:660:3006:1::1:1. Z předchozí tabulky vidíme, že se jedná o adresu alokovanou RIPE NCC (DNS jméno serveru se uvádí za parametrem -h):

```
# whois -h whois.ripe.net 2001:660:3006:1::1:1
inet6num: 2001:0660:3006::/48
netname: FR-AFNIC-SFINX
descr: AFNIC
```

```
country: FR
admin-c: BT261-RIPE
admin-c: NFC1-RIPE
tech-c: NFC1-RIPE
status: ASSIGNED
mnt-by: RENATER-MNT
source: RIPE # Filtered

irt: IRT-CERT-Renater
address: c/o ENSAM
address: 151 boulevard de l'H.pital
address: 75013 Paris
address: France
phone: +33 1 5394 2044
fax-no: +33 1 5394 2031
signature: PGPKEY-D754E8CD
encryption: PGPKEY-D754E8CD
admin-c: TI123-RIPE
tech-c: TI123-RIPE
auth: PGPKEY-D754E8CD
...
...
```

## Domácí cvičení

Zjistěte informace:

- ◆ o doméně vaší firmy,
- ◆ o veřejných IP adresách přidělených vaší firmě.

## Kapitola 14

# Telnet

Protokol Telnet slouží pro interaktivní (terminálovou) práci na vzdáleném počítači. Je jedním z nejstarších aplikačních protokolů používaných v Internetu. Kořeny má v síti ARPANET. První prameny jsou z roku 1969, kdy slovo Telnet vzniklo jako akronym z „*Telecommunications network protocol*“. Byl standardizován v roce 1980 normou RFC-764, která byla v roce 1983 nahrazena dodnes platnou normou RFC-854.

### Charakteristika protokolu

Protokol Telnet slouží pro emulaci klasického terminálu v počítačových sítích na bázi protokolu TCP/IP.

Klasický terminál je vstupně-výstupní zařízení, pomocí kterého může člověk komunikovat s počítačem. Klasický terminál je hardware obsahující klávesnici a výstupní zařízení. Výstupním zařízením je buď tiskárna, nebo displej. Klasický terminál je propojen zpravidla sériovou asynchronní linkou s počítačem.

Na PC je možné emulovat klasický terminál programem Terminal (resp. HyperTerminal u novějších systémů Microsoft). Často kladenou otázkou začátečníků je: „A jaký je rozdíl mezi těmi programy Terminal (Hyperterminal) a Telnet? Vždyť to dělá to samé!“ Tato otázka mne vždy zaskočí. Rozdíl je naprosto zásadní. Zatímco program HyperTerminal z PC dělá terminál připojený sériovou linkou k serveru (tj. buď nulovým modemem, či přes dvojici modemů), program Telnet emuluje terminál přes TCP/IP. Odpověď pak zní: „Zatímco při použití programu HyperTerminal komunikujete přes COM-port vašeho PC bez nějakého TCP/IP, programem Telnet komunikujete přes síťovou kartu (abstrahuji od případu, že je PC připojeno protokolem SLIP nebo PPP do sítě TCP/IP).“ Ani to většinou začátečníky neuspokojí, proto nasadím další argument: „No a na straně serveru, pakliže komunikujete přes HyperTerminal, každý potřebujete svou zásuvku (svůj sériový komunikační port), kdežto když se použije Telnet, stačí serveru i jedno síťové rozhraní (např. Ethernet) pro vás pro všechny.“

Použití Telnetu není omezeno pouze na práci na vzdáleném počítači. Jak uvidíme v dalších částech, správce sítě ocení Telnet při testech mnoha dalších protokolů: FTP, POP3, SMTP, HTTP, NNTP apod.

I když mnozí používáte program Telnet denně, při popisu protokolu Telnet vám bude připadat, že snad popisují nějaký jiný protokol. Nikoliv, Telnet je poměrně komplikovaný protokol, to aby vám zajistil vaše každodenní pohodlí při práci s ním.



**Poznámka:** K zásadní změně došlo ve Windows Vista. Zde již není k dispozici ani program Terminal, ani program HyperTerminal (v případě potřeby je nutné využít programy třetích stran – např. program putty). A aby toho nebylo až tak málo, tak ve Windows Vista není standardně k dispozici ani klient, ani server protokolu Telnet. Avšak jak klient, tak i server protokolu Telnet je možné ve Windows Vista využívat v případě, že je zahrneme v menu „Zapnout nebo vypnout funkce systému Windows“.

## Bezpečnost

Na bezpečnostně citlivých serverech je třeba po instalaci okamžitě zkontrolovat, zdali není spuštěn server pro Telnet. V případě, že ano, je třeba jej okamžitě zastavit a zajistit, aby se náhodou automaticky nenastartoval po opětovném startu systému. Naopak klient protokolu Telnet je mnohdy velice užitečný.

Telnet přes SSL (resp. TLS) je tč. málo rozšířen. V praxi se v případech vyžadujících bezpečnou komunikaci používá spíše SSH.

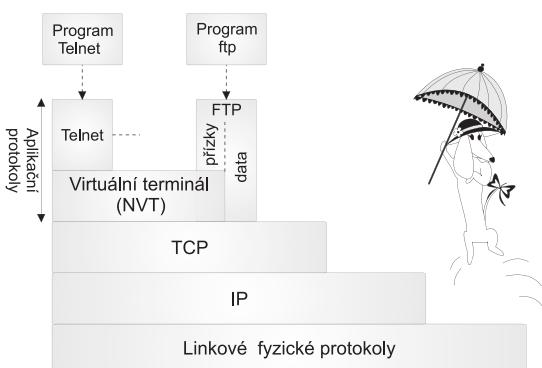
## Protokol Virtuální terminál (NVT)

NVT je zkratka vzniklá z *network virtual terminal*. NVT je podmnožinou protokolu Telnet, tj. jako by se protokol Telnet skládal ze dvou vrstev: ze spodní vrstvy – NVT – a horní vrstvy – vlastního protokolu Telnet. Protokol NVT se zabývá prezentací dat, tj. v jaký bajt se má změnit písmeno A, aby na druhém konci síťového spojení bylo interpretováno opět jako A, či jaký příkaz protokolu Telnet se má vygenerovat po stisku známého ^C pro abnormální ukončení programu spouštěného z terminálu.

Právě protokol NVT je použit v omezené míře pro prezentaci dat v mnoha dalších protokolech, jako jsou FTP, POP3, SMTP, NNTP i HTTP apod. MIME je v podstatě rozšířením této filozofie.



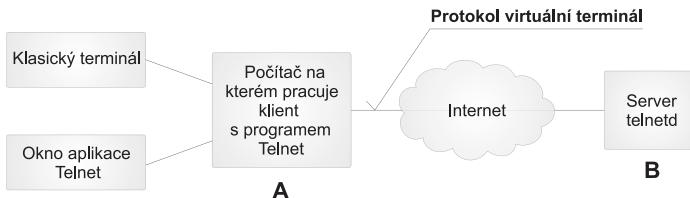
**Tip:** Mnohé aplikacní protokoly rovněž používají k prezentaci dat protokol NVT, a proto tyto protokoly lze testovat programem Telnet.



**Obrázek 14.1:** Síťová architektura protokolu NVT

Na obr. 14.2 je znázorněna základní filozofie práce s protokolem Telnet. Klient pracuje na počítači A budě:

- ◆ z klasického terminálu, ze kterého spouští program Telnet na počítači A ( operační systém počítače A je např. Unix či Windows),
- ◆ nebo sedí přímo u počítače A (např. s Windows) a spustí si na počítači A program Telnet, který mu otevře své okno přímo na počítači A.



**Obrázek 14.2:** Protokol virtuální terminál (NVT)

Nyní klient sedící u počítače A potřebuje pracovat na počítači B. Klient spustí program Telnet s parametrem jména vzdáleného počítače, na který se chce spojit – tj. s parametrem B. Program Telnet naváže spojení protokolem TCP na port 23 počítače B. Na tomto portu čeká server protokolu Telnet (program telnetd).

Problém spočívá v tom, že reprezentace dat na počítači A může být různá oproti reprezentaci dat na počítači B, tj. počítač A může používat např. kód ASCII, kdežto počítač B zcela jiný kód. Protokol NVT proto specifikuje reprezentaci dat, která se přenáší síť. Tato reprezentace dat se právě nazývá protokolem NVT. Protokol Telnet tak nepředepisuje ani počítači A, ani počítači B, jakou mají používat prezentaci dat lokálně (např. pro ukládání dat na disky), ale specifikuje, jakou mají oba počítače používat prezentaci dat při vkládání (resp. přijímání) dat do sítě.

Protokol Telnet proto nezajímá, jakou reprezentaci dat mají počítače A a B. Počítač A je povinen do TCP segmentů vkládat data v reprezentaci NVT. Obdobně počítač B si data konvertuje z NVT do své vlastní reprezentace.

Základ reprezentace dat v protokolu NVT je vzato prvních 128 znaků kódu ASCII (nejvyšší 8. bit je roven nule). Přitom prvních 32 znaků jsou řídicí znaky (nový řádek – NL, návrat vozíku – CR, zvonek – BEL atd.). Konec řádku je reprezentován dvojicí znaků „návrat vozíku (CR)“ a „nový řádek (NL)“. Na tomto případě je krásně vidět práce s protokolem NVT. Pokud jsou např. oba systémy, A i B, s operačním systémem Unix, je konec řádku reprezentován pouze znakem NL. Takže pokud se odesílá konec řádku z počítače A, musí počítač A do TCP segmentu na místo jednoho znaku NL vložit dva znaky CR a NL. Naopak počítač B musí dvojici CR a NL nahradit jedním znakem NL. Pokud počítač B obdrží pouze NL (bez CR), pak neobdržel data, která nereprezentují konec řádku, ale něco jiného.

## Příkazy protokolu Telnet

Zvláštní význam má znak „IAC“ (*Interpret as command*) s dekadickou hodnotou 255 (šestnáctkově FF). Pokud se opravdu má přenést znak s touto hodnotou, musí být zdvojen. Znak IAC se interpretuje jako začátek příkazu protokolu Telnet.

Příkazy protokolu Telnet jsou tak uvozeny znakem IAC. Za znakem IAC mohou následovat příkazy uvedené v následující tabulce.

**Tabulka 14.1:** Příkazy protokolu Telnet

Desítkově	Šestnáctkově	Označení	Význam
236	EC	EOF	Konec souboru ( <i>End of file</i> )
237	ED	SUSP	Suspenduj proces – vyšli proces signál SUSP
238	EE	ABORT	Ukonči proces signálem ABORT ( <i>Abort process</i> )
239	EF	EOR	Konec věty ( <i>End of record</i> )
240	F0	SE	Konec parametrů příkazu ( <i>Suboption end</i> )
241	F1	NOP	Prázdná operace ( <i>No operation</i> )
242	F2	DM	Značka pro označení urgentních dat v segmentu TCP ( <i>Data mark</i> )
243	F3	BRK	Ukonči proces signálem BREAK ( <i>Break process</i> )
244	F4	IP	Ukonči proces signálem INTERRUPT ( <i>Interrupt process</i> )
245	F5	AO	Zahodí výstup ( <i>abort output</i> )
246	F6	AYT	Jsi tam? ( <i>Are you there?</i> )
247	F7	EC	Únik do příkazového rámku ( <i>Escape character</i> )
248	F8	EL	Zruš řádek ( <i>Erase line</i> )
249	F9	GA	„Příjem“ ( <i>Go ahead</i> )
250	FA	SB	Počátek parametrů příkazu ( <i>Suboption begin</i> )
251	FB	WILL	Viz tab. 14.2
252	FC	WONT	
253	FD	DO	
254	FE	DONT	
255	FF	IAC	Datový bajt o hodnotě dekadicky 255 (šestnáctkově FF)

Vraťme se k obrázku 14.2 a zamysleme se nad příkazy v předchozí tabulce: Když bude uživatel chtít např. zrušit běžící proces příkazem ABORT, BREAK či INTERRUPT, stiskne na terminálu příslušné klávesy. Na základě jejich stisku se vygeneruje příslušný příkaz ke zrušení procesu (např. příkaz IAC ABORT – tj. FF EE). Jenže otázkou zůstává, mají-li se příslušné příkazy interpretovat lokálně na počítači A (tj. mají-li se z nich vygenerovat příkazy IAC protokolu Telnet), nebo mají-li se tyto informace přenést jako data na server B, který je bude interpretovat.

V praxi se setkáváme s oběma variantami. Lokální interpretaci příkazů a vygenerování IAC-příkazů lze v Unixu nastavit v příkazovém řádku programu Telnet-příkazem (není implementováno u klienta firmy Microsoft):

```
Telnet> toggle localchars
```

Prakticky se avšak lokální interpretace ve znakovém režimu nepoužívá. Naopak v řádkovém režimu se toto nastavení ignoruje. Lokální interpretace se použije vždy.

Z příkazového řádku lze odesílat i jednotlivé IAC-příkazy „ručně“ pomocí příkazu send. Např. zrušit proces signálem BREAK lze pomocí příkazu (není implementováno u klienta firmy Microsoft):

```
Telnet> send brk
```

Velice zajímavým příkazem je příkaz AYT („Jsi tam, servere?“). Je jakousi obdobou příkazu ping. Pokud je spojení se serverem OK, server odpoví příkazem YES (není implementováno u klienta firmy Microsoft):

```
Telnet> send ayt
[Yes]
```

kde řetězec [Yes] vrátil server – pakliže se takovýto řetězec neobjeví, je v komunikaci se serverem něco v nepořádku nebo komunikace vůbec není navázána.

Základním nástrojem protokolu Telnet jsou následující složené příkazy, kterými se odesilatel s příjemcem dohadují na nastavení voleb pro vzájemnou komunikaci:

- ◆ WILL, kterým odesilatel říká, že by rád používal nějakou volbu.
- ◆ DO, kterým odesilatel příjemci sděluje, aby příjemce používal nějakou volbu.
- ◆ WONT, kterým odesilatel příjemci sděluje, že odesilatel nepoužije nějakou volbu.
- ◆ DONT, kterým odesilatel příjemci sděluje, aby příjemce nepoužil nějakou volbu.

Je možných šest případů vzájemné komunikace:

**Tabulka 14.2:** Základní schéma komunikace složenými příkazy

	Odesilatel		Příjemce	Popis
1	WILL	→ ←	DO	Odesilatel si přeje používat nějakou volbu. Příjemce potvrzuje použití volby.
2	WILL	→ ←	DONT	Odesilatel si přeje používat nějakou volbu. Příjemce zamítá použití volby odesilatelem.
3	DO	→ ←	WILL	Odesilatel si přeje, aby příjemce použil nějakou volbu. Příjemce souhlasí.
4	DO	→ ←	WONT	Odesilatel si přeje, aby příjemce použil nějakou volbu. Příjemce odmítá.
5	WONT	→ →	DONT	Odesilatel si přeje nepoužívat nějakou volbu. Příjemce je povinen potvrdit nepoužití této volby.
6	DONT	→ ←	WONT	Odesilatel si přeje, aby příjemce nepoužíval nějakou volbu. Příjemce je povinen potvrdit nepoužití této volby.

Mnohé volby mohou mít i parametry. Seznam některých voleb je uveden v následující tabulce (podrobný seznam najeznete např. v RFC-2400):

**Tabulka 14.3:** Volby protokolu Telnet

Název	desítkově	šestnáctkově	Význam	RFC
ECHO	1	1	Pokud na terminálu zmáčkneme např. klávesu „A“, pak očekáváme, že se nám znak „A“ objeví na obrazovce terminálu, abychom měli jistotu, že jsme příslušnou klávesu opravdu zmáčkli. Poslat znak na obrazovku může lokálně nás terminál nebo server protokolem TELNET. Vybou ECHO žádáme o tuto službu server.	857
SUPPRESS GO AHEAD	3	3	Potlačení poloduplexního režimu.	858
STATUS	5	5	Status.	859
TIMING MARK	6	6	Značka. Dnes se nejčastěji používá pro komunikaci pseudořádkovém režimu.	860
TERMINAL TYPE	24	18	Typ terminálu tato volba může mít následující parametry: 1 = „Pošli“ – požadavek serveru na zaslání typu terminálu. 0 = „Posílám“ – klient zasílá typ terminálu. Po volbě 0 bezprostředně následuje řetězec obsahující typ terminálu.	1091
NAWS	31	1F	Dohoda na velikosti okna terminálu (tj. klient informuje server o počtu řádků a sloupců svého okna terminálu). Tato volba může mít dva dvoubajтовé parametry: počet sloupců a počet řádků.	1073
TSPEED	32	20	Rychlosť pripojenia terminálu (sériové linky klasického terminálu). Volba může obsahovať dva parametry: rychlosť vysílania a rychlosť príjmu.	1079
LFLOW	33	21	Řízení toku dat (tj. zpracování ^S a ^Q – pozastavení a opětovný start výstupu terminálu) tato volba může mít následující parametry: 0 = OFF (používá např. editor vi), 1 = ON, 2 = RESTART-ANY, 3 = RESTART-XON.	1372
LINEMODE	34	22	Řádkový režim tato volba může použít řadu parametrů.	1184
XDISPLOC	35	23	Umístění X-serveru v X-Windows. Význam parametrů je stejný jako u volby TERMINAL TYPE. Rozdílem je, že místo typu terminálu se odesílá umístění X-serveru ve formátu host:display[.obrazovka].	1096
OLD ENVIRON	36	24	Proměnné prostředí tato volba byla inovována normou RFC-1572, viz další řádek.	1408
NEW ENVIRON	39	27	Proměnné prostředí tato volba používá řadu parametrů (viz RFC-1572). V praxi se přenáší dvě proměnné prostředí: DISPLAY a PRINTER.	1572
STARTTLS	46	2E	Telnet Start TLS.	
KERMIT	47	2F	Telnet Kermit.	
EXTOP	255	FF	Rozšířený seznam voleb.	861

Příkladem použití složených příkazů je vyžádání si typu terminálu. Pokud totiž server zná typ (a tím i vlastnosti) terminálu, u kterého uživatel sedí, může uživateli nabídnout např. celoobrazovkový režim editoru vi apod. Jednou z možností komunikace vedoucí k získání typu terminálu je, že server iniciuje vyžádání typu terminálu. V prvním kroku se server klienta zeptá, zdali je mu vůbec klient typ terminálu schopen poskytnout – komunikace DO/WILL (když klient odpovídá kladně). V dalším kroku pak server vyšle složený příkaz (a to s parametry), kterým si vyžádá řetězec obsahující konkrétní typ terminálu. Složený příkaz s parametry je vždy uzavřen mezi dvojici „závorek“ SB a SE. Příklad této komunikace obsahuje následující tabulka (inverzně je uvedena šestnáctková hodnota následovaná příkazem v textové formě):

**Tabulka 14.4** Využití složených příkazů pro vyžádání typu terminálu

Klient		Server	Význam
	←	FF FD 18 <IAC DO TERMINAL-TYPE>	Klient, pošli mi typ tvého terminálu.
FF FB 18 <IAC WILL TERMINAL-TYPE>	→		Souhlasím s poskytnutím typu terminálu.
	←	FF FA 18 01 FF FO <IAC SB TERMINAL-TYPE pošli> <IAC SE>	Zašli mi typ svého terminálu. Parametr 01 = Pošli.
FF FA 18 00 76 74 31 30 30 FF FO <IAC SB TERMINAL-TYPE posílám> vt100 <IAC SE>	→		Klient posílá serveru typ svého terminálu, který je vt100. Parametr 00 = Zasílám typ terminálu (vt100).

### Signál pro synchronizaci (SYNCH)

Signál pro synchronizaci použije klient, chce-li, aby server zahodil ještě nezpracovaná data odeslaná klientem na server. Postup je následující:

- Do toku odesílaných dat z klienta na server se vloží příkaz protokolu Telnet <IAC DM>, tj. FF F2.
- TCP segment obsahující příkaz <IAC DM> se v záhlaví TCP segmentu označí příznakem „Urgentní data“ s tím, že urgentními daty je právě příkaz <IAC DM>.
- Jestliže server obdrží TCP segment s urgentními daty, všechna nezpracovaná data ve své vyrovnavací paměti (*buffer*) přeskočí až po příkaz <IAC DM>.

Synchronizaci lze vynutit příkazem z příkazového řádku programu Telnet (není implementováno u klienta firmy Microsoft):

```
Telnet> send synch
```

### Úniková sekvence a příkazový rádek programu Telnet

Uživatel pracující s programem Telnet pracuje v lokálním operačním systému a je díky protokolu Telnet připojen do operačního systému vzdáleného stroje. Úniková sekvence je zpravidla znak ^]

(tj. současný stisk klávesy Ctrl a klávesy ]). Stiskne-li uživatel tento znak, získá příkazový řádek programu Telnet na lokálním stroji:

Telnet>

respektive ve Windows:

Microsoft Telnet>

Na tento příkazový řádek pak může uživatel zadávat příkazy programu Telnet. Stejný příkazový řádek obdržíme, spustíme-li program Telnet bez parametru.

Spojení (relaci) lze navázat příkazem:

Telnet> open server [port]

kde port protokolu TCP je nepovinný – neuvede-li se, předpokládá se implicitní port 23. Avšak je v tom háček. Navázání spojení (např. v Unixu) na implicitní port z hlediska protokolu Telnet bývá jiné, než je navázání spojení na jiný port. V případě, že klient navazuje spojení na implicitní port, může věřit tomu, že na portu 23 opravdu čeká server protokolu Telnet, tj. klient může klidně odeslat IAC-příkazy: DO SUPPRESS-GO-AHEAD, WILL TERMINAL-TYPE, WILL LFLOW apod., protože server těmto příkazům bude s největší pravděpodobností rozumět.

Avšak pokud uživatel použil program Telnet pro komunikaci např. se serverem SMTP, server může být z takového příkazu WILL TERMINAL-TYPE zmaten. V navazování spojení na jiném portu program Telnet pouze naváže spojení na úrovni protokolu TCP a počká, jak zareaguje server. Pokud server začne s IAC-příkazy (např. IAC DO TERMINAL-TYPE), klient pochopí, že komunikuje se serverem protokolu Telnet, a rovněž pošle IAC-příkazy na server. V případě, že server nepošle žádný IAC-příkaz, pak předpokládá, že nejde o žádný Telnet, a IAC-příkazy server neobtěžuje. (Pokud uživatel chce, aby klient od počátku komunikoval IAC-příkazy na explicitně uvedeném portu, pak před číslo portu uvede znak „-“.)

Obdobně příkaz close slouží k ukončení spojení.

Příkazem set lze nastavit proměnné programu Telnet. Např.:

Telnet> set escape ^G

se nastaví únikové sekvence na ^G (není implementováno u klienta firmy Microsoft).

Telnet> set tracefile soubor

se výstup ladicích informací přesměruje do uvedeného souboru (není implementováno u klienta firmy Microsoft).

Příkazem toggle lze mj. aktivovat tisk ladicích informací (není implementováno u klienta firmy Microsoft). Ladicí informace jsou velmi zajímavé, např. pro pochopení protokolu Telnet. I já budu demonstrovat některé funkce protokolu Telnet na základě těchto ladicích informací. Výpis ladicích informací jsem obdržel takto:

Telnet> toggle options

tak jsem zapnul vypisování IAC-příkazů v textovém tvaru. Např. výpis:

SENT DO SUPPRESS-GO-AHEAD

znamená, že klient odeslal (SENT) IAC-příkaz DO SUPPRESS-GO-AHEAD (tj. šestnáctkově FF FD 03).

Jiným typem výpisu je výpis celého datového paketu (bez záhlaví linkového protokolu i bez záhlaví protokolů IP a TCP) příkazem:

```
Telnet> toggle netdata
```

který nadále bude vypisovat přenášená data. Příklad výpisu jednoho paketu:

```
> 0x0 fffa2000393630302c39363030ffff0ffa230074312e7076742e637a3a302e30  
> 0x20 fff0ffa270000444953504c41590174312e7076742e637a3a302e30ffff0ffa  
> 0x40 180044545445524dff0
```

Zde znak „>“ znamená, že paket byl odeslán klientem na server. Znak „menší než“ by obdobně znamenal příjem paketu ze serveru. Prostřední sloupec (začínající 0x0) určuje posunutí prvního znaku řádku od počátku výpisu paketu v šestnáctkové soustavě. A pravý (nejširší) sloupec obsahuje vlastní data.

Příkazem status lze získat aktuální nastavení relace.

Příkazem send lze „ručně“ odeslat IAC-příkaz. Příklady jsem již uvedl. Za zmínu snad ještě stojí příkaz getstatus. Příkazem IAC STATUS SEND klient požádá server o zaslání stavových informací (aby výpis měl smysl, předem si nastavte toggle options (příkaz není implementován u klienta firmy Microsoft)):

```
Telnet> send getstatus  
Sent suboption STATUS SEND (odeslání žádosti o status)  
RCVD IAC SB  
Received suboption STATUS IS (server podporuje:  
    WILL ECHO - znakový režim  
    WILL SUPPRESS GO AHEAD  
    WILL STATUS - odpovídат na žádost o status  
    DO TERMINAL TYPE - zpracovávat typ terminálu klienta  
    DO NAWS - zpracovávat velikost okna klienta  
    DO TSPEED - zpracovávat rychlosť linky k.l.  
    DO LFLOW - zpracovávat řízení toku dat  
    DO LINEMODE - řádkový režim  
    DO XDISPLOC - umístění X-serveru klienta  
    DO NEW-ENVIRON - proměnné prostředí podle RCF-1572)
```

Příkazem mode lze měnit režimy komunikace.

## Režimy komunikace

Komunikace v protokolu Telnet může použít jeden ze čtyř režimů:

1. Poloduplexní režim (*Half-duplex*). Tento režim je obdobou rádiové komunikace, kdy spolu komunikují dva subjekty přes společně sdílenou frekvenci. Pokud jedna strana spojení vysílá, druhá musí naslouchat. Strana vysílá až do signálu „přepínám“, což znamená přechod z vysílání na příjem. Místo slova „přepínám“ používá protokol Telnet příkaz <IAC GA>. S tímto režimem se dnes již nesetkáváme.
2. Znakový režim (*Character at a time*). Je dnes nejčastějším režimem. Bývá implicitně nastaven na klientech i serverech. V tomto režimu odesílá klient na server znak po znaku (většinou klient odesílá každý znak v samostatném TCP segmentu, tj. jeden znak je obalen dalšími minimálně 20 bajty IP záhlaví a minimálně 20 znaky záhlaví TCP segmentu).

Přechod do znakového režimu se provádí dialogem:

```
DO SUPPRESS-GO-AHEAD      >
                            <      WILL SUPPRESS-GO-AHEAD
                            <      WILL ECHO
DO ECHO                  >
```

Obě volby (SUPPRESS-GO-AHEAD) a ECHO jsou tedy aktivní.

3. Pseudořádkový režim (*Line at a time nebo „kludge“ line mode*) je odvozen od znakového režimu. Znakový režim vyžaduje, aby byly současně aktivní volby SUPPRESS-GO-AHEAD a ECHO. Pseudořádkový režim je obdobou znakového režimu, kdy jedna z uvedených voleb není aktivní.

Klient provádí lokálně ECHO sám. Problém je se zadáním hesla, které si nepřejeme, aby bylo zobrazováno. Proto se před zadáním hesla přejde do znakového režimu, kdy server odešle WONT ECHO a klient potvrdí DONT ECHO. Po zadání hesla server přejde zpět do původního režimu příkazem WILL ECHO, který klient potvrdí DO ECHO.

4. Řádkový režim (*Linemode*). Celý vstupní řádek se zpracovává na straně klienta (včetně případných úprav řádku). Pak je teprve celý řádek odeslán na server. Přechod do řádkového režimu se provede:

```
WILL LINEMODE      >
                    <      DO LINEMODE
```

V řádkovém režimu je opět problém se zadáváním hesla, který se obdobně jako u pseudořádkového režimu řeší dočasním přechodem do řádkového režimu. Obdobně se např. řeší i práce s obrazovkovým editorem vi.

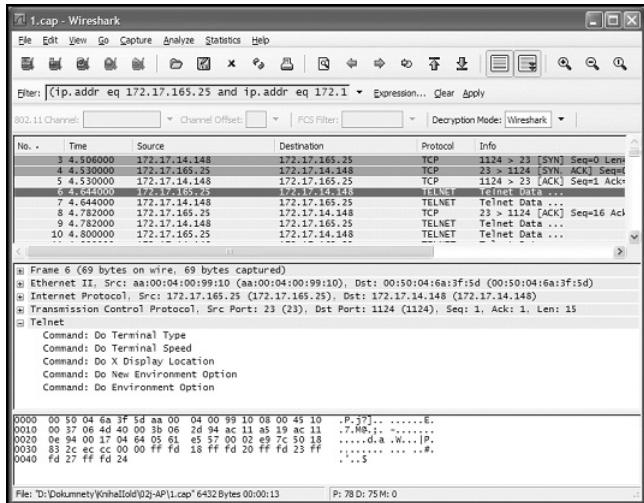
## Příklad komunikace klienta z Windows

Náš příklad komunikace klienta protokolu Telnet z Windows je se serverem s operačním systémem Unix. Ve Windows nemáme k dispozici příkaz toggle, takže nemůžeme zkoumat protokol Telnet pomocí ladicího výpisu.

To nás však neodradí. Vzpomeneme si na program Wireshark, spustíme jej a odchytíme si jednotlivé pakety protokolu Telnet (viz obr. 14.3).

Předpokládáme, že komunikace byla navázána na úrovni protokolu TCP. Klient Microsoft na implicitním portu nezačíná s komunikací pomocí IAC-příkazů protokolu Telnet, ale počká, vyklube-li se ze serveru protokolu Telnet.

Tab. 14.5 obsahuje vyčerpávající výpis této komunikace. Např. aplikační data z obr. 14.3 jsou rozpitívána do pěti jednotlivých IAC-příkazů. Nejprve je uveden IAC-příkaz šestnáctkově a vždy následuje jeho textový ekvivalent.



Obrázek 14.3: Paket protokolu Telnet obsahuje IAC-příkazy odesílané serverem

Tabulka 14.5: Komunikace klient-server protokolu Telnet (klient Windows)

Klient		Server	Význam
	←	<b>FF FF FD 18</b> <IAC DO TERMINAL-TYPE> <b>FF FD 20</b> <IAC DO TSPEED> <b>FF FD 23</b> <IAC DO XDISPLOC> <b>FF FD 27</b> <IAC DO NEW-ENVIRON> <b>FF FD 24</b> <IAC DO OLD-ENVIRON>	<ul style="list-style-type: none"> <li>Klient, pošli mi typ tvého terminálu</li> <li>Klient, pošli mi rychlosť své linky</li> <li>Klient, pošli mi umístění svého okna (pro X-Windows)</li> <li>Pošli mi proměnné svého prostředí (vlastně výpis příkazu SET ve Windows)</li> <li>Pošli mi proměnné prostředí (tato volba je zastaralá)</li> </ul>
<b>FF FB 18</b> <IAC WILL TERMINAL-TYPE>	→		<ul style="list-style-type: none"> <li>Souhlasím s poskytnutím typu terminálu (odpověď na DO)</li> </ul>
<b>FF FC 20</b> <IAC WONT TSPEED> <b>FF FC 23</b> <IAC WONT XDISPLOC> <b>FF FC 27</b> <WONT NEW-ENVIRON> <b>FF FC 24</b> <WONT OLD-ENVIRON>	→	<ul style="list-style-type: none"> <li>Nesouhlasím s poskytnutím rychlosti své linky (uživatel nepracuje s klasickým terminálem)</li> <li>Nesouhlasím s poskytnutím umístění svého okna (uživatel nepracuje v X-Windows)</li> <li>Neposkytnu proměnné svého prostředí</li> </ul>	
	←	<b>FF FA 18 01</b> <IAC SB TERMINAL-TYPE pošli> <b>FF FO</b> <IAC SE>	<ul style="list-style-type: none"> <li>Zašli mi typ svého terminálu. Jelikož se tato volba použije s parametrem, je nutné volbu uložit mezi závorky SB a SE</li> </ul>

Klient		Server	Význam
FF FA 18 00 76 74 31 30 30 <IAC SB TERMINAL-TYPE posílám> <b>vt100</b> FF F0 <IAC SE>	→		<ul style="list-style-type: none"> <li>Klient posílá serveru typ svého terminálu, který je vt100. Jelikož se tato volba použije s parametrem, je nutné volbu uložit mezi závorky SB a SE</li> </ul>
	←	FF FB 03 <IAC WILL SUPPRESS -GO-AHEAD> FF FD 01 <IAC DO ECHO> FF FD 1F <IAC DO NAWS> FF FB 05 <IAC WILL STATUS> FF FD 21 <IAC DO LFLOW>	<ul style="list-style-type: none"> <li>Server si přeje nepoužívat poloduplexní režim</li> <li>Server chce, aby klient prováděl ECHO</li> <li>Server chce, aby klient používal NAWS</li> <li>Server si přeje STATUS</li> <li>Server chce, aby klient používal RFC</li> </ul>
FF FD 03 <IAC DO <IAC DO SUPPRESS-GO-AHEAD>	→		<ul style="list-style-type: none"> <li>Klient potvrzuje nepoužití poloduplexního režimu</li> </ul>
FF FB 01 <IAC WILL ECHO> FF FC 1F <IAC WONT NAWS> FF FC 05 <IAC DONT STATUS> FF FC 21 <IAC WONT LFLOW>	→		<ul style="list-style-type: none"> <li>Klient souhlasí s tím, že bude provádět ECHO</li> <li>Klient odmítá NAWS</li> <li>Klient odmítá STATUS</li> <li>Klient odmítá RFC</li> </ul>
	←	FF FE 01 <IAC DONT ECHO> FF FB 01 <IAC WILL ECHO>	<ul style="list-style-type: none"> <li>Server nechce, aby klient prováděl ECHO</li> <li>Server chce sám provádět ECHO</li> </ul>
FF FC 01 <IAC WONT ECHO>	→		<ul style="list-style-type: none"> <li>Klient nebude provádět ECHO</li> </ul>
	←	<b>Login:</b>	
FF FD 01 <IAC DO ECHO>	→		<ul style="list-style-type: none"> <li>Klient souhlasí s tím, že server bude provádět ECHO</li> </ul>
<b>Jméno uživatele</b>	→		
	←	<b>Password:</b>	

## Příklad komunikace klienta z Unixu

Náš klient byl spuštěn v okně X-Windows též na systému UNIX. Server byl rovněž UNIX.

Příklad komunikace obsahuje obr. 14.4: Nejprve jsem spustil program Telnet bez parametru. Tak jsem obdržel příkazový řádek programu Telnet. Z příkazového řádku jsem spustil výpis ladících informací, a to jak výpis IAC-příkazů v textovém tvaru (1), tak i výpis celých aplikačních dat v šestnáctkovém tvaru (2).

Příkazem open jsem navázal spojení se serverem *t1.pvt.cz* (3). Program Telnet nastavil únikovou sekvenci na znak ^] (4). Jelikož jsem v příkazu open nespecifikoval číslo portu, klient (na rozdíl od klienta Windows) předpokládá, že se navazuje spojení se serverem protokolu Telnet. Klient proto odešle okamžitě po navázání spojení IAC-příkazy protokolu Telnet:

```
$ telnet
telnet> toggle options 1
Will show option processing.
telnet> toggle netdata 2
Will print hexadecimal representation of network traffic.
telnet> open t1.pvt.cz 3
Trying t1.pvt.cz...
Connected to t1.pvt.cz.
Escape character is '^]'.
4 SENT do SUPPRESS GO AHEAD
5 SENT will TERMINAL TYPE
6 SENT will NAWS
7 SENT will TSPEED
8 SENT will LFLOW
9 SENT will LINEMODE
10 SENT will NEW-ENVIRON
11 SENT will XDISPLOC
12 SENT do STATUS
13 SENT will XDISPLOC
14 > 0x0 fffbd03 fffbd18 fffbd1f fffbd20 fffbd21 fffbd22 fffbd27 fffbd05 fffbd23
15 < 0x0 fffbd18 fffbd20 fffbd23 fffbd27 fffbd24
16 RCVD do TERMINAL TYPE
17 RCVD do TSPEED
18 RCVD do XDISPLOC
19 RCVD do NEW-ENVIRON
20 RCVD do OLD-ENVIRON
21 SENT won't OLD-ENVIRON
22 > 0x0 fffc24
< 0x0 fffbd03ffffd1ffffd21ffffe22ffffb05 23
RCVD will SUPPRESS GO AHEAD 24
RCVD do NAWS 25
Sent suboption NAWS 0 80 (80) 0 24 (24) 26
RCVD do LFLOW 27
RCVD don't LINEMODE 28
RCVD will STATUS 29
> 0x0 fffa1f00500018ffff
< 0x0 fffa2001ffff0ffffa2301ffff0ffffa2701ffff0ffffa1801ffff
RCVD IAC SB 30
Received suboption TERMINAL-SPEED SEND
Sent suboption TERMINAL-SPEED IS 9600,9600 31
RCVD IAC SB 32
Received suboption X-DISPLAY-LOCATION SEND
Sent suboption X-DISPLAY-LOCATION IS "t1.pvt.cz:0.0" 33
RCVD IAC SB
Received suboption NEW-ENVIRON SEND 34
Sent suboption NEW-ENVIRON IS VAR "DISPLAY" VALUE "t1.pvt.cz:0.0" 35
RCVD IAC SB
Received suboption TERMINAL-TYPE SEND 36
Sent suboption TERMINAL-TYPE IS "DTTERM" 37
> 0x0 fffa2000393630302:39363030fff0ffffa230074312e7076742e637a3a302e30
> 0x20 fff0ffffa270006444953504c41590174312e7076742e637a3a302e30ffff0ffffa
> 0x40 18004454545524dff0
< 0x0 rffffd01
RCVD do ECHO 38
SENT won't ECHO 39
> 0x0 fffbd01
< 0x0 fffbd01
RCVD will ECHO 40
SENT do ECHO 41
> 0x0 fffbd01
< 0x0 0d000d0a0d000d0a4469676974616c20554e4958202874312e7076742e637a29
< 0x20 20287474797033290d000d0a0d000d000d0a0d00
Digital UNIX (t1.pvt.cz) (ttyp3) 42
< 0x0 6c6f67696e3a20
login: 43
```

Obrázek 14.4: Komunikace klient-server v UNIXu

- (5) Nekomunikuj v poloduplexním režimu.
- (6) Chci ti poslat typ svého terminálu.
- (7) Chci ti poslat počet řádků a sloupců svého okna.
- (8) Chci ti poslat rychlosť linky svého terminálu.
- (9) Chci se s tebou dohodnout na řízení toku dat.
- (10) Chtěl bych pracovat v řádkovém režimu.
- (11) Chtěl bych zaslat proměnné prostředí podle RFC-1572.
- (12) Zašli mi status.

(13) Chtěl bych ti poslat umístění svého X-serveru.

Příkazy (5 až 13) odesal klient na server v paketu, který je zobrazen na řádku (14). Server pak odpovídá IAC-příkazy v paketu, který je zobrazen šestnáctkově na řádku (15). Jednotlivé IAC-příkazy jsem v obou řádcích oddělil mezerou. Paket na řádku (15) obsahuje následující IAC-příkazy:

(16) Tak mi tedy, kliente, pošli typ terminálu, který používáš (odpověď na řádek (6)).

(17) Pošli mi rychlosť své linky (odpověď na řádek (8)).

(18) Pošli mi informace o umístění svého X-serveru.

(19) Pošli mi proměnné prostředí podle nové normy.

(20) Pošli mi proměnné prostředí podle staré normy.

Na řádku (21) klient odmítá poslat proměnné prostředí podle staré normy v paketu šestnáctkově znázorněném na řádku (22). Server vrací paket (23) obsahující následující příkazy:

(24) Nebudu komunikovat v poloduplexním režimu (odpověď na řádek (5)).

(25) Pošli mi počet řádků a sloupů svého terminálu. Na řádku (26) je okamžitě zformulovaná odpověď obsahující dva dvoubajtové parametry:

První parametr (první dva bajty z parametrů) obsahuje ve svém prvním bajtu nulu a v druhém bajtu  $50_{16}=80_{10}$ , tj. klient informuje server o tom, že má k dispozici 80 sloupců.

Druhý parametr informuje server o tom, že je k dispozici 24 řádků.

(27) Podporuji řízení toku dat.

(28) Nepodporuji řádkový režim.

(29) Podporuji zaslání stavových informací.

V následujícím paketu server zasílá:

(30) Zašli mi typ terminálu. Klient si okamžitě na řádku (31) zformuluje odpověď, že rychlosť vysílání a příjmu je 9 600 b/s.

(32) Zašli mi umístění svého X-serveru, klient formuluje na řádku (33) odpověď, že umístění je t1.pvt.cz:0.0.

(34) Zašli obsah proměnných prostředí, klient na řádku (35) zasílá obsah pouze proměnné prostředí DISPLAY.

(36) Zašli typ svého terminálu, klient na řádku (37) sděluje, že jeho typ je DTERM.

Na řádku (38) se server pokouší nabídnout pseudořádkový režim dotazem, nechce-li klient náhodou sám provádět ECHO. Klient na řádku (39) pseudořádkový režim odmítne.

Na řádku (40) server nabídne, že bude provádět ECHO, na což klient na řádku (41) odpoví, že ano.

Nyní už skončí martyrium IAC-příkazů. Server se na řádku (42) představí a na řádku (43) vyzve uživatele k zadání hesla. Dále už to znáte...

## Domácí cvičení

- ◆ Zjistěte, na kterých počítačích vaši lokální sítě (např. 10.1.1.0/26) je pravděpodobně spuštěn server protokolu Telnet.
- ◆ Co zjistíme následujícím příkazem (příkaz nmap viz. str. 40):

```
nmap -sS -p T:23 10.1.1.0/26
```

## Kapitola 15

# FTP

Protokol FTP má obdobně jako protokol Telnet také bohatou historii. V normách RFC se objevil v RFC-114 z 10. dubna 1971. Nyní je standardizován RFC-959 s úpravami v RFC-2228 a RFC-2640.



**Poznámka:** V této kapitole jsou uvedeny příklady za využití programu Telnet. Pokud chcete program Telnet využívat ve Windows Vista, pak je nutné jeho použití povolit volbou „Zapnout nebo vypnout funkce systému Windows“.

## Charakteristika

Protokol FTP slouží pro přenos souborů v počítačových sítích na bázi protokolu TCP/IP.

Protokol FTP má prakticky tato použití:

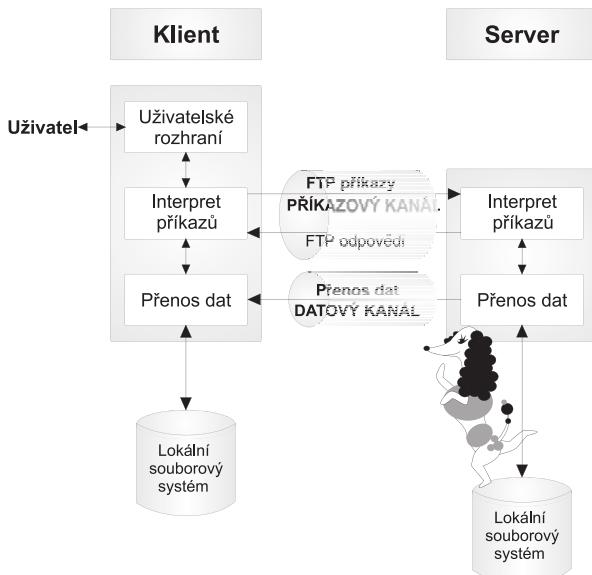
1. Používají jej uživatelé a správci serverů, kteří pracují přímo v operačním systému serveru.
2. Masově FTP používají běžní klienti pro stahování internetovým prohlížečem dat z FTP serverů. V tomto případě se však používá tzv. anonymní FTP server. Tomu je věnován samostatný odstavec.
3. Mnoho uživatelů na střední úrovni používá FTP ve zvláštních grafických klientech pro Windows (např. WS\_FTP) nebo jako součást grafické aplikace typu Commander (např. Norton Commander, Windows Commander).

## Architektura

Architektura protokolu FTP je znázorněna na obr. 15.1. Uživatel pracuje s uživatelským rozhraním. To bývá buď jako příkazový řádek programu ftp, nebo jako grafická utilita či internetový prohlížeč.

Uživatelské rozhraní bývá zpravidla vytvořeno podle zvyklostí operačního systému, ve kterém je implementováno. Uživatelské rozhraní předává požadavky vrstvě interpretu příkazů. Interpret příkazů pak komunikuje se serverem v příkazech definovaných normou FTP. Přehled příkazů je uveden v tabulace 15.1 (v odstavci 15.5). Uživatel má však k dispozici ještě bohatší rejstřík příkazů, protože interpret příkazů zpravidla interpretuje i několik příkazů pro práci s lokálním souborovým systémem klienta (minimálně příkaz lcd – nastavení aktuálního adresáře na straně klienta).

V případě, že se vrstvy interpretu příkazů klienta i serveru na příkazovém kanálu dohodnou na přenosu dat mezi klientem a serverem, požadavek na přenos dat předá vrstvě přenosu dat. Vrstva přenosu dat pracuje s lokálním souborovým systémem. Je schopna soubor na jedné straně číst a na druhé straně přenesená data zapisovat (do souborového systému).



Obrázek 15.1: Architektura FTP

Architektura protokolu FTP je zvláštní. Protokol FTP používá dva kanály typu klient-server:

- ◆ **Příkazový kanál**, kterým klient zaslá na server své požadavky, např. na výpis adresáře či stažení souboru. Tento kanál používá jako prezentační protokol virtuální terminál (NVT – viz kap. 14). Server vždy očekává požadavky příkazového kanálu na portu 21/tcp.
- ◆ **Datový kanál**, kterým se přenášejí požadovaná data (výpis adresáře či obsah souboru). U datového kanálu je právě jistá zvláštnost v tom, že se může role serveru a klienta obrátit. Proto rozděláváme dva režimy komunikace protokolu FTP: **aktivní** a **pasivní**.



**Poznámka:** FTP je mezi aplikačními protokoly výjimečně zejména používáním dvou kanálů: příkazového a datového. To velice komplikuje filtraci protokolu FTP (viz kap. 18).

Datový kanál je vždy otevřán s konkrétními vlastnostmi přenosu dat:

**Typ přenášeného souboru.** Protokol FTP rozeznává až čtyři typy přenášených souborů (implementovány bývají však pouze ASCII a binární):

- ◆ Implicitním typem přenášeného souboru je ASCII. Přenášená data jsou reprezentována v protokolu NVT, tj. odesilatel konvertuje odesílaná data do NVT a příjemce je na své straně konvertuje z NVT do prezentace v operačním systému příjemce. Tento typ přenášených dat také používá příkazový kanál.
- ◆ EBCDIC – alternativa pro případ, že oba konce spojení jsou systémy s kódem EBCDIC.
- ◆ Binární typ, který data přenáší jako souvislý tok bajtů. Běžně se tento typ používá pro přenos binárních dat.
- ◆ Lokální typ souboru dnes nebývá implementován. Je určen pro přenos binárních dat mezi systémy s různou reprezentací dat.

Pouze pro typy přenášených souborů ASCII a případně EBCDIC může být nastaveno **řízení formátu pro zobrazení** (resp. tisk) souboru:

- ◆ Soubor neobsahuje formátovací znaky (*format non-print*).
- ◆ Soubor obsahuje vertikální formátovací znaky pro tisk, jak jsou definovány v protokolu Telnet.
- ◆ První znak každé řádky obsahuje formátovací znak jazyka Fortran.

**Struktura přenášeného souboru** může být:

- ◆ Souborová struktura (*structure file*): soubor obsahující tok bajtů (implicitní hodnota).
- ◆ Struktura tvořená větami. Pouze pro typ přenášeného souboru ASCII nebo EBCDIC může být struktura přenášeného souboru tvořená větami. Přenášený soubor obsahuje řídicí znaky EOR (konec věty) a EOF (konec souboru). Oba řídicí znaky jsou uvozeny únikovou sekvencí FF (významový bajt o hodnotě FF musí být zdvojen). Za únikovou sekvencí FF následuje buď 01 (pro EOR), nebo 02 (pro EOF). Dvojice EOR EOF na konci souboru může být zkrácena na FF 03.
- ◆ Stránková struktura může být podporována v některých operačních systémech. Každá stránka je přenášena s číslem stránky, aby sekvence stránek mohla být zachována, i kdyby během přenosu došlo k záměně stránek.

**Mód přenášených dat:**

- ◆ Mód ve tvaru datového toku (*mode stream*) je implicitní hodnotou. Konec souboru je v případě souborové struktury přenášených dat signalizován uzavřením datového kanálu odesilatelem. (Pro strukturu tvořenou větami je konec souboru signalizován dvojicí znaků FF 03.)
- ◆ Blokový mód přenáší data po blocích. Každý blok obsahuje informační záhlaví.
- ◆ Kompresní mód umožňuje kompresi přenášených dat.

Pokud si vypíšeme status navázaného spojení protokolem FTP stejnojmenným příkazem status, dostaneme následující:

```
ftp> status
Connected to infoserv.ripe.net.
No proxy connection.
Mode: stream; Type: binary; Form: non-print; Structure: file
Verbose: on; Bell: off; Prompting: on; Globbing: on
Store unique: off; Receive unique: off
Case: off; CR stripping: on
Ntrans: off
Nmap: off
Hash mark printing: off; Use of PORT cmds: on
Interpretation of "|" in filenames: off
```

Čtvrtý řádek pak říká:

1. **Mode: stream**, tj. mód dat přenášených datovým kanálem je ve tvaru datového toku.
2. **Type: binary**, tj. typ přenášeného souboru je binární přenos; to znamená, že data se přenášejí jako souvislý tok bitů.
3. **Form: non-print**, tj. soubor neobsahuje formátovací znaky pro řízení výstupu (např. na tiskárnu).
4. **Structure: file**, tj. přenášená data mají souborovou strukturu – data jsou přenášena jako tok bajtů.

## Aktivní režim komunikace protokolu FTP



**Poznámka:** V aktivním režimu si při otevřání datového kanálu klient vymění roli se serverem. To začalo dělat velké potíže, když se začalo s ochranou intranetu pomocí filtrace (viz kap. 18).

Aktivní režim komunikace protokolu FTP je tím „klasickým režimem komunikace protokolu FTP“. Nejčastější otázkou je, jak může uživatel ovlivnit volbu režimu komunikace. Odpověď je „těžko“, protože takovou volbu nám musí umožnit tvůrce klienta ftp. Jen málokterý klient má však takovou volbu. Takže pokud používáme klienta z příkazového řádku, nejspíš budeme používat právě aktivní režim. Na druhou stranu mnohé verze internetových prohlížečů používají pasivní režim. Naopak serverům musíme přiznat, že zpravidla podporují oba režimy komunikace.

Nejprve si v krátkosti objasníme princip aktivní komunikace na obr. 15.2. Klient si přeje navázat spojení se serverem `infoserv.ripe.net`. Klient nejprve požádá správu volných portů na svém lokálním počítači o přidělení volného portu. Je mu přidělen nějaký port větší než 1023 (klientský port). Na tomto portu naváže TCP spojení s portem 21/tcp serveru. Vytvoří se tak příkazový kanál.

Jelikož si chceme podrobně popisovat komunikaci protokolem FTP, na straně klienta zadáme interní příkaz klienta `ftp: debug`. Tento příkaz nic nemění na komunikaci mezi klientem a serverem. Pouze nám klient bude vypisovat detailní informace o komunikaci protokolem FTP. Např. vypíše:

```
--> PORT 194,149,105,131,4,11
```

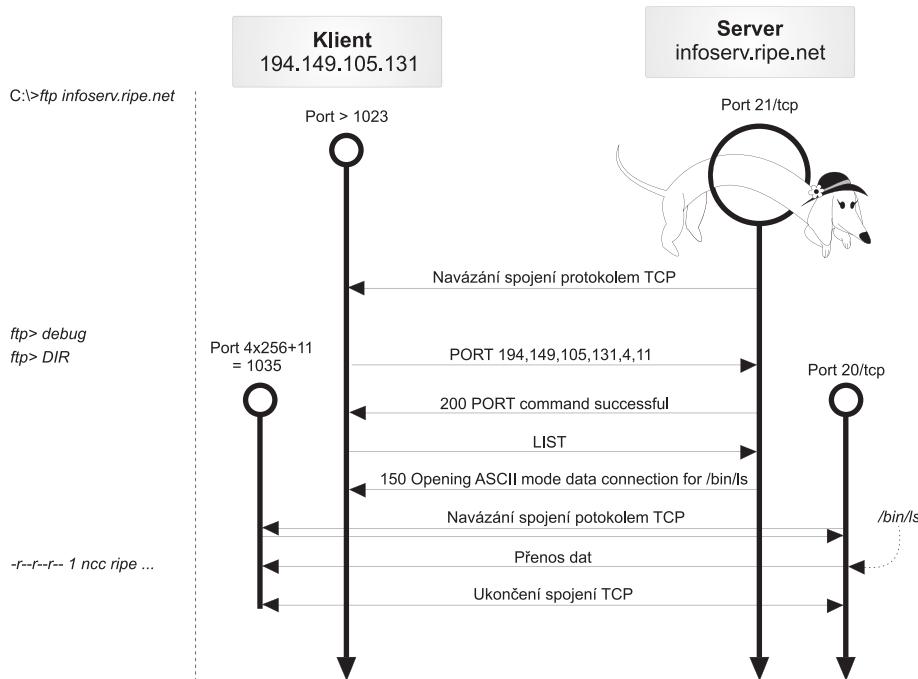
To znamená, že klient odesal na server (šipka  $\rightarrow$ ) příkaz protokolu FTP: „PORT 194,149,105,131,4,11“.

Nyní již můžeme zadat příkaz pro podrobný výpis aktuálního adresáře na serveru: „DIR“. Pro zaslání výpisu adresáře ze serveru na klienta se zřídí datový kanál. Právě aktivní režim je charakteristický tím, že datový kanál je zřízen zvláštním způsobem (obr. 15.2):

1. Klient požádá správu volných portů svého operačního systému o přidělení volného portu. Je mu přidělen např. port 1035 desítkově. (Číslo portu je pak přenášeno ve dvou bajtech: tj. 1035 je 04 0B šestnáctkově. V protokolu FTP se však každý bajt uvádí desítkově, tj. 4 a 11.)
2. Klient odešle příkazem PORT šest desítkových čísel obsahujících jeho IP adresu a port, tj. odešle 194, 149, 105, 131, 4,11.
3. Server naváže protokolem TCP spojení s klientem o IP adrese a portu uvedeném v příkazu PORT. Server tedy navazuje spojení s klientem! – jako by si server a klient v datovém kanálu vzájemně vyměnili svou roli.

Nyní může již klient odeslat příkaz LIST pro výpis adresáře. Všimněte si, že zatímco uživatel v uživatelském rozhraní zadal příkaz „DIR“, tento příkaz byl pro interpret příkazů konvertován do příkazu LIST protokolu FTP. Výpis adresáře na straně unixového serveru znamená spustit program /bin/ls, protože server vypisuje obsah adresáře právě tímto programem. Standardní výstup programu /bin/ls je přesměrován do datového kanálu.

Jelikož data byla přenášena v módu ve tvaru datového toku, je konec výpisu (= konec souboru přenášených dat) způsobem uzavřením datového kanálu.

**Obrázek 15.2:** Aktivní spojení FTP

V praxi si komunikaci protokolem FTP nebudeme kreslit, ale budeme si ji zobrazovat pomocí podrobného výpisu, který se aktivuje příkazem `debug`. Demonstraci FTP provedeme na příkladu 15.1, který se skládá z následujících fází:

1. Klient spustí program `ftp` s parametrem jméno cílového serveru (např. `ftp infoserv.ripe.net`). V příkladu 15.1 je v kroku 1 vidět, že spojení bylo navázáno.
2. Server vyžaduje autentizaci uživatele. Jako uživatel je zadán uživatel `ftp` (tj. anonymní uživatel – řetězec `ftp` je ekvivalentní s řetězcem `anonymous`).
3. V kroku 3 je vyžádáno heslo. Pro anonymního uživatele se zadává uživatelská e-mailová adresa, sloužící správci serveru pro statistiky.
4. V kroku čtvrtém řádku se server představí.
5. V pátém kroku se konečně klientovi zobrazí příkazový řádek programu `ftp`. Jako první příkaz byl zadán příkaz `debug`, který zapne zobrazování podrobných informací o spojení (zapne režim ladění).
6. V šestém kroku uživatel zadal příkaz `DIR`, který je konvertován do protokolu FTP. Nejprve je však třeba vytvořit datový kanál, takže klient alokuje volný klientský port 1035 (tj. 4, 11). O této alokaci je příkazem `PORT` informován server.
7. Příkaz `PORT` proběhl úspěšně, takže je možné odeslat požadavek na výpis adresáře příkazem `LIST`. Server pro výstup programu `/bin/ls` zřizuje datový kanál, do kterého vkládá výstup.
8. Následuje výpis obsahu adresáře (příklad 15.1 obsahuje jen část výpisu).

9. Úspěšné ukončení přenosu je signalizováno příkazovým kanálem.
10. Následuje krátká statistika o přenosu.
11. Nakonec uživatel může zadat další příkaz.

### Příklad

#### Příklad 15.1: Aktivní spojení

```
1. C:\ WINNT\ system32>ftp infoserv.ripe.net
Systém je připojen k infoserv.ripe.net.
220 infoserv.ripe.net FTP server (Version wu-2.6.1(1) ...
2. Uživatel (infoserv.ripe.net:(none)): ftp 331 Guest login ok, send your
complete e-mail address as password.
3. Heslo:
4. 230-Welcome 194.149.105.131,
230-
230-This is the ftp-server of the RIPE Network Coordination Centre (NCC).
...
230-
230 Guest login ok, access restrictions apply.
5. ftp> debug
Ladění Zapnuto
6. ftp> dir
--> PORT 194,149,105,131,4,11
7. 200 PORT command successful.
--> LIST 150 Opening ASCII mode data connection for /bin/ls.
total 922
8. -r-r-r-      1 ncc  ripe     86 Mar 24      1992 .msg.logout
-r-r-r-      1 ncc  ripe     89 Apr 24      1996 .msg.toomany
-rw-r-r-     1 ncc  ripe    1205 Jan 11     2000 .msg.welcome
-rw-rw-r-     1 ncc  ripe    2093 Sep 16     1998 About-ripe
dr-xr-xr-x    2 ncc  ripe     512 Jul 22     1996 bin
...
drwxr-xr-x    3 ncc  ripe    3072 Jun  7     09:15 ww-connectivity
9. 226 Transfer complete.
10. ftp: 1356 bajtů přijato za 0,92sekundy 1,47kB/s
11. ftp>
```

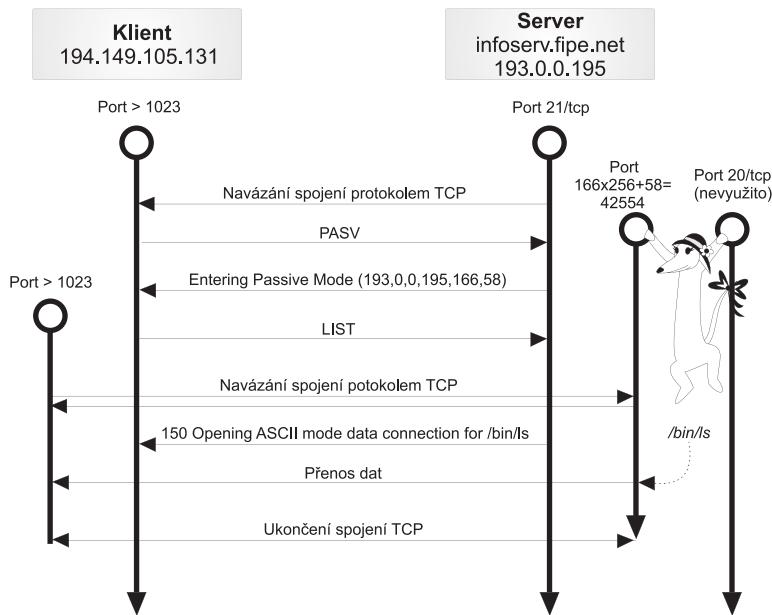
## Pasivní režim komunikace protokolu FTP



**Poznámka:** V pasivním režimu zůstávají role klienta i serveru zachovány.

Mnohdy může vadit, že v aktivním režimu se spojení pro datový kanál navazuje ze strany serveru na klienta. Tento problém odstraňuje pasivní režim FTP. V tomto režimu klient navazuje spojení jak pro příkazový kanál, tak i pro datový kanál. To může být velice užitečné, chceme-li např. síť chránit filtrací na přístupovém směrovači.

Komunikace je jednoduchá: klient příkazem *pasv* požádá server o alokaci portu, server jej alokuje a klientovi odpoví šesticí čísel. První čtyři jsou IP adresa serveru a druhá dvě čísla jsou číslem alokování portu. Klient následně na tuto IP adresu a port naváže spojení pro datový kanál.



Obrázek 15.3: Pasivní spojení FTP

Zatímco v případě aktivního spojení jsme pro zkoumání protokolu FTP nemohli použít program Telnet, protože pro navázání datového kanálu bychom museli program Telnet spustit na serveru, tak naoček pasivní režim komunikace si ukážeme nikoliv za využití programu ftp, ale programu Telnet.

Nejprve si v jednom okně otevřeme příkazový kanál protokolu FTP.

### Příklad

Příklad 15.2: Navázání pasivního spojení protokolem FTP pomocí dvojitého (simultánního) spuštění programu Telnet (opět na server *infoserv.ripe.net*):

1. Telnet> open infoserv.ripe.net 21  
Trying 193.0.0.195...  
Connected to infoserv.ripe.net.  
Escape character is '^]'.  
220 infoserv.ripe.net FTP server (Version wu-2.6.1(1) ... .
2. user ftp  
331 Guest login ok, send your complete e-mail address as password.
3. pass dostalek@pvt.cz
4. 230-Welcome ...  
230-  
230-This is the ftp-server of the RIPE Network Coordination Centre (NCC).  
230-You are user number 11 out of maximum 100.

```

...
230-
230 Guest login ok, access restrictions apply.
5. Pasv
227 Entering Passive Mode (193,0,0,195,166,58)
6. List
150 Opening ASCII mode data connection for /bin/ls.

```

```

$ telnet 193.0.0.195 42554
Trying 193.0.0.195...
Connected to infoserv.ripe.net.
Escape character is '^].
total 922
-r--r--r--    1 ncc  ripe     86 Mar 24  1992 .msg.logout
-r--r--r--    1 ncc  ripe     89 Apr 24  1996 .msg.toomany
-rw-r--r--    1 ncc  ripe   1205 Jan 11  2000 .msg.welcome
-rw-rw-r--    1 ncc  ripe   2093 Sep 16  1998 About-ripe
...
drwxr-xr-x    3 ncc  ripe   3072 Jun  7 09:15 ww-connectivity
Connection closed by foreign host.
$ 

```

```

226 Transfer complete.
7. quit
221-You have transferred 0 bytes in 0 files.
221-Total traffic for this session was 3246 bytes in 1 transfers.
221-Thank you for using the FTP service on infoserv.ripe.net.
221 Goodbye.
8. Connection closed by foreign host.

```



**Poznámka:** Pokud se vám povede tento příklad zopakovat, pak vězte, že jste kandidátem na znalce TCP/IP – klobouk dolů.

#### Průvodce příkladem 15.2:

- Na prvním řádku bylo otevřeno spojení příkazovým kanálem se serverem infoserv.ripe.net. To, že se jedná o FTP server, je dáno portem 21, který je vyhrazen pro příkazový kanál pro tokolu FTP.
- Autentizace uživatele.
- Zadání hesla.
- Server se nám představil – vypsal tzv. *banner*.
- Tento bod je rozhodující – zde klient říká, že komunikace má probíhat v pasivním režimu, tj. přeje si, aby port pro datový kanál alokoval server. Jelikož nemáme klienta ftp, ale pouze program Telnet, musíme příkazy přímo zadávat v protokolu FTP (viz tab.15.1). Server v pasivním režimu nepoužije port 20, ale požádá správu volných portů o nějaký volný port a je mu přidělen port. Následně klientovi vrátí IP adresu a číslo portu (227 Entering Passive Mode (193,0,0,195,166,58)). Číslo portu se skládá ze dvou bajtů. První bajt má desítkově hodnotu 166 a druhý 58. Nyní si číslo portu převedeme do desítkové soustavy jako jedno číslo: **166 x 256 + 58 = 42554**.

6. Nyní můžeme vydat příkaz pro výpis adresáře, kterým je v protokolu FTP příkaz *List* (viz tab. 15.1). Server okamžitě očekává navázání spojení pro datový kanál, takže v dalším okně musíme rychle spustit další instanci programu Telnet a navázat se serverem spojení, ale tentokrát na portu 42554 a okamžitě vidíme výstup datového kanálu.
7. Příkazem *quit* (viz tab. 15.1) vydáme příkaz k ukončení spojení příkazovém kanálu.
8. Datový kanál je uzavřen.

## Příkazy FTP

Příkazy protokolu FTP jsou přenášeny textově (v ASCII). Příkaz je vždy tvořen klíčovým slovem (např. *USER*). Klíčové slovo mohou následovat parametry oddělené mezerou. Příkaz je ukončen znaky CR (návrat vozíku) a LF (nový řádek).

**Tabulka 15.1:** Příkazy FTP (přenášené sítí)

Příkaz FTP (přenášený sítí)	Obvyklý příkaz příkazového řádku <b>ftp</b>	Význam
USER jméno-uživatele	user	Jméno uživatele.
PASS heslo		Heslo.
ACCT účet	account	Některé servery mohou vyžadovat kromě jména uživatele i název účtu, který může být využit pro přístup k některým datům.
CWD	cd	Nastavení aktuálního adresáře na serveru.
CDUP	cdup	Nastavení nadřízeného adresáře jako aktuálního adresáře na serveru.
SMNT cesta	–	Zavěšení souborového systému ( <i>mount</i> ).
QUIT	quit	Ukončení spojení.
REIN	–	Reinicializace – uživatel je odhlášen, ale spojení příkazovým kanálem zůstává. Poté může následovat příkaz <i>USER</i> .
PORT port	–	Viz text.
PASV		Přechod do pasivního režimu.
TYPE kód	ascii, binary	Specifikace typu přenášeného souboru (první parametr) a řízení formáty pro zobrazení (druhý parametr). Příklad: <i>TYPE A N</i> . Požaduje nastavení typu ASCII bez formátovacích znaků.
STRU struktura	–	Specifikuje strukturu přenášeného souboru. Např. <i>STRU F</i> požaduje nastavení struktury přenášeného souboru na souborovou strukturu ( <i>file</i> ).
MODE mód	–	Specifikuje mód přenášených dat. Např. <i>MODE S</i> požaduje nastavení módu na tvar datového toku ( <i>stream</i> ).
RETR cesta	get	Stažení souboru ze serveru.
STOR cesta	put	Přenesení souboru na server.

Příkaz FTP (přenášený sítí)	Obvyklý příkaz příkazového řádku ftp	Význam
STOU cesta	rununique + get	Obdoba STOR, avšak přenesený soubor bude mít jednoznačné jméno v adresáři (ftp nepřepíše existující soubor, ale pozmění název).
APPE cesta	append	Přenesete soubor na server. Jestliže na serveru existuje stejnojmenný soubor, pak není soubor přepsán, ale o přenesená data rozšířen.
REST značka		Restart přenosu dat.
RNFR cesta	rename	Přejmenování souboru se provede dvěma příkazy FTP.
RNTO cesta		Příkazem RNFR se zjistí, zdali původní soubor vůbec na serveru existuje, a příkazem RNTO se zadá nové jméno souboru.
ABOR		Abnormální ukončení předchozího příkazu (pokud neskončil).
DELE cesta	delete	Zrušení souboru.
RMD cesta	rmdir	Zrušení adresáře.
MKD cesta	mkdir	Vytvoření adresáře.
PWD	pwd	Zjištění aktuálního adresáře.
LIST [cesta]	ls	Výpis obsahu adresáře.
NLIST [cesta]	dir	Podrobný výpis obsahu adresáře.
SITE řetězec		Příkaz pro jiné služby serveru. Jaké služby server poskytuje, se zjistí příkazem: SITE HELP. Např. SITE IDLE zjistí nastavení limitů nečinného spojení.
SYST	system	Dotaz na informace o systému serveru.
STAT [cesta]	status	Zobrazení aktuálního stavu.
HELP [příkaz]	remotehelp	Zjištění podporovaných příkazů serveru.
NOOP		Prázdný příkaz.

Je však třeba rozlišovat mezi příkazy zadávanými uživatelem pomocí uživatelského rozhraní a příkazy protokolu FTP. Např. pro výpis adresáře uživatel programu ftp zadá příkaz DIR, avšak ten je konvertován na příkaz LIST protokolu FTP.

Na druhou stranu program ftp má mnohé interní příkazy, které často nesouvisejí se síťovou komunikací, ale slouží pro větší komfort uživatele. Takovými příkazy jsou např.:

- ◆ Příkaz *hash*, kterým může uživatel nastavit software klienta tak, že po přenesení 1 či 2 kB dat datovým kanálem se na obrazovce objeví znak dvojitý kříž a uživatel tak má iluzi, že se opravdu něco přenáší – že program „nezmrzl“.
- ◆ Příkaz *lcd*, kterým uživatel může změnit aktuální adresář na lokálním stroji. Někdy bývá implementováno více příkazů týkajících se lokálního systému (začínajících proto na l). Obecně lze provádět všechny (externí) příkazy lokálního systému (lokálního interpretu příkazů) tak, že na příkazový řádek programu ftp napíšeme před příkaz vykřičník.

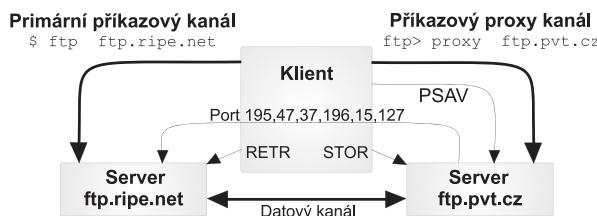
- ◆ Velice důležitým příkazem je *literal*. Pomocí něj lze odeslat libovolný příkaz protokolu FTP, tj. vše, co je argumentem příkazu *literal*, se přímo vloží do protokolu TCP. Např.:
   
literal PWD
- ◆ Zvláštní skupinou jsou příkazy začínající znakem m (jako *multiple*). Tyto příkazy umožňují používat znaky hromadného výběru, jako je znak hvězdička, v názvech souborů. Je tak možné jedním příkazem přenést více souborů. Nutno však podotknout, že tyto příkazy pouze za uživatele vygenerují několik příkazů protokolu FTP, které se pak postupně provádějí. Jako první příkaz použijí výpis vzdáleného adresáře, na jehož základě generují jednotlivé příkazy. Jedná se tedy pouze o vlastnost softwaru klienta (byť komfortní), nikoliv o další sadu příkazů protokolu FTP.
- ◆ Problémem přenosu mezi různými operačními systémy (např. Microsoft a Unix) je skutečnost, že pro jména souborů se u Microsoftu používají zpravidla velká písmena, kdežto na Unixu malá písmena. Je tedy otázkou, zda se ve jménech souborů nemají konvertovat písmena (velká na malá, resp. malá na velká). Tato konverze se zpravidla ovládá příkazem *case*.

Kromě uvedených příkazů protokolu FTP se mohou ještě používat experimentální příkazy začínající znakem X, např. XCWD je experimentální obdobou příkazu CWD. Pro bližší informace doporučuje RFC-1123.

## Proxy

Na úvod musím upozornit, že zde popisované proxy nesouvisí s proxy, jak se s ním setkáváme v oblasti firewallů. Myšlenka spočívá v tom, že klient může zprostředkovat spojení mezi dvěma FTP servery. Na 15.4 je znázorněna situace, kdy klient zprostředkuje přenos souboru ze serveru *ftp.ripe.net* na server *ftp.pvt.cz*.

Klient nejprve zřídí příkazový kanál se serverem *ftp.ripe.net*. Pomocí příkazu proxy zřídí další kanál (tzv. proxy – kanál) se serverem *ftp.pvt.cz*. Nyní odešle na server *ftp.pvt.cz* příkaz PROXY (příkaz protokolu FTP). Server *ftp.pvt.cz* alokuje port, na kterém bude čekat spojení pro datový kanál (IP adresu a číslo vyalokovaného portu předá příkazovým kanálem klientu). Vtip je v tom, že server si může myslet, že čeká na spojení od klienta. Avšak klient obratem předá IP adresu a alokovaný port serveru *ftp.ripe.net*. Nyní již může server *ftp.ripe.net* navázat spojení pro datový kanál se serverem *ftp.pvt.cz* a do takto vytvořeného datového kanálu může vložit kopii příslušného souboru. V tomto případě musí klient vydat jednomu serveru příkaz RETR (čti soubor) a druhému příkaz STOR (zapiš soubor).



Obrázek 15.4: Proxy FTP

A konečně výpis komunikace v ladícím režimu:

```
ftp ftp.ripe.net
Connected to ftp.ripe.net.
220 ftp.ripe.net FTP server (Version wu-2.6.1(1) Tue Jul 18 14:18:18 CEST 2000)
ready.
Name (ftp.ripe.net:root): ftp
331 Guest login ok, send your complete e-mail address as password.
Password:
230 Guest login ok, access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> debug
Debugging on (debug=1).
ftp> proxy open ftp.pvt.cz
Connected to ftp.pvt.cz
220 ftp.pvt.cz FTP server (Digital UNIX Version 5.60) ready.
Name: ftp
--> USER ftp
331 Guest login ok, send ident as password.
Password:
--> PASS XXXX
230 Guest login ok, access restrictions apply.
--> SYST
215 UNIX Type: L8 Version: Digital UNIX V4.0 (Rev. 878)
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> proxy get soubor
--> TYPE I
ftp.pvt.cz:200 Type set to I.
--> PASV
ftp.pvt.cz:227 Entering Passive Mode (195,47,37,196,15,127)
--> TYPE I
ftp.ripe.net:200 Type set to I.
--> PORT 195,47,37,196,15,127
ftp.ripe.net:200 PORT command successful.
--> RETR soubor
ftp.ripe.net:150 Opening BINARY mode data connection for soubor (2093 bytes).
--> STOR soubor
ftp.pvt.cz:150 Opening BINARY mode data connection for soubor (0.0.0.0,0).
ftp.pvt.cz:226 Transfer complete.
ftp.ripe.net:226 Transfer complete.
local: soubor remote: soubor
ftp>
```

Proxy FTP není implementováno u klientů Microsoft.

## Návratové kódy

Na jednotlivém klientem zadávané příkazy protokolu FTP server odpovídá hláškou s trojciferným návratovým kódem následovaným volitelným textovým objasněním návratového kódu. Trojciferný návratový kód má tvar:

xyz

kde:

x může nabývat hodnot:

- 1 Pozitivní odpověď při zahájení nějaké akce. Před tím, než bude moci klient odeslat další příkaz, lze očekávat další hlášku (o ukončení akce).
- 2 Pozitivní odpověď na provedení příkazu. Klient může zadat další příkaz.
- 3 Pozitivní odpověď, na kterou musí klient udělat konkrétní akci. Např. po zadání jména uživatele je vyžadováno heslo.
- 4 Negativní odpověď při chybě. Po odstranění chyby je možné příkaz zadat znovu.
- 5 Permanentní negativní odpověď. Např. o nepodporovaném příkazu.

y může nabývat hodnot:

- 0 Syntaktická chyba
- 1 Informativní hláška
- 2 Chyba spojení
- 3 Chyba autentizace
- 4 Nespecifikovaná chyba
- 5 Chyba signalizována souborovým systémem

z blíže určuje chybu.

Příklady chybových kódů:

- 125 Data connection already open; transfer starting.
- 230 User logged in, proceed.
- 331 User name okay, need password.
- 452 Insufficient storage space in system.
- 502 Command not implemented.

## Abnormální ukončení příkazu

Abnormální ukončení právě prováděného příkazu (zpravidla datového přenosu) je prakticky jediným využitím příkazů protokolu Telnet (resp. protokolu NVT).

V tabulce 15.1 (str. 353) můžeme najít pro abnormální ukončení prováděného příkazu příkaz ABOR. Mohlo by se stát, že by server zpracovával jednotlivé příkazy klienta postupně za sebou, takže server by příkaz ABOR zpracoval, až předchozí příkaz skončí. Ale my chceme, aby předchozí příkaz skončil dříve, tj. aby příkaz ABOR byl zpracován okamžitě, jakmile je obdržen.

Software klienta je většinou nastaven tak, že je citlivý na stisk nějaké klávesové zkratky pro abnormální ukončení. Nechť je touto klávesovou zkratkou např. ^C (grafičtí klienti mohou mít tlačítko „ukončit“). Po stisku ^C klient zastaví odesílání dat v datovém kanálu (pokud se data odesírají od klienta na server) a odešle dvojici příkazů:

1. Příkaz protokolu NVT pro abnormální ukončení procesu: <IAC> <IP><IAC> <DM>. Jsou to příkazy dva, protože za příkazem IP musí následovat ještě DM (datová značka). Server čte svou vstupní vyrovnávací paměť až po datovou značku. Vše, co je po datovou značku, ve zpracování přeskočí. TCP segment nesoucí datovou značku označí jako urgentní a ukazovátko urgentních dat (v záhlaví TCP segmentu) ukazuje právě na značku DM. Značka DM se zpracovává tak, že server jde po vyrovnávací paměti zpět od datové značky DM, až najde odpovídající příkazy IAC. Tentokráté najde <IAC> <IP>.
2. Vlastní příkaz ABOR protokolu FTP.

Celkově tedy klient odešle 10 bajtů: <IAC><IP><IAC><DM>ABOR <CR> <LF>.

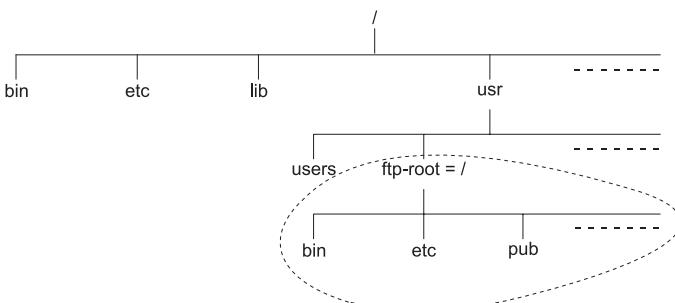
Otázkou je, budou-li jednotlivé implementace klienta FTP nastavovat příznak urgentních dat a odesílat příkazy IAC. To už je věcí tvůrců softwaru klienta.

## Anonymní FTP

Myšlenka anonymního serveru se nepoužívá pouze pro FTP servery, ale zejména pro webové servery (HTTP servery). Anonymní server umožňuje přístup libovolnému klientu, tj. přístup bez autentizace klienta.

Jelikož protokol FTP ve své podstatě s anonymními servery příliš nepočítal, i anonymní uživatelé zadávají při svém přístupu na FTP server jméno uživatele a heslo. Jako jméno uživatele se zpravidla používá buď řetězec *anonymous* nebo řetězec *ftp*. Oba řetězce mají stejný význam. Jako heslo je často vyžadována uživatelská e-mailová adresa. Ta provozovateli anonymního FTP serveru slouží pro vytváření statistik přístupu na server.

Některé anonymní servery umožňují přístup pouze klientům, jejichž počítač má správný záznam v reverzním DNS, tj. těm, k jejichž IP adresě se v DNS nalezne jméno jejich počítače. Toto omezení má však velice sporný význam.



**Obrázek 15.5:** Anonymní FTP

Anonymní FTP server slouží uživatelům pro stahování souborů z něj, tj. na anonymním FTP serveru mohou firmy nabízet své informace. Na anonymní FTP servery se velice pohodlně přistupuje pomocí internetových prohlížečů.

Jsou-li anonymní FTP servery připojeny k Internetu, jsou vystaveny velkému nebezpečí útoků. Implementace FTP serverů proto velice často pro zvýšení bezpečnosti provádějí trik znázorněný na obr. 15.5. Při spuštění FTP serveru jako procesu je na jeho počátku použito systémové volání pro změnu kořenového adresáře na adresář pouze s daty, která chce FTP server zpřístupnit. Operační systém serveru se bude tedy pro tento proces uměle tvářit tak, že kořenovým adresářem je pouze kořenový adresář s daty FTP serveru. Uživatelé přihlášení k tomuto serveru tak nebudou mít možnost dostat se do jiných adresářů operačního systému serveru.

V tomto případě je nutné ještě upozornit na skutečnost, že server nebude mít ani přístup k adresářům, kde je např. program *ls*, používaný pro výpis adresáře, proto pod falešným kořenovým adresářem ještě nesmíme zapomenout navést např. adresář *bin* s programem *ls*. Bývá nutné zavést též adresář *etc* s falešným souborem *passwd*. Adresář *bin* však nemusí umožňovat uživatelům čtení – stačí možnost spustit program *ls*. Program *ls* sám pro sebe nesmí využívat ani sdílené knihovny operačního systému, protože k nim nebude mít přístup (musí mít staticky sestavené knihovny). Proto je zajímavé si zobrazit velikost programu *ls* z adresáře, ve kterém se běžně používá v operačním systému, a z adresáře, ze kterého jej používá anonymní FTP server.

Anonymní FTP server by rozhodně neměl být spuštěn pod superuživatelem, ale na počátku programu serveru je kromě změny kořenového adresáře použito volání pro změnu uživatele a skupinu uživatelů. Jako uživatel (resp. skupina uživatelů) se používá uživatel s minimálními přístupovými právy v operačním systému serveru, umožňující ale korektní provoz FTP serveru.

Běžně se anonymní FTP servery používají pouze pro čtení souborů, proto správce anonymního FTP -serveru vždy pečlivě u všech adresářů ruší právo zápisu do adresáře. V praxi se někdy anonymní FTP server používá i pro výměnu souborů mezi uživateli, tj. je potřeba, aby anonymní uživatelé měli možnost zápisu. Většinou se to povolí pouze do jednoho adresáře. Tomuto adresáři se pak naopak nastaví možnost zápisu, ale znemožní se čtení adresáře (to nesouvisí s právem čtení souboru). Takže jeden uživatel tam může uložit data, ale nikdo si nepřečepte, co v adresáři je. Jiným kanálem (např. telefonem) se pak domluví s druhým uživatelem, aby si soubor stáhl.

## Domácí cvičení

- ◆ Pomocí inspirace z domácího cvičení v kap. 14 zjistěte, na jakém počítači běží FTP server (tj. procvičte si program *nmap*).
- ◆ Programem Wireshark odchytěte komunikaci protokolem FTP a zjistěte, jestli se jednalo o pasivní nebo aktivní FTP.
- ◆ Pokuste se simulovat příklad 15.2.



## Kapitola 16

# HTTP

Protokol HTTP (*Hypertext Transfer Protocol*) je výrazně mladší protokol než protokoly Telnet a FTP. Jeho počátky jsou někde kolem roku 1990, kdy s myšlenkou přišli vědci z CERNu (*European Organization for Nuclear Research*). Způsobili tím zvrat ve vývoji Internetu, ale dále se tím příliš nechtěli odvádět o vlastního jaderného výzkumu, tak vše, co následovalo kolem webů, předali konsorciu W3C (viz [www.w3.org](http://www.w3.org)).

Předchůdcem protokolu HTTP byl dnes již zapomenutý protokol Gopher. Pak následovala historicky velice významná verze 0.9 protokolu HTTP. Ta se díky své jednoduchosti okamžitě dočkala velkého množství implementací, a založila tak slávu protokolu HTTP. Dnes je aktuální verze 1.1 z roku 1999 specifikovaná v RFC-2616.



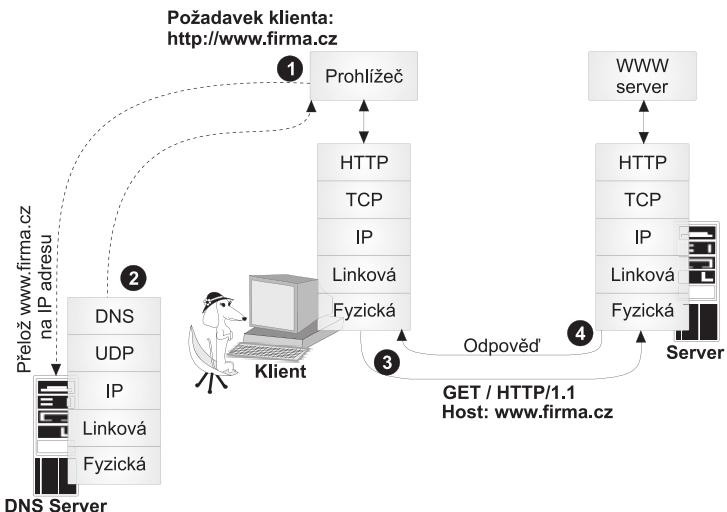
**Poznámka:** V této kapitole jsou uvedeny příklady za využití programu telnet. Pokud chcete program telnet využívat ve Windows Vista, pak je nutné jeho použití povolit volbou „Zapnout nebo vypnout funkce systému Windows“.

## Klient-server

Základní architekturou komunikace v protokolu HTTP je komunikace klient-server. V případě, že se navazuje přímé spojení protokolem TCP mezi klientem a serverem (obr. 16.1), uživatel zapíše do okna prohlížeče identifikátor objektu (URI), jež chce prohlížet. Klient nejprve z identifikátoru objektu vyřízne jméno serveru, jež přeloží za pomoci DNS na IP adresu (1 a 2) a poté klient naváže s takto získanou IP adresou serveru spojení protokolem TCP.

Do takto vytvořeného kanálu vloží prohlížeč HTTP dotaz (3), na který v témže spojení server odpoví HTTP odpověď (4). Prohlížeč následně zobrazí odpověď uživateli. Důležité je, že prohlížeč zobrazuje uživateli webové stránky. Každá webová stránka se většinou skládá z řady objektů. Každý objekt je nutné z webového serveru stáhnout jedním HTTP dotazem. V protokolu HTTP starších verzí se pro každý dotaz vždy navazovalo nové TCP spojení. Takže prvním dotazem se stáhl základní text stránky, ve kterém byla řada dalších odkazů, jež bylo nutné stáhnout pro zobrazení stránky. V dalším kroku se pokud možno najednou navázalo s webovým serverem pro stažení každého objektu TCP spojení. Tato strategie vede k vytvoření špičky v zátěži přenosové cesty.

Protokol HTTP verze 1.1 implicitně předpokládá, že TCP spojení bude mezi klientem a serverem navázáno jedno pro celou sadu dotazů („pro celou webovou-stránku“). Je možné jej uzavřít po jednom dotazu i po více dotazech. Klient může odeslat v jednom spojení více dotazů, aniž by vždy čekal na vyřízení předchozího dotazu (*Pipelining*).



**Obrázek 16.1:** Základní architektura protokolu HTTP

Protokol HTTP verze 1.1 implicitně předpokládá, že do vytvořeného spojení se vkládá více dotazů a odpovědí na ně. Pakliže je požadováno spojení explicitně ukončit, je třeba do záhlaví vložit hlavičku:

Connection: Close

Komunikace v protokolu HTTP se zásadně skládá z dotazu a odpovědi. Relace mezi klientem a serverem je tvořena vždy pouze dotazem a odpovědí na tento dotaz. Starší verze protokolu HTTP dokonce navazovaly spojení protokolem TCP pouze na jednu relaci dotaz-odpověď. Novější verze umožňují využít navázané spojení na více relací dotaz-odpověď. Avšak i v případě, že jedním TCP spojením prochází více relací, tyto relace spolu nikterak nesouvisí.

```
Záštupce - cmd
GET / HTTP/1.1
Host: www.playboy.com

HTTP/1.1 200 OK
Server: Netscape-Enterprise/3.6 SP3
Date: Wed, 28 Dec 2000 16:54:44 GMT
Expires: Sat, 31 Dec 2005 00:00:00 GMT; path '/';
Content-type: text/html
Set-cookie: RMID=c32c25c03a48e450; expires=Fri, 31-Dec-2010 23:59:59 GMT; path('/');
Content-length: 1223
Connection: close

<html>
  <head>
    <title>playboy.com</title>
    <meta http-equiv="pics-label" content="pi
    ce-1.1 "http://www.rsac.org/ratingv01.html" 1 gen true comment "RSACI North Am
    erica Server" for "http://www.playboy.com" on "1996.05.04T06:51:00Z" r (n 4 o 3
    v 0)"/>
    <meta http-equiv="pics-label" content="pics-1.1 "http://WWW.census.gov
    /prod/sais/0001/1.htm" for "http://www.playboy.com" (ss "http://WWW.census.gov
    /prod/sais/0001/1.htm")>
    <meta name="Description" content="playboy.com - your destination on the web.">
    <meta name="Keywords" content="Playboy, Playboy.com, Playmates, Hef, Bunny, Playbo
    y Enterprises Inc. entertainment for men, Playboy Mansion, Playmate of the Year,
    Playboy catalog, Playboy TU">
```

**Obrázek 16.2:** Dotaz (GET...) následovaný odpovědí (HTTP/1.1 200...) protokolu HTTP, který byl proveden programem Telnet (zobrazení stránky [www.playboy.com](http://www.playboy.com) pro odborníky)

Skutečnost, že protokol HTTP neumožnuje delší dialog než jeden dotaz a bezprostřední odpověď na něj, je jistým omezujícím limitem protokolu HTTP. Prakticky může nastat následující situace: uživatel chce pomocí protokolu HTTP nakupovat ve virtuálním obchodním domě na Internetu. Vybere si zboží, které si uloží do virtuálního nákupního košíku, a ten si nese během svého nákupu. Jenže zákazník bude další relací vybírat další zboží a jak uchovat o zákazníkovi informaci, že má již nějaké zboží v košíku (kde ten košík udržovat)? Tímto problémem se budeme zabývat v kapitole Cookies.

Na obr. 16.2 je programem Telnet navázáno spojení se serverem [www.playboy.com](http://www.playboy.com) na portu 80. Postupoval jsem následujícím způsobem: Ve Windows jsem zadal:

```
C: C:> telnet  
C: telent> set localecho  
C: telent> open www.playboy.com 80
```

Nyní se navázalo spojení protokolem HTTP na port 80 serveru [www.playboy.com](http://www.playboy.com). Z klávesnice jsem pak pokračoval – zadal dotaz protokolu HTTP:

```
GET / HTTP/1.1      ... metodou GET požaduji obsah kořenového adresáře  
Host: www.playboy.com  ... zadání virtuálního webového-serveru  
                      ... prázdný řádek oddělující záhlaví od dat
```

Okamžitě přišla odpověď:

```
HTTP/1.1 200 OK ...
```

Kupodivu některým uživatelům takováto komunikace s webovým serverem nepostačuje, požadují příslušné informace graficky zobrazit (obr. 16.3). Avšak to je podstatně komplikovanější problém. Komunikace na obr. 16.2 zobrazila text formátovaný v jazyce HTML, který sám obsahuje pouze odkazy na obrázky, takže každý obrázek z webové stránky 16.3 musí být dodatečně z webového serveru stažen samostatnou komunikací dotaz-odpověď. Tyto komunikace jsou naprostě samostatné (byť by běžely jedním TCP spojením), tj. obrázek může být použit i na jiných stránkách či může být stažen samostatně (pravým tlačítkem myši v prohlížeči).



Obrázek 16.3: Zobrazení příslušné webové stránky pro běžné uživatele

Jiným omezením protokolu HTTP je použití architektury klient-server. Ta neumožňuje odesílat asynchronní události ze serveru klientovi. Protokolem HTTP se tak těžko vytvářejí aplikace typu „burza“, kdy by pro uživatele bylo praktické v případě zajímavé směny cenného papíru, aby server klienta na tuto skutečnost upozornil. Server tak může tuto skutečnost sdělit klientovi nejdříve v okamžiku, kdy klient odešle nějaký dotaz na server.

Uživatel si většinou nastaví prohlížeč (klienta) tak, že získaný požadavek nejenom uživateli zobrazí, ale pokud je to možné, též jej uloží na disk do paměti Cache. Při opakování dotazu pak může být uživateli informace zobrazena přímo z lokálního disku. Jsou nejrůznější strategie, kdy zobrazovat tyto informace a kdy má klient přenést informace ze serveru. Dokonce existuje i možnost, že se klient dotáže serveru, zda se tím nezměnily informace, když ze serveru přenese pouze záhlaví odpovědi. Některé odpovědi serveru mohou být označeny jako neuložitelné do paměti Cache; pak je tam klient uložit nesmí.

Internetový prohlížeč nebývá klientem pouze protokolu HTTP. Zpravidla se jedná o integrovaného klienta, který „umí“ i protokol FTP a případně umí vyvolat klienty dalších protokolů:

- ◆ protokolů elektronické pošty,
- ◆ protokolu LDAP (tj. adresář),
- ◆ protokolu NNTP (tj. diskusní skupiny),
- ◆ Telnet apod.

Protokol HTTP zavádí proxy, bránu a tunel. Jedná se o mezilehlé systémy, které mohou ležet na cestě mezi klientem a serverem. Na cestě od klienta k serveru může ležet libovolné množství těchto mezilehých systémů. Z hlediska protokolu TCP se navazuje TCP spojení vždy mezi dvěma sousedními uzly, tj. mezi klientem a první proxy, mezi první proxy a druhou proxy atd. Při popisu proxy, brány a tunelu se omezíme nejprve na situaci, že mezi klientem a serverem je pouze jeden mezilehlý systém. Následně pak zjistíme, že umístěním více mezilehlých systémů se vůbec nic nezmění.

Nejčastěji se proto tyto systémy používají tam, kde není možné přímo navázat TCP spojení mezi klientem a serverem, tj. např. na firewallu oddělujícím intranet od Internetu.

## Domácí cvičení

Programem telnet vypište homepage svého oblíbeného webového serveru.

## Proxy

Proxy je systém skládající se ze dvou částí:

1. Ze serverové části, která přijímá požadavky klienta, jakoby je přijímal cílový server. Požadavky však v zálepě předá klientské části.
2. Z klientské části, která převeze požadavky od serverové části, naváže TCP spojení s cílovým serverem a předá jménem klienta požadavky cílovému serveru k vyřízení.

Takto se proxy jeví uživateli. Avšak uprostřed proxy mezi serverovou a klientskou částí je ještě skryta vlastní logika proxy. Proxy totiž rozumí aplikačnímu protokolu (v našem případě protokolu HTTP) a s požadavkem přijatým od klienta může provést několik operací:

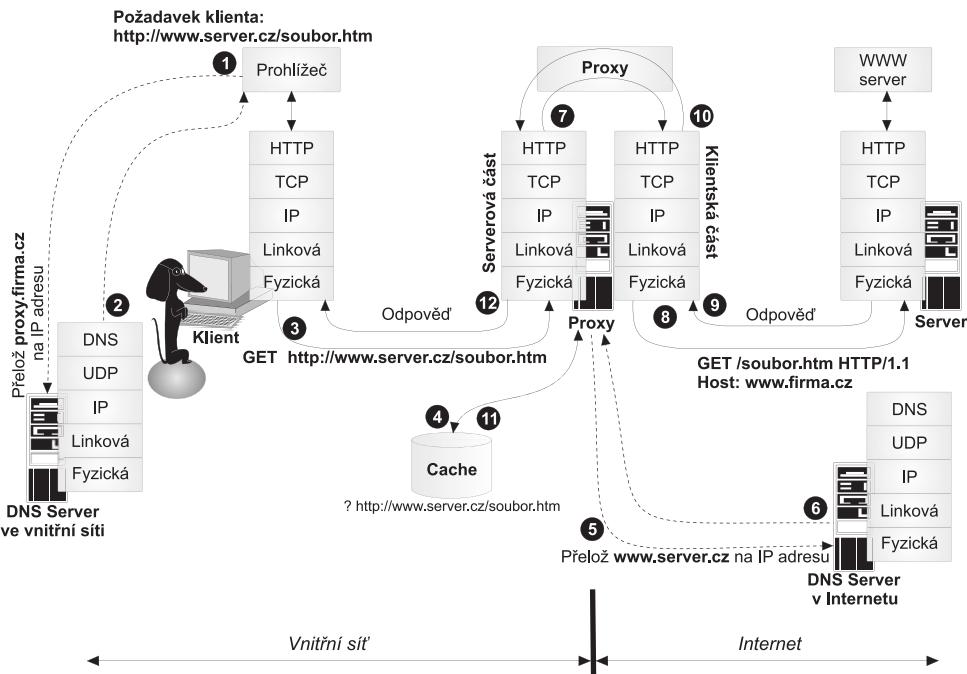
1. Může přepsat požadavek (resp. odpověď), tj. změnit data aplikačního protokolu.
2. Odpovědi může ukládat do paměti Cache (např. na disk). Pokud proxy obdrží v budoucnu stejný požadavek (např. i od jiného klienta), může tento požadavek vrátit rychleji přímo z paměti, aniž by navazovala spojení s cílovým serverem. Vypadá to sice efektivně, ale zásadní otázkou takto uchovávaných odpovědí je jejich aktuálnost. Ukládání dat do paměti se tak stává komplikovanou záležitostí, které věnujeme samostatný odstavec. V dnešní době již máme k dispozici velice sofistikované algoritmy pro práci s pamětí Cache. Díky používání dynamických stránek, stavových transakcí a zabezpečovaných spojení a v neposlední řadě zvýšení propustnosti komunikačních linek začíná význam paměti Cache na proxy ustupovat.
3. Může zjišťovat, zdali je klient oprávněn takový požadavek provést.

Proxy může prověřovat oprávněnost klienta provést nějaký požadavek z několika hledisek:

- ◆ Může zjišťovat, zda klient nepřistupuje na nějaký nežádoucí server. Např. zaměstnavatel může nechat na proxy nastavit seznam serverů, na které nebudou moci jeho zaměstnanci přistupovat. V praxi je běžné, že zaměstnavatel zakáže přístup např. na [www.playboy.com](http://www.playboy.com) (zaměstnanci si pak ale na Internetu najdou 10 jiných serverů s pro ně ještě zajímavější tematikou, o kterých zaměstnavatel neví).
- ◆ Může zjišťovat, zda je uživatel oprávněn proxy vůbec používat. V takovém případě vyžaduje autentizaci uživatele. Nejčastější typy autentizace uživatele jsou:
  1. Pomocí IP adresy stanice, na které uživatel pracuje. Tato autentizace není příliš bezpečná, proto slouží spíše k administrativnímu omezení některých klientů nepoužívat proxy (např. nepřistupovat skrze proxy do Internetu).
  2. Pomocí jména uživatele a stálého hesla.
  3. Pomocí jména uživatele a jednorázového hesla.
- ◆ Proxy běžící na firewallu mohou od operačního systému požadovat, aby prováděl kontrolu, z jakého síťového rozhraní přichází uživatelův požadavek na proxy, tj. zda uživatel přistupuje ze síťového rozhraní vnitřní sítě, či ze síťového rozhraní do Internetu. To pochopitelně standardní implementace TCP/IP v operačním systému neumí. To bývá právě jedním z důvodů, proč firewalls při své instalaci zásadně zasahují do operačního systému. Podle těchto informací pak proxy zjišťuje, zda je požadavek z intranetu či jde např. o útok z Internetu. Útokům z Internetu může být také bráněno tím, že serverová část proxy naslouchá pouze na IP adresu vnitřní sítě proxy.
- ◆ V případě, že proxy ví, odkud požadavek přišel (jestli z vnitřní sítě, nebo z Internetu), může použít jiný autentizační mechanismus pro požadavky z vnitřní sítě a pro požadavky z Internetu. Např. z vnitřní sítě vyřizuje všechny požadavky, kdežto z Internetu vyžaduje autentizaci jednorázovým heslem.
- ◆ I proxy může data před tím, než je předá (i uloží do paměti Cache), zkонтrolovat, zda neobsahuje viry. Většinou to proxy nedělá sama, ale volá jiný proces, který tuto speciální kontrolu provede. Tento proces může běžet i na specializovaném počítači. Pak se takový počítač často označuje jako Viruswall (odvozeno od Firewall).

Na obr. 16.4 je schematicky znázorněna činnost proxy. Na počátku uživatel zapíše do okna svého prohlížeče identifikátor objektu (URI), jež chce prohlížet. Např. klient do okna prohlížeče vloží požadavek:

<http://www.server.cz/soubor.htm>



Obrázek 16.4: Proxy (na obrázek se již v dotazu klienta nevešla verze protokolu a hlavička Host)

Klient však bude tento požadavek vyřizovat skrze proxy, tj. v konfiguraci svého prohlížeče (obr. 16.5) zapsal jméno proxy, skrze kterou se bude požadavek vyřizovat. Prohlížeč z identifikace objektu zjistí pouze aplikační protokol. Jak je totiž vidět z obr. 16.4, pro každý protokol může klient použít jinou proxy.

V prvním kroku klient přeloží jméno proxy na IP adresu (na obr. 16.4 akce 1 a 2). Jméno cílového serveru totiž nemusí být v intranetu ani převozitelné. Nyní klient naváže TCP komunikaci se serverovou částí proxy na port uvedeném v konfiguraci klienta (viz obr. 16.5). Do takto vytvořeného TCP spojení vloží klient svůj HTTP požadavek (3):

```
GET http://www.server.cz/soubor.htm HTTP/1.1
Host: www.server.cz
```

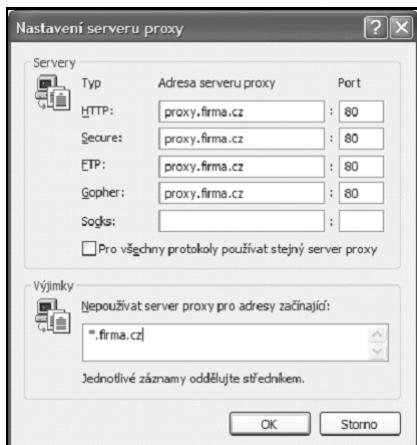
Proxy ve své paměti Cache prověří, nemá-li odpověď na tento požadavek náhodou k dispozici (4).

V případě, že požadavek v paměti Cache nebyl nalezen, předá požadavek klientské části na vyřízení. Klientská část musí z URI-požadavku vyříznout jméno serveru (`www.server.cz`) a přeložit jej v DNS (5 a 6). Jelikož proxy má již přístup do Internetu, je schopna tento požadavek nechat v Internetu přeložit.

Klientská část proxy nejprve přepíše požadavek na:

```
GET /soubor.htm HTTP/1.1
Host: www.server.cz
```

Následně klientská část naváže spojení s cílovým serverem protokolem TCP a předá mu požadavek (8) jménem klienta. Server vrátí odpověď (9), kterou obdrží proxy (10). Pokud je odpověď přípustné uložit do paměti Cache, uloží ji tam (11). Proxy předá odpověď klientovi (12), který ji zobrazí uživatelům a případně si ji též uloží do své paměti Cache.



**Obrázek 16.5:** Konfigurace proxy, brány a tunelu v internetovém prohlížeči

Na obr. 16.5 je zobrazena konfigurace internetového prohlížeče. Je tam konfigurována proxy pro protokol HTTP. Dále je tam konfigurace brány pro protokoly FTP a Gopher. Řádek začínající „Secure“ je konfigurací tunelu pro SSL.

## Brána

Brána (*gateway*) je mezilehlý uzel, který pracuje obdobně jako proxy, avšak s tím rozdílem, že mění (konvertuje) aplikační protokol. Nejběžnějším případem je brána, jejíž serverová část od klientů přijímá požadavek v protokolu HTTP a mění jej na komunikaci v protokolu FTP (viz obr. 16.6).

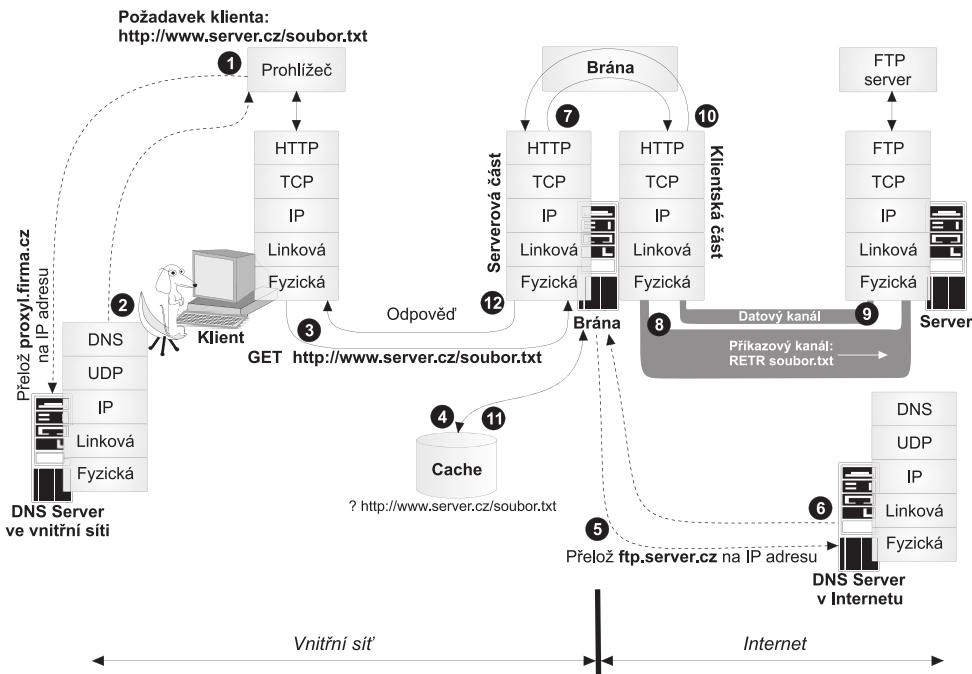
Zajímavostí brány je, že pokud si uživatel přeje zobrazit obsah adresáře, brána protokolem FTP zjistí pouze obsah adresáře jako vypsaný programem `ls` (na Unixu) či `DIR` v systémech Microsoft, tj. obdrží datovým kanálem informace o jednotlivých souborech v adresáři. Tento obsah adresáře avšak musí vrátit klientovi jako webovou stránku, tj. ve formátu jazyka HTML. Takže musí mít v zásobě obrázky ikonek pro soubor, ikonky pro adresář apod., které pak zobrazí klientovi. To má za následek, že zobrazíte-li si obsah adresáře skrze proxy od jednoho dodavatele a posléze skrze proxy jiného dodavatele, grafická úprava může být jiná.



**Poznámka:** Brána nemůže předpokládat, že by se klient spokojil s `dr-xr-xr-x` či něčím podobným – běžný klient chce ikonku adresáře.

Není to překvapující, protože pokud prohlížeč navazuje spojení přímo s cílovým serverem (bez proxy), obsah adresáře získaný přes protokol FTP musí obdobně přepracovat do grafického tvaru,

takže výpis téhož adresáře zobrazený různými prohlížeči (resp. jinými verzemi téhož prohlížeče) může mít jinou grafickou úpravu.



Obrázek 16.6: Brána (na obrázek se již v dotazu klienta nevešla verze protokolu a hlavička Host)

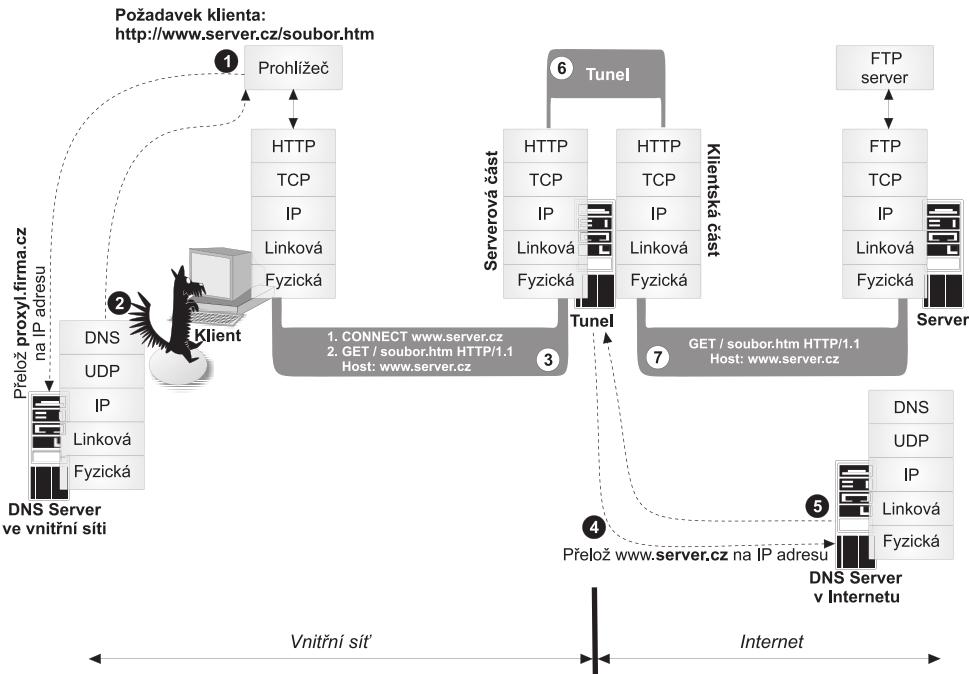
## Tunel

Tunel je dalším typem mezilehlého uzlu. Klient naváže s tunelem TCP spojení (obr. 16.7). Do takto vytvořeného spojení klient vloží příkaz (metodu) CONNECT s parametrem, kterým je jméno a port cílového serveru. Tunel přeloží jméno cílového serveru na IP adresu a sám jménem klienta naváže s cílovým serverem TCP spojení na portu uvedeném v příkazu CONNECT. Nyní má tunel navázána dvě obousměrná spojení:

- ◆ Spojení klient – tunel.
- ◆ Spojení tunel – cílový server.

Každé z těchto TCP spojení si představíme jako dvojici rour – jedna roura je pro přenos dat tam a druhá pro přenos dat zpět (duplexní spoj). Funkce tunelu je jednoduchá – tunel totiž neudělá nic jiného než to, že roury „mechanicky“ navaří na sebe“, tj. aniž by věděl, co přenáší. Vše, co přijde od klienta, předá mechanicky do roury na cílový server. Obdobně vše, co přijde ze serveru, předá klientovi.

V takovémto spojení pak klient může klidně začít i protokolem SSL navazovat šifrované spojení. Základní vlastností tunelu tak je, že tunel „nerozumí“ přenášeným datům. Jelikož tunel se nejčastěji používá právě pro spojení zabezpečená protokolem SSL/TLS, tak ani nepřekvapí, že se IP adresa a port tunelu v prohlížečích konfiguruje v rádku *Secure* (obr. 16.5). Po ukončení spojení se celý tunel rozpadá.

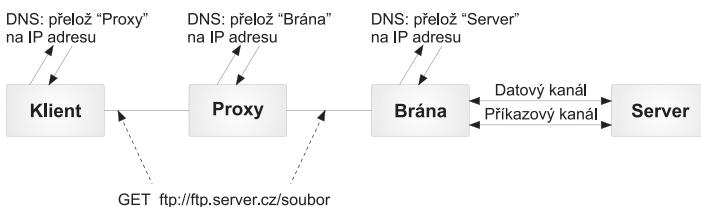


Obrázek 16.7: Tunel

Je vcelku pochopitelné, že tunel nevidí do přenášených dat, takže nemůže kontrolovat, co klient ze serveru stahuje za data – jestli se např. nejedná o Java applety, ActiveX komponenty, viry, červy apod.

## Více mezilehlých uzlů

Na cestě od klienta až po cílový server jsou možné nejrůznější kombinace mezilehlých systémů.



Obrázek 16.8: Řetězec mezilehlých systémů

Je vcelku běžné používat „proxy on proxy“ v podnicích, jejichž intranet obsahuje nějaké vnitřní jádro oddělené firewallem od zbylého intranetu. Vnitřní intranet je oddělen proxy od zbytku intranetu, který je od Internetu oddělen další proxy. Z vnitřního intranetu je pak do Internetu spojení přes dvojí proxy.

V případě zdvojené proxy se jméno cílového serveru na IP adresu musí přeložit až na poslední proxy před cílovým serverem. Obdobně je praktické bránu umístit až na konec řetězce (obr. 16.8).

## URI

Identifikátorem objektů ve webovém světě jsou tzv. URI (*Uniform Resource Identifiers*). URI mohou být: *Uniform Resource Names* (URN), *Uniform Resource Locators* (URL) a *Uniform Resource Characteristics* (URC). V praxi se setkáváme pouze s URL.

Jednotlivé aplikační protokoly mají své schéma URI. URI specifikuje RFC-1738 jako

<schéma>:<na\_schématu\_závislá\_část>

kde <schéma> může mj. být:

```
http Protokol HTTP
ftp Protokol FTP
mailto Odeslání elektronické pošty (protokol SMTP)
nntp Protokol NNTP (diskusní skupiny)
Telnet Odkaz na relaci protokolem Telnet
file Soubor (např. na disku)
imap Protokol IMAP
ldap Protokol LDAP
pop Protokol POP3
```

Ve schématu (nikoliv v celém URI) nezávisí na tom, použijeme-li velké nebo malé písmeno, tj. ftp je totéž jako FTP či Ftp.

V celém URI se mohou vyskytovat pouze znaky kódu US-ASCII. V případě, že je nutné zadat jiný znak, položí se před něj znak % a za ním následuje šestnáctkové vyjádření znaku. V šestnáctkovém vyjádření znaku můžeme psát jak velká písmena, tak i malá písmena pro šestnáctkové číslice A až F. Znaky „,“, „/“, „?“, „:“, „@“, „=“ a „&“ jsou vyhrazeny pro speciální použití, tj. mají-li se použít v nějakém řetězci, musí být konvertovány do šestnáctkové soustavy a uvozeny procentem.

### Schéma http

`http://<uživatel:heslo><server>:<port>/<cesta>?<dotaz>#<fragment>`

Uživatel a heslo slouží pro autentizaci klienta. Heslo se obecně nedoporučuje zadávat v URI. Není-li heslo zadáno, protokol HTTP si jej vyžádá dodatečným dialogem. V dialogu se při zadávání heslo nezobrazuje. V tomto případě nesmí být jméno uživatele následováno dvojtečkou.

V případě, že se za jménem uživatele uvede dvojtečka, za kterou okamžitě následuje název serveru, zadali jsme jméno uživatele a prázdný řetězec jako heslo.

Je-li zadáno jméno uživatele (resp. jméno uživatele a heslo), pak jméno serveru musí začínat znakem @. Port protokolu TCP je uveden za dvojtečkou. Chceme-li použít implicitní port 80, neuvedeme ani dvojtečku, ani číslo portu.

Cesta obsahuje identifikaci souboru skládající se z adresářů oddělených opět lomítkem a jména souboru.

Za otazníkem následuje řetězec dotazů. Dvojitý kříž specifikuje odkaz na fragment webové stránky. Webová stránka může být větší, než je okno na obrazovce. Šoupátkem se můžeme po webové stránce pohybovat. Webovou-stránku je možné rozdělit na fragmenty, jejichž počátek je identifikován jako <fragment>. Pokud chceme, aby od levého horního okraje obrazovky byl zobrazován právě <fragment>, aniž bychom museli pohybovat šoupátkem, použijeme uvedený odkaz.

### Příklady:

1. *http://server.firma.cz*  
Je dotazem anonymního klienta na obsah kořenového adresář serveru *server.firma.cz*.
2. *http://novak@server.firma.cz*  
Je dotazem uživatele novak na obsah kořenového adresáře serveru *server.firma.cz*. Pokud je pro tohoto uživatele požadováno serverem heslo, bude vyžádáno následným dialogem se serverem.
3. *http://server.firma.cz/cgi-bin/formular.exe?pole1=%20&pole2=hod2*  
Anonymní uživatel spustí program formular.exe z adresáře cgi-bin. Tomuto programu bude předán řetězec pole1=%20&pole2=hod2 (pomocí proměnné prostředí QUERY\_STRING). Jedná se o formulář obsahující dvě pole: pole1 obsahuje mezera (mezera je šestnáctkově 20) a pole2 obsahuje hodnotu hod2.
4. *http://server.firma.cz/adr1/adr2/adr3/dokument.html#odstavec5*  
Anonymní uživatel požaduje zobrazit obsah dokumentu document.html z adresáře /adr1/adr2/adr3. Přitom si při zobrazování tohoto dokumentu přeje, aby byl zobrazen začátek fragmentu odstavec5.

### Schéma ftp

*ftp://<uživatel:heslo><server>:<port>/<cd<sub>1</sub>>/<cd<sub>2</sub>>/.../<cd<sub>n</sub>>/<soubor>;type=y*

Význam uživatele, hesla a jména serveru je obdobný jako v případě schématu pro protokol HTTP. Rozdíl je jen v tom, že pokud se nezadá jméno uživatele, automaticky se dosadí uživatel „anonymous“. Pokud je známa e-mailová adresa uživatele, dosadí se jako heslo. Implicitním portem je port 21. Typ přenášeného souboru může být buď „i“, nebo „a“, avšak tento parametr se prakticky nepoužívá.

Základní otázkou je, jak je interpretován řetězec <cd<sub>1</sub>>/<cd<sub>2</sub>>/.../<cd<sub>n</sub>>/<soubor>. Ten se interpretuje tak, že se postupně protokolem FTP provádějí příkazy „CWD cd<sub>1</sub>“, „CWD cd<sub>2</sub>“ až „CWD cd<sub>n</sub>“. Poté se provede příkaz RETR-soubor.

Tato interpretace neumožňuje snadno specifikovat absolutní cestu od kořenového adresáře. Např.:

*ftp://ftp.firma.cz/etc/passwd*

způsobí „CWD etc“ a „RETR passwd“. Pokud bychom chtěli provést „CWD /etc“, musíme zadat:

*ftp://ftp.firma.cz/%2Fetc/passwd*

protože 2F je šestnáctkově lomítko.

### Schéma mailto

`mailto:<rfc822-addr-spec>`

Toto schéma má jediný parametr, kterým je e-mailová adresa, na niž má být odeslána pošta.

#### Příklad

`mailto:dostalek@pvt.cz`

### Schéma nntp:

`nntp://<server>:<port>/<skupina>/<číslo_příspěvku>`

### Schéma Telnet

`Telnet://<uživatel>:<heslo>@<server>:<port>/`

V tomto schématu může být vynecháno: poslední lomítka, jméno uživatele a heslo.

### Schéma file

`file://<server>/<cesta>`

Toto schéma používáme zejména pro odkaz na data na lokálním disku. Pak místo jména počítače použijeme prázdný řetězec (nebo jméno localhost), takže při specifikaci souboru nejčastěji uvádíme tři lomítka za sebou. Cesta je pak úplná specifikace souboru.

#### Příklady

`file:///C|/WINNT/system32/eula.txt`

Zajímavá je situace spíše v operačních systémech, kde adresářová struktura je jiná než u Microsoftu či v Unixu. Vezměme jako příklad operační systém OpenVMS. Zde např. soubor DISK\$USER:[MY.NOTES]NOTE123456.TXT vyjádříme:

`file:///disk$user/my/notes/note12345.txt`

### Schéma pop

`pop://<user>;auth=<auth>@<host>:<port>`

Blíže viz RFC-2384.

Schéma imap specifikuje RFC-2192, schéma ldap specifikuje RFC-2255 atd.

## Relativní URI

Nezačíná-li URI názvem schématu, jedná se o relativní URI. Relativní URI je vždy vztaženo k nějaké bázi – nějakému absolutnímu URI. Bází může být URI zobrazovaného dokumentu nebo případně i předešlého dokumentu, tj. URI dokumentu, ze kterého byl odkaz proveden. V případě, že žádné takové URI neexistuje, může být použito implicitní URI celé aplikace.

Relativní URI může obsahovat v cestě dokonce znak „.“ a znak „...“ pro odkaz na aktuální a nadřízený adresář. Při zpracovávání relativního URI se na URI hledí jako na několik samostatných částí: jméno serveru, cesta se jménem souboru, dotaz a fragment, kterými se nahrazuje báze zprava. Vychází se z toho, že fragment začíná znakem #, otázka znakem ?, cesta lomítkem a server dvěma lomítky.

Např. ve zobrazené stránce <http://www.firma.cz/cesta/soubor.htm> mohou být tedy např. relativní hypertextové odkazy:

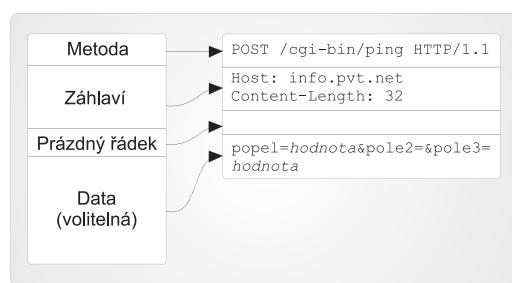
- ◆ #odstavec1, který lze převést na absolutní URI:  
<http://www.firma.cz/cesta/soubor.htm#odstavec1>.
- ◆ soubor2.htm, který lze převést na absolutní URI:  
<http://www.firma.cz/cesta/soubor2.htm>.
- ◆ ../soubor3.htm, který lze převést na absolutní URI:  
<http://www.firma.cz/soubor3.htm>.

## HTTP dotaz

Struktura HTTP dotazu (i odpovědi) připomíná strukturu e-mailu. Na první pohled vidíme rozdíl pouze v prvním řádku. První řádek dotazu obsahuje tzv. metodu a první řádek odpovědi obsahuje tzv. stavový řádek.

HTTP dotaz se skládá (viz obr. 16.9) z následujících částí:

- ◆ Metody. Protokol HTTP verze 1.1 podporuje metody: GET, POST, HEAD, OPTIONS, TRACE, CONNECT, PUT a DELETE. Metody PUT a DELETE nebyvají implementovány.
- ◆ Záhlaví, které je tvořeno jednotlivými hlavičkami. Každá hlavička začíná klíčovým slovem (např. Host:). Klíčové slovo je ukončeno dvojtečkou následovanou mezerou. Za mezerou pak mohou následovat parametry hlavičky. Celá hlavička je vždy ukončena koncem řádku CRLF. Pouze jediná hlavička je povinná, a to hlavička Host.
- ◆ Prázdného řádku (CRLF CRLF, kde první CRLF ukončuje poslední řádek hlavičky).
- ◆ Přenášených dat (volitelně).



**Obrázek 16.9:** Struktura a příklad HTTP dotazu

Metoda má v protokolu HTTP verze 1.1 vždy tvar:

<Název metody> <URI> HTTP/1.1

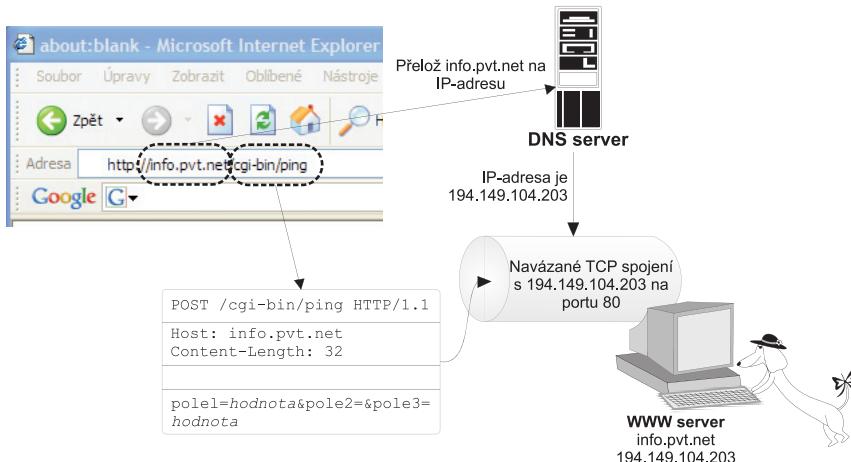
kde verze protokolu HTTP je 1.1, což je tč. aktuální verze. Uvedení verze je povinné. Pokud bychom uvedli jen název metody a URI, jednalo by se o dotaz v protokolu HTTP verze 0.9. Nesmíme se pak divit, že v odpovědi nedostaneme např. stavový řádek, ten byl totiž zaveden až ve verzi 1.0 (verze 1.0 neměla zase mj. hlavičku Host:).

### Příklad

Příklad metody GET:

GET / HTTP/1.1

Pro mne bylo překvapujícím zjištěním, že webové servery absolutní URI neakceptují. Přijmou ho jen proxy a brány (tunel už URI vůbec nechce, ten chce jen metodu CONNECT). A pokud uživatel za jménem serveru lomítko nenapíše, musí si ho tam prohlížeč domyslet.



**Obrázek 16.10:** Rozbor URI prohlížečem (na cestě mezi klientem a serverem není proxy, brána ani tunel)

Pokud poprvé kontaktuji nějaký server, nemá smysl uvažovat u relativního URI o bázi ze zobrazeného dokumentu či z dokumentu, ze kterého byl dokument zobrazen. Absolutní URI se tak rozloží na model (např. protokol HTTP, který implicitně navazuje spojení na port 80 serveru), jméno počítače, které se překládá v DNS (aby spojení vůbec mohlo být navázáno), a zbytek, tj. cestu případně dotaz a fragment.

Server naopak relativní URI doplní na absolutní. Aplikací je HTTP server (bylo navázáno spojení na portu 80), takže schéma je http. Jméno serveru se doplní z hlavičky host a zbytek server obdržel jako parametr metody.

### Metoda GET

Metoda GET slouží k dotazování klienta na konkrétní informace uložené na serveru. Dotaz je formulován jako součást URI v řetězci <dotaz> (za otazníkem). Metoda GET může teoreticky obsahovat i data, avšak tato možnost bývá zřídka využívána (data se přenesou jako součást dotazu).

Příklady si ukážeme pomocí programu Telnet ve Windows Vista. Jako server poslouží info.pvt.net (můžete zkoušet třeba [www.vase-firma.cz](http://www.vase-firma.cz)) běžící na portu 80:

```
C:\ > telnet  
C: telnet> set localecho  
C: telnet> connect info.pvt.net 80
```

Řádky, které jsem napsal z klávesnice, uvozuji řetězcem „C:“ a řádky, které mi odpověděl server, jsou uvozeny „S:“. Prvním příkladem je požadavek na výpis obsahu kořenového adresáře serveru (adresáře `/`). Webové servery bývají nakonfigurovány tak, že většinou při požadavku na obsah adresáře nevrací výpis souborů. Vrátí zpravidla soubor, který se jmenuje `index.html` a který obsahuje nějakou zprávu. V kořenovém adresáři bude tento soubor tedy obsahovat titulní stránku webového serveru.

### Příklad

Příklad 16.1:

```
C: GET / HTTP/1.1  
C: Host: info.pvt.net  
C:                                         Prázdný řádek ukončující záhlaví  
S: HTTP/1.0 200 OK  
S: Date: Wed, 20 Dec 2000 17:20:18 GMT  
S: Last-Modified: Mon, 10 Apr 2000 07:14:43 GMT  
S: Content-Length: 2855  
S: Server: Apache/1.2b10  
S: ETag: "e9ed-b27-38f17f63"  
S: Accept-Ranges: bytes  
S: Content-Type: text/html  
S:                                         Prázdný řádek ukončující záhlaví odpovědi  
S: <HTML>  
S: <HEAD>  
S:   <TITLE>info.pvt.net </TITLE>  
S: </HEAD>  
S: <TABLE>  
S:   <TR><TD> Vitame Vas na serveru ....
```

V protokolu HTTP verze 1.1 je povinná hlavička `Host`, proto i nás dotaz musel tuto hlavičku obsahovat. Hlavička `Host` obsahuje jméno serveru (viz část webový server).

Odpověď obsahuje několik zajímavých hlaviček:

- ◆ **Date:** Datum a čas, kdy byla odpověď vytvořena.
- ◆ **Last-Modified:** Čas poslední modifikace zdroje (webové stránky).
- ◆ **Content-Length:** Délka dat odpovědi v bajtech.
- ◆ **Accept-Ranges:** Informace o tom, že server akceptuje dotazy, které jsou zaslány v několika částech (velikost části se udává v bajtech).
- ◆ **ETag:** Jednoznačně identifikuje verzi odpovědi. Pokud je v paměti Cache odpověď stejné verze jako na serveru, jsou obě odpovědi shodné.

Metodou GET můžeme pokládat též podmíněné dotazy za využití hlaviček **If-Modified-Since**, **If-Unmodified-Since**, **If-Match**, **If-None-Match** a **If-Range**. Tyto dotazy umožňují přenášet data odpovědi jen v případě, že podmínka v dotazu je pravdivá. Hlavičky If-Match a If-None-Match vyhod-

nocují identifikaci verze odpovědi (ETag), kdežto hlavičky If-Modified-Since a If-Unmodified-Since vyhodnocují datum poslední modifikace. Hlavička If-Range vyhodnocuje buď identifikaci, nebo verzi odpovědi, požaduje však od serveru jen tu část odpovědi, která byla změněna.

### Příklad

Příklad 16.2:

V příkladu 16.1 jsme získali odpověď o identifikaci „e9ed-b27-38f17f63“ (uvozovky je třeba uvádět!). Můžeme tedy server požádat, aby nám vrátil odpověď v případě, že se odpověď nezměnila (tj. server poskytuje stále stejnou odpověď):

```
C: GET / HTTP/1.1
C: Host: info.pvt.net
C: If-Match: "e9ed-b27-38f17f63"
C:
S: HTTP/1.1 200 OK
S: Date: Sat, 23 Dec 2000 18:09:38 GMT
S: Server: Apache/1.2b10
S: Last-Modified: Mon, 10 Apr 2000 07:14:43 GMT
S: ETag: "e9ed-b27-38f17f63"
S: Content-Length: 2855
S: Accept-Ranges: bytes
S: Content-Type: text/html
S:
S: <HTML>
S: <HEAD>
S:   <TITLE>info.pvt.net </TITLE>
S: </HEAD>
S: <TABLE>
S:   <TR><TD> Vitame Vas na serveru ....
```

Jelikož na serveru se příslušná webová stránka nezměnila, server nám ji zaslal. Takový dotaz asi není příliš praktický. Praktičtější by byl dotaz vracející data odpovědi naopak jen v případě, že se webová stránka změnila (v opačném případě by byla webová stránka zobrazena z paměti Cache). K tomuto účelu slouží hlavička **If-None-Match**.

Hlavička **If-Modified-Since** žádá server o odpověď jen v případě, že se příslušná stránka změnila. Netestuje se však na identifikaci stránky, ale na čas poslední modifikace.

### Příklad

Příklad 16.3:

```
C: GET / HTTP/1.1
C: Host: info.pvt.net
C: If-Modified-Since: Mon, 10 Apr 2000 07:14:43 GMT
C:
S: HTTP/1.1 304 Not Modified
S: Date: Sat, 23 Dec 2000 19:07:48 GMT
S: Server: Apache/1.2b10
S: ETag: "e9ed-b27-38f17f63"
```

A jelikož se příslušná stránka nezměnila, vrátí nám server pouze hlavičku s informací o tom, že se stránka nezměnila. V opačném případě použijeme hlavičku **If-Unmodified-Since** (obdoba příkazu 16.2).

## Metoda POST

Metodou POST odesíláme na server data (např. pole HTML-formuláře). Zasílání metody POST programem Telnet má háček. Celý problém spočívá ve faktu, že v HTTP požadavku následují za prázdným rádkem data, která odesílá klient na server. Přitom server musí poznat, kolik dat bude klient odesílat. Z klávesnice nedokážu porizovat data tak rychle, aby na mne server nečekal. Pokud mu v záhlaví nesdělím množství odesílaných dat, bude si po prázdném rádku ukončujícím záhlaví myslit, že žádná data již odesílat nechci. Něco jiného je, když dotaz odešle prohlížeč, který vloží záhlaví i data do jednoho TCP segmentu, takže server má celý dotaz k dispozici. Z klávesnice však dostává bajt po bajtu takovou rychlosť, jak mačkám klávesy.

### Příklad

Příklad 16.4:

```
C: POST /cgi-bin/ping HTTP/1.1
C: Host: info.pvt.net
C: Content-Length: 32
C:                                         - prázdný řádek oddělující záhlaví od dat
C: ping=info.pvt.net&pign=3&pocet=3

S: HTTP/1.1 100 Continue

S: HTTP/1.1 200 OK
S: Date: Thu, 21 Dec 2000 07:21:11 GMT
S: Server: Apache/1.2b10
S: Transfer-Encoding: chunked
S: Content-Type: TEXT/HTML
S:
S: 188
S: PING info.pvt.net (194.149.104.203): 56
S: data bytes<BR>64 bytes from 194.149.104.
S: 203: icmp_seq=0 ttl=64 time=0 ms<BR>64 bytes from 194.149.104.203: icmp_seq=1 tt
S: l=64 time=0 ms<BR>64 bytes from 194.149.104.203: icmp_seq=2 ttl=64 time=0 ms<BR>
S: <BR><BR>—info.pvt.net PING Statistics—<BR>3 packets transmitted, 3 packets
S: received, 0% packet loss<BR>round-trip (ms) min/avg/max = 0/0/0 ms<BR>
S: 0
S:
```

Spočítal jsem proto, kolik bajtů bude mít datová část mého dotazu, a do záhlaví jsem přidal hlavičku Content-Length. Nedočkavý server mne zprávou „HTTP/1.1 100 Continue“ („no tak dělej!“) informuje o tom, že můj dotaz neodmítá a že jej zpracuje (to se prohlížeči většinou nestává, protože ten stačí svými daty rychle zásobovat server). Na odpovědi je zajímavá hlavička „Transfer-Encoding: chunked“. Jedná se opět o méně běžnou hlavičku, kterou jsem si pravděpodobně vykoledoval u serveru pomalostí zadávání svého dotazu z klávesnice. Tato hlavička sděluje, že server odesílá odpověď po částech (chunked). Tato forma odpovědi se skládá z jednotlivých částí, jež začínají rádkem obsa-

hujícím od první pozice délku části v šestnáctkové soustavě ( $18816=39210$ ). Vždy musí být uvedena poslední část o velikosti nula. Naše odpověď se tak skládá ze dvou částí: první je dlouhá 392 bajtů a druhá prázdná – signalizující konec odpovědi. Za poslední částí následuje prázdný řádek, za kterým mohou následovat hlavičky zápatí HTTP odpovědi (v praxi málo využívány).

Ještě bych se chtěl zmínit o údajích, které jsem zaslal na server „ping=info.pvt.net&pingn=&pocet=3“. Jedná se o tři pole webového formuláře. První pole se jmenuje „ping“ a uživatel do něj vyplnil řetězec „info.pvt.net“. Druhé pole se jmenuje „pingn“ a uživatel jej nevyplnil. Nakonec třetí pole se jmenuje „pocet“ a je v něm trojka. Jednotlivá pole jsou od sebe oddělena speciálním znakem & určeným k oddělování polí formuláře. Důležité je, že metodou POST byla tato data zaslána v datové části HTTP dotazu. Pokud budou tato data na serveru zpracovávána CGI-skriptem /cgi-bin/ping, budou předána na standardní vstup tohoto skriptu. V případě, že bychom obsah stejného formuláře chtěli odeslat pomocí metody GET, pak by metoda dotazu měla tvar:

```
C: GET /cgi-bin/ping?ping=info.pvt.net&pingn=&pocet=3 HTTP/1.1
C: Host: info.pvt.net
C:
```

V tomto případě bychom hlavičku Content-Length nepotřebovali, protože data jsou předána jako dotaz. Pokud by byla tato data na straně serveru opět zpracovávána CGI-skriptem, muselo by se jednat o jiný skript, protože data bych metodou GET CGI-skript neobdržel na standardní vstup, ale v proměnné prostředí QUERY\_STRING. (*Je pochopitelně možné napsat i skript, který umí zpracovat jak standardní vstup, tak i proměnnou QUERY\_STRING.*)

## Metoda HEAD

Metoda HEAD požaduje pouze hlavičku bez dat.

### Příklad

Příklad 16.5:

```
C: HEAD / HTTP/1.1
C: Host: info.pvt.net
C:
S: HTTP/1.0 200 OK
S: Date: Wed, 20 Dec 2000 17:20:18 GMT
S: Last-Modified: Mon, 10 Apr 2000 07:14:43 GMT
S: Content-Length: 2855
S: Server: Apache/1.2b10
S: ETag: "e9ed-b27-38f17f63"
S: Accept-Ranges: bytes
S: Content-Type: text/html
```

## Metoda TRACE

Je jakousi obdobou příkazu tracert. Tentokrát však nejistíme, kolik je mezi naším a cílovým počítačem směrovačů, ale kolik je tam proxy nebo bran.

V případě, že komunikujeme s proxy nebo bránou, nesmíme zapomenout, že do metody (TRACE) musíme zapsat celé absolutní URI (<http://info.pvt.net>).

### Příklad

#### Příklad 16.6:

```
C: TRACE http://info.pvt.net HTTP/1.1
C: Host: info.pvt.net
C:
S: HTTP/1.0 200 OK
S: Date: Wed, 20 Dec 2000 17:24:04 GMT
S: Server: Apache/1.2b10
S: Content-Type: message/http
S:
S: TRACE / HTTP/1.0
S: Host: info.pvt.net
S: Cache-Control: Max-age=259200
S: Via: 1.1 proxy.pvt.cz:8080 (Squid/1.1.22)
```

V datové části odpovědi je pak statistika, která nás zajímá, uvedena v hlavičce **Via**. V této hlavičce jsou uvedeny jednotlivé proxy nebo brány oddělené čárkou. V našem případě jsme komunikovali pouze přes jednu proxy.

O každé proxy či bráně mohou být uvedeny čtyři údaje:

1. Protokol (v případě protokolu HTTP může chybět).
2. Verze protokolu (1.1).
3. Mezilehlý systém (proxy nebo brána – v našem příkladu proxy.pvt.cz:8080).
4. Komentář v kulatých závorkách obsahující zpravidla informace o softwaru mezilehlého systému (Squid/1.1.22).

Pokud by na cestě bylo více mezilehlých systémů, pak by za uvedeným systémem následovala čárka a další systém atd.

Zajímavá hlavička je Cache-Control, která v našem případě udává, že informace mají být udržovány v paměti Cache maximálně 259 200 sekund.

### Metoda OPTIONS

Metoda OPTIONS se používá ke zjištění komunikačních vlastností serveru či požadovaného URI. V případě, že se dotazujeme na vlastnosti celého serveru, použijeme místo URI hvězdičku:

### Příklad

#### Příklad 16.7:

```
C: OPTIONS * HTTP/1.1
C: Host: info.pvt.net
C:
S: HTTP/1.1 200 OK
S: Date: Sat, 23 Dec 2000 19:11:22 GMT
S: Server: Apache/1.2b10
S: Content-Length: 0
S: Allow: GET, HEAD, OPTIONS, TRACE
```

Server nám sděluje, že podporuje metody GET, HEAD, OPTIONS a TRACE. Zarazilo mne, že server nevrátil metodu POST. Tak to si neodpustím serveru zadat URI, na které jsem metodu POST již použil, (příklad 16.4) a počkám, co server odpoví:

### Příklad

Příklad 16.8:

```
C: OPTIONS /cgi-bin/ping HTTP/1.1
C: Host: info.pvt.net
C:
S: HTTP/1.1 200 OK
S: Date: Sat, 23 Dec 2000 21:23:20 GMT
S: Server: Apache/1.2b10
S: Content-Length: 0
S: Allow: GET, HEAD, POST, OPTIONS, TRACE
```

Kupodivu se překvapení nekonalo, u tohoto URI server umožní i metodu POST. Vysvětlení je prosné: v příkladech 16.7 a 16.8 je pokaždé použité jiné URI. A metoda POST není pravděpodobně podporována pro všechny možné URI tohoto serveru.

## HTTP odpověď'

HTTP odpověď začíná stavovým rádkem ve tvaru:

<Verze> <Výsledkový kód> <Poznámka>

kde verze je verzí protokolu HTTP, ve které je odpověď formulována. Výsledkový kód specifikuje úspěšnost/neúspěšnost operace a poznámka textově objasňuje výsledek operace. Za stavovým rádkem následuje opět záhlaví tvorené hlavičkami. Záhlaví je ukončeno prázdným rádkem, který odděluje záhlaví od přenášených dat. V případě, že záhlaví obsahuje hlavičku „Transfer-Encoding: chunked“, může být za daty opět prázdný rádek následovaný zápatím, které je opět tvořeno hlavičkami. Nesešlo se mi jsem se v praxi s případem, kdy by se zápatí použilo.

### Příklad

Příklad stavového rádku (kladná odpověď):

HTTP/1.1 200 OK

Výsledkové kódy jsou trojciferné. První cifra určuje druh odpovědi:

- ◆ 1xx – informativní odpověď, zpracování pokračuje dále.
- ◆ 2xx – akce proběhla úspěšně.
- ◆ 3xx – přesměrování, tj. další akce se bude týkat jiného URI.
- ◆ 4xx – chyba klienta (např. syntaktická chyba v dotazu).
- ◆ 5xx – chyba serveru (např. chyba CGI-skriptu).

## Přehled výsledkových kódů

Pokud se poznámka nevejde do stavového řádku, je do odpovědi přidána hlavička **Warning**. Jedná se v podstatě o doplňující stavový řádek. Hlavička Warning má dva parametry oddělené mezerou: výsledkový kód a poznámku.

Nejčastěji se hlavička Warning používá k doplnění informací podávaných z paměti Cache, tj. nikoliv z první ruky (ze serveru). Může se totiž stát, že cache vrací prošlé informace, protože např. proxy není schopna navázat spojení dále směrem k serveru (to jsou výsledkové kódy 110 až 112).

Přehled výsledkových kódů používaných v hlavičce Warning:

```
110 Response is stale
111 Revalidation failed
112 Disconnected operation
113 Heuristic expiration
199 Miscellaneous warning
```

# Ostatní hlavičky

## Hlavičky Accept

Hlavičkami Accept, Accept-Charset, Accept-Encoding a Accept-Language sděluje klient ve svém dotazu své možnosti. Každá z těchto hlaviček může obsahovat několik eventualit oddělených čárkou. U každé možnosti může být za středníkem uvedena kvalita (q). Kvalita je číslo mezi 0 a 1. Čím vyšší má eventualita kvalitu, tím více ji klient preferuje (implicitně se předpokládá q=1). Pro specifikaci eventuality je možné uvést hvězdičku označující všechny možnosti.

Hlavičkou **Accept** klient specifikuje podporované typy médií (viz hlavička Content-Type). Např.:

```
Accept: text/*;q=0.3, text/html, image/jpeg;q=0.7, model/vrml, */*;q=0.1
```

Říká, že klient upřednostňuje:

1. model/vrml s kvalitou 1,
2. text/html s kvalitou 1,
3. image/jpeg s kvalitou 0,7,
4. libovolný text s kvalitou 0,3,
5. libovolné médium s kvalitou 0,1.

Jenž v praxi se server nebude pravděpodobně rozhodovat mezi médiem model a text, ale mezi texty či modely různé kvality.

Hlavičkou **Accept-Charset** klient specifikuje podporované znakové sady:

```
Accept-Charset: iso-8859-5, unicode-1-1;q=0.8, *;q=0.1
```

Říká, že klient upřednostňuje znakovou sadu iso-8859-5 (s kvalitou 1), dále s kvalitou 0,8 podporuje znakovou sadu unicode-1-1. Jinak podporuje libovolnou znakovou sadu s kvalitou 0,1.

Hlavičkou **Accept-Encoding** klient sděluje podporované typy kódování dat:

```
Accept-Encoding: compress;q=0.5, gzip
```

Klient preferuje metodu gzip, avšak s kvalitou 0,5 podporuje i metodu compress.

Hlavičkou **Accept-Language** klient sděluje podporované jazyky:

Accept-Language: cz, en;q=0.5

Klient preferuje češtinu, ale podporuje i angličtinu.

Hlavička Accept-Ranges je používána v odpovědi serveru klientovi. Viz příklad 16.1.

## Autorizace klienta

Klient sice může přímo do URI zapsat jméno uživatele a heslo, to je však málo běžné. Běžnější je dialog, kdy klient nezadá své autentizační informace. V tomto případě server vrátí:

```
HTTP/1.1 401 Unauthorized
WWW-authenticate: autent_metoda realm="řetězec", případně_další_parametry
```

První parametr je typ autentizační metody, kterou server vyžaduje. Řetězec „realm“ bude zobrazen klientovi, aby věděl, k jakému objektu se má autentizovat. Konečně některé autentizační metody mohou používat další parametry. Autentizační metoda Basic např. další parametry nepoužívá.

RFC-2617 rozeznává dva typy autentizace:

- ◆ Basic, kde autentizační metoda je „Basic“.
- ◆ Digest, kde autentizační metoda je „Digest“

Obě metody používají autentizaci jménem uživatele a heslem.



**Poznámka:** Autentizaci Basic je třeba nezaměňovat s autentizací jménem a heslem pomocí webového formuláře. V takovém případě si autentizaci řeší sám tvůrce aplikace – autentizační informace se nepřenáší v hlavičce Authorization, ale v datech webové stránky!

Autentizace Basic přenáší síti jméno a heslo v textovém (nezabezpečeném tvaru). Autentizační dialog pak probíhá např. takto :

```
C: GET /soubor HTTP/1.1
C: Host: info.pvt.net
C:
S: HTTP/1.1 401 Unauthorized
S: WWW-authenticate: Basic realm="info.pvt.net"
S: ...                                     ...další hlavičky
```

```
C: GET /soubor HTTP/1.1
C: Host: info.pvt.net
C: Authorization: Basic RG9zdGFsZWs6aGVzbG8=
C:
S: HTTP/1.1 200 OK
S: ...
```

Zde klient po obdržení „HTTP/1.1 401 Unauthorized“ provede autentizaci jménem a heslem. Jenže server může nabízet větší množství objektů („virtuálních serverů“) a ke každému z nich můžeme použít jinou autentizaci. Proto server vrací řetězec „realm“, aby prohlížeč mohl uživateli do dialogového okna zobrazit, k jakému objektu má zadat jméno uživatele a heslo. Ze jména a hesla je vytvořen

řetězec tím, že se mezi ně vloží dvojtečka (např. Dostalek:heslo). V hlavičce Authorization se nepřenáší řetězec přímo, ale kódován Base64, tj.

Base64(Dostalek:heslo)=RG9zdGFsZWs6aGVzbG8=

Komukoliv, kdo na cestě od klienta k serveru odchytí hlavičku Authorization, stačí pustit na řetězec RG9zdGFsZWs6aGVzbG8= dekódování Base64 (např. programem OpenSSL viz část 20.1.2), a získá heslo.

Tomu se snaží zabránit autentizace typu Digest. Tento typ autentizace rovněž používá heslo uživatele, avšak nepřenáší heslo samotné, ale kontrolní součet (počítaný např. algoritmem MD5) z:

- ◆ Čísla „nonce“ generovaného serverem jako kontrolní součet z časového razítka, identifikace odpovědi (ETag) a soukromého klíče serveru.
- ◆ Čísla „opaque“ generovaného rovněž serverem. Čísla „nonce“ a „opaque“ jsou předána klientovi v hlavičce WWW-authenticate jako další parametry.
- ◆ Jména uživatele.
- ◆ Hesla.
- ◆ Řetězce z parametru „realm“.
- ◆ Požadovaného URI.

Další možnosti autentizace přináší RFC-2831. Kromě toho některé firewally používají autentizaci Basic, avšak umožňují používat jednorázové heslo generované autentizačním kalkulátorem.

Microsoft zavedl typ autentizace SPNEGO (*Simple and Protected Negotiate*), která byla později standardizována v RFC-4559. Princip spočívá v tom, že server v hlavičce WWW-authenticate nabídne metodu Negotiate. Klient pak ve své odpovědi uvede hlavičku Authorization opět s dvěma parametry:

- ◆ Prvním parametrem je název autentizační metody, tj. v tomto případě řetězec „Negotiate“.
- ◆ Druhým parametrem je tzv. SPNEGO-token, který obsahuje lístek protokolu Kerberos (variantně je podporována i klasická NTLM-autentizace používaná ve starších verzích Windows).

Využití lístků protokolu Kerberos elegantně umožňuje klientům přihlášeným do domény Windows se přihlašovat i na webové servery běžící na UNIXu metodou *Single Sign On* – aby uživatel přihlášený do domény Windows nemusel znova zadávat přihlašovací informace při přístupu na webové servery (i UNIXové).

## Proxy autentizace

Autentizace klienta vůči proxy je zcela obdobná autentizaci klienta vůči serveru. V případě, že proxy vyžaduje autentizaci, zašle odpověď:

407 Proxy Authentication Required

Proxy-authenticate: autent\_metoda realm=řetězec, případně\_další\_parametry

Klient se autentizuje pomocí hlavičky Proxy-Authorization. Hlavičky Proxy-Authenticate a Proxy-Authorization mají stejnou syntaxi jako hlavičky WWW Authorization a Authenticate.

## Hlavičky Content

Hlavičky Content vycházejí z filozofie MIME, avšak se samotným MIME nejsou zcela kompatibilní. Protokol HTTP nepodporuje např. hlavičku Content-Transfer-Encoding a pochopitelně ani hlavičku Mime-Version.

Hlavička **Content-Type** je obdobou stejnojmenné hlavičky v MIME. Popisuje typ přenášených dat. Např.:

```
Content-Type: text/html; charset=ISO-8859-4
```

Tato hlavička specifikuje, že přenášená data jsou text formátovaný v jazyce HTML a použitá znaková sada je ISO-8859-4.

Hlavička **Content-Length** obsahuje délku přenášených dat.

Hlavička **Content-Encoding** specifikuje kódovací algoritmus použitý před odesláním dat. Jelikož protokol HTTP je osmibitový, kódováním se rozumí např. komprese dat.

## Příklad

```
Content-Encoding: gzip
```

Hlavička **Content-Language** specifikuje jazyk. Např.

```
Content-Language: cz
```

Hlavička **Content-MD5** obsahuje kontrolní součet algoritmem MD-5 z přenášených dat.

Hlavička **Content-Range** se použije v případě, že zpráva obsahuje pouze část přenášených dat. Např. je-li odpověď serveru je příliš velká, rozdělí ji server na několik částí:

```
S: HTTP/1.1 206 Partial content
S: Content-Range: bytes 21010-47021/47022
S: Content-Length: 26012
S: ... (za lomítkem je celková délka zprávy)
```

Hlavička **Content-Location** obsahuje URI s přenášenými daty. Tato hlavička má význam zejména v případě, když požadovaná data jsou uložena v několika lokalitách. Tato hlavička nesouvisí s přesměrováním! Podobný význam má hlavička **Referer**, kterou může klient serveru sdělit, odkud získal informace o požadovaném URI. Server si pak může dělat statistiky o odkazech na něj. Prohlížeče však většinou do hlavičky Referer vyplní URI právě zobrazené stránky. To je v případě, že je na příslušné stránce hypertextový odkaz na inkriminovanou stránku, v pořádku. Avšak v případě, že uživatel explicitně do dialogu zapíše nové URI, může být hlavička Referer zavádějící. Máte-li např. v prohlížeči zobrazenu stránku nějaké veřejně prospěšné společnosti a do dialogu zapíšete [www.playboy.com](http://www.playboy.com), do hlavičky Referer se dostane odkaz na veřejně prospěšnou společnost a v Playboym se ve statistice objeví, že na jejich stránky má odkaz tato veřejně prospěšná společnost.

## Přesměrování a dočasná nedostupnost

Může se stát, že požadovaný objekt byl přemístěn na jiné URI (např. na jiný server či do jiného adresáře). V takovém případě server vrátí stavový řádek s výsledkovým kódem 3xx a hlavičkou:

```
Location: nové-URI
```

Např.

```
HTTP/1.1 301 Moved Permanently  
Location: http://www.firma.cz/soubor  
...
```

Avšak může se i stát, že objekt nebyl přesměrován, ale je dočasně nedostupný. Pak může server klientovi sdělit nejen špatnou zprávu (že je objekt nedostupný), ale pomocí hlavičky **Retry-After** může klientovi poradit, za jak dlouho se má pokusit znova na objekt dotázat. Např. mu poradí, aby se dotázel za 1 minutu:

```
HTTP/1.1 503 Service Unavailable  
Retry-After: 60  
...
```

Hlavička **Retry-After** může mít význam i v případě přesměrování:

```
HTTP/1.1 301 Moved Permanently  
Location: http://www.firma.cz/soubor  
Retry-After: 60  
...
```

V tomto případě server sděluje klientovi, aby přesměrování provedl až za 60 sekund.

## **Hlavička Upgrade**

Hlavičkou Upgrade sděluje klient serveru, že by chtěl v existujícím TCP spojení přejít na „lepší“ protokol. Tj. např. protokol novější verze či protokol bezpečnější. Např.:

```
Upgrade: HTTP/2.0, SHHTTP/1.3
```

Server potvrzuje přechod do jednoho z uvedených protokolů stavovým kódem „101 Switching Protocols“.

## **Cache**

U paměti Cache si nejprve musíme uvědomit, že může být realizována u klienta, na proxy, u brány i na serveru. Nemůže být realizována pouze na tunelu, neboť tunel neví, co přenáší.

Paměť Cache je být buď sdílená, nebo privátní. Sdílená cache slouží pro ukládání informací, které jsou nezávislé na tom, jakému uživateli jsou určeny. V případě, že informace je závislá na uživateli (např. autentizační informace klienta, stavové informace atd.), nesmí být uložena do sdílené paměti Cache, avšak může být uložena do privátní cache.

Cache může urychlit komunikaci. Základním problémem však je, jak zajistit, aby cache neodpověděla prošlymi daty (*stale data*). Pokud server nechce od cache nic mimořádného, zpravidla nám v odpovědi vyplní hlavičky Date, Last-Modified a Expires. V případě mimořádných nároků navíc vyplní hlavičku Cache-Control, kterou však též může použít klient v dotazu.

Hlavička **Expires** určuje datum a čas, po jehož uplynutí se považuje informace za prošlou. Do té doby je informace považována za čerstvou (*fresh*). Cache si počítá dobu, po kterou může informaci považovat za čerstvou, v sekundách (*freshness lifetime*).

U informací se v paměti Cache udržuje stáří (*age*), tj. doba, po kterou byla informace uložena v paměti Cache či putovala sítí. Stáří se udává ve vteřinách. V případě, že informace je podána z paměti Cache, tj. není z první ruky (ze serveru), použije se hlavička **Age**, udávající stáří informace.

### Příklad

Příklad 16.9:

V případě, že odpověď obsahující následující hlavičky dorazí 23. 12. 2000 ve 20:11:22:

```
Date: Sat, 23 Dec 2000 19:11:22 GMT
Expires: Sat, 23 Dec 2000 22:11:22 GMT
Pak:
age=3600
freshness lifetime=7200
```

Pokud by se tato informace okamžitě (23. 12. 2000 ve 20:11:22) předala dále, doplnila by se o hlavičku:

```
Age: 3600
```

Problém je s informacemi neobsahujícími hlavičku Expires. Zde záleží na implementaci. Jako docela praktická se jeví úvaha o použití hlaviček Date a Last-Modified. Rozdíl hodnot uvedených v hlavičce Date a Last-Modified vyjadřuje dobu, po kterou leží informace nezměněna na serveru. Jako rozumné se pak může jevit udržovat tuto informaci v paměti Cache např. po 10 % tohoto rozdílu.

Hlavička Cache-Control může mít celou řadu parametrů. Klient může do dotazu v hlavičce Cache-Control uvést mj. následující parametry:

- ◆ **no-cache** – na požadavek nesmí být odpovězeno z paměti Cache,
- ◆ **no-store** – jedná se pravděpodobně o citlivé informace, proto nesmí být informace uloženy,
- ◆ **max-age=s** – klient si nepřeje informace starší (*age*) než **s** sekund,
- ◆ **min-fresh=s** – klient si přeje pouze informace, které budou považovány za čerstvé ještě minimálně **s** vteřin,
- ◆ **max-stale=s** – klient je ochoten akceptovat i informace, které jsou prošlé, ne však o více jak **s** sekund.

Server ve své odpovědi může uvést parametry:

- ◆ **public** – odpověď může být uložena i do sdílené paměti Cache,
- ◆ **private** – odpověď může být uložena pouze do privátní paměti Cache (odpověď obsahuje privátní informace – jiný uživatel může mít jiné informace ze stejného URI),
- ◆ **no-cache** – odpověď nesmí být uložena do cache,
- ◆ **no-store** – odpověď nesmí být uložena na disk (může např. obsahovat citlivé informace); výjimkou je uložení na disk (mimo cache) explicitně uživatelem (např. pravým tlačítkem myši),
- ◆ **must-revalidate** – server vyžaduje, aby veškeré paměti Cache na cestě byly tímto požadavkem občerstveny,
- ◆ **proxy-revalidate** – obdoba must-revalidate, avšak cache na klientovi nemusí být obnovena, tím se např. mohou v klientově paměti cache zachovat informace pro autentizaci klienta na proxy,

- ◆ **max-age=s** – server explicitně specifikuje maximální dobu, po kterou má být informace udržována v paměti Cache (*age*); pokud server do své odpovědi zahrne parametr max-age=0, donutí všechny paměti Cache na cestě občerstvit tuto informaci,
- ◆ **s-maxage=s** – obdoba max-age, avšak platí pouze pro sdílenou cache.

Je třeba ještě upozornit, že protokol http verze 1.0 podporoval jedinou hlavičku, a to:

Pragma: no-cache

Tou server přizkazoval, aby odpověď nebyla ukládána do paměti Cache. Proto v případě, že data nemají být uložena do paměti Cache, se používá tato hlavička též pro zpětnou kompatibilitu (ne všechny proxy a brány na cestě musí podporovat verzi 1.1).

## Informace o softwaru

Hlavičku **User-Agent** může vložit klient do svého dotazu, aby informoval server o svém softwaru. Tato informace může sloužit serveru pro statistické využití. Hlavička User-Agent se také používá pro formulaci odpovědi serverem v případě, že server chce využít nějaké speciální vlastnosti softwaru klienta. V současné době např. aplikace běžící na serveru zjišťují, jestli klient nepoužívá MS Explorer, a pokud ano, může aplikace na klienta odesílat ActiveX komponenty.

Hlavičku **Server** může použít server ve své odpovědi, aby informoval klienta o software, kterým je realizován.

## Cookie

Jak jsem již uvedl, protokol HTTP používá pouze dotaz a bezprostřední odpověď na dotaz. To neumožňuje udržet mezi klientem a serverem relaci skládající se z řady odpovědí a dotazů. Není možné předat nějakou informaci mezi dvěma dotazy na server.

RFC-2965 řeší tento problém zřízením jednoduché relace, během které je možné, aby informace z odpovědi byly předány do dalšího dotazu. Tyto informace se nazývají cookie. Server ve své odpovědi hlavičkou **Set-Cookie2** předá klientovi stavovou informaci. Klient pak může tuto informaci ve svém dalším dotazu zopakovat.

Klient může například pomocí cookie nakupovat ve virtuálním obchodním domě. Klient prochází virtuálním obchodním domem a vybírá si zboží do svého nákupního košíku. Jenže informace o tom, co má klient v nákupním košíku, musí být někde udržována. Tato informace o stavu relace klienta s obchodním domem bude udržována pomocí cookies:

1. Klient vejde úvodní stránkou <http://www.virtualni.obchodni.dum.cz/obchod/vstup> do virtuálního obchodního domu [www.virtualni.obchodni.dum.cz](http://www.virtualni.obchodni.dum.cz), tj. klient odešle HTTP dotaz:

C: POST /obchod/vstup HTTP/1.1

...

2. Server přiřadí klientovi identifikaci při vstupu pomocí hlavičky Set-Cookie2:

S: HTTP/1.1 200 OK

S: Set-Cookie2: Zakaznik="007"; Version="1"; Path="/obchod"

...

3. Klient si vybere zboží:

```
C: POST /obchod/saka HTTP/1.1  
C: Cookie: Version="1"; Zakaznik="007"; Path="/obchod"  
...
```

4. Server vrátí identifikaci vybraného zboží:

```
S: HTTP/1.1 200 OK  
S: Set-Cookie2: Version="1"; Zbozi="sako_05"; Path="/obchod"  
...
```

5. Klient může dále vybírat, ale nakonec si musí vybrat způsob dopravy zboží:

```
C: POST /obchod/doprava HTTP/1.1  
C: Cookie: Version="1";  
Zakaznik="007"; Path="/obchod";  
Zbozi="sako_05"; Path="/obchod"  
...
```

6. Server vrátí identifikaci zvolené dopravy:

```
S: HTTP/1.1 200 OK  
S: Set-Cookie2: Version="1"; Doprava="dobirka"; Path="/obchod"  
...
```

7. Klient dorazil na pokladnu a přeje si potvrdit objednávku (platbou se nyní nezabýváme):

```
C: POST /obchod/doprava HTTP/1.1  
C: Cookie: Version="1";  
Zakaznik="007"; Path="/obchod";  
Zbozi="sako_05"; Path="/obchod";  
Doprava="dobirka"; Path="/obchod"  
...
```

8. Serveru už zbývá jen potvrdit objednávku:

```
S: HTTP/1.1 200 OK  
...
```

Tím je relace skončena.

Složitost použití cookie spočívá ve svobodě klienta procházet virtuálním obchodním domem a uprostřed nákupu si odskočit na zcela jiný server, který shodou okolností může též používat cookie, a může mít dokonce některé údaje v URI stejné (např. cestu).

Klient si proto musí pro každý server zaznamenávat cookie. Jenže na témže serveru může běžet několik různých aplikací využívajících cookie, takže klient musí odděleně zaznamenávat jednotlivé komunikace pomocí cookies. Klient musí jednotlivé komunikace rozlišovat podle:

- ◆ Jména serveru, které nesmí obsahovat tečku uprostřed jména, tj. nesmí to být jméno serveru se subdoménou. (V našem případě se server jmenuje „www“.)
- ◆ Domény, ve které server leží. (V našem případě „virtualni.obchodni.dum.cz“.)
- ◆ Portu serveru. (V našem případě 80 – implicitní port HTTP serveru.)
- ◆ Cesty, která musí být začátkem v cestě uvedené v URI. Ve virtuálním obchodním domě mohu vybírat saka např. skriptem /obchod/saka; kravaty /obchod/kravaty či dámské boty skriptem /obchod/damske. Celý mechanismus cookie použité aplikace se pak týká celého podstromu /obchod. Jako cesta se uvede jméno tohoto podstromu.

## Hlavička Set-Cookie2

Tato hlavička může obsahovat následující parametry:

- ◆ **Parametr=hodnota**, tj. vlastní aplikační hodnoty. Např. Zbozi=tsako\_05. Hlavička musí obsahovat alespoň jeden výskyt tohoto parametru.
- ◆ **Version=1**, tj. verzi mechanismu cookies. Hlavička je povinná. Ostatní hlavičky jsou již nepovinné.
- ◆ **Comment=komentář**, může obsahovat komentář.
- ◆ **Discard**, cookie nemají být udržovány po ukončení běhu programu klienta (prohlížeče).
- ◆ **Domain=doména**, specifikuje doménu, jíž se tato cookie týká.
- ◆ **Max-Age=sekundy**, specifikuje maximální dobu, po kterou může klient tuto informaci udržovat. Nastavením „Max-Age=0“ může server ukončit relaci.
- ◆ **Path=cesta**, specifikuje cestu (odvozenou z URI), které se cookie týká.
- ◆ **Port=port**, specifikuje port serveru, ze kterého byly cookie odeslány.
- ◆ **Secure**, specifikuje požadavek na klienta, aby komunikoval zabezpečeným kanálem.

## Hlavička Cookie

Tato hlavička má následující syntaxi:

Cookie: \$Version=1; par1; par2; ...

Kde parx má syntaxi:

Parametr=hodnota [; \$Path=cesta] [; \$Domain=doména] [; \$Port=port]

## Domácí cvičení

- ◆ Programem `nmap` zjistěte, na jakých počítačích vaší sítě běží webové servery (nejspíše poběží na portu 80/tcp).
- ◆ Programem `telnet` vypište homepage těchto serverů.
- ◆ Pokud jste situováni na intranetu (za proxy), pokuste se programem vypsat `telnet`. Vypište homepage nějakého známého serveru v Internetu.



## Kapitola 17

# Elektronická pošta

Základní představa architektury internetové elektronické pošty (obr. 17.1) pochází z poloviny 70. let minulého století. Tehdy uživatelé seděli u terminálů, ze kterých spouštěli poštovní klienty. Poštovní klient neměl nic společného se síťovou komunikací. Poštovní klient byl v podstatě pouze specializovaný textový editor, který uměl uživateli zobrazit obsah poštovní zprávy a navíc uměl manipulovat se zprávami v jeho poštovní schránce. Totéž někdy uměl i s případnými dalšími (privátními) poštovními schránkami uživatele, které uživatelům dodnes slouží k dalšímu utřídění jejich elektronické pošty. Poštovním klientem bylo dále možné zprávu pořídit a odeslat. Odesláním se opět nerozumí nějaká síťová komunikace, ale uložení zprávy do fronty zpráv poštovního serveru.

Dodnes jsou základem internetové poštovní komunikace dvě normy:

- ◆ RFC-821, která byla v roce 2001 doplněna, vyšla jako RFC-2821. Tyto normy specifikují základní komunikační protokol pro přenos elektronické pošty v Internetu, kterým je protokol SMTP (*Simple Mail Transfer Protocol*).
- ◆ RFC-822, která byla v roce 2001 rovněž doplněna a vyšla jako RFC-2822. Tyto normy specifikují formát nejenom poštovní zprávy v Internetu. Odtud také pramení termín „**zpráva je formátu RFC-822**“.

Fronta zpráv na poštovním serveru (obr. 17.1) je pravidelně procházena SMTP klientem, který navazuje spojení se vzdálenými SMTP servery, jimž zprávu předá.



**Poznámka:** Všimněte si, že rozlišujeme mezi poštovním klientem a klientem SMTP.

SMTP server přijme zprávu a zjišťuje, je-li určena pro jeho lokální uživatele. V případě, že nikoliv, pak zprávu opět uloží do poštovní fronty. Tu obsluhuje jeho poštovní klient, jenž se pokouší zprávu doručit směrem k adresátovi.

V případě, že adresát je lokálním uživatelem systému, pak SMTP server uloží přijatou zprávu do poštovní schránky adresáta. Zde je třeba upozornit, že poštovní server mívá přístupová práva ke všem uživatelům svého systému, tj. SMTP server bývá spuštěn pod privilegovaným uživatelem. Pokud by se útočník dokázal prolomit do poštovního serveru, může získat neomezený přístup do systému. Proto je správnější jiný postup: poštovní server běží pod jiným uživatelem, než je superuživatel. Přístup do poštovních schránek uživatelů se mu pak zajistí např. přes skupinová práva.

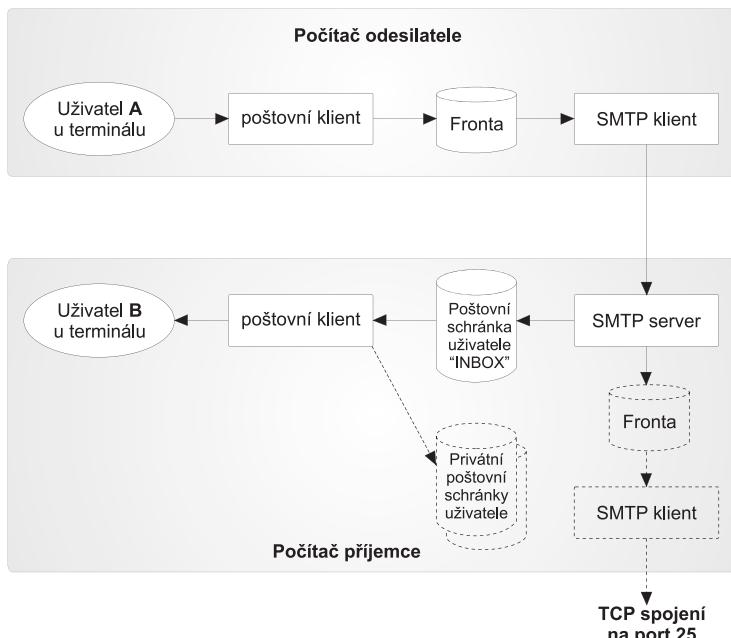
Uživatel má v systému zpravidla jednu poštovní schránku INBOX, kam SMTP server ukládá jeho příchozí poštu. Poštovní schránka se nejmíne jako soubor INBOX, ale zpravidla má jméno shodné se jménem uživatele (v UNIXu např. /var/spool/mail/uživatel). Název INBOX se pro ni používá zejména v protokolu IMAP4.

Kromě poštovní schránky pro příchozí poštu (INBOX) si uživatel může zřídit i privátní poštovní schránky. Do téhoto schránek si přesouvá zprávy ze schránky INBOX. Privátní poštovní schránky nejsou obsluhovány SMTP serverem. Bývají zřizovány např. v domovském adresáři uživatele. Cílem je přimět uživatele k tomu, aby si v „systémové“ schránce INBOX příchozí poštu nearchivovali. Tohoto cíle dosahují i někteří poštovní klienti tím, že pokud si příchozí poštu uživatel zobrazí, automaticky ji přenesou do privátní poštovní schránky, kterou – pokud ji uživatel nepojmenuje jinak – pojmenují INBOX nebo inbox, ale v domovském adresáři uživatele.

Internetová pošta má díky ukládání odchozí pošty do fronty a ukládání příchozí pošty do poštovní schránky adresáta jednu zásadní vlastnost. Tou je skutečnost, že uživatel může „odeslat“ e-mail, který si příjemce může vyzvednout ze své schránky, až bude chtít, tj. není tedy nutné okamžitě navazovat spojení odesilatele na příjemcův systém v době odeslání pošty. Příjemcův systém může být i vypnut v době, kdy odesílatel zprávu odesílá. Nepodaří-li se klientovi SMTP poštu odeslat, ponechá ji ve frontě. Zpráva pochopitelně nebude ve frontě do nekonečna. Správce systému má většinou nastavenu maximální dobu položky v poštovní frontě na 2–7 dní. Poté se pošta vrací odesílateli jako nedoručená.

Je to ještě komplikovanější. SMTP klient může neodeslat zprávu z mnoha důvodů. Je na správné konfiguraci, aby v podstatě rozlišovala mezi dvěma situacemi:

- ◆ Momentálně např. nelze poštu dále doručit, ale je pravděpodobné, že za jistou dobu to půjde (např. jméno cílového systému se přeložilo v DNS, ale systém je nedostupný).
- ◆ Zprávu nelze doručit pro chybu, kterou nelze odstranit (např. jméno vzdáleného systému je správné, ale uvedený adresát na systému neexistuje). V takovém případě je třeba zprávu neponechávat ve frontě, ale okamžitě vrátit odesílateli.



**Obrázek 17.1:** Architektura SMTP

Na obr. 17.1 chce uživatel A odeslat zprávu elektronickou poštou uživateli B. Uživatel A pomocí svého poštovního klienta pořídí zprávu. Nakonec pořízenou zprávu „odešle“, avšak odeslání je pouze uložení zprávy do fronty. SMTP klient prochází frontu, až dorazí na naši zprávu. Zprávu se pokusí doručit na adresátův systém; pokud se mu to nepodaří, ponechá zprávu ve frontě.

V případě, že server zprávu přijme, zkoumá, je-li adresát zprávy lokálním uživatelem systému; pak mu zprávu uloží do jeho poštovní schránky INBOX. V případě, že adresát není lokálním uživatelem systému, uloží zprávu do fronty k odeslání dále.

Pokud zpráva dorazí až do poštovní schránky INBOX adresáta, pak si ji adresát může zpracovat svým poštovním klientem.

S příchodem osobních počítačů přišel zvrat i v používání elektronické pošty. Ve svém jádru elektronická pošta zůstala zachována, avšak uživatel již nechce sedět u terminálu poštovního serveru (byť emulovaného protokolem Telnet nebo ssh na svém PC), ale chce využívat aplikace na svém PC. Otázkou je, jak z PC poštu odeslat a jak na PC poštu přijmout.

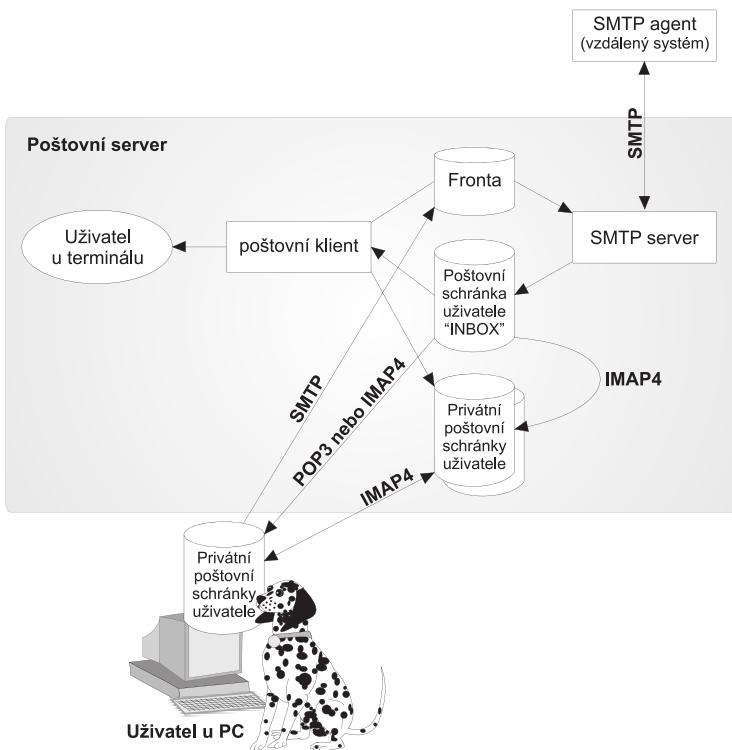
Zatímco při odesílání pošty z PC lze vcelku bez problému opět použít protokol SMTP, pro příjem pošty z poštovního serveru na PC není protokol SMTP vhodný. Co by to totiž znamenalo? Příjemce má své PC zapnuto zpravidla jen několik hodin. Kromě této doby by zůstávala pošta v odesilatelově frontě a příjemcův systém by se jevil jako nedostupný. Dalším problémem je, že na příjemcově PC by musel běžet SMTP server. Zvolila se proto jiná strategie, znázorněná na obr. 17.2.

Uživatel má svou příchozí poštovní schránku (INBOX) na poštovním serveru. To znamená, že z hlediska protokolu SMTP je poštovní server cílovou stanicí. Zůstává vyřešit, jak si vybírat INBOX ze svého PC.

Pro práci s poštovní schránkou uživatele na poštovním serveru jsou k dispozici dva protokoly (oba jsou podporovány jak klienty Microsoftu, tak i ostatními poštovními klienty):

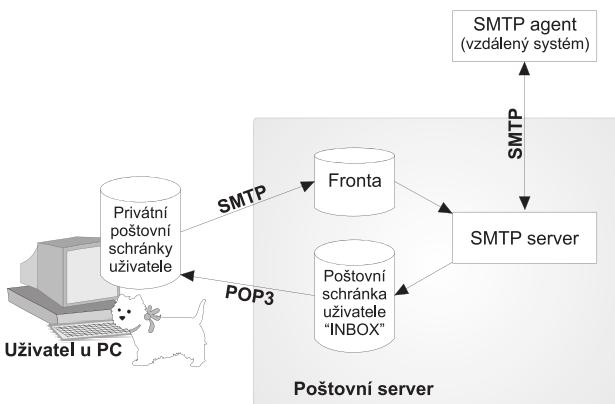
- ◆ Protokol POP3. Jedná se o velice jednoduchý protokol, pomocí kterého pracuje uživatel Offline. Z poštovního serveru si uživatel stáhne příchozí poštu na své PC a ukončí spojení se serverem. Teprve po stažení pošty uživatel pracuje s jednotlivými poštovními zprávami. V případě, že chce uživatel poštu odeslat, použije protokol SMTP.
- ◆ Protokol IMAP4. Jedná se o komplikovaný protokol, který umožňuje pracovat nejen offline, ale i online. Uživatel může mít navázáno spojení s poštovním serverem delší dobu a být serverem průběžně informován o změnách ve své poštovní schránce. Protokolem IMAP4 je možno také synchronizovat poštovní schránky na serveru s jejich kopiami na PC. Schránky na serveru tak zůstávají zálohovou schránek na PC. V případě, že chce uživatel poštu odeslat, použije také protokol SMTP. Použití protokolu IMAP4 je praktické zejména v případě, že někdy chceme pracovat z PC a někdy z terminálu serveru nebo také pokud chceme pracovat z více různých PC.

Otázkou je, kdy zvolit POP3 a kdy IMAP4. Pro velké poskytovatele internetových služeb je velice výhodný protokol POP3, protože pošta nezůstává na serveru (obr. 17.3). Uživatelé si ji stahují na svá PC. V případě, že sto tisíc uživatelů má trvale svou poštu na serveru, není žádná disková kapacita dostatečná pro takové ohromné množství dat. To může být příležitost pro malé poskytovatele, kteří mohou nabídnout některým svým klientům nadstandardní službu privátních poštovních stránek na serveru.



**Obrázek 17.2:** Protokoly POP3 a IMAP4

Naopak pro menší firmy je výhodný protokol IMAP4, protože se jím provádí záloha poštovních schránek. A je jednodušší zálohovat jednu diskovou kapacitu, než aby si každý uživatel zajíšťoval sám. Přitom ztráta obsahu poštovní schránky může pro vedoucího pracovníka způsobit i velké ekonomické škody. Je proto nutné při použití protokolu POP3 dbát na to, aby pošta byla zálohována.



**Obrázek 17.3:** Pošta v případě velkých poskytovatelů internetových služeb

Na obrázcích neuvádíme slovo poštovní server, ale SMTP agent. Je to proto, že na poštovním serveru je vždy SMTP klient pro odesílání pošty a SMTP server pro příjem pošty. Navíc software poštovního agenta musí v pravidelných intervalech procházet frontu a snažit se její položky odeslat. Jedná se tedy o službu (resp. démona), která běží stále. V okamžiku odesílání položky fronty se tento démon pak chová jako klient (z hlediska protokolu TCP).

Otázkou je, jak zorganizovat poštu ve vnitřní síti firmy. V případě, že máme více poštovních serverů, osvědčí se použít jednoho centrálního poštovního serveru označovaného jako „mail hub“. Centrálním poštovním serverem pak prochází veškerá poštovní komunikace firmy. Na centrálním poštovním serveru se také nazývá průchod všech poštovních zpráv firmy. Je možné na jednom místě dohledat, zda e-mail odešel a kdy. Podobně jako u proxy je i na centrálním poštovním serveru možné kontrolovat, zda poštovní zprávy neobsahují viry nebo spamy. Kdyby byla možná přímá komunikace mezi jednotlivými systémy, pak by taková ochrana byla nutná na každém systému. To by mohlo být ekonomicky neúnosné.

Konfigurací poštovních serverů se zabývá postmaster. Veškeré konfigurační lahůdky se soustředí na konfiguraci centrálního poštovního serveru. Konfigurace lokálních poštovních agentů bývá jednoduchá (vše odesílájí na centrální poštovní server). Často lze pro konfiguraci lokálního poštovního agenta použít nastavení vzniklé při instalaci systému.

Je několik možností, jak nakonfigurovat centrální poštovní server firmy. Na počátku je třeba stanovit, jaké budou zaměstnanci používat poštovní adresy. Jsou v podstatě dvě filozofie:

1. Poštovní adresy typu *novak@obchod.firma.cz* či *dvorak@provoz.firma.cz*.
2. Poštovní adresy typu *Frantisek.Novak@firma.cz*.

V prvním případě vnitřní síť firmy rozdělíme na DNS domény *obchod.firma.cz*, *provoz.firma.cz* atd. Pro doménu budeme provozovat lokální poštovní server (lokální poštovní agent). Přitom je možné obsluhovat na fyzicky jednom serveru i více domén.

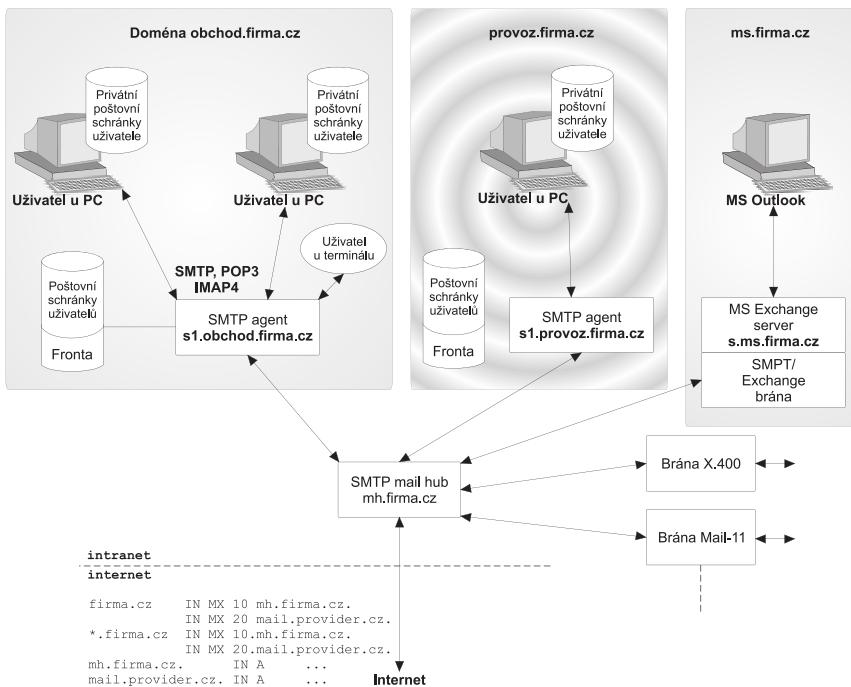
Z Internetu nám přicházejí poštovní zprávy pro *novak@obchod.firma.cz* na centrální poštovní server. Centrální poštovní server je nakonfigurován tak, aby všechny e-maily do domény *obchod.firma.cz* předával na server *s1.obchod.firma.cz*.

Na serveru *s1.obchod.firma.cz* má pak uživatel *novak* svou poštovní schránku (INBOX). Uživatel *novak* tak pravděpodobně bude mít dvě poštovní adresy:

- ◆ *novak@s1.obchod.firma.cz*,
- ◆ *novak@obchod.firma.cz*.

SMTP agent bude na serveru *s1.obchod.firma.cz* pravděpodobně nakonfigurován tak, aby akceptoval pro své lokální uživatele obě poštovní adresy.

Ve druhém případě uživateli přichází poštovní zpráva pro adresáta *Frantisek.Novak@firma.cz* na centrální poštovní server. Zde se tato adresa přeloží na adresu *novak@obchod.firma.cz* nebo na adresu *novak@s1.obchod.firma.cz*. To je již adresa doručitelná v rámci vnitřní síť. Jinou možností centrálního poštovního serveru je nepřekládat adresu příjemce a držet informaci, že tento příjemce má poštovní schránky na serveru *s1.obchod.firma.cz*. Ten pak může jméno *Frantisek.Novak* akceptovat či přeložit na *novak* – to již závisí na konfiguraci lokálního e-mailového serveru.



**Obrázek 17.4:** Centrální poštovní server (mail hub)

Uživatel tak může mít už i tři poštovní adresy:

- ◆ novak@s1.obchod.firma.cz
- ◆ novak@obchod.firma.cz
- ◆ Frantisek.Novak@firma.cz

Centrální poštovní server je většinou nakonfigurován tak, že u odchozí pošty do Internetu přepisuje adresu odesilatele tak, aby každý odesilatel měl jen jednu adresu. A také adresy typu novak@s1.obchod.firma.cz do Internetu sdělují něco o vnitřní struktuře firmy, což může být potenciálním zdrojem informací i pro útočníky.

To, že uživatel má více adres, ve vnitřní síti celkem nevadí. Nepříjemné je to u konferencí v Internetu. Např. uzavřené konference (na bázi systému listserv) kontrolují adresu odesilatele. Pokud se uživatel zaregistruje do konference pod jedním jménem a pod jiným jménem tam zaše příspěvek, může být příspěvek odmítnut.



**Poznámka:** Překlad poštovních adres na centrálním poštovním severu byl sice krásnou ideou, ale s příchodem bezpečné pošty (S/MIME) vzala za své. Pokud totiž např. přeložíte (změníte) adresáta u šifrované pošty, cílový poštovní klient nebude schopen podle adresáta nalézt příslušný certifikát, a proto ani příslušný dešifrovací klíč. (Více k S/MIME viz kniha *Velký průvodce infrastrukturou PKI a technologií elektronického podpisu*, Computer Press, 2006).



**Poznámka:** V této kapitole jsou uvedeny příklady za využití programu telnet. Pokud chcete program telnet využívat ve Windows Vista, pak je nutné jeho použití povolit volbou „Zapnout nebo vypnout funkce systému Windows“.

## Elektronická pošta a DNS

Máme mnoho eventualit poštovních adres. Např. *novak@s1.obchod.firma.cz*, *novak@obchod.firma.cz*, *Frantisek.Novak@firma.cz*, či dokonce *František.Novák@s1.obchod.firma.cz*. V Internetu je třeba poštovní zprávy se všemi těmito adresami dopravit na poštovní server *s1.obchod.firma.cz*. Toho lze dosáhnout řádným záznamem v DNS. Např.:

```
firma.cz.          IN MX 10 mh.firma.cz.  
                  IN MX 20 mail.provider.cz.  
*.firma.cz.       IN MX 10 mh.firma.cz.  
                  IN MX 20 mail.provider.cz.  
mh.firma.cz.      IN A ...  
mail.provider.cz. IN A ...
```

MX-záznamy slouží k přesměrování pošty na konkrétní poštovní server – v našem případě centrální poštovní server *mh.firma.cz*. První řádek říká, že veškerá pošta, která končí řetězcem „@*firma.cz*“, se má směrovat na server *mh.firma.cz*. Druhý řádek pak říká, že pokud náhodou není server *mh.firma.cz* momentálně dostupný, má se pošta dopravit na server *mail.provider.cz*, odkud bude později přepravena na *mh.firma.cz*.

Třetí řádek říká, že veškerá pošta končící řetězcem „*firma.cz*“ se má směrovat na server *mh.firma.cz*. Pokud nechceme v Internetu přepravovat adresy typu *novak@s1.obchod.firma.cz*, *novak@obchod.firma.cz* či *Frantisek.Novák@s1.obchod.firma.cz*, ale pouze adresy typu *Frantisek.Novak@firma.cz* či *novak@firma.cz*, pak třetí a čtvrtý řádek s hvězdičkou nepoužijeme.

Jinou eventualitou je, že nepoužijeme hvězdičku, ale v DNS vyjmenujeme subdomény druhé úrovně, pro které chceme v Internetu dopravovat poštu:

```
firma.cz.          IN MX 10 mh.firma.cz.  
                  IN MX 20 mail.provider.cz.  
obchod.firma.cz.  IN MX 10 mh.firma.cz.  
                  IN MX 20 mail.provider.cz.  
provoz.firma.cz. IN MX 10 mh.firma.cz.  
                  IN MX 20 mail.provider.cz.  
ms.firma.cz.       IN MX 10 mh.firma.cz.  
                  IN MX 20 mail.provider.cz.
```

## Formát poštovní zprávy

Formát poštovní zprávy je specifikován normou RFC-2822. Zpráva se skládá ze záhlaví a těla zprávy. Záhlaví je od těla zprávy odděleno jedním prázdným řádkem (tj. ASCII-znaky: CRLF CRLF). Záhlaví i tělo zprávy jsou tvořeny pouze ASCII-znaky!

Záhlaví se skládá z jednotlivých hlaviček. Hlavička začíná klíčovým slovem ukončeným dvojtečkou. Za klíčovým slovem mohou následovat parametry. Hlavička se ukončuje koncem řádku, tj. CRLF.

Mezi jednotlivými částmi hlavičky mohou být vkládány mezery a tabelátory. Hlavička může pokračovat na další řádek. Další řádek však v takovém případě musí začínat mezerou nebo tabelátorem (klíčové slovo hlavičky musí být odsazeno od první pozice řádku).

Zejména v adrese mají důležitý význam následující znaky:

- ◆ Středník a dvojtečka mají význam oddělovače v seznamu. Užívají se např. při oddělování jednotlivých adresátů v hlavičce To.
- ◆ Špičaté závorky < > mají speciální význam v adrese. Vyskytuje-li se v adrese řetězec ve špičatých závorkách, pak se vše mimo tento řetězec ignoruje – za adresu se vezme řetězec ze špičatých závorek.
- ◆ Hranaté závorky [ ] mají význam ve jméně počítače; signalizují, že jméno počítače nemá být překládáno v DNS.
- ◆ Do kulatých závorek ( ) se uzavírá komentář.

### Příklad

```
From: Libor Dostalek
<dostalek@pvt.cz>
To: Frantisek.Novak@firma.cz;
Novak@[195.47.40.4] (to je primo IP-adresa)
```

### Přehled základních hlaviček z RFC-822

Hlavička	Význam
Received:	Tuto hlavičku připisuje na počátek e-mailu každá e-mailová gateway (e-mailový server), kterou zpráva prochází. <b>Takže čteme-li hlavičky Received: odspodu nahoru, zjistíme celou trasu, přes které e-mailové servery zpráva šla.</b> V této hlavičce se mohou vyskytovat slova: <ul style="list-style-type: none"> <li>• from – počítač, ze kterého byla zpráva přijata,</li> <li>• by – počítač, kterým byla zpráva přijata,</li> <li>• via – fyzická cesta,</li> <li>• with – síťový nebo poštovní protokol,</li> <li>• id – příjemcova identifikace zprávy,</li> <li>• for – pro koho je zpráva určena (naříkadel je-li adresátem distribuční list, pak se zde zachová původní adresát – tj. distribuční list).</li> </ul>
From:	Od
Sender:	Vyřizuje (sekretářka) – prakticky zde bývá např. informace o konferenci, přes kterou zpráva přšla.
Date:	Datum odeslání (den, datum, čas a časová zóna).
Reply-To:	Odpověď zasílejte na.
In-Reply-To:	Odpovídáme vám na vaši zprávu: Identifikace původní zprávy.
To:	Adresát.
Cc:	Na vědomí ( <i>Carbon Copy</i> ).
Bcc:	Tajná kopie – tato hlavička se před odesláním smaže ( <i>Blind Carbon Copy</i> ).
Message-Id:	Identifikace zprávy.

Hlavička	Význam
Keywords:	Klíčová slova charakterizující obsah.
References:	Další odkazy.
Subject:	Věc (krátká charakteristika obsahu zprávy).
Comments:	Komentář.
Encrypted:	Šifrováno (zastaralé).
X-	Uživatelsky definovaná hlavička (uživatelem se rozumí autor softwaru). Např. X-Mailer se často používá pro specifikaci programu, kterým odesílatel odesílá zprávu.
Resent-	Při automatickém předávání zprávy (např. vrácení nedoručitelné zprávy) se před původní hlavičkou vloží řetězec Resent- (např. Resent-From nebo Resent-Cc apod.).

### Příklad

```

Received: from mh.pvt.cz (mh.pvt.cz [194.149.105.36])
        by cbu.pvt.net (8.8.5/8.7.3)
        with SMTP id PAA23028;
        Wed, 16 Apr 1997 15:10:14 +0200 (MET DST)
Received: from ncc.ripe.net by mh.pvt.cz
        with SMTP id AA08008 (5.67b/IDA-1.5
        for <registr@pvt.cz>);
        Wed, 16 Apr 1997 15:05:09 +0200
Received: by ncc.ripe.net
        id AA00228 (5.65a/NCC-2.41);
        Wed, 16 Apr 1997 13:55:02 +0200
Received: from mail.freedotnet.net
        by ncc.ripe.net with SMTP
        id AA00215 (5.65a/NCC-2.41);
        Wed, 16 Apr 1997 13:55:00 +0200
Received: from jon.freedotnet.net ([208.215.186.129])
        by homer.freedotnet.com (Netscape Mail Server v2.02)
        with ESMTP
        id AAA105 for <local-ir@ripe.net>;
        Wed, 16 Apr 1997 13:09:15 +0100
Reply-To: <jon.brody@freedotnet.net>
From: "Dr Jon Brody" <jon.brody@freedotnet.net>
To: <local-ir@ripe.net>
Subject: Re: Role?
Date: Wed, 16 Apr 1997 12:58:51 +0100
X-Mailer: Microsoft Internet Mail 4.70.1155

```

### Text e-mailu

Hlavičky této zprávy se musí rozdělit na hlavičky začínající Received a ostatní hlavičky. Hlavičky Received přidávají na začátek zprávy poštovní servery, kterými zpráva prochází. Proto u hlaviček Received záleží na pořadí. Hlavička Received, kterou přidal první server, je poslední. Hlavička Received před ní je hlavičkou, kterou přidal další server atd.

Čtěme-li hlavičky Received zdola nahoru, vidíme, že zpráva byla odeslána z počítače jon.freedotnet.net, pak ji zpracovával počítač homer.freedotnet.net (počítač mail.freedotnet.net je jen jiné jméno

téhož počítače). Dále zpráva putovala na ncc.ripe.net. Ten ji předal na mh.pvt.cz a nakonec skončila na cbu.pvt.net.cz.



**Tip:** O rozšířeném formátu poštovní zprávy – MIME – se blíže můžete dozvědět v publikaci „Velký průvodce infrastrukturou PKI a technologií elektronického podpisu“, Computer Press, 2006.

## SMTP

Vlastní protokol SMTP (*Simple Mail Transfer Protocol*) je poměrně jednoduchý protokol. Jednotlivé příkazy jsou textové v kódu ASCII (podobně jako u protokolu FTP). Je proto snadné využívat program Telnet např. pro odeslání poštovní zprávy protokolem SMTP.

Klient navazuje spojení protokolem TCP se serverem na dobré známém portu 25. Klient vkládá do takto vytvořeného kanálu příkazy a server odpovídá odpovědí obsahující trojciferný stavový kód následovaný textovým popisem chyby.

Příkazy klienta jsou čtyřznaková slova. Nezávisí na tom, zda se použijí velká či malá písmena (či jejich kombinace). Příkaz může být následován parametrem, který je oddělen mezerou. Příkaz je zakončen koncem řádku CRLF.

Princip protokolu SMTP si můžeme ukázat na příkladu odeslání e-mailu pomocí programu Telnet z Windows Vista (doplněno o C: a S:, tj. o signalizaci, že se jedná o příkaz klienta či odpověď serveru):

```
C:\ > telnet  
C: telnet> open smtp.nextra.cz 25  
S: 220 dns.terminal.cz ESMTP  
  
C: MAIL FROM: fantom@peklo.org  
S: 250 fantom@peklo.org... Sender ok  
  
C: RCPT TO: dostalek@pvt.cz  
S: 250 dostalek@pvt.cz... Recipient ok  
  
C: DATA  
S: 354 Enter mail, end with "." on a line by itself  
  
C: Prijdu si pro tebe.  
C: Fantom  
C: .  
C:  
S: 250 UAA91875 Message accepted for delivery  
  
C: QUIT  
S: 221 dns.terminal.cz closing connection
```



**Poznámka:** Pokud si předchozí příklad chcete vyzkoušet ze svého PC, pak nezapomeňte zaměnit řetězec „smtp.nextra.cz“ za DNS jméno SMTP serveru vašeho poskytovatele Internetu.

Po navázání TCP spojení se server představil (stavový kód 220). Klient mohl zadat první příkaz. Pokud chce odeslat e-mail, pak zahájí dialog příkazem MAIL, který jako parametr musí mít slovo FROM: následované odesilatelem (prázdný odesilatele se zadá jako FROM: <>). Server si ověří odesilatele a stavovým kódem 250 mne informuje, že odesilatele akceptuje.

Dále klient musí zadat adresáta příkazem RCPT. Adresát se zadává jako parametr příkazu RCPT za slovem TO:. Server opět stavovým kódem 250 souhlasí s adresátem. Nyní mohu přejít k odeslání vlastní poštovní zprávy. Postupuji tak, že zadám příkaz DATA. Server mne stavovým kódem 354 upozorní, že kdybych nevěděl, zpráva se ukončuje tečkou na novém řádku, po které musí následovat opět nový řádek (tj. CRLF.CRLF). Zpráva se vezme celá, jak je (včetně záhlaví), a odešle se. Server nám opět zprávou 250 sděluje, že zprávu přijal.

Nakonec zadám příkaz QUIT pro ukončení spojení. Ukončení spojení potvrďí server zprávou 221.

Příkaz	Význam
HELO klient	Klient se představuje serveru jménem počítače. Příkaz by se měl používat na počátku dialogu klienta se serverem: C: helo libor.pvt.net S: 250 dns.terminal.cz Hello Libor.pvt.net, pleased to meet you
MAIL FROM: odesilatele	Odesilatele.
RCPT TO: příjemce	Příjemce (tentto příkaz se opakuje pro každého příjemce).
DATA	Chci odeslat zprávu.
RSET	Aktuální translace bude abnormálně ukončena (veškeré přenesené informace ve FROM a TO budou zahrozeny).
SEND FROM: odesilatele	Obdoba příkazu MAIL, ale zpráva má být zobrazena na terminálu příjemce. Nepoužívá se.
SOML FROM: odesilatele	Obdoba příkazu MAIL, ale zpráva má být zobrazena na terminálu nebo uložena do poštovní schránky příjemce. Nepoužívá se.
SAML FROM: odesilatele	Obdoba příkazu MAIL, ale zpráva má být zobrazena na terminálu a také uložena do poštovní schránky příjemce. Nepoužívá se.
VRFY adresa	Dotaz, zda příjemce zná uvedenou adresu. Server vrací plné jméno uživatele a jeho přesnou poštovní identifikaci.
EXPN adresa	Obdoba VRFY, ale umí pracovat nejen s jednotlivými uživateli, ale i s celými seznamy uživatelů.
HELP [příkaz]	C: help S: 214-This is Sendmail version 8.9.3 S: 214-Topics: S: 214- HELO EHLO MAIL RCPT DATA S: 214- RSET NOOP QUIT HELP VRFY S: 214- EXPN VERB ETRN DSN S: 214-For more info use "HELP <topic>"
QUIT	Ukončení spojení.
TURN	Přepnutí role serveru a klienta. Pokud server tento příkaz potvrdí, klient očekává, že server začne s odesíláním poštovních zpráv ze serveru na klienta. Jelikož se tento příkaz nepovažuje za bezpečný, nepoužívá se. Nebezpečí spočívá v tom, že kdokoliv bez jakékoliv autentizace by mohl „vycucnout“ ze serveru frontu mailů.

Asi jste si všimli, že některé údaje, jako je odesilatel a příjemce, se zadávají dvakrát. Jednou jsou zadány v záhlaví zprávy (v hlavičkách), podruhé v příkazech protokolu SMTP (např. v příkazech MAIL a RCPT).

Údaje ze záhlaví zprávy jsou z hlediska transportu zprávy Internetem druhořadé. Protokol SMTP totiž neprepravuje pouze záhlaví zprávy, prázdný rádek a text zprávy. Přepravuje ještě tzv. SMTP -obálku zprávy (nezaměňovat s elektronickou obálkou v protokolu S/MIME!). Obálka zprávy právě obsahuje údaje např. z příkazů MAIL a RCPT.

Pro přepravu zprávy mezi SMTP servery je tak důležitá SMTP obálka. V předchozím příkladě jsme odeslali programem Telnet zprávu, která neobsahovala žádné hlavičky (tj. neobsahovala ani hlavičky To: a From:). Avšak pokud byste si prohlédli tuto zprávu po doručení adresátovi, uvidíte, že poštovní server dodělal hlavičku From:, tj. vzal údaje ze SMTP obálky a udělal z nich hlavičku.

Na druhou stranu pokud nějakým programem vložíte do fronty poštovního serveru (nebo přímo předáte programu sendmail) poštovní zprávu bez obálky, musí si vzít z hlaviček údaje do obálky. Hlavičky v záhlaví jsou druhořadé, ale někdy se hodí pro přepravu poštovní zprávy.

Tím lze také vysvětlit, jak je možné, že adresátovi dojde zpráva adresovaná pomocí hlavičky Bcc:. Je to jednoduché. Pokud poštovní server načítá zprávu a např. prvotně pro ni vytváří SMTP obálku a narazí na hlavičku Bcc, pak tuto hlavičku ze zprávy vyoperuje. To však neznamená, že informace z ní zahodí – vloží je do SMTP obálky. Tím způsobí, že zpráva je doručena adresátovi, ale tento adresát není zmíněn v záhlaví zprávy.

### Přehled stavových kódů

```
211 System status, or system help reply
214 Help message
220 <domain> Service ready
221 <domain> Service closing transmission channel
250 Requested mail action okay, completed
251 User not local; will forward to <forward-path>

354 Start mail input; end with <CRLF>.<CRLF>
421 <domain> Service not available,
450 Requested mail action not taken: mailbox unavailable
451 Requested action aborted: local error in processing
452 Requested action not taken: insufficient system storage
500 Syntax error, command unrecognized
501 Syntax error in parameters or arguments
502 Command not implemented
503 Bad sequence of commands
504 Command parameter not implemented
550 Requested action not taken: mailbox unavailable
551 User not local; please try <forward-path>
552 Requested mail action aborted: exceeded storage allocation
553 Requested action not taken: mailbox name not allowed
```

## ESMTP

Rozšířením protokolu SMTP vznikl protokol ESMTP (*Extensions SMTP*). Princip tohoto rozšíření původně specifikoval samostatný standard RFC-1869, ale následně toto rozšíření bylo více či méně včleněno i do RFC-2821. Základním problémem jakéhokoliv rozšíření je jeho zpětná komptabilita. ESMTP v tomto směru přichází s velice vtipným řešením.

Zatímco protokol SMTP zpravidla začíná svůj dialog příkazem HELO, tak ESMTP použije příkaz EHLO. Server budé odpoví:

- ◆ že se klient musel splést ve jméně příkazu. Odesilatel si okamžitě uvědomí, že server je „pouze“ SMTP, a pokračuje příkazem HELO;
- ◆ stavovým kódem 250 („že je vše v pořádku“). Klient okamžitě pozná, že server je ESMTP. Navíc server ve své odpovědi uvede, které rozšiřující příkazy podporuje:

```
C: ehlo Libor.pvt.net
S: 250-dns.terminal.cz Hello Libor.pvt.net, pleased to meet you
S: 250-EXPN
S: 250-VERB
S: 250-8BITMIME
S: 250-SIZE 8388608
S: 250-DSN
S: 250-ONEX
S: 250-ETRN
S: 250-HELP
```

Zbývá popsat jednotlivé rozšiřující příkazy. Příkazy jsou většinou zkratkou slov vystihujících jejich význam (ONEX = *One message transaction only*).

## VERB

Příkaz VERB (*Verbose*) způsobí, že server začne vypisovat podrobný protokol o komunikaci.

### Příklad

```
C:\ > telnet
C: telent> smtp.nextra.cz 25
S: 220 dns.terminal.cz ESMTP
C: verb
S: 250 Verbose mode
C: mail from: fantom@pvt.cz
S: 250 fantom@pvt.cz... Sender ok
C: rcpt to: dostalek@pvt.cz
S: 250 dostalek@pvt.cz... Recipient ok
C: data
S: 354 Enter mail, end with "." on a line by itself
C: test
S: .

S: 050 dostalek@pvt.cz... Connecting to fw.pvt.cz. via esmtp...
S: 050 220 fw.pvt.cz SMTPXD version 141 ready at Thu, 28 Dec 2000 ...
S: 050 >>> EHLO dns.terminal.cz
```

```
S: 050 500 Command unrecognized
S: 050 >>> HELO dns.terminal.cz
S: 050 250 fw.pvt.cz Hello dns.terminal.cz, pleased to meet you
S: 050 >>> MAIL From:<fantom@pvt.cz>
S: 050 250 <fantom@pvt.cz>... Sender ok
S: 050 >>> RCPT To:<dostalek@pvt.cz>
S: 050 250 <dostalek@pvt.cz>... Recipient ok
S: 050 >>> DATA
S: 050 354 Enter mail, end with "." on a line by itself
S: 050 >>> .
S: 050 250 MAA14445 Message accepted for delivery
S: 050 dostalek@pvt.cz... Sent (MAA14445 Message accepted for delivery)
S: 250 MAA43965 Message accepted for delivery
S: 050 Closing connection to fw.pvt.cz.
S: 050 >>> QUIT
S: 050 221 fw.pvt.cz closing connection
```

Jedná se o dialog serveru s následujícím serverem na cestě k adresátovi. Řetězec „>>>“ vyjadřuje „odesílám“.

## BITMIME

Toto rozšíření je určeno pro přenos zpráv MIME (viz kapitola 3) obsahujících osmibitová data. Pokud server nepotvrdí, že podporuje tento osmibitový přenos, SMTP klient nesmí odeslat na server zprávu, jejíž tělo obsahuje jiné znaky než ASCII. Může však před odesláním zprávu kódovat např. Base64, čímž ji převede do sedmibitového tvaru.

Rozšíření 8BITMIME rozšiřuje příkaz MAIL o další parametr BODY=8BITMIME. Příklad:

```
C: MAIL FROM: fantom@peklo.cz BODY=8BITMIME
S: 250 fantom@peklo.cz ... Sender and 8BITMIME ok

C: RCPT TO: dostalek@pvt.cz
S: 250 dostalek@pvt.cz ... Recipient ok

C: DATA
S: 354 Send 8BITMIME message, ending in CRLF.CRLF.
...
C: .
S: 250 OK
```

## SIZE

Toto rozšíření je určeno pro specifikaci délky zprávy. Server ve své odpovědi na příkaz EHLO zpravidla vrací toto rozšíření s číselným parametrem, který desítkově specifikuje, jakou maximálně dlouhou zprávu je ochoten akceptovat (do délky zprávy se započítávají konce řádků, ale nikoliv samotný příkaz DATA).

Následně může klient rozšíření SIZE použít jako další parametr příkazu MAIL, kde může specifikovat, jak dlouhou zprávu posílá. Server si tak může alokovat příslušnou velikost paměti pro uložení

zprávy. Nebo naopak server může tak velkou zprávu odmítnout (např. pro nedostatek paměti) dříve, než se začne s přenosem dat.

## ETRN

Toto rozšíření ocení zejména malé firmy, které mají svůj poštovní server za komutovanou linkou. Příchozí poštovní zprávy zůstávají na poštovním serveru poskytovatele, který se je snaží doručit na poštovní server firmy, jenž je však po většinu doby nedostupný.

V případě, že se firma připojí komutovanou linkou, očekává, že se od poskytovatele okamžitě pohrne příchozí pošta. Avšak nic takového se nekoná. Poštovní agent poskytovatele zprávy odešle, až na ně při procházení frontou přijde řada. A jelikož tam zprávy ležely dlouho, agent u těchto položek mohl značně prodloužit dobu, po uplynutí které se opětovně bude pokoušet je doručit. Nebo dokonce pro uvedenou firmu (tj. uvedenou doménu DNS) je agent nakonfigurován, aby se neobtěžoval položky automaticky doručit a ponechal je ve frontě, až obdrží příkaz ETRN.

Příkaz ETRN má jediný parametr, a to je doména, pro kterou má server nastartovat prohledávání fronty. Je třeba upozornit, že – na rozdíl od příkazu TURN – pokud příkaz ETRN provede útočník, tak se nejvýše nepodaří poštu doručit (fronta se nastartuje zbytečně). V případě použití příkazu TURN by požadovanou poštu obdržel útočník.

Rozšíření ETRN specifikuje RFC-1985.

## Potvrzení o doručení zprávy

Elektronická pošta v Internetu negarantuje doručení zprávy. Z nejrůznějších příčin může být zpráva na své pouti Internetem ztracena. Pro odesilatele zprávy je tak někdy zajímavé si nechat doručení zprávy potvrdit. Takové potvrzení může např. napsat ručně příjemce. Avšak nás zajímá možnost automatizace této činnosti.

Automatizovat notifikaci o doručení poštovní zprávy lze udělat třemi způsoby:

- ◆ Rozšířením ESMTP označovaným **DSN**, které notifikuje doručení poštovní zprávy do schránky uživatele na serveru.
- ◆ Rozšířením MIME realizovaným hlavičkou Disposition-Notification-To. Toto rozšíření notifikuje zobrazení zprávy příjemcem. Zobrazení však neznamená, že by si příjemce zprávu skutečně přečetl nebo jí snad porozuměl.
- ◆ S/MIME umožňuje odeslat notifikaci v případě korektně verifikovaného elektronického podpisu adresátem. Blíže viz publikace Velký průvodce PKI a technologií elektronického podpisu.

Prakticky je rozdíl v těchto mechanismech v tom, že rozšíření DSN interpretuje ESMTP server, tj. ESMTP server generuje notifikační zprávu, kdežto rozšíření MIME je interpretováno až poštovním klientem (např. MS Outlook), tj. notifikační zprávu generuje poštovní klient při otevření zprávy adresátem (S/MIME notifikaci rovněž generuje software klienta).

Skutečnost, zda váš poštovní klient podporuje jedno z těchto rozšíření, či dokonce obě, závisí na tvůrci tohoto softwaru (jedná se přece jenom o rozšíření). V případě použití obou notifikací obdržíte zpět dvě zprávy. První, že zpráva byla doručena do schránky, a druhou, že zpráva byla příjemci zobrazena.

## DSN (Delivery Status Notifications)

Toto rozšíření umožňuje notifikaci doručení zprávy. Rozšiřuje příkaz MAIL protokolu SMTP o dva parametry:

- ◆ Parametr RET sloužící k indikaci, zdali se má v notifikační poštovní zprávě vrátit celý obsah původní zprávy (RET=FULL), nebo jen záhlaví zprávy (RET=HDRS).
- ◆ Parametr ENVID určuje identifikaci zprávy, aby bylo možné párovat notifikace s původními zprávami.

Dále rozšíření DSN rozšiřuje příkaz RCPT také o dva parametry:

- ◆ Parametr NOTIFY specifikuje podmínu, za které se má notifikace generovat. Podmínky mohou být:
  - NOTIFY=NEVER (nikdy neposílat notifikaci)
  - NOTIFY=SUCCESS (zaslat notifikaci v případě úspěšného doručení)
  - NOTIFY=FAILURE (zaslat notifikaci v případě neúspěšného doručení)
  - NOTIFY=DELAY (zaslat notifikaci při zpoždění doručení)
- ◆ Podmínky SUCCESS, FAILURE a DELAY mohou být kombinovány. Např.:
  - NOTIFY=SUCCESS, FAILURE (zaslat notifikaci při neúspěšném i úspěšném doručení).
- ◆ Parametr ORCPT určuje původního (originálního) adresáta. Původní adresa může být změněna např. při předávání zprávy (*forward*). Tento parametr obsahuje dvě hodnoty: typ adresy (např. rfc822) a vlastní adresu. Např.  
ORCPT=rfc822; dostalek@pvt.cz

Notifikuje se v okamžiku zápisu do adresátovy poštovní schránky (INBOX), tj. v okamžiku, po kterém bude zpráva dostupná protokoly POP3 či IMAP4.

### Příklad

Odeslal jsem následující zprávu:

```
C: MAIL FROM: skol00@t1.pvt.cz RET=HDRS ENVID=007
S: 250 <skol00@t1.pvt.cz>... Sender ok
C: RCPT TO: dostalek@ica.cz NOTIFY=SUCCESS,FAILURE
    ORCPT=rfc822;dostalek@pvt.cz
S: 250 <dostalek@ica.cz>... Recipient ok
C: DATA
S: 354 Enter mail, end with do" on a line by itself
```

Text zprávy

```
S: 250 QAA64601 Message accepted for delivery
```

Zpráva dorazila k adresátovi a byla uložena do jeho poštovní schránky. To způsobilo odeslání notifikační e-mailové zprávy odesílateli *skol00@t1.pvt.cz*. Zajímavá je právě notifikační zpráva, která je v následující tabulce. Notifikační zpráva je typu multipart, tj. skládá se z několika samostatných částí:

```
Received: from localhost (localhost)
    by t1.pvt.cz (8.9.3/8.9.3) with internal id QAA31130;
    Thu, 28 Dec 2000 16:20:57 +0100 (MET)
```

Date: Thu, 28 Dec 2000 16:20:57 +0100 (MET)  
From: Mail Delivery Subsystem <MAILER-DAEMON>  
Message-Id: <200012281520.QAA31130@t1.pvt.cz>  
To: <sko100@t1.pvt.cz>  
MIME-Version: 1.0  
Content-Type: multipart/report; report-type=delivery-status;  
boundary="QAA31130.978016857/t1.pvt.cz"  
Subject: Return receipt  
Auto-Submitted: auto-generated (return-receipt)  
This is a MIME-encapsulated message

*-QAA31130.978016857/t1.pvt.cz*  
The original message was received at Thu, 28 Dec 2000 16:20:57 +0100  
(MET)  
from server.ica.cz [195.47.13.11]

— The following addresses had successful delivery notifications—  
<dostalek@t1.pvt.cz> (successfully delivered to mailbox)

— Transcript of session follows —  
<dostalek@t1.pvt.cz>... Successfully delivered

*-QAA31130.978016857/t1.pvt.cz*  
Content-Type: message/delivery-status

Original-Envelope-Id: 007  
Reporting-MTA: dns; t1.pvt.cz  
Received-From-MTA: DNS; server.ica.cz  
Arrival-Date: Thu, 28 Dec 2000 16:20:57 +0100 (MET)

Original-Recipient: rfc822;dostalek@pvt.cz  
Final-Recipient: RFC822; <dostalek@t1.pvt.cz>  
Action: delivered (to mailbox)  
Status: 2.1.5  
*-QAA31130.978016857/t1.pvt.cz*  
Content-Type: text/rfc822-headers

Return-Path: <sko100@t1.pvt.cz>  
Received: from server.ica.cz (server.ica.cz [195.47.13.11])  
by t1.pvt.cz (8.9.3/8.9.3) with ESMTP id QAA30941  
for <dostalek@t1.pvt.cz>; Thu, 28 Dec 2000 16:20:57 +0100 (MET)  
Received: from dns.terminal.cz (dns.terminal.cz [195.70.130.1])  
by server.ica.cz (8.9.2/8.8.7) with ESMTP id QAA25234  
for <dostalek@ica.cz>; Thu, 28 Dec 2000 16:20:26 +0100 (CET)  
From: sko100@t1.pvt.cz Received: from [195.47.37.200] ([195.47.37.200])  
by dns.terminal.cz (8.9.3/8.9.3) with SMTP id QAA64601  
for dostalek@ica.cz; Thu, 28 Dec 2000 16:20:18 +0100 (CET)  
Date: Thu, 28 Dec 2000 16:20:18 +0100 (CET)  
Message-Id: <200012281520.QAA64601@dns.terminal.cz>

*-QAA31130.978016857/t1.pvt.cz-*

Tuto zprávu Content-Type: multipart/report; report-type=delivery-status specifikuje RFC-1984. Jádrem této zprávy je druhá část Content-Type: message/delivery-status, kterou pro zdůvodnění ještě zopakuji:

```
Content-Type: message/delivery-status
Original-Envelope-Id: 007
Reporting-MTA: dns; t1.pvt.cz
Received-From-MTA: DNS; server.ica.cz
Arrival-Date: Thu, 28 Dec 2000 16:20:57 +0100 (MET)

Original-Recipient: rfc822;dostalek@pvt.cz
Final-Recipient: RFC822; <dostalek@t1.pvt.cz>
Action: delivered (to mailbox)
Status: 2.1.5
```

Tělo zprávy obsahuje:

1. Informace o zprávě:

Original-Envelope-Id: ... Původní identifikace zprávy zadaná parametrem ENVID.  
Reporting-MTA: ... Poštovní agent, který provedl vlastní doručení; obsahuje typ jména a jméno.  
Received-From-MTA: ... Poštovní agent, ze kterého zpráva přišla (předposlední agent).  
Arrival-Date: ... Datum a čas doručení zprávy do poštovní schránky příjemce.

2. Informace o příjemci a o doručení zprávy:

Original-Recipient: ... Původní adresát, jak jej specifikoval odesílatel.  
Final-Recipient: ... Adresát, kterému byla zpráva doručena ve skutečnosti.  
Action: ... Akce specifikující, jak to s doručením zprávy dopadlo:  
Failed – Zpráva nebyla doručena.  
Delayed – Zpráva se zpozdila.  
Delivered – Zpráva byla doručena.  
Relayed – Zpráva byla předána bránou do jiného poštovního systému.  
Expanded – Zpráva byla rozeslána seznamu uživatelů (adresou příjemce byl seznam – tj. alias či mail list).  
Status: ... Obsahuje informaci (tři čísla oddělená tečkou) o doručení zprávy (viz RFC-1893). První číslo indikuje, jestli byla zpráva doručena (2 = doručena, 4 = nedoručena pro dočasnou chybu, 5 = nedoručena pro neodstranitelnou chybu). Druhé číslo specifikuje příčinu potíží při doručení (1 = adresa, 2 = poštovní schránka, 3 = poštovní systém, 4 = síť, 5 = poštovní protokol, 6 = typ média, 7 = bezpečnost). Třetí číslo podrobně specifikuje konkrétní příčinu. Podrobný seznam stavových kódů viz RFC-1893.

## POP3

*Post Office Protocol* verze 3 je jednoduchý protokol, kterým si uživatel může ze své poštovní schránky na poštovním serveru stáhnout zprávy do lokálních poštovních schránek na svém PC. Je určen pro práci offline s poštovním serverem. POP3 je specifikován v RFC-1939.

Klient navazuje spojení na TCP, port 110 serveru. Po navázání spojení se server představí a je v tzv. autentizačním stavu, tj. čeká na autentizaci uživatele. Např.:

```
+OK QPOP (version 2.1.4-R4-b5a) at t1.pvt.cz starting. <3774.978040846@t1.pvt.cz>
```

Základní autentizace je jménem a heslem uživatele. V případě, že autentizace proběhne s kladným výsledkem, komunikace přechází do transakčního stavu, kdy klient může pracovat se zprávami ve své poštovní schránce na serveru. I když klient např. zruší některé zprávy ve své poštovní schránce na serveru, během transakčního stavu je takové zrušení ještě odvolatelné. Na závěr relace přejde klient do stavu UPDATE, kdy se provedou veškeré změny v jeho poštovní schránce na serveru trvale.

Server na zadávané příkazy vrací odpověď začínající buď znakem + v případě, že je kladná, nebo znakem – v případě signalizace chyby.

Jelikož se příkazy zadávají v ASCII, je komunikace s POP3-serverem snadná i pomocí programu Telnet.

### Příklad

Nejprve si však probereme jednotlivé příkazy (jsou vždy čtyřznakové):

1. Autentizační stav:

Příkazem USER zadává uživatel své jméno. Příklad:

```
C: USER dostalek
```

```
S: +OK Password required for dostalek.
```

Příkazem PASS zadává uživatel své heslo (příkaz je volitelný). Příklad:

```
C: PASS heslo
```

```
S: +OK dostalek has 2 message(s) (3605 octets).
```

(Tj. přihlášení proběhlo dobře; na serveru máš 2 zprávy dlouhé dohromady 3 605 bajtů.)

Příkazem QUIT může uživatel ukončit spojení.

2. Transakční stav:

Příkaz STAT udává počet zpráv v poštovní schránce a celkovou velikost poštovní schránky:

```
C: STAT
```

```
+OK 2 3605
```

Příkaz LIST vrací seznam jednotlivých zpráv v poštovní schránce (co zpráva, to jeden řádek).

U každé zprávy je uvedeno její pořadové číslo a její velikost:

```
C: LIST
```

```
S: +OK 2 messages (3605 octets)
```

```
S: 1 1196
```

```
S: 2 2409
```

Příkazem RETR stahujeme zprávu ze serveru na PC; jako parametr se použije číslo zprávy, která se má stahovat:

```
C : RETR 2
```

Příkazem DELE se ruší zpráva v poštovní schránce na serveru. Jako parametr se zadává číslo rušené zprávy.

```
C: DELE 2
```

```
S: +OK Message 2 has been deleted.
```

NOOP je prázdný příkaz:

C: NOOP  
S: +OK

Příkazem RSET lze opět oživit zprávy zrušené během aktuální relace:

C: RSET  
S: +OK Maildrop has 2 messages (3605 octets)

Příkazem TOP lze vypsat počátek zprávy. Syntaxe je:

TOP číslo\_zprávy počet\_řádek těla zprávy.

### 3. Stav UPDATE:

Ukončení relace se provede opět příkazem QUIT. V tomto okamžiku se provede fyzické zrušení všech zpráv označených jako zrušené.

## Příklad

Příklad (stažení zprávy ze serveru):

```
C:\ > telnet t1.pvt.cz 110
S: +OK QPOP (version 2.1.4-R4-b5a) at t1.pvt.cz starting.
<3774.978040846@t1.pvt.cz>

C: USER dostalek
S: +OK Password required for dostalek.

C: PASS heslo
S: +OK dostalek has 2 message(s) (3605 octets).

C: LIST
S: +OK 2 messages (3605 octets)
S: 1 1196
S: 2 2409

C: RETR 2
S: +OK 1196 octets
S: X-UIDL: b27991db2a4199a85f593d76b58338c7
S: Received: from dns.terminal.cz (dns.terminal.cz [195.70.130.1])
by t1.pvt.cz (8.9.3/8.9.3) with ESMTP id XAA04273
for <dostalek@t1.pvt.cz>; Thu, 28 Dec 2000 23:21:10 +0100 (MET)

... záhlaví a text zprávy
```



**Poznámka:** Opět si to nezapomeňte vyzkoušet vůči vašemu serveru POP3!

Ačkoliv existuje celá řada možných rozšíření protokolu POP3, bývají servery POP3 v praxi koncipovány často jako velice jednoduché servery umožňující, aby v jednom okamžiku byl ke své poštovní schránce přihlášen uživatel protokolem POP3 pouze jednou. Po přihlášení uživatele se uživatelská schránka na serveru zkopíruje. Původní schránka zůstane a mohou do ní přicházet další e-maily např. protokolem SMTP. S kopíí schránek pracuje protokol POP3. Režim UPDATE je právě ten okamžik, kdy dojde ke zpětnému slití obou poštovních schránek do jedné.

Při přihlášení uživatele k serveru POP3 se proto nejprve testuje, zda již neexistuje kopie poštovní schránky. V případě, že ano, pak se předpokládá, že uživatel pracuje protokolem POP3 se svou schránkou, a takový požadavek na přihlášení je odmítnut. Proto když se uživatel nemůže k serveru POP3 přihlásit, nejprve zkонтrolujeme, zda nějakým omylem nezůstala zachována kopie poštovní schránky z minulého přihlášení uživatele.

Jiný problém u serverů POP3 vzniká v okamžiku, kdy chceme na operačním systému serveru aktivovat bezpečnostní režim C2 (či dokonce B). Po takové aktivaci někdy přestane server POP3 pracovat (nikdo se k němu nepřihlásí). Problém spočívá v tom, že server POP3 musí najít hesla uživatelů, aby je mohl prověřovat. A právě při přechodu k bezpečnostnímu režimu C2 může v operačním systému dojít k uložení hesel na „bezpečnější místo“. Server POP3, který neumí pracovat s tímto „bezpečnějším“ uložením hesel, není schopen přihlásit jediného uživatele.

## IMAP4

Protokol IMAP (Internet Message Access Protocol) verze 4 je specifikován RFC-3501. Tato specifikace je někdy též označována jako IMAP4rev1, protože je revizí původní normy protokolu IMAP4, specifikované v RCF-1730.

IMAP4 je sofistikovaný protokol určený pro práci s poštovními schránkami z PC na serveru v režimu online (i offline). V jednom okamžiku můžeme pracovat se svými poštovními schránkami z více aplikací. Dokonce některé aplikace navazují současně dvě TCP spojení se serverem IMAP4 (např. MS Outlook): jedno pro práci s poštovními schránkami a druhé pro práci s jednotlivými položkami (poštovními zprávami). Server protokolu IMAP používá port 143/tcp.

Během práce s poštovní schránkou (během navázaného TCP spojení) může jiná aplikace změnit obsah poštovní schránky (např. SMTP server zapíše do schránky nově příchozí poštu). Tyto události (jako připsání nových zpráv na konec poštovní schránky) jsou do navázaného spojení serverem signalizovány. Prakticky může klient zjišťovat příchod nové zprávy tak, že na server odešle prázdný příkaz NOOP. To server vybudí k akci podívat se, zda nedošlo ke změnám v poštovní schránce. Pokud ano, tak na klienta vysype příslušné změny.

Pokud má jedna aplikace otevřenu poštovní schránku pro čtení a zápis a jiná aplikace si chce otevřít schránku rovněž pro čtení a zápis, musí být první aplikaci otevření schránky změněno pouze pro čtení. První aplikaci musí být okamžitě signalizováno, že nadále má schránku otevřenu pouze pro čtení, a pokud chce provést změnu v poštovní schránce, musí si ji znova otevřít. Takto pracovat s poštovními schránkami zpravidla umí server protokolu IMAP4, ale co jiné aplikace? Aby mohlo několik aplikací pracovat s jednou poštovní schránkou, nesmí si aplikace vzájemně „škodit“, jinak dojde k havárii a server IMAP4 ukončí TCP spojení. Nekorektní jednání lze často simulovat např. tím, že schránku otevřenou pro IMAP4 otevřeme např. programem mail na serveru a provedeme v ní nějaké změny – např. zrušíme některou ze zpráv (obdobně by to dopadlo, kdybychom chtěli schránku současně otevřít protokolem IMAP4, POP3).

Nejdříve však musím popsat, jak vůbec příkazy protokolu IMAP4 vypadají. Poté co klient naváže TCP -spojení se serverem na portu 143, server se představí, např. programem Telnet ve Windows Vista:

```
C:\ WINNT\ System32> telnet  
C: telnet> server.firma.cz 143  
S: * OK server.firma.cz IMAP4rev1 v10.170 server ready
```

Nyní může klient zadávat příkazy. Příkazy se opět zadávají v ASCII (obdobně jako v případě protokolů NVT, SMTP, POP3 apod.). Příkazy jsou však odlišné od příkazů protokolu POP3. Odlišnost není jen v názvech příkazů, ale zejména v jejich filozofii. V protokolu IMAP4 může být zadávána řada příkazů a odpovědi na ně mohou přicházet ze serveru v libovolném pořadí. Proto klient příkazy čísluje a server do své odpovědi zopakuje číslo příkazu, na který odpovídá. Je věci klienta, jak příkazy identifikuje (čísluje). Identifikace příkazu je obecně řetězec (nemusí to být ani číslo). Jednoznačnost identifikace je rovněž věcí klienta.

Po navázání spojení si můžeme ukázat formát příkazů a odpovědí na jednoduchém příkazu CAPABILITY, kterým se klient ptá serveru, jaké funkčnosti umí server poskytnout:

```
C: 0000 CAPABILITY
S: * CAPABILITY IMAP4 IMAP4REV1 SCAN SORT AUTH=LOGIN
S: 0000 OK CAPABILITY completed
```

Klient očísloval svůj dotaz řetězcem 0000, za kterým následuje vlastní příkaz (v našem případě příkaz CAPABILITY). Za příkazem mohou následovat jeho parametry, avšak příkaz CAPABILITY parametry nemá.

Server odpovídá dvěma druhy odpovědí:

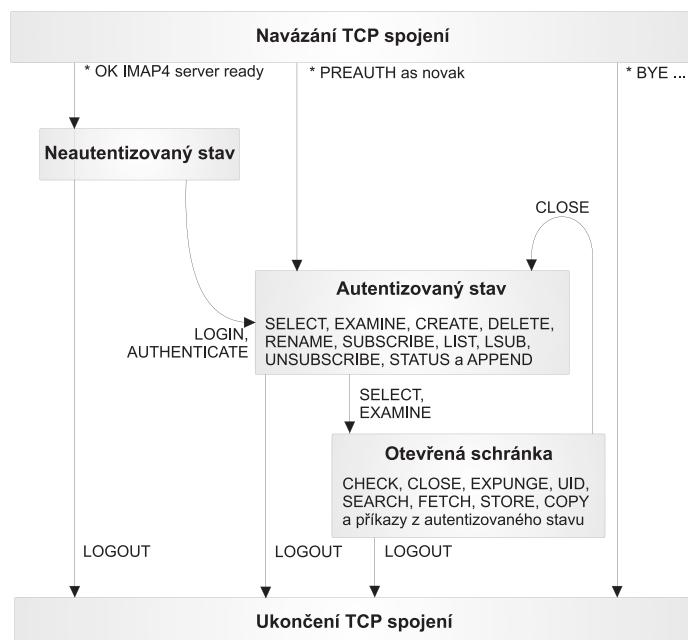
1. Nečíslovanými odpověďmi, které mají na místě čísla příkazu/odpovědi hvězdičku. Tyto nečíslované řádky v podstatě nesou informaci, kterou klient požadoval. V našem případě server podporuje: protokol IMAP4, protokol IMAP4 revize 1, rozšíření SCAN a SORT a autentizaci uživatele příkazem LOGIN (tj. jménem a heslem).
  2. Číslovanými odpověďmi, které začínají číslem příkazu a sdělují, jak příkaz dopadl. Číslovaná odpověď se skládá z:
    1. Čísla příkazu.
    2. Výsledku:
      - a) OK – příkaz proběhl úspěšně,
      - b) NO – příkaz proběhl neúspěšně,
      - c) BAD – chybný příkaz, např. pro syntaktickou chybu příkazu,
      - d) výsledkem PREAUTH může server po navázání spojení signalizovat, že klient je přihlášen jako konkrétní uživatel bez nutnosti se přihlásit příkazy LOGIN či AUTHENTICATE,
      - e) výsledek BYE je naopak vrácen v případě, že server se s klientem nechce již dále bavit.  
Např. klient je příliš dlouho přihlášen a je nečinný:  
\* BYE Autologout; idle for too long
    - nebo tento výsledek může být i součástí odhlašovací sekvence při ukončování spojení:
- ```
C: 4 LOGOUT
S: * BYE p30x01 IMAP4rev1 server terminating connection
S: 4 OK LOGOUT completed
```
- či naopak na server se pokusil přihlásit klient, který je na černé listině, proto mu server místo svého zdvořilého představení pošle výsledek BYE apod.
3. Další upřesňující textové informace.

Na obr. 17.5 jsou popsány jednotlivé stavy v protokolu IMAP4. Po navázání spojení zpravidla nastane „neautentizovaný stav“, kdy je nutná autentizace klienta (nebo ukončení příkazem LOGOUT).

Výjimkou je situace, kdy server okamžitě po navázání klientovi sdělí, že je již předem s přihlášením automaticky autentizován (PREAUTH). V praxi jsem se však s předautentizací nesetkal.

V autentizovaném stavu může klient pracovat s poštovními schránkami na serveru jako se soubory, tj. může poštovní schránku vytvořit, zrušit, přejmenovat apod. Příkazem SELECT (nebo EXAMINE) klient otevře konkrétní poštovní schránku a přejde do režimu „Otevřená schránka“, ve kterém může pracovat s jednotlivými položkami otevřené poštovní schránky (např. položky přenést ze serveru na klienta apod.).

Příkazy CAPABILITY, NOOP a LOGOUT jsou na stavu nezávislé, proto je možné je zadávat kdykoliv.



Obrázek 17.5: Stavy v protokolu IMAP4

## Neautentizovaný stav

### LOGIN

Příkazem LOGIN se uživatel přihlašuje pomocí jména a hesla. Po úspěšném provedení příkazu LOGIN se přejde do autentizovaného stavu.

C: 1 login uživatel heslo  
S: 1 OK LOGIN completed

Pro jiné mechanismy autentizace než jménem a heslem (např. autentizace systémem Kerberos) se používá příkaz AUTHENTICATE. Obecně klient jako parametr příkazu AUTHENTICATE předá autentizační mechanismus, který navrhuje použít pro svou autentizaci. V případě, že server tento mechanismus podporuje, odpoví řádkem začínajícím znakem „+“, sdělujícím „tak pokračuj“

v autentizaci“. Klient pak odpovídá nějakou autentizační informací. V případě, že autentizační dialog vyžaduje další komunikaci mezi serverem a klientem, server opět odpovídá řádkem začínajícím znakem „+“. Veškeré autentizační informace přenášené mezi klientem a serverem jsou kódovány Base64. Příklad této komunikace je uveden v části 4.2.4.

## Autentizovaný stav

### Příkazy CREATE, DELETE, RENAME a LIST

Příkazem CREATE se vytváří poštovní schránka jako soubor, příkazem DELETE se ruší, příkazem RENAME se přejmenovává a příkaz LIST slouží k výpisu adresáře.

```
C: 2 LIST "" "*"
S: * LIST (\ NoInferiors) "/" .profile
S: * LIST (\ NoInferiors \ UnMarked) "/" .login
S: * LIST (\ NoInferiors \ UnMarked) "/" .cshrc
S: * LIST (\ NoInferiors) NIL INBOX
S: 2 OK LIST completed
```

Výpisem adresáře v operačním systému Unix lze zjistit přibližně totéž (INBOX je „systémová“ poštovní schránka – v méém případě /var/spool/mail/dostalek):

```
$ ls -a
... .cshrc .login .profile
```

Syntaxe příkazu LIST je poněkud nezvyklá. Příkaz LIST má dva parametry. První parametr bych nazval cestou a druhý jménem poštovní schránky. Ve jméně poštovní schránky mohou být znaky hromadného výběru hvězdička a procento. Hvězdička rozvíjí vše, ale procento pouze jméno poštovní schránky – nikoliv adresářovou strukturu (nebo chcete-li strukturu poštovních schránek). Příklad rozdílu je:

### Příklad

```
C: 1 list "" "%"
S: * LIST (\ NoInferiors) "/" .profile
S: * LIST (\ NoInferiors \ UnMarked) "/" .login
S: * LIST (\ NoInferiors \ UnMarked) "/" .cshrc
S: * LIST (\ NoSelect) "/" mail
S: * LIST (\ NoInferiors) NIL INBOX
S: 1 OK LIST completed

C: 1 list "" "*"
S: * LIST (\ NoInferiors) "/" .profile
S: * LIST (\ NoInferiors \ UnMarked) "/" .login
S: * LIST (\ NoInferiors \ UnMarked) "/" .cshrc
S: * LIST (\ NoSelect) "/" mail
S: * LIST (\ NoInferiors) "/" mail/schranka2
S: * LIST (\ NoInferiors) NIL INBOX
S: 1 OK LIST completed
```

Odpověď serveru na příkaz LIST obsahuje na každém řádku informace o jednom vypisovaném souboru skládající se ze tří údajů:

1. Příznaky, které jsou uzavřeny v kulatých závorkách. Příznaky máme:
  - \ Noinferiors – nejdá se o adresář, tj. ve struktuře poštovních schránek nemůže pod touto položkou existovat další položka.
  - \ Noselect – tento soubor nelze otevřít jako poštovní schránku.
  - \ Marked – tento soubor je označen, že by případně mohl být otevřen jako poštovní schránka.
  - \ Unmarked – tento soubor neobsahuje žádné označení, že by případně mohl být poštovní schránkou.
2. Oddělovače v hierarchii poštovních schránek (např. v Unixu se v cestě k souboru používá lomítko jako oddělovač mezi názvy adresářů). Slovo NIL znamená, že žádná hierarchie neexistuje.
3. Jméno poštovní schránky (jméno souboru).

Příkazem CREATE si v domovském adresáři („~dostalek“) vytvořím podadresář mail a v něm poštovní schránky schranka1 a schranka2:

```
C: 3 CREATE "~dostalek/mail/schranka1"
S: 3 OK CREATE completed
```

```
C: 4 create "~dostalek/mail/schranka2"
S: 4 OK CREATE completed
```

Příkazem LIST opět mohu ověřit, zdali byla schránka vytvořena:

```
C: 5 list "mail" "*"
S: * LIST (\ NoSelect) "/" mail
S: * LIST (\ Noinferiors) "/" mail/schranka1
S: * LIST (\ Noinferiors) "/" mail/schranka2
S: 5 OK LIST completed
```

Těhož výpisu lze dosáhnout příkazem operačního systému:

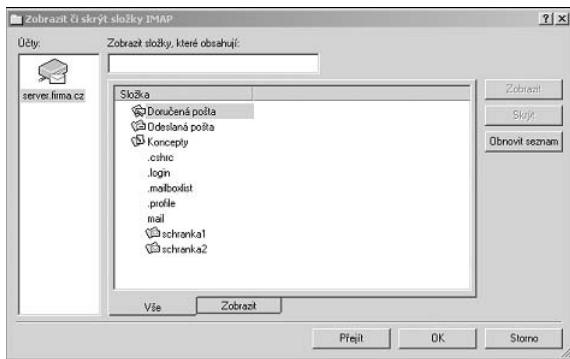
```
$ ls mail/*
mail/schranka1 mail/schranka2
```

Celou mou adresářovou strukturu lze vypsat příkazem:

```
1 OK LOGIN completed
1 list "" "*"
* LIST (\ Noinferiors) "/" .profile
* LIST (\ Noinferiors \ UnMarked) "/" .login
* LIST (\ Noinferiors \ UnMarked) "/" .cshrc
* LIST (\ NoSelect) "/" mail
* LIST (\ Noinferiors \ Marked) "/" mail/schranka1
* LIST (\ Noinferiors \ Marked) "/" mail/schranka2
* LIST (\ Noinferiors \ UnMarked) "/" .mailboxlist
* LIST (\ Noinferiors \ UnMarked) "/" "Odeslan&AOE- po&AWE-ta"
* LIST (\ Noinferiors \ Marked) "/" Koncepty
* LIST (\ Noinferiors) NIL INBOX
1 OK LIST completed
```

S takovýmto výpisem by však běžný uživatel nebyl spokojen, proto mu jej uživatelská poštovní aplikace zobrazí v přívětivější formě. Na obrázku 17.6 je příklad grafické interpretace našeho výstupu v okně

aplikace MS Outlook Express (MS Outlook 2000 má obdobné okno). Jen je třeba upozornit, že poštovní schránka INBOX se vcelku logicky na obrázku neoznačuje „INBOX“, ale „Doručená pošta“.



**Obrázek 17.6:** Výpis MS Outlook Express

Příkazem RENAME mohu soubor mail/schránka2 přejmenovat např. na mail/schránka3:

```
C: 6 RENAME mail/schránka2 mail/schránka3
S: 6 OK RENAME completed
```

Konečně příkazem DELETE mohu schránku zrušit:

```
C: 7 DELETE mail/schránka3
S: 7 OK DELETE completed
```

Nakonec si opět zřídíme schránku mail/schránka2, abychom pro další výklad měli na serveru alespoň dvě poštovní schránky.

### Příkazy SUBSCRIBE, LSUB, UNSUBSCRIBE

Příkazem SUBSCRIBE klient sdělíl serveru, aby označil soubor jako poštovní schránku.

```
C: 1 subscribe mail/schránka1
S: 1 OK SUBSCRIBE completed
```

Příkazem LIST si můžeme vypsat, zdali je schránka1 opravdu označena jako poštovní schránka:

```
C: 2 list mail *
S: * LIST (\ NoSelect ) "/" mail
S: * LIST (\ NoInferiors \ Marked ) "/" mail/schránka1
S: * LIST (\ NoInferiors ) "/" mail/schránka2
S: OK LIST completed
```

Implementace našeho serveru si pro informace o tom, která schránka je označena, vytvoří soubor .mailboxlist v domovském adresáři uživatele, což můžeme ověřit výpisem adresáře:

```
$ ls -a
. .cshrc .mailboxlist mail ...
```

Obdobně si označíme i schránku schránka2. Poté můžeme vypsat obsah souboru .mailboxlist:

```
$ cat .mailboxlist
mail/schránka1
mail/schránka2
```

Příkaz UNSUBSCRIBE slouží ke zrušení označení schránky. Příkaz LSUB je obdobou příkazu LIST, ale vypisuje jen označené schránky:

```
C: 3 lsub "mail" "*"
S: * LSUB () "/" mail/schranka1
S: * LSUB () "/" mail/schranka2
S: 3 OK LSUB completed
```

## STATUS

Příkazem STATUS lze získat informace o poštovní schránce, aniž by schránka byla otevřena. Příkaz STATUS má dva parametry: prvním parametrem je jméno poštovní schránky a druhým parametrem v kulatých závorkách uvedený seznam stavů položek.

Stavy položek v poštovní schránce jsou:

- ◆ MESSAGES – server vrátí počet zpráv v poštovní schránce.
- ◆ RECENT – server vrátí počet položek majících příznak \ Recent.
- ◆ UIDNEXT – server vrátí číslo zprávy, která bude přiřazena nové zprávě v poštovní schránce (až zpráva přijde).
- ◆ UIDVALIDITY – server vrátí jednoznačnou identifikaci UID poštovní schránky jako takové.
- ◆ UNSEEN – server vrátí počet položek nemajících příznak \ Seen.

V následujícím příkladu se pro změnu nepodíváme na stav privátní poštovní schránky, ale schránky INBOX (/var/spool/mail/dostalek):

### Příklad

```
C: 4 STATUS INBOX (MESSAGES RECENT UIDNEXT UIDVALIDITY UNSEEN)
S: * STATUS INBOX (MESSAGES 6 RECENT 0 UNSEEN 4 UIDNEXT 7
    UIDVALIDITY 978588855)
S: 4 OK STATUS completed
```

## Příkazy SELECT a EXAMINE

Příkazem SELECT otevřeme poštovní schránku, tj. přejdeme do režimu „Otevřená poštovní schránka“. Příkaz EXAMINE je obdobou příkazu SELECT, otevírá však poštovní schránku pouze pro čtení. Jako parametr se uvede jméno poštovní schránky:

### Příklad

```
C: 5 select INBOX
S: * 6 EXISTS
S: * OK [UIDVALIDITY 978588855] UID validity status
S: * FLAGS (\ Answered \ Flagged \ Deleted \ Draft \ Seen)
S: * OK [PERMANENTFLAGS (\ Answered \ Flagged \ Deleted \ Draft \ Seen)]
    Permanent flags
S: * OK [UNSEEN 3] 3 is first unseen
S: * 0 RECENT
S: 5 OK [READ-WRITE] SELECT completed
```

Odpověď serveru je složitější – skládá se z několika druhů rádků:

- ◆ Řádek „6 EXITS“ říká, že právě otevřená poštovní schránka obsahuje 6 zpráv.
- ◆ Řádek „OK [UIDVALIDITY...“ sděluje jednoznačnou identifikaci poštovní schránky.
- ◆ Řádek „FLAGS...“ popisuje minimální množství příznaků, které udržuje poštovní schránka o svých zprávách:
  - \ Seen – zpráva byla přečtena.
  - \ Answered – na zprávu bylo odpovězeno.
  - \ Flagged – zpráva je označena jako urgentní.
  - \ Deleted – zpráva je označena jako zrušená. Zrušení lze provést příkazem EXPUNGE.
  - \ Draft – zpráva ještě není dokončená.
  - \ Recent – nová zpráva. Při příštím otevření poštovní schránky již tato zpráva nebude označena příznakem \ Recent.
- ◆ Řádek „OK [PERMANENTFLAGS...“ vypisuje příznaky, které u zpráv může měnit klient a zůstanou u zpráv změněny trvale, tj. i po ukončení relace.
- ◆ Řádek „OK [UNSEEN 3]...“ specifikuje číslo první nepřečtené zprávy v poštovní schránce, tj. zprávy před touto zprávou jsou již všechny přečteny.
- ◆ Řádek „0 RECENT“ udává počet zpráv s příznakem \ Recent.

## Otevřená schránka

Každá poštovní schránka má svou jednoznačnou identifikaci, kterou server vrací v parametru UIDVALIDITY. Tato identifikace je např. nezávislá na jménu poštovní schránky jako souboru.

Obdobně každá zpráva v poštovní schránce má rovněž svou identifikaci, která je též jednoznačně garantována. Identifikační sekvence je rostoucí. Spojením identifikace poštovní schránky s identifikací zprávy vznikne jednoznačná identifikace zprávy v systému.

Tato jednoznačná identifikace je sice garantována, ale není příliš praktická. Zprávy se proto sekvenčně číslují od jedničky v rámci poštovní schránky. Dojde-li tedy např. ke zrušení zprávy v poštovní schránce příkazem EXPUNGE, musí dojít i k přečíslování zpráv v poštovní schránce.

Poštovní schránky se u nás nazývají i jmény s českou diakritikou, např. „Odeslaná pošta“. Má-li se vytvořit takováto schránka na serveru jako soubor, pak to některým operačním systémům může vadit. Proto pro znaky, které nejsou ASCII, server kóduje Base64. Postup je takový, že znak & znamená přechod do Base64 a znak – zase přechod do ASCII. Například pro „Odeslaná pošta“ vznikne soubor „Odeslan&AOE- po&AWE-ta“.

## COPY

Příkaz COPY zkopíruje zprávy z otevřené poštovní schránky do schránky uvedené jako druhý parametr. Prvním parametrem je číslo přesouvané zprávy. Místo jedné zprávy můžeme uvést interval zpráv. Např. 3:6 je interval zpráv číslo 3, 4, 5 a 6. Následující příklad kopíruje zprávy 3 až 6 do poštovní schránky mail/schranka1:

### Příklad

```
C: 3 COPY 3:6 mail/schranka1
S: 3 OK COPY completed
```

## SEARCH

Příkazem SEARCH lze v poštovní schránce vyhledat zprávy. Vyhledává se podle vyhledávacích kritérií, která se uvádějí jako parametr příkazu SEARCH. Pokud je uvedeno více vyhledávacích kritérií, vyhledávají se zprávy, které vyhovují všem kritériím současně (AND). Je však třeba dodat, že existuje i kritérium OR, které vyhodnocuje dílčí kritéria operací OR. Obdobně existuje i kritérium negace NOT.

Kritéria:

Interval zpráv, např. 3:6.

ALL – všechny zprávy v poštovní schránce.

ANSWERED – všechny zprávy s nastaveným příznakem \ Answered.

DELETED – všechny zprávy s nastaveným příznakem \ Deleted.

DRAFT – všechny zprávy s nastaveným příznakem \ Draft.

FLAGGED – všechny zprávy s nastaveným příznakem \ Flagged.

SEEN – všechny zprávy s nastaveným příznakem \ Seen.

NEW – všechny zprávy s nastaveným příznakem \ Recent, ale nemají nastaven příznak \ Seen.

RECENT – všechny zprávy s nastaveným příznakem \ Recent.

UNANSWERED – všechny zprávy, jež nemají nastaven příznak \ Answered.

UNDELETED – všechny zprávy, jež nemají nastaven příznak \ Delete.

UNDRAFT – všechny zprávy, jež nemají nastaven příznak \ Draft.

UNFLAGGED – všechny zprávy, jež nemají nastaven příznak \ Flagged.

UNSEEN – všechny zprávy, jež nemají nastaven příznak \ Seen.

BODY řetězec – všechny zprávy, jejichž tělo obsahuje uvedený řetězec.

TO řetězec – všechny zprávy obsahující v hlavičce To uvedený řetězec.

CC řetězec – všechny zprávy obsahující v hlavičce Cc uvedený řetězec.

BCC řetězec – všechny zprávy obsahující v hlavičce Bcc uvedený řetězec.

SUBJECT řetězec – všechny zprávy obsahující v hlavičce Subject uvedený řetězec.

FROM řetězec – všechny zprávy obsahující v hlavičce From uvedený řetězec.

HEADER hlavička: řetězec – všechny zprávy mající v uvedené hlavičce uvedený řetězec.

ON datum – všechny zprávy odeslané v uvedeném datu.

BEFORE datum – všechny zprávy, jejichž interní datum je starší než uvedené datum.

SINCE datum – všechny zprávy, jejichž interní datum je stejné nebo mladší než uvedené datum.

SENTBEFORE datum – všechny zprávy odeslané před uvedeným datem.

SENTON datum – všechny zprávy odeslané v uvedeném datu.

SENTSINCE datum – všechny zprávy odeslané v uvedeném datu nebo po něm.

LARGER velikost – všechny zprávy, jejichž velikost je větší, než je uvedený počet bajtů.

SMALLER velikost – všechny zprávy, jejichž velikost je menší, než je uvedený počet bajtů.

UID id – všechny zprávy o uvedené jednoznačné identifikaci.

NOT – negace vyhledávacího kritéria.

OR kritérium1 kritérium2 – operátor OR.

### Příklad

```
C: 789 SEARCH UNSEEN NOT FROM dostalek SINCE 4-Jan-2000
S: * SEARCH 7 8
S: 789 OK SEARCH completed
Zprávy 7 a 8 vyhovují zadaným kritériím.
```

### FETCH

Příkazem FETCH lze ze serveru stáhnout zprávu nebo její část. Příkaz FETCH má syntaxi:

```
FETCH zprávy (co)
```

Příčemž „zprávy“ je buď číslo zprávy, nebo interval zpráv (např. 2:93) a „co“ říká, jaká část zprávy má být podána. V kulatých závorkách můžeme specifikovat řadu informací, které mají být serverem podány. Např.:

### Příklad

- ◆ BODY[sekce] – server vrátí obsah sekce zprávy uvedené v hranatých závorkách. Sekce jsou HEADER (záhlaví), HEADER.FIELDS (hlavičky, jejichž jména jsou uvedena v kulatých závorkách), HEADER.FIELDS.NOT, MIME a TEXT (text zprávy). Např.

```
C: 89 FETCH 9 BODY[HEADER.FIELDS (FROM DATE)]
S: * 9 FETCH (BODY[HEADER.FIELDS ("FROM" "DATE")] { 83 }
S: Date: Fri, 5 Jan 2001 15:28:39 +0100
S: From: Libor Dostalek test user <dostalek>
S:
S: )
S: 89 OK FETCH completed
```

Příkaz FETCH vrátil ze zprávy o pořadovém čísle 9 hlavičky Date a From. Data odeslaná ze serveru jsou uzavřena v kulatých závorkách (za řetězcem „FETCH“). Na prvním řádku vrácených dat je zopakováno, která data server vrací „BODY[HEADER.FIELDS („FROM“ „DATE“)]“, následováno řetězcem „{ 83 }“. Ve složených závorkách se uvádí délka následujících dat, která se nevešla na tento řádek, tj. dále je posíláno 83 bajtů dat a vše je ukončeno na konci zasílaných dat pravou kulatou závorkou.

- ◆ BODY[sekce]<od.do> je obdobou předchozí možnosti, kde ve špičatých závorkách je navíc uvedeno, jak velké množství dat se má přenést. Následující příklad přenese prvních 20 bajtů z datové části zprávy číslo 6:

```
C: 90 FETCH 6 BODY[TEXT]<1.20>
S: * 6 FETCH (BODY[TEXT]<1> { 20 }
S: 20 bajtů zprávy
S: )
S: 90 OK FETCH completed
```

- ◆ BODY.PEEK[sekce]<od.do> je obdobou BODY s tím, že nevrací zprávy s příznakem \ Seen.
- ◆ FLAGS – vrací příznaky uvedených zpráv:

```
C: 81 FETCH 9:10 FLAGS
S: * 9 FETCH (FLAGS (\ Seen))
S: * 10 FETCH (FLAGS (\ Seen))
S: 81 OK FETCH completed
```

- ◆ RFC822 – vrací zprávu ve formátu RFC-822:

```
C: 83 fetch 9 RFC822
S: * 9 FETCH (RFC822 { 277}
S: Received: by P30X01.cbu.pvt.cz; (5.65/1.1.8.2/23Jun99-9.1MPM)
    id AA18388; Fri, 5 Jan 2001 15:28:39 +0100
S: Date: Fri, 5 Jan 2001 15:28:39 +0100
S: From: Libor Dostalek test user <dostalek>
S: Message-Id: 0101051428.AA18388@P30X01.cbu.pvt.cz
S: To: dostalek
S: Subject: pokus
S:
S: text zprávy
S: )
S: 83 OK FETCH completed
```

- ◆ RFC822.SIZE – vrací délku zprávy:

```
C: 84 fetch 9 RFC822.SIZE
S: * 9 FETCH (RFC822.SIZE 277)
S: 84 OK FETCH completed
```

- ◆ UID – vrací číslo zprávy.

- ◆ BODYSTRUCTURE – vrací strukturu zprávy. Např. zpráva obsahující text ve dvou alternativách (textové a HTML) a přílohu se souborem ve formátu MS Word má např. strukturu:

```
C: 91 FETCH 11 BODYSTRUCTURE
S: * 11 FETCH (BODYSTRUCTURE ((( "TEXT" "PLAIN" ("CHARSET" "iso-8859-2") NIL NIL
    "QUOTED-PRINTABLE" 10 1 NIL NIL NIL) ("TEXT" "HTML" ("CHARSET" "iso-8859-2")
    NIL NIL "QUOTED-PRINTABLE" 345 10 NIL NIL NIL) "ALTERNATIVE" ("BOUNDARY" "=_
    NextPart_001_0008_01C07734.EBEF7F00") NIL NIL) ("APPLICATION" "MSWORD" ("NAME"
    "soubor.doc") NIL NIL "BASE64" 26628 NIL ("ATTACHMENT" ("FILENAME" "soubor.
    doc")) NIL) "MIXED" ("BOUNDARY" "=_NextPart_000_0007_01C07734.EBEDF860") NIL
    NIL)
S: 91 OK FETCH completed
```

Uvedenou zprávu jsem pro zajimavost vypsal ve fomátu RFC-822:

### Příklad

```
Message-Id: <000b01c0772c$8b1046e0$950e11ac@libor>
From: "Libor Dostalek" <dostalek@pvt.cz>
To: <dostalek>
Subject: test
Date: Fri, 5 Jan 2001 16:31:14 +0100
Mime-Version: 1.0
Content-Type: multipart/mixed;
    boundary="=_NextPart_000_0007_01C07734.EBEDF860"
```

```
Toto je zprava ve formatu MIME obsahujici vice casti.
____=_NextPart_000_0007_01C07734.EBEDF860
Content-Type: multipart/alternative;
    boundary="=_NextPart_001_0008_01C07734.EBEF7F00"
```

```
----=_NextPart_001_0008_01C07734.EBEF7F00
Content-Type: text/plain;
    charset="iso-8859-2"
Content-Transfer-Encoding: quoted-printable

Text zprávy

----=_NextPart_001_0008_01C07734.EBEF7F00
Content-Type: text/html;
    charset="iso-8859-2"
Content-Transfer-Encoding: quoted-printable

<HTML>
<BODY bgColor=3D#fffff>
<DIV><FONT face=3D"Arial CE" =size=3D2>
text zprávy
</FONT></DIV></BODY></HTML>

----=_NextPart_001_0008_01C07734.EBEF7F00

----=_NextPart_000_0007_01C07734.EBEDF860
Content-Type: application/msword;
    name="soubor.doc"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
    filename="soubor.doc"

Base64 kódovaný dokument v MS Word
----=_NextPart_000_0007_01C07734.EBEDF860-
```

## **STORE**

Příkaz STORE slouží ke změně atributů položek (viz příkaz SELECT) v poštovní schránce. Je pochopitelně možné u položky nastavit např. atribut \ Draft. Avšak asi nejjazdavějším atributem je atribut \ Delete, jímž se připraví položka ke zrušení. Následujícím příkazem EXPUNGE se pak položka zruší.

Příkaz STORE má tři parametry:

- ◆ Číslo položky nebo interval čísel položek v poštovní schránce, u kterých má dojít ke změně atributů.
- ◆ Druhým parametrem je jedno z klíčových slov: FLAGS, +FLAGS nebo -FLAGS. Pomocí klíčového slova +FLAGS přidáváme atributy uvedené jako třetí parametr. Pomocí slova -FLAGS ubíráme parametry a pomocí slova FLAGS nastavíme parametry položky na parametry uvedené jako třetí parametr.
- ◆ Posledním parametrem je seznam atributů uzavřený v kulatých závorkách.

### **Příklady:**

```
C: 35 store 9:10 +FLAGS (\ Deleted)
S: * 9 FETCH (FLAGS (\ Seen \ Deleted))
```

```
S: * 10 FETCH (FLAGS (\ Seen \ Deleted))
S: 35 OK STORE completed

C: 36 store 10 -FLAGS (\ Deleted)
S: * 10 FETCH (FLAGS (\ Seen))
S: 36 OK STORE completed

C: 37 store 9 FLAGS (\ Deleted)
S: * 9 FETCH (FLAGS (\ Deleted))
S: 37 OK STORE completed
```

### **EXPUNGE**

Příkazem EXPUNGE se v poštovní schránce zruší položky označené příznakem \ Delete. V poštovní schránce máme 11 zpráv a zprávy 9 a 10 jsou označeny příznakem \ Delete. Pak příkaz EXPUNGE způsobí:

```
C: 38 EXPUNGE
S: * 9 EXPUNGE
S: * 9 EXPUNGE
S: * 9 EXISTS
S: * 0 RECENT
S: 38 OK Expunged 2 messages
```

Asi si myslíte, že jsem udělal chybu, protože se měly zrušit řádky 9 a 10, a nikoliv dvakrát řádek 9. Ono to ale pracuje následovně: Okamžitě po zrušení zprávy 9 dojde k přečíslování poštovní schránky, protože zprávy v poštovní schránce musí být číslovány nepřerušenou sekvencí čísel, takže ta původní zpráva číslo 10 má nyní číslo 9. Následně tak dojde opětovně ke zrušení zprávy číslo 9 (původně 10).

### **CLOSE**

Příkazem CLOSE uzavřeme poštovní schránku a přejdeme do režimu „Autentizovaný stav“.

```
C: 100 close
S: 100 OK CLOSE completed
```

## **Domácí cvičení**

- ◆ Programem nmap vyhledejte na vaší síti (může se jednat i o vzdálenou síť) servery SMTP, POP3 a případně IMAP4.
- ◆ Programem telnet odešlete testovací e-mail svým kolegům.
- ◆ Programem telnet si protokolem POP3 (případně i IMAP4) prohlédněte e-maily ve své poštovní schránce.

(Uvedená cvičení lze provádět jen vůči serverům, které nevyžadují zabezpečení pomocí SSL/TLS nebo neomezují přístup klientů jen z některých IP adres apod.)



## Kapitola 18

# Filtrace, proxy a NAT

Tato kapitola se zabývá aktivními prvky rozsáhlých počítačových sítí, kterými se oddělují vnitřní podnikové sítě od Internetu nebo jednotlivé podnikové sítě mezi sebou, aby se zamezilo útokům proti systémům umístěným ve vnitřních sítích. Těmito prvky mohou být odděleny i jednotlivé sítě v rámci téže firmy, které požadují jinou úroveň zabezpečení.

Filtrace, proxy i firewall je také možné použít pro vybudování extranetu. Extranet je způsob, jak zprostředkovat nadstandardní komunikaci mezi dvěma vnitřními sítěmi (obvykle partnerských organizací), která však nemá charakter úplné virtuální privátní sítě.

Příklady filtrů jsou uváděny pro CISCO směrovače.



**Poznámka:** V této kapitole jsou uvedeny příklady za využití programu telnet. Pokud chcete program telnet využívat ve Windows Vista, pak je nutné k jeho použití povolit volbou „Zapnout nebo vypnout funkce systému Windows“.

## Filtrace

Filtrací rozumíme kontrolu procházejících paketů aktivním prvkem sítě na základě jeho obsahu a následné rozhodnutí, může-li být paket poslan dále, nebo ne. Filtr nemění obsah datových paketů (což neplatí o některých dále popisovaných prostředcích ochrany, jako je např. NAT). Filtraci lze přirovnat k cenzuře prováděné v některých případech s poštovními zásilkami. Filtrace se může provádět na různých úrovních:

- ◆ **Na linkové vrstvě** – zde se zpravidla provádí na přepínačích (*switch*). Mohou ji však do jisté míry provádět i některé firewally.
- ◆ **Filtrace protokolů IP a TCP**, kterou lze provádět na směrovačích (*router*), ale rovněž ji provádějí některé firewally.
- ◆ **Filtrace aplikačních protokolů** – je však otázkou, zdali se ještě jedná o filtraci, protože při filtrace na aplikativní vrstvě již mnohdy dochází ke změně přenášených paketů, zejména v okamžiku, kdy je realizována proxy nebo brána. Existují však i firewally, které provádí „čistou“ filtrace na aplikativní vrstvě (nepoužívají proxy či bránu), ale zase filtrace kombinují s NAT, který přepisuje záhlaví IP datagramu či TCP segmentu.

V případě, že se provádí filtrace na úrovni linkového protokolu, pak příslušný filtr musí „umět“ linkové záhlaví, protože se filtruje na základě informací uvedených v linkovém záhlaví každého přenášeného linkového rámce. V případě filtrace na úrovni protokolu IP musí zase filtr „umět“ záhlaví IP datagramu, v případě filtrace na úrovni protokolu TCP zase filtr musí „rozumět“ záhlavím TCP

-segmentu. Jiná je situace u aplikačních protokolů, které se často neskládají ze záhlaví a přenášených dat. V takovém případě musí filtr „rozumět“ celému aplikačnímu protokolu a filtrovat veškerá aplikační data.

Filtrace **linkového protokolu** v podstatě nepatří do problematiky Internetu. Mnohé firmy dnes totiž budují nákladné firewalls a často zapomínají, že zdaleka největší procento úspěšných útoků není od nějakých tajuplných uživatelů Internetu, ale od vlastních zaměstnanců. A právě v případě LAN je velice snadné se podívat na obsah přenášených rámci a přečíst si, jaká mají kolegové hesla (používají-li protokoly telnet, ftp, POP3, HTTP atp.).

V případě, že se ve firmě používají právě programy telnet, ftp nebo snad rlogin a všichni uživatelé sedí na neprepínacím Ethernetu, pak mnozí tvrdí, že snad ani nemá cenu používat hesla.

Filtrací na přepínači (*switch*) lokální sítě lze i zajistit, aby přepínač kontroloval, že za daným rozhraním přepínače je pouze ve filtru uvedená linková (šestibajtová) adresa – tj. aby nebylo možné, že si útočník přinese svůj počítač, který zapojí do zásuvky lokálního rozvodu namísto řádně zapojeného počítače\*), a začne útočit.

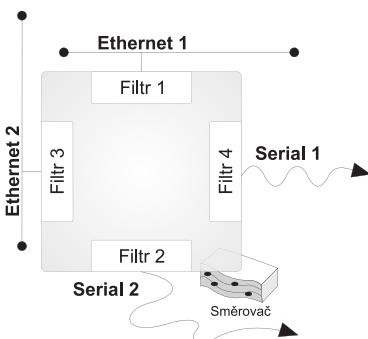


**Obrázek 18.1:** Ochrana vnitřní sítě filtrací

Obecně je cílem filtrace vytvořit polopropustný filtr, který umožnuje vlastním zaměstnancům přístup do Internetu, ale zamezuje přístup cizím uživatelům do naší sítě. Proto se často takovýto filtr přirovnává k diodě (pro elektrikáře) nebo k semipermeabilní membráně (pro biology).

Na obr. 18.1 je šipkami znázorněn cíl filtrace: umožnit přístup z vnitřní sítě do Internetu a zamezit útokům z Internetu na zdroje na vnitřní sítě. Na obrázku šipky nevyjadřují přenos dat, ale pouze směr přístupu. Přenos dat je vždy obousměrný. Klient posílá do Internetu datové pakety s požadavky na získání informací z Internetu a zpět z Internetu chce získávat data.

Směrovač může mít více síťových rozhraní, takže předávání IP datagramů může být mezi všemi rozhraními. Pokud je to technicky možné, přiřazují se filtry jednotlivým rozhraním. Na obr. 18.2 je rozhraní „Ethernet1“ přiřazen filtr „Filtr 1“, rozhraní „Serial1“ je přiřazen „Filtr 4“ atd.

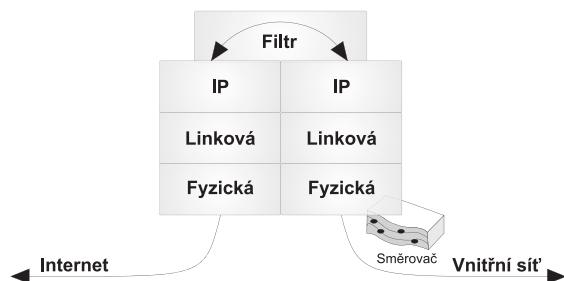


**Obrázek 18.2:** Filtry na směrovači se přiřazují jednotlivým síťovým rozhraním

V klasickém operačním systému (např. ve starších verzích systému UNIX) nelze pro aplikační vrstvu snadno zjistit, ze kterého linkového rozhraní IP datagram přišel, protože linková vrstva „vybalí“ z linkového rámce IP datagram a předá jej vrstvě IP. Při tomto předávání se již nepředává informace, z jakého linkového rozhraní data přišla – k dispozici je pouze IP adresa. To je důvod, proč si firewalls upravují standardní operační systémy. V případě, že není možné důvěryhodně v operačním systému zjistit, ze kterého rozhraní IP datagram přichází, filtry se v takových systémech realizují, jako by ležely ve středu systému, a identifikace linkových rozhraní ve filtroch nefiguruje.

## Filtrace na úrovni protokolu IP

Filtrace se realizuje filtrem, který se vkládá mezi síť, kterou chceme chránit, a Internet.



**Obrázek 18.3:** Filtr na úrovni protokolu IP

Filtr je zpravidla jednou z funkcí přístupového směrovače. Podobně jako poštmaster klasického poštovního úřadu třídí a předává poštu na základě adresy uvedené na obálce, směrovač předává IP datagramy pouze na základě informací uvedených v záhlaví IP datagramu. Korektně pracující poštmaster nikdy neotevře poštu a nepřečte si ji, stejně tak dobře vychovaný směrovač by nikdy neměl pracovat s jinými informacemi než s těmi uvedenými v IP záhlaví. Uvidíme však, že filtrace pouze na základě informací ze záhlaví IP datagramu nestačí a směrovač-filtr potřebuje pracovat i s informacemi ze záhlaví TCP segmentu či UDP datagramu. A existují i filtry pracující s aplikačními informacemi.

Máte-li zakoupen rozumný přístupový směrovač, pak vás filtrace na IP vrstvě zpravidla nebude stát žádné další investice. Stačí ji pouze nastavit. Pro úplnost je třeba dodat, že v praxi se můžete setkat i s filtry realizovanými na serverech a firewallech.

Filtrovat na úrovni protokolu IP lze pouze na základě údajů, které jsou v záhlaví IP datagramu (obr. 18.4). Ať uvažujeme nad IP záhlavím jakkoliv, filtrovat budeme nejspíše na základě IP adresy odesilatele a příjemce.



**Poznámka:** Velice často se též zahazují IP datagramy s vyplňenými volitelnými položkami „Explitní směrování“. V tomto případě se rovněž jedná o klasický případ filtrace.

Princip filtrování (obr. 18.3) spočívá v tom, že na rozdíl od správně vychovaného směrovače se filtr podílí do IP datagramu na jeho IP záhlaví a případně i do záhlaví TCP (resp. UDP) paketu a rozhodne se, zda takový paket do příslušného rozhraní předá, nebo zda jej zahodí.



**Obrázek 18.4:** Záhlaví IP datagramu

Filtr se rozhoduje na základě obsahu záhlaví IP datagramu (případně též záhlaví TCP/UDP paketu). Jistou zvláštností je filtrace protokolu ICMP, o které se také zmíníme.

Nyní se vraťme k filtraci na úrovni IP protokolu. Samotná filtrace na úrovni pouze IP protokolu mnoho nezmůže, ale ve spojení s filtrací na úrovni protokolu TCP je poměrně mocným nástrojem. Nyní si popíšeme filtraci pouze na základě IP adresy odesilatele a příjemce.

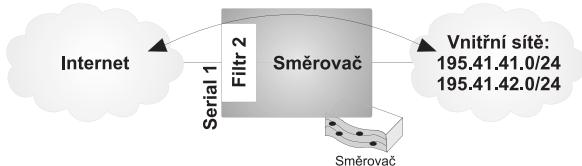
Při tvorbě filtrování jsou teoreticky možné dvě varianty. Buď se vše implicitně povolí a dopisují se pravidla specifikující, které počítáče kam nesmí. Nebo naopak se implicitně vše zakáže a pouze explicitně se něco povoluje. Z bezpečnostních důvodů se většinou dává přednost druhé variantě (považuje se obecně za bezpečnější).

I když je filtrace možná na nejrůznějších prvcích, s největší pravděpodobností se s ní v praxi setkáte na směrovačích CISCO. Proto si popíšeme filtraci na těchto směrovačích, které podporují následující typy filtrování:

- ◆ „Standardní filtry“ filtrující pouze na základě IP adresy odesilatele (odesilatelem se rozumí odesíatel IP datagramu).
- ◆ „Rozšířené filtry“ filtrující na základě jak IP adresy odesilatele, tak i IP adresy příjemce. Rozšířené filtry umí filtrovat i na základě informací z TCP záhlaví (resp. UDP záhlaví).
- ◆ „Dynamické filtry“ umožňující otevřít specifikovaný průchod směrovačem uživateli, který se vůči směrovači autentizoval. Praktické využití tohoto filtrování je omezeno spíše na správce systému, protože uživatel se nejprve musí autentizovat např. programem Telnet. To však asi není pro běžné uživatele příliš schůdné.
- ◆ „Reflexivní filtry“, které sledují relaci vyššího protokolu (TCP, resp. UDP) a povolují směrem ven zřídit relaci a směrem dovnitř propouštět pouze pakety patřící k této relaci.

### Standardní filtr

Standardní filtr prakticky nelze použít k nějaké bezpečnostní ochraně. Můžeme si představit podle obr. 18.5, že naše vnitřní síť není chráněna před útoky z Internetu, a ani tuto ochranu nevyžadujeme. Vnitřní síť je tvořena dvěma sítěmi třídy C: síť 195.41.41.0/24 a síť 194.41.42.0/24. Cílem naší filtrace je, abychom přístup do Internetu povolili pouze ze sítě 195.41.41.0/24 a uživatelům vnitřní sítě 195.41.42.0/24 tento přístup naopak zakázali.



**Obrázek 18.5:** Standardní filtr

Nejprve je třeba vysvětlit, jak se nelogicky u směrovačů CISCO zadává síť 195.41.41.0/24. Pokud bychom očekávali, že zapíšeme tuto síť jako síť 195.41.41.0 s maskou 255.255.255.0, je to omyl. CISCO v masce zaměňuje nuly a jedničky (mají na to několik důvodů, proč tomu tak je, ale nejlépe je o tom nepřemýšlet a mechanicky to přijmout). Takže ve světě filtrů CISCO je:

Síť 195.41.41.0/24 síť 195.41.41.0 s maskou 0.0.0.255

Tvorba filtru se skládá ze dvou částí. Nejprve se definuje filtr (access-list) a ve druhém kroku se filtr aktivuje na konkrétním rozhraní směrovače. Toto elegantní řešení tak umožní jednou vytvořený filtr aktivovat i na více síťových rozhraních směrovače.

### Příklad

Náš filtr číslo 2 tak bude mít tvar:

```
access-list 2 permit 195.41.41.0 0.0.0.255
access-list 2 deny 195.41.42.0 0.0.0.255
```

kde slovo „permit“ znamená „povolit“ a slovo „deny“ znamená „zakázat“.

Nyní nám zbývá druhý krok: aktivovat filtr na rozhraní „Serial 1“. To se provede příkazem:

```
interface serial 1 ip access-group 2 out
```

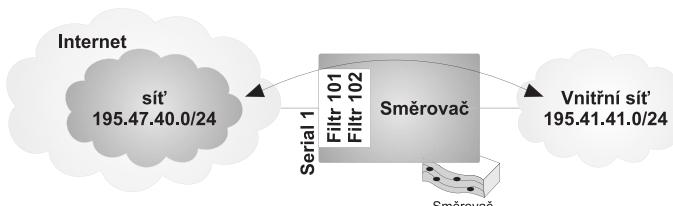
kde číslo filtru (2) se uvede jako parametr za klíčovým slovem „access-group“. Zajímavé je poslední klíčové slovo „out“, které říká, že k filtraci dochází ve směru přenosu datových paketů z rozhraní směrovače do sítě (*outgoing*). Obdobně slovo „in“ by znamenalo, že k filtraci dochází ve směru ze sítě do směrovače (*incoming*).

Je tak možné napsat dva filtry: jeden pro tok dat ze směrovače do sítě a druhý pro tok dat ze sítě do směrovače.

### Rozšířený filtr

Rozšířený filtr umožňuje filtraci na základě adresy odesilatele i příjemce.

Pro další úvahy si vybudujeme hypotetickou podnikovou síť 195.41.41.0/24 a propojíme ji pomocí směrovače k Internetu (obr. 18.6). Náš hypotetický podnik má ještě další podobně k Internetu při-



**Obrázek 18.6:** Filtr omezující komunikaci na komunikaci pouze mezi dvěma sítěmi

pojené pracoviště se sítí 195.47.40.0/24. Obě pracoviště chtějí komunikovat spolu a pro vzájemnou komunikaci chtějí využít Internet.

Nejprve si popíšeme filtr 101 pro komunikaci ze sítě 195.41.41.0/24 do sítě 195.47.40.0/24:

### Příklad

```
access-list 101 permit ip 195.41.41.0 0.0.0.255 195.47.40.0 0.0.0.255
access-list 101 deny ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
```

Tento filtr se skládá ze dvou řádků: v prvním rádku říkáme, že povolujeme komunikaci IP protokolem (klíčové slovo „ip“) z jedné sítě do druhé, a v zápětí na druhém rádku však musíme zakázat jakoukoliv jinou komunikaci. (Druhý rádek je jakousi obdobou rádku „default“ ve směrovači tabulce – „vše ostatní je zakázáno“). V případě, že na vašem směrovači budete používat kombinaci více filtrov, rádek „vše ostatní je zakázáno“ uvedete až v posledním filtrov.

A obdobně napíšeme filtr 102 pro komunikaci v opačném směru.

```
access-list 102 permit ip 195.47.40.0 0.0.0.255 195.41.41.0 0.0.0.255
access-list 102 deny ip 0.0.0.0 255.255.255.255 0.0.0.0 255.255.255.255
```

Nyní se již dostanu k tomu, proč jsme vytvořili filtry dva. Filtr 101 použijeme k filtraci toku dat ze směrovače do Internetu (out) a filtr 102 v opačném směru:

```
interface serial 1
    ip access-group 101 out
    ip access-group 102 in
```

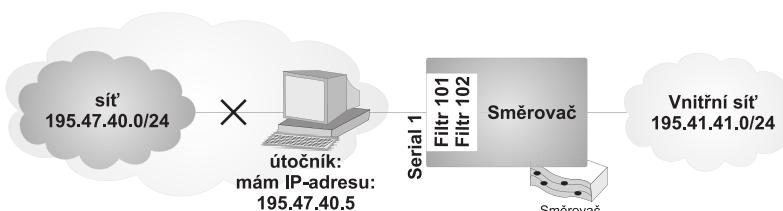
Proč jsem však aktivoval filtr na rozhraní „Serial 1“, tj. rozhraní směrem do Internetu? Bylo by přece možné aktivovat filtr na rozhraní směrem do vnitřní sítě (to se jmenuje „ethernet 0“):

```
interface ethernet 0
    ip access-group 101 in
    ip access-group 102 out
```

Pravidlem však je, že filtry aktivujeme vždy na rozhraní, které je blíže k útočníkovi, aby se zamezilo útočníkovi i útokům vedeným proti samotnému směrovači. V tomto případě očekáváme útočníka z Internetu (nekomentuji zde, že k většině útoků dochází od vlastních zaměstnanců).

Bezpečnostní riziko při propojení dvou sítí přes Internet s filtrací na úrovni IP adres spočívá v tom, že někdo v Internetu, kdo se nachází mezi oběma sítěmi, se nelegálně může prohlásit za IP adresu, která patří jedné ze sítí.

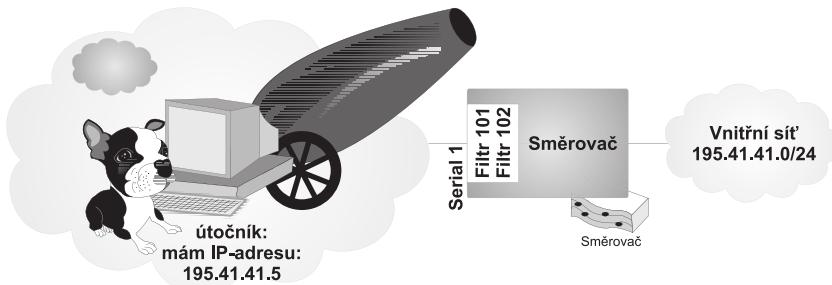
Tak může útočník předstírat, že je naším protějškem, což takto postavený filtr nemůže zjistit (tzv. *man in the middle attack*).



**Obrázek 18.7:** Man in the middle attack

Proto v případě, že jsou dvě sítě propojeny pomocí Internetu takovým způsobem, aby tvořily z hlediska přístupu jeden celek, zabezpečuje se toto spojení ještě dalšími prostředky. Příkladem řešení je vytvoření VPN např. pomocí IPsec.

Dalším útokem je *address-spoofing attack*. Při něm útočník odesílá IP datagramy s adresou odesilatele patřící do intervalu vnitřní sítě. Obr. 18.8 znázorňuje, jak se dostat skrze náš filtr na počítač vnitřní sítě.



Obrázek 18.8: Address-spoofing attack

Útočník v tomto případě volí adresu z vnitřní sítě. Zdánlivě se to může zdát neefektivní, protože útočník nemůže dostávat žádné odpovědi. Pro útočníka je komplikovanější provést útok, aniž by od napadeného počítače dostával nějaké reakce. Avšak v mnoha případech může útočník reakci napadeného předpokládat, a tak odpovědi ani nepotřebuje. Takový útok je komplikovaný např. už při použití protokolu TCP, kde napadený generuje náhodné číslo, kterým začíná číslovat svá odesílaná data. Útočník tak musí ve svém útoku předpovědět toto náhodné číslo, aby mohl v TCP relaci pokračovat. Zdá se neuvěřitelné předpovědět náhodné číslo, ale v minulosti se ukázalo, že mnohé generátory náhodných čísel nebyly příliš „náhodné“ a k takovým útokům došlo. Nedávná studie prokázala, že mnoho obvyklých operačních systémů stále nemá dostatečně kvalitní generátory náhodných čísel a útoky tohoto druhu jsou stále možné. Avšak útočit UDP datagramy či ICMP signalizací není komplikované ani pro méně zdatného útočníka. Např. podvrhnout v DNS přiřazení jména počítače na jinou IP adresu či změnit čas pomocí protokolu NTP jsou velmi snadné úlohy.

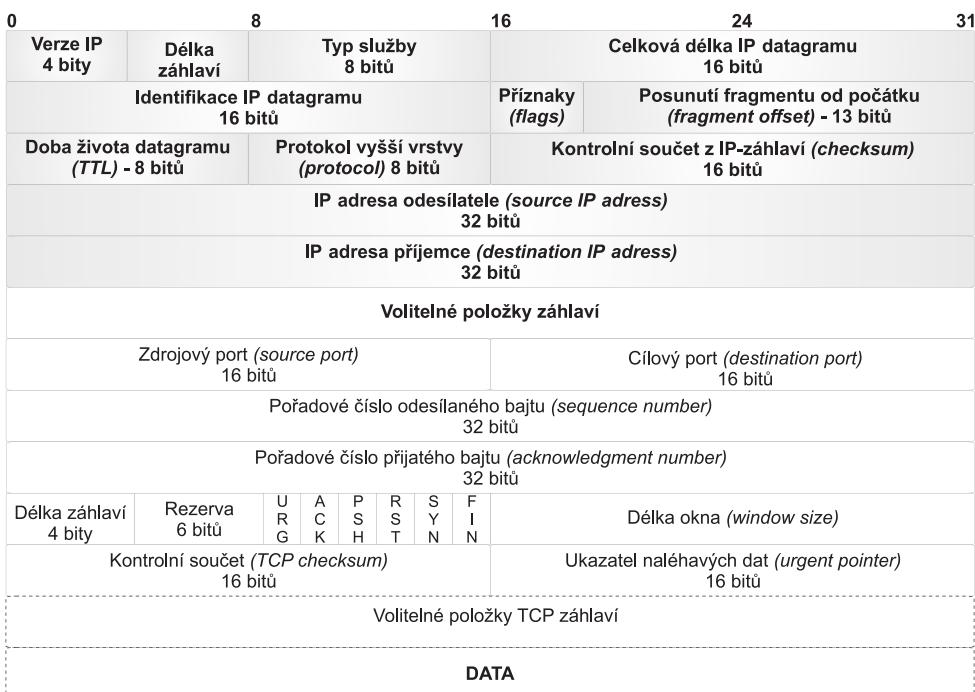
IP adresu volí útočníci tak, aby našli skulinku ve vašem filtrování. Nefiltrující směrovače se nezabývají adresou odesilatele, takže IP datagram s chybnou adresou odesilatele snadno putuje Internetem. Řeknete si, že cílem útoku je získat nějaké informace. Když však útočník nedostane žádnou odpověď, nemůže dostat ani žádné informace. To však útočníkovi nevadí, je vcelku snadné si odpověď nechat poslat elektronickou poštou na jiný počítač ve vnitřní síti (program sendmail ve vnitřní síti přeče přijme a dále pošle většinou vše, co dostane).

Filtry vytvářené v některých operačních systémech umožňují uvádět jména počítačů, ve filtrování však uvádějte zásadně IP adresy, nikoliv jména počítačů. Vyhýbejte se uvádění jmen počítačů, protože překlad jména počítače na IP adresu pomocí DNS není nijak zabezpečen a lze jej snadno podvrhnout (viz *address-spoofing attack*).

## Filtrace na úrovni TCP

### Rozšířený filtr – pokračování

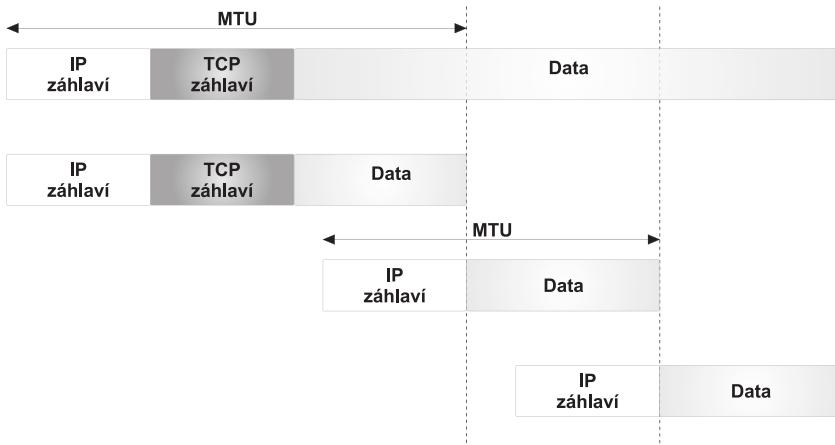
Zatím jsme rozšířený filtr použili pouze pro protokol IP; nyní jej použijeme i pro protokoly vyšší vrstvy.



Obrázek 18.9: Záhlaví TCP segmentu předcházené záhlavím IP datagramu

Zatímco filtrací protokolu IP se určovalo, které počítače mohou mezi sebou komunikovat, filtrací na úrovni protokolu TCP (resp. UDP) se tato komunikace omezuje jen na některé aplikace běžící na těchto počítačích. Kombinací filtrace protokolu IP a protokolů TCP/UDP je možné stanovit, aby konkrétní počítače mohly mezi sebou komunikovat pouze konkrétními aplikacemi. Navíc je možné filtrací zajistit, aby klienti určité aplikace z vnitřní sítě mohli využívat serverů v Internetu, a přitom naopak, aby cizí klientům byl znemožněn přístup do vnitřní sítě – např., aby se klienti z vnitřní sítě dostali na WWW servery v Internetu, ale aby se cizí klienti nedostali na WWW servery vnitřní sítě.

Začněme však od začátku. TCP segment (resp. UDP datagram) paket se vkládá (balí) do IP datagramů (obr. 18.9). IP datagram nesoucí TCP segment může být během své cesty Internetem fragmentován na několik IP datagramů. To se stane v případě, že na nějaký směrovač na cestě dorazí IP datagram, jehož délka je větší než největší možná velikost dat linkového rámce (tzv. MTU), kterým mají být data dále přepravována. Např. na obr. 18.10 nám z jednoho TCP segmentu fragmentací vznikly tři IP datagramy.



Obrázek 18.10: Fragmentace

Fragmentací vzniklé IP datagramy se dopravují Internetem samostatně. Všimněte si, že TCP záhlaví je jen v prvním z nich! Jak může provádět směrovač filtraci na základě záhlaví TCP segmentu, když v některých IP datagramech TCP záhlaví vůbec není? Směrovač to dělá jednoduše. V případě, že chce filtrovat protokol TCP, filtrouje pouze ty IP datagramy, které obsahují TCP záhlaví. Ostatní propouští.

Příjemci pak nedorazí pouze první fragment. Příjemce se snaží sestavit z jednotlivých doslých částí TCP segmentu celý paket. Ale to se mu nepodaří, protože mu vždy schází začátek. Po dosažení časového limitu to příjemce vzdá. Jediný problém je v tom, že příjemce, který neustavil TCP paket, o tom protokolem ICMP (*packet reassembly time expired*) informuje odesilatele TCP paketu. Tato informace vysvětuje útočníkovi, proč se nedostal dále. Doporučuje se tedy takovéto odpovědi také filtrovat, aby útočník nedostával další informace.

Filtrace na úrovni protokolu TCP se proto často spojuje s filtrací protokolu ICMP, o které bude řeč později.

Modernější filtry nepropouští mechanicky IP datagramy, které neobsahují TCP záhlaví. Postupují jednoduše tak, že si počkají na všechny IP datagamy konkrétního TCP segmentu a pak se rozhodnou. Tato technika by nebyla možná, pokud by od odesilatele k příjemci existovaly dvě cesty. To však není nás případ.

Z hlediska filtrace je v záhlaví TCP (resp. UDP) zejména využitelný port odesilatele a port příjemce. V TCP záhlaví je navíc využitelný ještě příznak ACK (obr. 5.9).

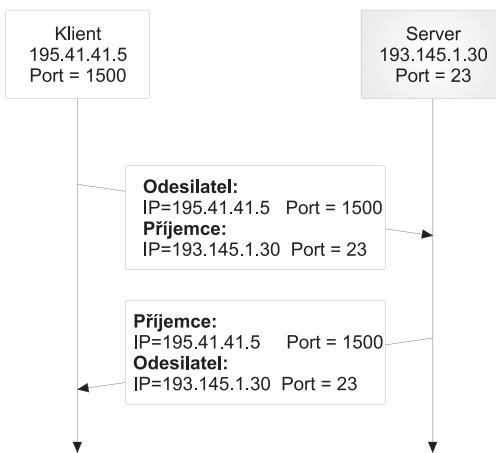
Servery používají všeobecně známá čísla portů, jejichž podmnožinu najdete v UNIXu v souboru /etc/services, ve Windows 2000 a výše v souboru C:\WINNT\system32\drivers\etc\services a kompletní oficiální informaci získáte na <http://www.iana.org/>.

Standardní klient si ještě před tím, než začne komunikovat, od správy portů (služba operačního systému) nechá dynamicky přidělit číslo portu. Po uzavření spojení se toto číslo vrací operačnímu systému k dalšímu použití – pro další klienty. Klienti obvykle používají dynamicky přidělovaná čísla portů platná pouze na dobu spojení. Z tohoto pravidla existuje několik málo výjimek. (Např. datové

spojení u aktivního režimu FTP navazuje FTP server, který v tomto datovém spojení hráje úlohu klienta, a používá k tomu obvyklý port 20. Mnoho FTP serverů však od toho upouští a používá libovolný port.)

Čísla portů se v unixových systémech historicky rozdělují hranicí 1023 na porty privilegované a neprivilegované. Privilegované (1–1023) mohou používat pouze procesy, které spustil superuživatel. Servery spouští většinou právě superuživatel. Takže porty 1–1023 se někdy označují jako serverové porty, porty s vyššími čísly jako klientské. Nemusí to však být pravidlem, některé servery je běžné spouštět na portech o vyšších číslech (např. X-servers, Archie, databázové servery atd.). Naopak např. na PC je snadné, aby klient použil jako klientský port např. port 23, který je běžně používán serverem protokolu telnet (programem telnetd). Je to snadné proto, že osobní počítač si spravuje klient sám a může si zde nainstalovat i např. systém UNIX, který obsluhuje jako superuživatel. Ve filtroch bude me však nadále předpokládat, že klient používá porty > 1023.

Zopakujme si výměnu paketů mezi klientem a serverem v protokolu TCP. Jako příklad si zvolíme protokol Telnet, jehož server používá port 23.



**Obrázek 18.11:** Komunikace klient-server v protokolu Telnet

Nejprve musí být spuštěn server na portu 23 (viz obr. 18.11). Budeme předpokládat, že server má IP adresu 193.145.1.30 a že IP adresa klienta je 195.41.41.5. Dříve, než může začít klient s komunikací, musí si nechat od svého operačního systému přidělit nějaký volný port > 1023. Předpokládejme, že systém klientovi přidělil port 1500.

Nyní již klient může protokolem TCP navazovat spojení se serverem.

Příslušný filtr, který by propouštěl pouze komunikaci protokolem Telnet z vnitřní sítě 195.41.41.0/24 do Internetu a vše ostatní zakazoval, se opět skládá ze dvou částí. Z vnitřní sítě ven pustíme:

### Příklad

```
ip access-list extended filtrOUT
permit tcp 195.41.41.0 0.0.0.255 any eq 23
deny ip any any
```

A obdobně zpět z Internetu do vnitřní sítě pustíme:

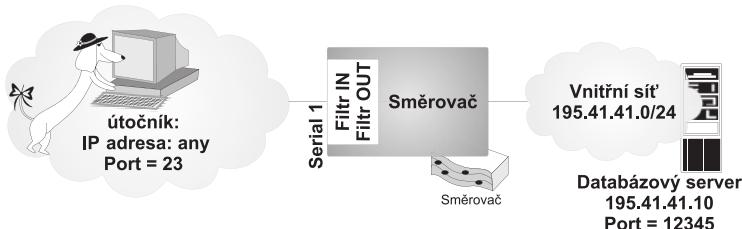
```
ip access-list extended filtrIN
permit tcp any 195.41.41.0 0.0.0.255
deny ip any any
```

Místo číselného označení filtrů jsem použil název filtru (tzv. pojmenovaný filtr). V tomto případě je nutné použít klíčové slovo „extended“ a na počátek vložit slovo „ip“ (při číselném označení se rozšířený filtr pozná podle toho, že má číslo větší než 99).

Asi jste si všimli dalšího klíčového slova „any“, které se používá jako synonymum pro „vše“, tj. pro:  
0.0.0.0 255.255.255.255

Filtry pak aktivujeme příkazem:

```
interface serial 0
  ip access-group filterOUT out
  ip access-group filterIN in
```



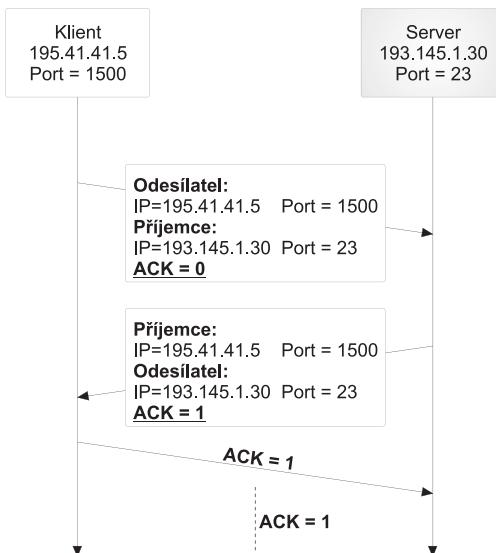
**Obrázek 18.12:** Útočník používající port 23 jako port klientský

Jak zjistíte z obr. 18.12, námi napsaný filtr nám umožní útočit na servery běžící na portech větších než 1023 a to jsou zejména databázové servery (databázové servery se spouští na těchto portech, aby správce databáze nemusel mít superuživatelská privilegia).

I tomuto útoku lze zabránit zdokonalením filtrace na úrovni protokolu TCP.

Protokol TCP vytváří spojení mezi dvěma aplikacemi. V tomto spojení si obě strany vzájemně potvrzují došlé pakety. K potvrzování došlých paketů se v TCP záhlaví používá příznak ACK. Tento příznak se nastavuje ve všech paketech TCP, kterými se potvrzují předchozí došlé pakety. Příznak ACK tedy není nastaven v prvním paketu, kterým klient navazuje spojení. Ve všech dalších paketech má již klient nastaven příznak ACK. Všechny pakety, které odesílá server, mají rovněž nastaven příznak ACK.

Jinými slovy: První paket odeslaný klientem nemá nastaven příznak ACK, kdežto všechny pakety, které odesílá server, mají nastaven příznak ACK (obr. 18.13). Zakáže-li se tedy některým směrem propouštět TCP pakety s nenastaveným příznakem ACK, pak tímto směrem znemožníme klientům navazovat spojení se servery.



Obrázek 18.13: Příznak ACK není nastaven pouze u prvního TCP segmentu

### Příklad

```
access-list extended FiltrOUT
permit tcp 195.41.41.0 0.0.0.255 any eq 23
deny ip any any
access-list extended FiltrIN
permit tcp any 195.41.41.0 0.0.0.255 established
deny ip any any

interface serial 0
    ip access-group FiltrOUT out
    ip access-group FiltrIN in
```

Kouzelné slovo „established“ právě sděluje, že se dovnitř (tj. filtrem FiltrIN) mají vpouštět pouze TCP-segmenty s nastaveným příznakem ACK (ACK = 1). Pro úplnost je třeba dodat, že slovem „established“ netestujeme pouze nastavení příznaku ACK, ale ošetruje se i průchod paketu s nastaveným příznakem RST (odmítnutí spojení).

### Reflexivní filtry

Protokoly UDP, ICMP, IGMP apod. nemají bit ACK. Proto u nich s technikou uvedenou v předchozím odstavci nevystačíme. Jedná se vesměs o protokoly, které nevytváří spojení.

Pokud nenajdeme nějakou jinou elegantní cestu k filtraci u těchto protokolů, nezbývá nám, než se bez těchto protokolů obejít nebo jejich použití omezit. Přitom se bez nich nemusíme obejít při komunikaci v rámci vnitřní sítě či v Internetu. Pouze je budeme omezovat při komunikaci přecházející mezi vnitřní sítí a Internetem.

Jinou možností je vybudovat demilitarizovanou zónu DMZ (obr. 18.15). DMZ je samostatná síť oddělená jak od Internetu, tak i od intranetu. Její základní vlastností je, že je dostupná jak z Internetu, tak i z intranetu. Pomocí filtrů lze pak omezit komunikaci „podezřelými“ protokoly pouze z Internetu do DMZ a rovněž z vnitřní sítě do DMZ. Tím se znemožní přímá komunikace mezi Internetem a vnitřní sítí „podezřelými“ protokoly. V tomto případě musíme na DMZ umístit server příslušného protokolu.

Elegantním řešením jsou reflexivní filtry, u jejichž popisu zůstaneme u systémů firmy CISCO.

Jakmile se začne uvažovat o tom, které rozhraní je rozhraním pro vnitřní síť a které rozhraní pro Internet, nenápadně se dostáváme do úplně jiné kategorie zařízení – do oblasti firewallů. Reflexivní filtr je již tak pokročilou vlastností směrovače, že začíná být otázkou, jestli takový směrovač už nenačázat firewalem.

Takže než začneme s popisem reflexivního filtru, jasně si na našem směrovači řekněme, které rozhraní je do Internetu, a budeme jej nazývat „vnějším rozhraním“, a které je do vnitřní sítě, a budeme jej nazývat „vnitřním rozhraním“. Pro úplnost musíme ještě dodat, že směrovače mívají více rozhraní – např. rozhraní pro DMZ či pro „extranety“.

Reflexivní filtr není nic složitého. Pro jeho vysvětlení se vraťme zcela na počátek, k obr. 18.1. Cílem ochrany vnitřní sítě je umožnit zaměstnancům z vnitřní sítě přistupovat do Internetu a zamezit útočníkům z Internetu útočit na systémy ve vnitřní sítí. Avšak jak komunikace zaměstnance, tak i atak útočníka jsou z hlediska přenosu dat obousměrné relace – tj. zaměstnanec navazuje relaci např. s webovým serverem v Internetu, zadá mu, jaké informace od něj žádá, a webový server mu je podává.

Klasické filtry se však na komunikaci zaměstnance s webovým serverem v Internetu dívají jako na dva samostatné kanály: jeden jde od klienta na server a druhým server vrací data. Reflexivní filtr se dívá na relaci jako celek: tj. klient zahajuje relaci a reflexivní filtr zpět propustí jen datové pakety náležející této relaci a žádné jiné. Jako relaci zde chápeme např. i překlad jména v DNS prováděný za využití protokolu UDP. Taková relace se pak skládá z dotazu a odpovědi.

Reflexivní filtr filtrouje odchozí datové pakety. Pokud zachytí paket, kterým se zahajuje nová relace, pak v opačném směru vygeneruje položku dočasného filtru, která umožní průchod paketům téže relace v opačném směru. Položka dočasného filtru získá své parametry z odchozího paketu:

- ◆ Protokol vyšší vrstvy je stejný jako v odchozím paketu.
- ◆ IP adresa odesilatele a příjemce je v položce dočasného filtru přehozena, protože příchozí paket bude mít tyto položky přehozeny.
- ◆ Porty odesilatele a příjemce jsou rovněž prohozeny. (Porty se uplatní pouze u paketů nesoucích protokol TCP nebo UDP.)

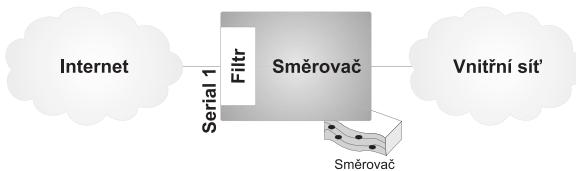
Dočasná položka je udržována po dobu relace. V případě protokolu TCP je udržována ještě 5 vteřin po průchodu druhého paketu s příznakem FIN nebo je ukončena po průchodu paketu s příznakem RST (odmítnutí spojení). Tuto taktiku lze uplatnit pouze u protokolu TCP, který vytváří spojení. Obecně se pomocí klíčového slova „timeout“ nastavuje ve vteřinách interval, po jehož uplynutí se položka dočasného filtru ruší, pokud po tu dobu byla relace nečinná.

Reflexivní filtry mají i svá omezení. Nelze je použít pro protokoly, jejichž odpověď může použít jiný port. Např. je nelze aplikovat na klasické FTP (avšak na pasivní FTP lze použít i reflexivní filtry).

### Reflexivní filtr na vnějším rozhraní

V prvním kroku vytvoříme reflexivní filtr, který bude zachycovat odchozí datové pakety, kterými bude klient vnitřní sítě zřizovat nové relace. Odchycení prvního paketu relace má za následek zřízení dočasného filtru v opačném směru. V druhém kroku musíme připravit dočasný filtr.

Reflexivní filtr definujeme pro odchozí provoz, kdežto dočasný filtr definujeme pro příchozí provoz. Oba filtry musí mít svá samostatná jména. Zatímco u reflexivního filtru definujeme jeho jednotlivé položky, u dočasného filtru definujeme pouze jeho základ – jeho položky budou vznikat dynamicky s tím, jak budou uživatelé vnitřní sítě zřizovat své relace.



**Obrázek 18.14:** Reflexivní filtr na vnějším rozhraní směrovače

Reflexivní filtr se vytvoří obdobně, jako jsme vytvářeli klasický pojmenovaný filtr:

```
ip access-list extended filtrREFLEX
```

Parametrem „permit“ se pak definují jednotlivé položky reflexivního filtru. Parametr „permit“ má následující tvar:

```
permit protokol any any reflect jméno [timeout vteřiny]
```

kde klíčové slovo „reflect“ právě specifikuje, že se jedná o reflexivní filtr. Jméno za slovem „reflect“ nám provede spojení mezi reflexivním filtrem a dočasným filtrem. Klíčové slovo „timeout“, definující časový interval, po který se udržuje položka dočasného filtru při nečinnosti relace, je nepovinné, proto je uvedeno v hranatých závorkách.

Při definici reflexivního filtru je nejdůležitějším parametrem parametr „protokol“, specifikující, pro jaký protokol vyšší vrstvy se reflexivní filtr definuje. Pro každý protokol vyšší vrstvy, pro který chceme zavést reflexivní filtr, musíme napsat samostatný příkaz „permit“.

Nyní se může připravit dočasný filtr, který se opět definuje obdobně jako klasický pojmenovaný filtr:

```
ip access-list extended filtrDOCASNY
```

Definice dočasného filtru se ukončuje příkazem „evaluate“, jehož parametrem je jméno uvedené při definici reflexivního filtru za klíčovým slovem „reflect“:

```
evaluate jméno
```

Tím se sváže dočasný filtr s příslušným reflexivním filtrem. Aktivace filtru na příslušném vnějším rozhraní se pak opět provede příkazem „ip access-group“. Celá definice reflexivního filtru může být komplikována ještě tím, že reflexivní filtr (i dočasný filtr) může obsahovat i klasické statické položky. Při takto komplikovaném filtru je třeba si uvědomit, že směrovač zpracovává jednotlivé filtry a jejich položky sekvenčně. Bude-li procházející paket vyhovovat nějaké statické položce, pak se již další filtry neuplatní a příslušná položka v dočasném filtru nemusí vzniknout.

Příklad reflexivního filtru umožňujícího uživatelům zřizovat vnitřní síť libovolné relace na bázi protokolů TCP a UDP (vstupní filtr má i statickou položku, kterou je znemožňována komunikace protokolem ICMP):

### Příklad

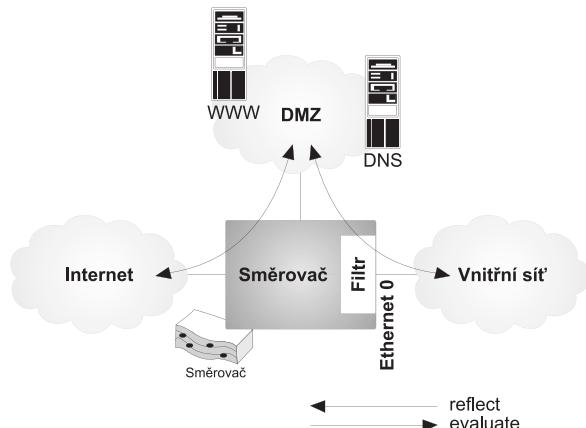
```
Interface Serial 1
ip access-group filtrRE out
ip access-group filtrDO in

ip access-list extended filtrRE
permit tcp any any reflect REFLEX
permit udp any any reflect REFLEX

ip access-list extended filtrDO
deny icmp any any
evaluate REFLEX
```

### Reflexivní filtr na vnitřním rozhraní

Reflexivní filtr není vhodné aktivovat na vnějším rozhraní přístupového směrovače v případě, že používáme demilitarizovanou zónu (DMZ). Tato aktivace by znemožnila přístup z Internetu k systémům na DMZ. Obzvláště v případě, kdy bychom měli na DMZ např. webový server či DNS server, by tato situace byla nepříjemná.



**Obrázek 18.15:** Reflexivní filtr na vnitřním rozhraní směrovače

Řešením je aktivovat reflexivní filtr na vnitřním rozhraní. V tomto případě se reflexivní filtr definuje na vstupu do vnitřního rozhraní směrovače a dočasný filtr vzniká na výstupu tohoto rozhraní. Náš příklad modifikovaný pro vstupní rozhraní by mohl vypadat např. následovně:

### Příklad

```
Interface Ethernet 0
ip access-group filtrRE in
ip access-group filtrDO out
```

```
ip access-list extended filtrRE
permit tcp any any reflect FEFLEX
permit udp any any reflect REFLEX

ip access-list extended filtrDO
deny icmp any any
evaluate REFLEX
```

## Filtrace protokolů UDP, ICMP a případně dalších protokolů

Zde již jde pouze o úvahuna uvedené téma, neboť metody jsme již popsali. Je nesporné, že teoreticky dovedou směrovače filtrovat nejrůznější protokoly – a to včetně protokolů, které s rodinou protokolů TCP/IP vůbec nesouvisí. Já si myslím, že je třeba se nejprve zamyslet nad tím, jaké protokoly budou naše aplikace využívat.

Přitom si musíme uvědomit, že řešíme využití těchto protokolů pro komunikace skrze přístupový směrovač, tj. mezi Internetem a vnitřní sítí.

Podle mého názoru je použití protokolu UDP v praxi spojeno pouze s DNS, různými multimediálními protokoly a chaty (RealAudio/Video, NetMeeting, ICQ) a nanejvýše s protokolem NTP (*Network Time Protocol*). Jenže protokol NTP nikdy do vnitřní sítě nepouštíme – maximálně protokolem NTP synchronizujeme přístupový směrovač a pomocí něj pak předáváme čas do vnitřní sítě. Jelikož náklady na přijímač času GPS jsou pouze v tisících Kč, myslím si, že je jednodušší si zřídit zdroj přesného času přímo ve vnitřní síti bez pomoci Internetu. Bylo by však dobré mít alespoň tři nezávislé zdroje času, a pak se jeden z Internetu hodí. Otázka je, zda firma opravdu takový důmyslný systém času efektivně využije.

Nepřipadá mi ani potřebné skrze přístupový směrovač propouštět jiné aplikační protokoly na bázi UDP. Stejná situace je i s protokolem ICMP.

V případě protokolu UDP i ICMP si lze pomocí reflexivním filtrem. Nemáme-li reflexivní filtr k dispozici, pak protokol ICMP prostě zcela zakážeme a zbývá jediný problém, a to DNS. Pokud na směrovači (uvědomte si, že to může být i unixový počítač) umíme spustit DNS server, vyhne se filtrování DNS stejně jako u NTP. DNS server nám totiž může zprostředkovat veškeré informace o DNS z Internetu. Na mnoha dedikovaných směrovačích to však není možné.

Já bych to vyřešil tak, že bych udělal klasický statický filtr umožňující klientům vnitřní sítě pouze přístup na primární a sekundární DNS server poskytovatele, na jeho port 53/udp.

## Zakázané adresy

Zjistíme-li, že z nějaké IP adresy nás atakuje útočník, zaneseme si tuto adresu do množiny zakázaných adres – na „černou listinu“. Filtry pak rozšíříme o explicitní zákaz komunikace s těmito adresami. Za zmínku stojí i to, že zakázané adresy můžeme mít i z vnitřní sítě.

## Aplikační protokoly a filtrace

Jsme u celkového shrnutí, jak jsou na tom s filtrací jednotlivé aplikační protokoly. Zabývám se pouze aplikačními protokoly: Telnet, SSH, FTP, HTTP, SSL/TLS, SMTP, POP3, IMAP4, NNTP, LDAP a o protokolech DNS a NTP jsem se již zmínil. Vím, že v rodině protokolů TCP/IP je aplikačních

protokolů podstatně více, ale podle mého názoru ostatní protokoly jsou pro komunikaci v Internetu postradatelné.

Závěr bude, že protokoly na bázi architektury klient-server, kde klient je umístěn ve vnitřní síti a server v Internetu, se pro filtraci hodí. Jedná se o protokoly: Telnet, SSH, HTTP, SSL/TLS, FTP, POP3, IMAP4, LDAP, klient SMTP, klient NNTP a diskutabilní pasivní FTP.

Ostatní protokoly se při ochraně vnitřní sítě pouze filtrací nedají použít. Jedná se o aktivní FTP a o umístění serveru SMTP či NNTP na vnitřní síť.

### Telnet a SSH

Protokol Telnet je pro filtraci snad úplně ideální. Otázkou je, proč přistupovat protokolem Telnet do Internetu. Vždyť je tak nebezpečný – kdekoli lze odchytit používané heslo a komunikace běží nezabezpečené. Takže závěr je, že protokol Telnet je sice výborný z hlediska filtrace, ale prakticky by se používat neměl.

V Internetu je oblíben program SSH, který uživateli poskytuje minimálně stejně možnosti jako Telnet, avšak spojení je provedeno zabezpečeným kanálem. Je třeba podotknout, že protokol SSH se stejně dobře filtruje jako Telnet (server SSH zpravidla používá port 22/tcp). Takže v praxi, když už je třeba interaktivně přistupovat do nějakého systému na Internetu, použije se SSH a bezpečnostní rizika plynoucí z používání Telnetu jsou eliminována.

Příklad rozšířeného filtru pro SSH (vnitřní síť je 195.41.41/24):

#### Příklad

```
access-list extended FiltOUT
permit tcp 195.41.41.0 0.0.0.255 any eq 22
deny ip any any

access-list extended FiltrIN
permit tcp any 195.41.41.0 0.0.0.255 eq 22 established
deny ip any any

interface serial 0
  ip access-group FiltOUT out
  ip access-group FiltrIN in
```

### FTP klient ve vnitřní síti

Protokol FTP je problém. Na aktivní FTP nepostačuje ani reflexivní filtr. Pokud chceme používat aktivní FTP, pak s filtrací nevystačíme – zde je jediný lék, proxy.

Použití pasivního FTP však také není bez potíží a asi by se v případě ochrany vnitřní sítě pouze filtrací také používat nemělo. Jenže internetové prohlížeče většinou využívají pasivní FTP a většina serverů pasivní FTP také podporuje, takže uživatelé vyvýjejí tlak na použití „alespoň“ pasivního FTP. Podle mne je rozumné povolit pasivní FTP pouze za využití reflexivního filtru.

Nyní musím vysvětlit, proč je FTP problémové. FTP používá dva kanály:

- ◆ Příkazový kanál, kde spojení navazuje klient. Příkazový kanál se chová jako protokol Telnet a jeho napadení je snadné.

- ◆ Datový kanál, který navazuje:
  - server v případě aktivního FTP,
  - klient v případě pasivního FTP.

V případě, že bychom filtrací povolili navazování datového kanálu z Internetu do vnitřní sítě (aktivní FTP), musíme otevřít možnost navázat spojení z Internetu na libovolný port větší než 1023 vnitřní sítě. Můžete namítnout, že se jedná o klientské porty, ale to není úplně pravda. Některé servery (zejména databáze) běží i na těchto „klientských“ portech. Vlámání se do databázového serveru je přitom jedním z obvyklých cílů hackerů.

Takže už o aktivním FTP a jeho případné filtrace nebudeme ani přemýšlet a podíváme se na pasivní FTP. Opět je zde problém pouze s datovým kanálem. Ten je sice navazován ze strany klienta, ale není navazován na nějaký všeobecně známý port serveru, ale na libovolný port větší než 1023, který serveru v daném okamžiku přidělí správa volných portů.

Pokud píšeme filtr pro pasivní FTP, napišeme filtr pro příkazový kanál (je navazován z portu většího než 1023 vnitřní sítě na port 21 serveru v Internetu) a dále musíme napsat filtr pro datový kanál, tedy propustit komunikaci z vnitřní sítě z libovolného portu většího než 1023 na libovolný port větší než 1023 – a to je pořádná díra ve filtru. To už nemáme filtr, ale cedník. Je pravdou, že u pasivního FTP lze u datového kanálu ještě kontrolovat, že spojení se navazuje z vnitřní sítě do Internetu a ne naopak (sledování příznaku ACK), ale stejně bych doporučoval se takovéto velké díře vyhnout.

Myslím si ale, že pokud se použije reflexivní filtr, bude ochrana ve většině konkrétních případů dostatečná.

## HTTP klient ve vnitřní síti

Protokol HTTP je na první pohled pro ochranu vnitřní sítě filtrací jako stvořený. Používá protokol TCP, kde klient je ve vnitřní síti a server v Internetu. Co více si můžeme přát?

Ovšem opět je tu nějaké „jenže“. Spočívá v tom, že na Internetu se nezřídka používají webové servery běžící na jiných portech, než je port 80/tcp. Firmy mívaly titulní stránky vystaveny zpravidla na standardním portu 80/tcp, ale nějakou aplikaci (která je zrovna pro klienta zajímavá) vystaví na portu např. 8080/tcp. Říkají si, vždyť my klienta neobtěžujeme, my ho nenutíme psát:

`http://www.firma.cz:8080`

my jsme to skryli do hypertextového odkazu. Klientovi stačí zmáčknout jen nějaké tlačítko... Jenže pokud je klient za filtrem, který předpokládá, že server běží na portu 80/tcp, zmáčknutí tlačítka je tím posledním, co klient mohl udělat, pak mu už aplikace nepoběží.

Další nevýhodou ochrany vnitřní sítě pomocí filtru v případě protokolu HTTP je, že filtr (na rozdíl od proxy) nemůže mít cache.

Opět obecný reflexivní filtr propouštějící spojení na libovolný port na Internetu se zdá být elegantním řešením, které obejde uvedené nedostatky (jen tu cache nevykouzlí).

## SSL/TLS klient

U protokolu HTTPS se jedná o stejnou situaci jako u protokolu HTTP.

Pro SSL (resp. TLS) se nepoužívá proxy – používá se tunel, který nemůže vidět do přenášených dat (jsou přece šifrována). Tunel tak nemůže mít cache. Proto z hlediska uživatele je filtrace např. refle-

xivním filtrem a použití tunelu přibližně stejné i z hlediska rychlosti odezvy. I z hlediska bezpečnosti v tom podle mého názoru není přílišný rozdíl.

### **SMTP klient ve vnitřní síti**

V tomto případě nemáme problém s filtrací (ve filtru uvedeném pro SSH stačí změnit port serveru na 25/tcp). Háček je v tom, že SMTP klient se dá použít pouze k odeslání elektronické pošty. Pro příjem pošty protokolem SMTP by ve vnitřní síti musel běžet SMTP server. To však filtrací nelze ošetřit.

Jinými slovy: Použití ochrany vnitřní sítě filtrací v případě protokolu SMTP znamená, že SMTP server nemůžeme mít ve vnitřní síti. Musíme použít SMTP server poskytovatele Internetu nebo umístit poštovní server na Internet či na DMZ. To však není odlišná podmínka od ostatních protokolů, ale pro management firmy je často nepříjemná.

### **Klient POP3 a klient IMAP4 ve vnitřní síti**

Zde opět není problém s filtrací. V tomto případě se předpokládá, že poštovní server je buď v Internetu, nebo na DMZ.

Jedná se o případ, kdy klient ze serveru vybírá poštu protokolem POP3 nebo IMAP4 a odesílá poštu protokolem SMTP. V obou případech se navazuje spojení ze strany klienta do Internetu a filtrace je velice jednoduchá.

Naopak s použitím proxy pro protokoly POP3 a IMAP4 mohou být potíže, protože tyto protokoly proxy přímo nepodporují. Naštěstí jsou natolik jednoduché, že je zvládne generická proxy. Avšak elegantním řešením pro POP3 a IMAP4 je transparentní proxy.

### **LDAP-klient ve vnitřní síti**

Přístup adresáře z vnitřní sítě na LDAP-server v Internetu lze opět dobře filtrovat. Naopak s použitím proxy pro protokol LDAP mohou být potíže, protože protokol LDAP proxy nepodporuje. Přitom podpora protokolu LDAP je většinou jednou ze základních funkcí internetových prohlížečů, které jsou od počátku upraveny pro práci s proxy. Nicméně i zde platí, že generická proxy poslouží dostatečně a situaci vyřeší. Opět elegantním řešením je transparentní proxy.

### **NNTP**

Přístup klienta protokolu NNTP na server v Internetu lze filtrovat filtrem popsaným pro SSH, změníme-li číslo portu na 119/tcp.

U protokolu NNTP snad ani nikdo nepředpokládá, že by měl server zapojený do celosvětové výměny NEWS na vnitřní síti. Znamenalo by to vyhradit linku o kapacitě alespoň 1 Mb/s jen pro tento účel.

Kdyby to bylo třeba, doporučoval bych přijímat světové NEWS ze satelitu. Bylo by to i lacinější a bez rizika. Odesílání NEWS vzniklých ve vnitřní síti je pak možné do Internetu realizovat, protože v tomto případě se systém chová jako klient.

## Servery

Zbývá se zastavit u problémů spojených s tím, že chceme mít ve firmě server, který je dostupný z Internetu. Jedná se zejména o servery protokolů: DNS, HTTP, HTTPS, SMTP, POP3 či IMAP4.

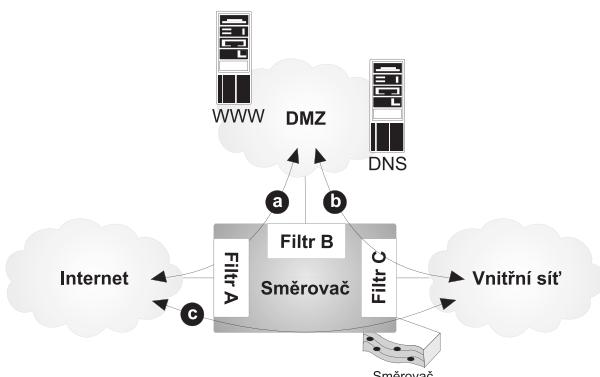
Obecně je nutno požadavek na umístění takových serverů do vnitřní sítě zamítnout (v případě, že se využívá výhradně ochrana filtrací). Výjimkou je použití reflexních filtrů.

Zámerně jsem však napsal, že požadavek je na umístění serveru v prostorách firmy – nikoliv ve vnitřní síti. Takový požadavek se totiž dá vyřešit pomocí DMZ. Stačí si obstarat směrovač ne s jedním sériovým rozhraním a jedním rozhraním pro LAN, ale s jedním sériovým rozhraním do Internetu a dvěma rozhraními pro LAN – jedno pro vnitřní síť a druhé pro DMZ.

Na DMZ pak umístíme server (obr. 18.16). Stačí fyzicky jeden server (např. server s Windows či Linuxem), na němž spustíme aplikační servery, které potřebujeme.

Budeme tak mít dvě lokální sítě: vnitřní síť a DMZ. Tyto sítě musí být fyzicky odděleny. Existují např. přepinače, na kterých můžete programově vytvořit dvě samostatné LAN. Toto řešení je nutné odmítnout, protože opomenutím správce se může nastavení vynulovat a hub vytvoří jednu lokální síť. Z hlediska uživatele to také pracuje, ale pro hackera to vytvoří ráj.

Máme-li na DMZ pouze jeden počítač a jako LAN používáme Ethernet TP, žádný hub nepotřebujeme. Stačí totiž použít křížený propojovací kabel.



**Obrázek 18.16:** Na server na DMZ se přistupuje jednak z Internetu přístupem (a) a jednak z vnitřní sítě přístupem (b). Některé jiné protokoly DMZ nevyužívají, proto jim filtry umožňují nutný přístup přímo do Internetu (c).

Na serveru na DMZ pak můžeme spustit:

- ◆ Poštovní server, tj. SMTP server, z něhož vybíráme poštu protokoly IMAP4 či POP3 a na který z vnitřní sítě odesíláme poštu protokolem POP3. Z vnitřní sítě používáme pro poštu pouze přístup (b) nakreslený na obr. 18.16. Sám poštovní server má konektivitu do Internetu pro protokol SMTP – přístup (c).
- ◆ Primární DNS server domény *firma.cz* (nebo alespoň pouze caching DNS server). To je zajímavé zejména v případě, že směrovač nepodporuje reflexivní filtry. Tím se totiž zamezí nutnosti povolit DNS komunikaci z vnitřní sítě do Internetu, která se filtry dá osetřit jen nedokonale (není-li k dispozici reflexivní filtr).

- ◆ Máme-li již vybudovaný takový server, pak bych na něm doporučil spustit jednoduché proxy pro HTTP, FTP a tunel pro HTTPS. Musí to být ale jednoduché proxy běžící na serverech majících jen jedno síťové rozhraní (např. MS proxy je pro tento případ příliš dokonalý nástroj, který vyžaduje dvě síťová rozhraní a pro tento případ se příliš nehodí).
- ◆ V takovém případě si již na nás server umístíme i webový server firmy, a pokud je třeba, tak i FTP server.

Naopak přímo do Internetu se filtrací propouští komunikace protokoly Telnet (resp. SSH) a LDAP.

## Závěr k filtraci

Závěr je tedy takový, že firma by si měla provést analýzu rizik, ze které vyplýne, zda je ochrana filtrace dostatečná, či nikoliv.

Z analýzy rizik pravděpodobně vyplýne, že ochrana vnitřní sítě pouze filtrace je dostatečný nástroj pro firmy, které nevyžadují ve svých prostorách provozovat servery dostupné z Internetu, tj. vystačí se servery poskytovatele Internetu nebo jinými servery v Internetu. V takovém případě je dobré připlatit si na systém směrovače, který podporuje reflexivní filtry, aby chom mohli s klidným spaním používat ve vnitřní síti i DNS alespoň pasivní FTP. Výhodou je, že poštu si můžeme vybírat jak z vnitřní sítě, tak i z Internetu.

V případě, že firma chce provozovat např. webový či poštovní server, umístí jej na DMZ. Toto řešení je v podstatě srovnatelné s firewallem. Bezpečnost tohoto řešení pak převážně závisí na nastavení bezpečnosti serveru na DMZ a správném nastavení přístupového směrovače (např. aby se směrovač nesprávoval protokolem Telnet z Internetu apod.).

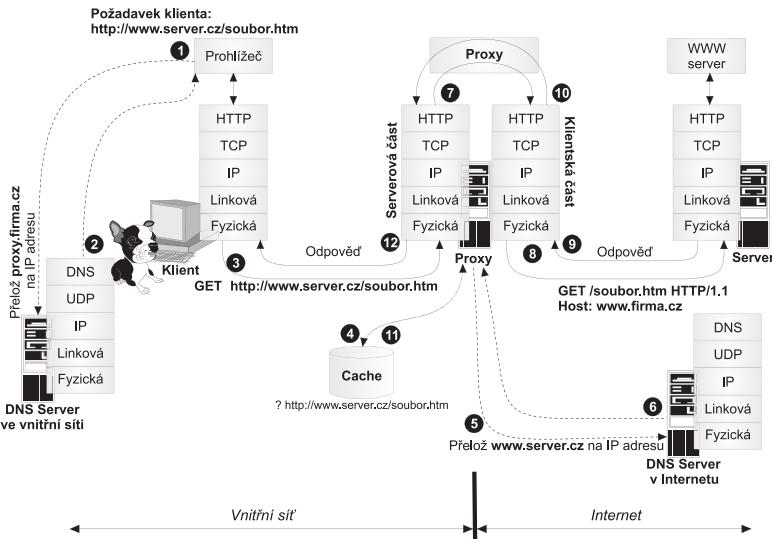
Zbývá jen dodat, že útočník nečihá pouze v Internetu, ale drtivá převaha útočníků hrozí z vnitřní sítě. Proto je nutné zabezpečit přístupový směrovač a server v DMZ i proti témtu útokům. Takže např. správu přístupového směrovače povolíme pouze z konzoly a server na DMZ spravujeme pouze protokolem SSH nebo z konzoly. Základním pravidlem DMZ je, že ze serverů umístěných v DMZ nepovolujeme navazovat spojení protokolem TCP do vnitřní sítě.

## Proxy

V 16. kapitole jsme se seznámili s proxy, bránou a tunelem v případě protokolu HTTP. Jedná se v podstatě také o filtry, které ale pracují na aplikační vrstvě. V této kapitole si uvedeme, že mechanismus proxy-brána-tunel se dá využít nejenom pro protokol HTTP. Problém je ale zejména u starších protokolů s tím, že musí být specifikován nejenom název cílového serveru, ale i mezilehlé proxy. Hledají se proto pro tyto protokoly náhradní řešení, kterými jsou „klasická proxy“, „generická proxy“ a „transparentní proxy“.

Vezměme si např. proxy v protokolu HTTP. Ta se skládá ze dvou částí. Na jedné straně pracuje jako server a na druhé straně jako klient. Serverová část proxy přijímá požadavky od klientů a předává je klientské části proxy, která jménem klientů předává požadavky cílovému serveru (obr. 18.17). Proxy si představujeme jako aplikaci běžící na počítači se dvěma síťovými rozhraními. Jedno síťové rozhraní je připojeno do vnitřní sítě a druhé do Internetu. Mnohdy jsou užitečné i jednodušší proxy, které používají pouze jedno síťové rozhraní.

Proxy se používá zejména na rozhraní dvou sítí, mezi nimiž není přímá konektivita. Proxy se pak používá jako oddělovač obou sítí. Z jedné sítě je tak možno připojit se na proxy a z ní pak dále do druhé sítě.



Obrázek 18.17: HTTP proxy (viz též obrázek 16.4)

Klasická představa proxy je počítač se dvěma sítěmi se síťovými rozhraními do každé z nich (např. jedno rozhraní do Internetu a druhé do vnitřní sítě). Cílem je, aby na vnitřním rozhraní pracovala serverová část proxy a na vnějším rozhraní klientská část. Nemusí to být jediným cílem. Někdy je požadováno, aby proxy pracovala i opačně, tj. aby pracovníci na cestách (klienti) si mohli např. vybírat poštu z vnitřní sítě. Tako pracující proxy je však nebezpečná. Pokud se vůbec něco takového povolí, pak zpravidla v kombinaci s autentizací jednorázovým heslem před propuštěním klienta do vnitřní sítě. Klienti bývají pro tyto účely vybavováni autentizačními kalkulátory. Dnes se stále více takovéto řešení opouští a nahrazuje řešením na bázi VPN.

Proxy pracuje tak, že serverová část proxy předává požadavky klientské části proxy a opačně. Při předávání požadavků mezi oběma částmi proxy nemusí dojít pouze k mechanickému předávání, ale předávání může být propracovanější. Zajímavé jsou zejména dvě následující vlastnosti:

- ◆ **Filtrace.** Při předávání může být definována sada pravidel specifikující, co se může a co nesmí předávat. Může se podobně jako při filtraci na směrovačích povolovat komunikace skrze proxy jen některým počítačům. Zajímavější je však typ filtrace, který na směrovačích není možný. Směrovač totiž nevidí do aplikačního protokolu, kdežto proxy ano. Takže proxy může filtrovat takové jemnosti, jako je povolování některých aplikačních příkazů a zakazování jiných či filtrování obsahu, jako jsou aplikace ActiveX a JavaScript. Běžné je také udržování černé listiny serverů, kam si zaměstnavatel nepřeje, aby jeho zaměstnanci přistupovali. Jedná se však většinou o zcela zbytečnou práci. Často bývá na černé listině např. `www.playboy.com`. Zaměstnavatel si myslí, že tak na zaměstnance vyzrál. Ale když někdo na Internetu něco najít chce, tak si najde vždy nějakou alternativu – nějaký jiný zdroj. Takže proxy může např. pro protokol HTTP určovat, které počítače mohou používat metodu GET a na jaká URL. Jiné počítače zase mají povolenu metodu PUT či POST. Obdobně u protokolu FTP je možné povolit příkaz GET, ale zakázat příkaz PUT, aby zaměstnanci vnitřní sítě nemohli zcizit data.
- ◆ **Cache.** Proxy si může zpracované požadavky a jejich odpovědi ukládat na disk. V případě opakování stejného požadavku pak proxy může odpovědět přímo z cache. Pochopitelně správce proxy může nastavit, bude-li např. odpovídat pouze z proxy, nebo ověřovat, zda nedošlo ke změně dat na originálním serveru atp. Jenže dnes v době dynamických stránek bývají v cache často firemní logo.

Některé aplikační protokoly ve své podstatě pracují jako proxy. Není třeba pro tyto protokoly vyvíjet nějaké speciální proxy – místo nich stačí použít server pro příslušný protokol. Jedná se např. o protokoly:

- ◆ SMTP – mezi dvě sítě (jejichž uživatelé nemají vzájemnou konektivitu) umístíme poštovní server. Pak se e-mailsy z jedné sítě směrované do druhé sítě odesírají na uvedený poštovní server, který je posílá dálé. Tak ale přece pracovaly už sítě UUCP (Unix to Unix copy), které byly před Internetem, tj. před protokoly TCP/IP.
- ◆ NNTP – komunikace mezi NEWS-servery. Princip je stejný jako u protokolu SMTP.
- ◆ NTP – Máme-li mezi Internetem a intranetem časový server, pak komunikace z Internetu může jít na tento časový server, který synchronizuje své hodiny. Vnitřní síť si nastavuje hodiny podle tohoto serveru. Není třeba komunikace protokolem NTP skrze tento server.
- ◆ DNS – této problematice se budeme věnovat v kap. 19 Firewally.

Přesto pracují-li servery zejména protokolů SMTP a DNS přímo na firewallu, pak – jak se ukáže později – s tím mohou nastat v případě protokolu SMTP bezpečnostní a v případě DNS technické potíže. Proto se na firewallech používají tzv. SMTP proxy a DNS proxy, mající specifický význam. Serverová část těchto proxy je spuštěna místo příslušného serveru, tj. SMTP proxy je spuštěna jako server na portu 25/tcp a DNS proxy je spuštěna na portech 53/udp a 53/tcp. Úkolem těchto proxy je prolustrovat příchozí požadavky, zda jsou oprávněné, či nikoliv, a pouze oprávněné požadavky přejdají příslušnému serveru k provedení. Příslušný server spuštěný na jiném než standardním portu pak přijímá požadavky pouze z IP adresy 127.0.0.1 (port větší než 1023) zasláné na adresu 127.0.0.1 a port, na kterém je spuštěn příslušný server. Alternativně může být skutečný server umístěn na jiném počítači na vnější síti. V případě protokolu SMTP může SMTP proxy zapsat příchozí poštovní zprávu přímo do fronty, tj. SMTP server není vůbec nutný. SMTP agent se pak startuje jako klient.

Zatímco u protokolů HTTP a HTTPS je proxy součástí jejich definice a u protokolů SMTP, NNTP, NTP a DNS již víme, jak na to, u protokolů Telnet, SSH, FTP, POP3, IMAP4, LDAP a vlastnoručně naprogramovaných aplikací nevíme, jak proxy postavit. U těchto protokolů je zásadní problém v tom, jak sdělit proxy jméno cílového serveru, na který má klientská část proxy navázat spojení. Řešení je v podstatě trojí:

- ◆ klasická proxy,
- ◆ generická proxy,
- ◆ transparentní proxy.

Stejný problém je i s protokoly SMTP a NNTP v případě, že na proxy nechceme spustit příslušný server, ale pouze chceme spustit jednoduchou proxy, která by umožnila přístup klientům vnitřní sítě např. na server poskytovatele Internetu.

## Klasická proxy

S klasickou proxy se nejspíše setkáte pro protokol FTP nebo TELNET. Představte si, že sedíte u klasického klienta pro TELNET.

Proxy je spuštěna na počítači *proxy.firma.cz* a její serverová část očekává požadavky od klientů na portu např. 1500. Na příkazový řádek tedy zadáte:

```
telnet  
telnet> open proxy.firma.cz 1500
```

Proxy vám odpoví:

\*\*\* Vitame Vas na proxy.firma.cz \*\*\*

Nyní potřebujete nějak sdělit své proxy jméno vzdáleného počítače, se kterým má její klientská část vaším jménem navázat spojení. A TELNET žádný takový příkaz nemá. Jenže u protokolu TELNET (i u FTP) se to dá udělat jednoduše – serverová část proxy jako server rozšíří množinu příkazů o příkaz:

`connect server.firma.cz`

Uživatel pak pracuje tak, že se nejprve programem Telnet připojí na proxy, kde zadá příkaz `connect` (zkráceně jen `c`), ve kterém specifikuje název nebo IP adresu skutečného serveru. Klientská část proxy naváže spojení s cílovým serverem zadaným v příkazu `connect` a začne předávat mezi klientem a tímto serverem data. Poté se klientovi spojení již jeví, jako by mezi ním a skutečným serverem žádná proxy nebyla.

Tabulka vyjadřující IP adresy a porty, které jsou použity v příkladu pro klasické proxy (v záhlaví IP datagramů jsou vždy IP adresy, přesto jsem v tabulce uvedl jména počítačů, aby byla tabulka snadno srozumitelná):

| Směr           | IP adresa odesílatele | Port odesílatele | IP adresa příjemce | Port příjemce | Proxy | IP adresa odesílatele | Port odesílatele | IP adresa příjemce | Port příjemce |
|----------------|-----------------------|------------------|--------------------|---------------|-------|-----------------------|------------------|--------------------|---------------|
| <b>ven</b>     | klient                | >1023            | proxy.firma.cz     | 1500          | →     | proxy.firma.cz        | >1023            | server             | 23            |
| <b>dovnitř</b> | proxy.firma.cz        | 1500             | klient             | >1023         | ←     | server                | 23               | proxy.firma.cz     | >1023         |



**Poznámka:** Co by se stalo, kdybyste nezadalí v příkazu Telnet port 1500? Použil by se pro protokol Telnet standardní port 23. V takovém případě by se na počítači `proxy.firma.cz` aktivoval přímo server pro protokol Telnet, tj. po přihlášení byste mohli pracovat jako interaktivní uživatelé počítače `proxy.firma.cz`. V praxi je to však málo pravděpodobné, protože správce proxy neumožní spouštět server protokolu Telnet (nanejvýš SSH).

Jednoduché je používat klasickou proxy pro řádkové klienty. Klienti FTP používající okna musí být upraveni pro takovouto práci s proxy a poprvadě řečeno většina dnešních grafických klientů FTP umožňuje vybrat z mnoha dialogů proxy nejrůznějších výrobců.

## Generická proxy

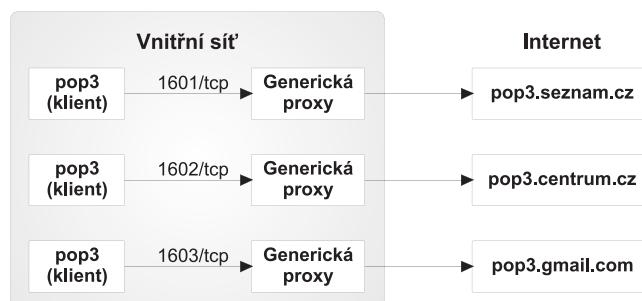
Mnoho klientů však není naprogramováno tak, aby mohly vést počáteční dialog s proxy, tj. aby jí sdělily IP adresu nebo jméno serveru, se kterým chtějí komunikovat.

Jedním z řešení je pak generická proxy. Generická proxy je obecný program, který může správce konfigurovat a spouštět podle potřeby. Generická proxy existuje nejenom pro protokol TCP, ale i pro protokol UDP!

Serverová část generické proxy je spuštěna (očekává požadavky od klientů vnitřní sítě) na konkrétním správcem nakonfigurovaném portu. Klientská část generické proxy je pevně nastavena na jeden cílový server.

Chceme-li umožnit klientům vnitřní sítě přístup na více serverů, pak musíme spustit generickou proxy pro každý cílový server zvlášť. Každá spuštěná proxy pak očekává požadavky na jiném portu.

Jinými slovy: Jelikož neumíme sdělit jméno a port cílového serveru, proxy je natvrdo nasměrována na konkrétní cílový server. Pokud chcete přistupovat na více serverů, pak co server, to jedna spuštěná proxy. Každá proxy je pak spuštěna na svém (jiném) portu.



**Obrázek 18.18:** Generická proxy pro POP3

Jako příklad si zvolme generickou proxy pro POP3. Na obrázku 18.18 je znázorněno, že chce-li klient stahovat poštu ze serveru *pop3.seznam.cz*, pak musí kontaktovat proxy na portu 1601, chce-li komunikovat se serverem *pop3.centrum.cz*, pak musí skrze port 1602. Obdobně server *pop3.gmail.com* je dostupný přes port 1603 (čísla portů byla zvolena náhodně).

Generická proxy nepotřebuje žádnou úpravu na straně klienta ani serveru a je vhodná zejména pro protokoly:

- ◆ POP3, IMAP4, SSH, LDAP a ostatní již hotové aplikace nepodporující proxy. Avšak pro každý server v Internetu je třeba aktivovat jednu generickou proxy.
- ◆ Protokoly SMTP a NNTP (*Network News Transfer Protocol*), nechceme-li spustit servery těchto protokolů, ale chceme-li umožnit klientům vnitřní sítě přístup na konkrétní servery v Internetu.



**Poznámka:** V parametrech při spuštění generické proxy zadávají parametry klientské části proxy, čili ty informace, které se u klasické proxy zadávají v úvodním dialogu při navazování spojení, tj. jméno a port cíleného serveru. Generická proxy předpokládá, že takový dialog není možný, proto jsou tyto parametry nastaveny „natvrdo“ v konfiguračním souboru a generická proxy je tak vždy nasměrována na jeden konkrétní počítač.

## Transparentní proxy

O transparentních proxy pojednává norma RFC-1918. Transparentní proxy funguje z pohledu uživatele jako směrovač. Základní předpoklady pro práci transparentní proxy jsou:

- ◆ Klient ve vnitřní síti má možnost přeložit jméno cílového serveru na jeho IP adresu v Internetu, tj. není oddělené DNS pro vnitřní síť a pro Internet. Klient ve vnitřní síti musí mít možnost přeložit jméno libovolného počítače z Internetu na IP adresu. Klient vyrobí IP datagram, kde adresa příjemce už odpovídá adrese cílového serveru – nikoliv IP adrese proxy.
- ◆ IP datagram odeslaný klientem do Internetu musí být vnitřní síť směrován skrze transparentní proxy.
- ◆ Mezi Internetem a vnitřní sítí může být sice více transparentních proxy, ale všechna dostupná spojení musí být vedena přes jednu transparentní proxy. Není možné, aby IP datagramy jednoho spojení procházely přes různé transparentní proxy.

Napsal jsem, že transparentní proxy se chová jako směrovač, ale ve skutečnosti je to velice nestandardní směrovač. Klasický směrovač IP datagramy, které přichází na jeho rozhraní, předává na jiné rozhraní. Transparentní proxy však příchozí pakety akceptuje, jako by byly určeny přímo pro ni. To znamená, že z IP datagramu vybalí TCP paket a vůči klientovi se tváří, že je tento TCP paket určen pro ni. Klient naváže spojení s transparentní proxy a má pocit, že navázal spojení se skutečným serverem. Transparentní proxy však obratem navazuje svou klientskou částí spojení s cílovým serverem.

S transparentní proxy nemusí vést klient žádný dialog. Transparentní proxy získá IP adresu a port cílového serveru z IP datagramu, který obdržela její serverová část od klienta.

Transparentní proxy může mít i tabulkou, kde je uvedeno, od koho a kam může akceptovat spojení. Dále může mít tabulku uvádějící transformaci IP adres či portů mezi tím, co obdrží od klienta, a IP adresou či portem, na který bude skutečně její klientská část navazovat spojení.

Transparentní proxy se nejčastěji používají pro protokoly TELNET, FTP a pro komunikaci klienta se serverem protokoly NNTP, IMAP, POP a POP3.

U klasické proxy jsme si uvedli tabulkou IP adres a portů pro protokol Telnet. V případě transparentní proxy klient použije v IP datagramu přímo adresu originálního serveru a port 23 (TELNET).

Tabulka má pak tvar:

| <b>Směr</b>    | <b>IP adresa odesilatel</b> | <b>Port odesilatele</b> | <b>IP adresa příjemce</b> | <b>Port příjemce</b> | <b>Proxy</b> | <b>IP adresa odesilatel</b> | <b>Port odesilatele</b> | <b>IP adresa příjemce</b> | <b>Port příjemce</b> |
|----------------|-----------------------------|-------------------------|---------------------------|----------------------|--------------|-----------------------------|-------------------------|---------------------------|----------------------|
| <b>ven</b>     | Klient                      | >1023                   | server                    | 23                   | →            | proxy.<br>firma.cz          | >1023                   | server                    | 23                   |
| <b>dovnitř</b> | Server                      | 23                      | klient                    | >1023                | ←            | server                      | 23                      | proxy.<br>firma.cz        | >1023                |

(V záhlaví IP datagramů jsou opět uvedeny IP adresy, přesto jsem v tabulce uvedl jména počítačů, aby byla tabulka snadno srozumitelná.)

Zamyslite-li se nad předchozí tabulkou, pak se vlastně transparentní proxy chová z hlediska klienta jako směrovač s filtrem a z hlediska cílového serveru jako klasická proxy.

Rozdíly mezi klasickou a transparentní proxy:

| Problém                | Klasická proxy                                                                                                                                                     | Transparentní proxy                                                                                                     |
|------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------|
| IP adresace            | Všechny sítě musí umět adresovat proxy.<br><i>firma.cz.</i>                                                                                                        | Klient musí umět adresovat cílový server.                                                                               |
| DNS                    | Je možná úplná izolace intranetu a Internetu.                                                                                                                      | V intranetu musí být převozitelná jména z Internetu. Opačně to není třeba.                                              |
| Směrování IP datagramů | Směrování v intranetu a Internetu může být nezávislé, ale v obou sítích musí být dostupná <i>proxy.firma.cz.</i>                                                   | Adresa cílového serveru musí být přepravitelná už od klienta vnitřní sítě.                                              |
| Skrytí IP adres        | IP adresy Internetu jsou skryty před adresami intranetu a naopak IP adresy intranetu jsou skryty před Internetem. Vnitřní síť může např. použít adresu 10.0.0.0/8. | IP adresy intranetu jsou skryty před Internetem, opačně tomu tak není. Vnitřní síť může např. použít adresu 10.0.0.0/8. |
| Software pro proxy     | Běžný software, který je přenositelný na úrovni zdrojových textů programu.                                                                                         | Speciální software, s jehož přenositelností na jiné platformy jsou potíže – nutné úpravy do zdrojových textů.           |
| Uživatelský software   | Musí být konfigurovatelný, aby se dalo zadat jméno a port proxy. Dále musí umět vést počáteční dialog s proxy.                                                     | Stačí běžný software.                                                                                                   |
| Znalosti uživatele     | Musí si osvojit počáteční dialog s proxy (pokud to za něho nedělá aplikace).                                                                                       | Žádné další znalosti nejsou třeba.                                                                                      |

## Autentizace

Proxy, tunel i brána zpravidla provádí autentizaci klienta před tím, než mu poskytnou/zamítnou své služby. Autentizace může být na základě:

- ◆ IP adresy klienta – na proxy může být vedena tabulka IP adres jednotlivých klientů, kteří mohou používat proxy. Nebo se vedou adresy sítí, ze kterých mohou klienti proxy používat (např. síť 10.0.0.0/8).
- ◆ Jména a hesla.
- ◆ Jednorázového hesla generovaného např. autentizačním kalkulátorem.
- ◆ Protokolu Kerberos atd.

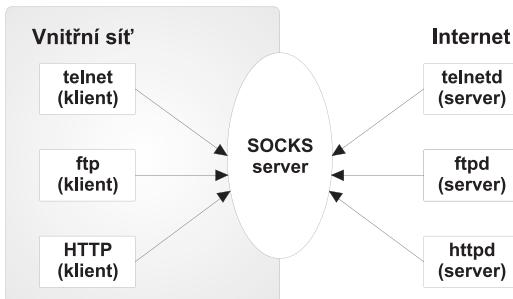
## SOCKS

Doposud jsme pro každý aplikační protokol spouštěli samostatnou proxy. Cílem protokolu SOCKS je jedna aplikace, která bude společná pro všechny proxy, tunely a brány.

Proxy řeší dva problémy:

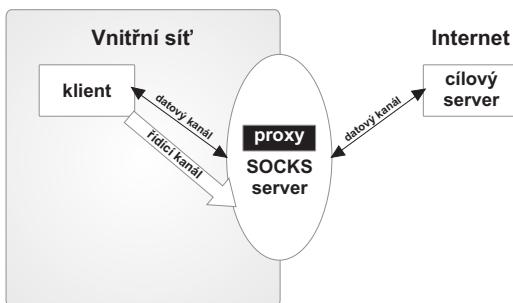
1. Autentizaci klienta. Pouze pro autentizované klienty se proxy otevře.
2. Zajištění přenosu informací mezi vnitřní sítí a Internetem.

SOCKS řeší problematiku proxy včetně autentizace zcela obecně a nezávisle na použitém operačním systému. SOCKS-server je proxy, která je společná pro všechny aplikační protokoly (obr. 18.19).



Obrázek 18.19: SOCKS-server

SOCKS je protokol, kterým komunikují klienti se SOCKS-serverem. Cílem této komunikace je otevřít příslušnou proxy pro daný požadavek. Klient zřídí řídicí kanál se SOCKS-serverem (zpravidla na portu 1080/tcp). Dialogem vedeným v řídicím kanálu server sdělí klientovi IP adresu a port serverové části proxy, kterou otevřívá pro klienta. Klient pak na uvedenou IP adresu a port naváže datové spojení protokolem TCP, jako by navazoval spojení s cílovým serverem v Internetu. Pokud se klient se SOCKS-serverem dohodl na UDP proxy, pak klient na uvedenou adresu a port protokolu UDP odešle datagram, který proxy předá do Internetu (obr. 18.20).



Obrázek 18.20: Řídicí proxy a datový kanál

Protokol SOCKS verze 5 zavedený v RFC-1928 řeší celou problematiku komplexně ve čtyřech krocích:

- 1. Dohoda na autentizační metodě.** Klient nabídne SOCKS-serveru autentizační metody, které podporuje. SOCKS-server z nich vybere jednu. Klient se pak na základě této metody autentizuje. Teoreticky v sobě může metoda zahrnovat i požadavek na šifrování a elektronické podepisování přenášených dat mezi klientem a SOCKS-serverem. V takovém případě po prokázání totožnosti běží veškerý provoz zabezpečeně (šifrován, případně elektronicky podepisován). Požadavek na zabezpečení dat mezi klientem a SOCKS-serverem je důležitý zejména v případech, kdy by bylo od SOCKS-serveru požadováno, aby umožňoval autentizovaným klientům Internetu přístup do vnitřní sítě.

2. **Autentizační dialog.** Autentizační dialog závisí na zvolené metodě. Tč. je normalizová metoda autentizace pomocí jména a hesla uživatele (RFC-1929), metoda GSS-API (RFC-1961) a pochopitelně metoda bez autentizace. Metoda bez autentizace je metoda, kdy od klienta není vyžadován žádný autentizační dialog. Avšak SOCKS-server je přesto schopen provést autentizaci na základě IP adresy klienta.
3. **Požadavek na zřízení příslušné proxy.** Teprve po autentizačním dialogu SOCKS-server přijme od klienta požadavek, na jehož základě zřídí příslušnou proxy. Klient protokolem SOCKS předá požadavky SOCKS-serveru. Klient vydává požadavek CONNECT či BIND pro zřízení proxy na bázi protokolu TCP a požadavek ASSOCIATE pro zřízení UDP proxy (*UDP relay*).
4. **Datová komunikace skrze proxy se vzdáleným serverem.**

Jinými slovy: Klient (např. program Telnet) musí nejprve vést dialog se SOCKS-serverem, kterému sdělí, na jaký server v Internetu se chce propojit. Volitelně je možná autentizace klienta. Podstatné je, že klient vede se SOCKS-serverem dialog. Programy na straně klienta (např. telnet) musí být upraveny tak, aby uměly vést tento dialog – tj. programy klientů musí být upraveny, aby komunikovaly se SOCKS serverem.



**Poznámka:** V konfiguraci webového prohlížeče můžete zvolit buď proxy-bránu-tunel, nebo SOCKS server. Volba SOCKS-serveru vyloučí tedy použití proxy, brány nebo tunelu!

Jsou vytvořeny knihovny podporující SOCKS-protokol, které jsou obdobou standardních socketových knihoven používaných při programování v prostředí TCP/IP. Stačí ve zdrojovém textu programu nahradit volání standardních funkcí socketových knihoven těmito voláními. Přitom obojí volání mají stejnou syntaxi. Stačí pouze mechanicky zaměnit názvy funkcí:

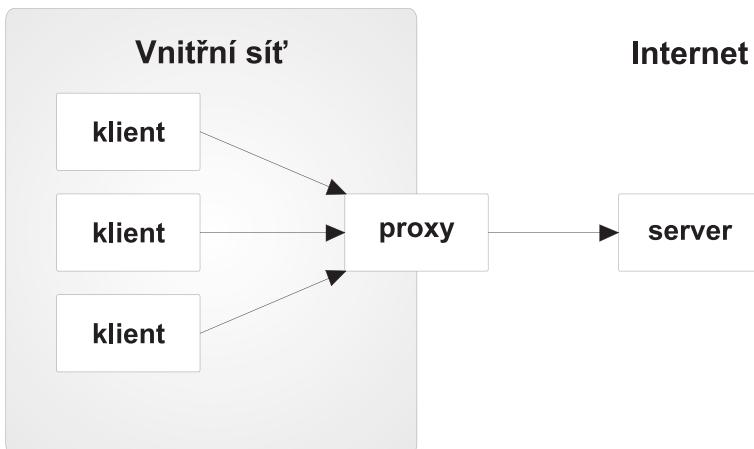
| Standardní knihovny | Knihovny SOCKS |
|---------------------|----------------|
| connect()           | Rconnect()     |
| getsockname()       | Rgetsockname() |
| bind()              | Rbind()        |
| accept()            | Raccept()      |
| listen()            | Rlisten()      |
| select()            | Rselect()      |

V podstatě ani není třeba opravovat zdrojové texty programů, protože uvedenou substituci je možno zadat při překladu programu (tj. specifikovat v souboru Makefile). Máte-li zdrojový text programu klienta, pak jej znova přeložíte s uvedenou substitucí. Získáte tak klienta pracujícího se SOCKS. Dnešní klienti jsou již konfigurovatelní tak, že je možné zvolit komunikaci přes SOCKS-server.

## Skryté sítě

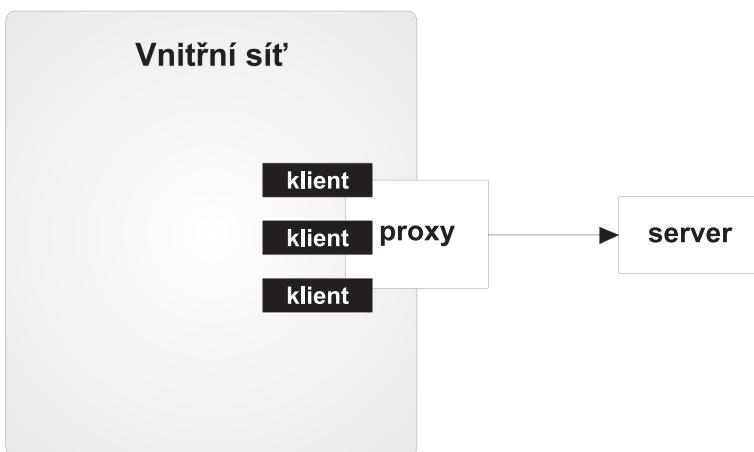
V případě, že používáme proxy, navazují se dvě TCP spojení (obr. 18.21):

1. Mezi klientem a proxy.
2. Mezi proxy a cílovým serverem.



**Obrázek 18.21:** Při proxy se navazují dvě TCP spojení

Z pohledu Internetu (např. z hlediska cílového serveru) není vnitřní síť vidět – je skryta. Klienti vnitřní sítě se jeví, jako by byli lokálními klienty proxy (obr. 18.22).



**Obrázek 18.22** Z hlediska Internetu se proxy jeví, jako by byla nějakým velkým počítačem s velkým množstvím terminálů a všichni klienti vnitřní sítě seděli u těchto terminálů

Spojení klienta s proxy může být realizováno protokolem TCP/IP, nebo dokonce i jinými protokoly, jako např. IPX/SPX apod. Budeme si však i pro spojení na vnitřní síti nadále představovat použití protokolu TCP/IP.

Z pohledu Internetu na IP adresách klientů nezáleží – jsou skryty za proxy. Avšak z hlediska proxy bychom ji zbytečně přivedli do svízelné situace, kdyby měl klient stejnou IP adresu jako některý z cílových serverů. Proto je důležité volit pro klienty vnitřní sítě IP adresy takové, které se na Internetu nevyskytují.

I když je ve světě mnoho vnitřních sítí (intranetů), klienti jednotlivých intranetů mezi sebou většinou nenavazují žádné spojení, tj. všechny intranety mohou používat stejné IP adresy. Pro intranety byly přiděleny následující intervaly IP adres:

10.0.0.0 až 10.255.255.255  
172.16.0.0 až 172.31.255.255  
192.168.0.0 až 192.168.255.255

Stejná úvaha platí i v případě, že pro oddělení vnitřní sítě od Internetu nepoužíváme proxy, ale filtrace s NAT.

## NAT

Proxy pracující na aplikační vrstvě nám dokáže skrýt vnitřní podnikovou síť před Internetem. Nešlo by to však udělat bez proxy? Co takhle přímo na směrovači převádět IP adresy? Když dokáže směrovač přepsat položku TTL v záhlaví IP datagramu, neuměl by přepsat IP adresu odesilatele a příjemce?

Řešením je „*Network Address Translator*“ (NAT), který takové převody realizuje. NAT specifikuje RFC-1631 a dále následující normy: RFC-2663, RFC-2709, RFC-2766 a RFC-2993.

NAT se konfiguruje jako funkce přístupového směrovače do Internetu, tj. směrovače na straně firmy (nikoliv směrovače na straně poskytovatele Internetu). Nadále budeme předpokládat, že naše vnitřní síť je propojena s Internetem pouze jedním směrovačem. V případě, že máme spojení vnitřní sítě s Internetem např. zdvojeno, je třeba velice úzkostlivě dbát na to, aby oba směrovače měly identickou konfiguraci. Avšak i tehdy, když mají oba směrovače identickou konfiguraci, některé dále popsané funkce jsou v případě dvou směrovačů problematické. Jedná se zejména o případ, kdy jsou IP adresy přidělovány dynamicky.

Pro vnitřní síť budeme používat adresy ze sítě 10.0.0.0/8. Poskytovatelem Internetu nám je přidělena síť 195.47.40.0/24.

### Jednoduchý NAT

Jednoduchý NAT je určen pro navazování relace z vnitřní sítě do Internetu. Ve vnitřní síti jsou směrovatelné nejenom adresy vnitřní sítě, ale i adresy Internetu. Směrovač má k dispozici blok adres přidělený poskytovatelem Internetu. V odchozích paketech pak směrovač přepisuje adresy vnitřní sítě na adresy z uvedeného bloku. U příchozích paketů pak zase adresy přepisuje opačně.

Pro jednoduchý NAT stačí mít ve směrovači převodní tabulkou („tabulku NAT“) obsahující jako položky vždy IP adresu vnitřní sítě a IP adresu, na kterou se má adresa vnitřní sítě konvertovat do Internetu.

Příklad tabulky NAT:

| IP adresa vnitřní sítě | Celosvětově jednoznačná IP adresa |
|------------------------|-----------------------------------|
| 10.0.0.3               | 195.47.40.13                      |
| 10.0.0.4               | 195.47.40.14                      |
| 10.0.0.5               | 195.47.40.18                      |

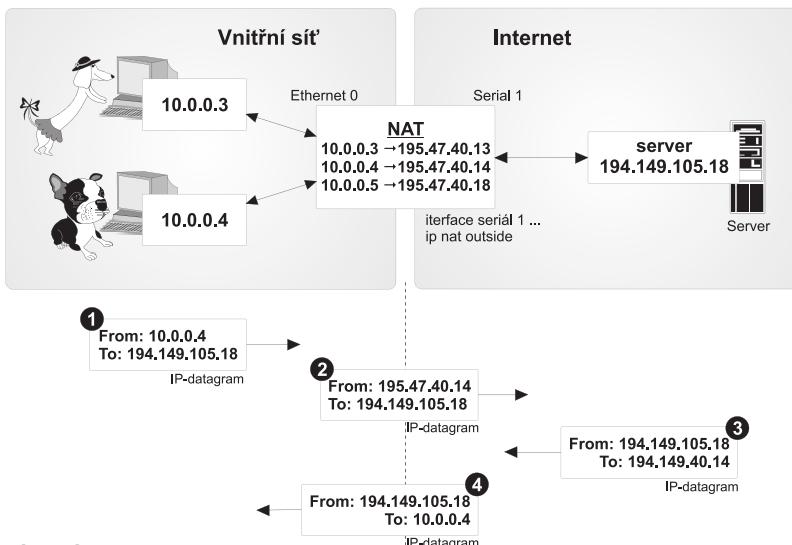
Z hlediska Internetu se počítač ve vnitřní síti o IP adrese 10.0.0.4 jeví, jako by měl IP adresu 195.47.40.14. V terminologii CISCO by překlad pomocí statické tabulky NAT vypadal např.:

### Příklad

```
ip nat inside source static 10.0.0.3 195.47.40.13
ip nat inside source static 10.0.0.4 195.47.40.14
ip nat inside source static 10.0.0.5 195.47.40.18

interface ethernet 0 ...
ip nat inside

interface seriál 1 ...
ip nat outside
```



Obrázek 18.23: NAT

NAT-tabulka může obsahovat statické položky, kdy je IP adresa vnitřní sítě jednoznačně trvale přiřazena celosvětově jednoznačná IP adresa. Kromě toho může mít směrovač k dispozici i pytel (*pool*) IP adres, které přiděluje dynamicky – V terminologii CISCO bychom dynamický překlad adres z vnitřní sítě na adresy 195.47.40.16 až 195.47.40.31 provedli – (pomocí příkazu access-list sdělujeme, jaké IP adresy vnitřní sítě mají přístup do Internetu):

### Příklad

```
ip nat pool pytel 195.47.40.16 195.47.40.31 netmask 255.255.255.240
access-list 1 permit 10.0.0.0 0.255.255.255
ip nat inside source list 1 pool pytel
interface serial 1 ...
ip nat outside
interface ethernet 0 ...
ip nat inside
```

Stačí tedy vzít několik přidělených IP adres od poskytovatele a dát je do pytle. Pytel adres pak stačí předat směrovači. Jestliže klient začne navazovat spojení se serverem v Internetu, přidělí se mu dynamicky z pytle jedna adresa. Po ukončení spojení se adresa vrací zpět do pytle.

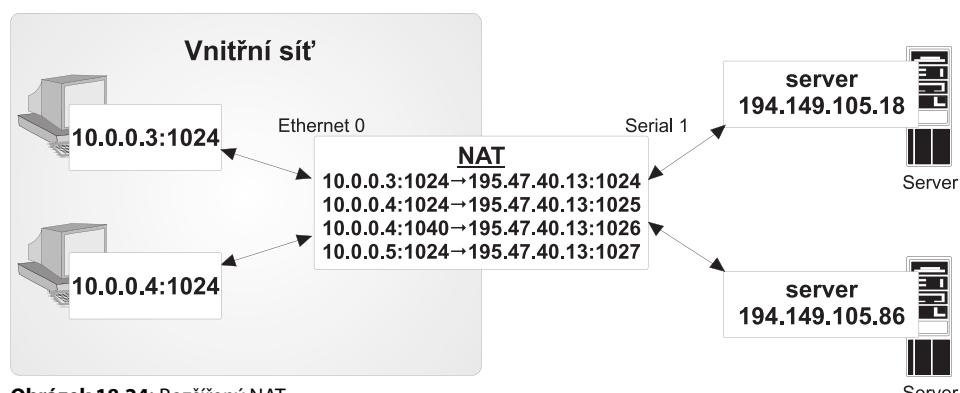
Pro dynamické přidělování IP adres potřebujeme mít v pytle připraveno kolik IP adres, kolik může v jednom okamžiku přistupovat do Internetu klientů.

## Rozšířený NAT

Rozšířený NAT neboli „*Network Address and Port Translation*“ (NAPT) umožňuje šetřit přidělené IP adresy. Umožňuje totiž, aby více IP adresám ve vnitřní síti byla přidělena z bloku adres od poskytovatele Internetu jedna společná IP adresa.

NAPT používá položky tabulky NAT rozšířené o čísla portů a typ protokolu. Na obr. 18.24 je znázorněn příklad takové komunikace. Počítač 10.0.0.3 chce z portu 1024 navázat spojení protokolem TCP na server 194.149.105.18. Pomocí rozšířené tabulky NAT je IP adresa odesilatele přeložena z 10.0.0.3 na 195.47.40.13 a port je ponechán 1024. Během tohoto spojení chce počítač 10.0.0.4 z portu 1024 navázat spojení se serverem 194.149.105.86. IP adresa 10.0.0.4 je přeložena rovněž na adresu 195.47.40.13, ale port 1024 je již použit, takže je přeložen na 1025.

Následující tabulka znázorňuje zmíněnou situaci, kdy jsou různí klienti vnitřní sítě překládáni na stejnou IP adresu, přesto je spojení určeno jednoznačně – obě spojení se liší číslem portu klienta.



Obrázek 18.24: Rozšířený NAT

| Protokol | IP adresa a port vnitřní sítě | Celosvětově jednoznačná IP adresa a port |
|----------|-------------------------------|------------------------------------------|
| TCP      | 10.0.0.3:1024                 | 195.47.40.13:1024                        |
| TCP      | 10.0.0.4:1024                 | 195.47.40.13:1025                        |
| TCP      | 10.0.0.4:1040                 | 195.47.40.13:1026                        |
| UDP      | 10.0.0.5:1024                 | 195.47.40.13:1027                        |

Z hlediska konfigurace směrovačů CISCO dochází pouze k přidání jediného slova „overload“:

### Příklad

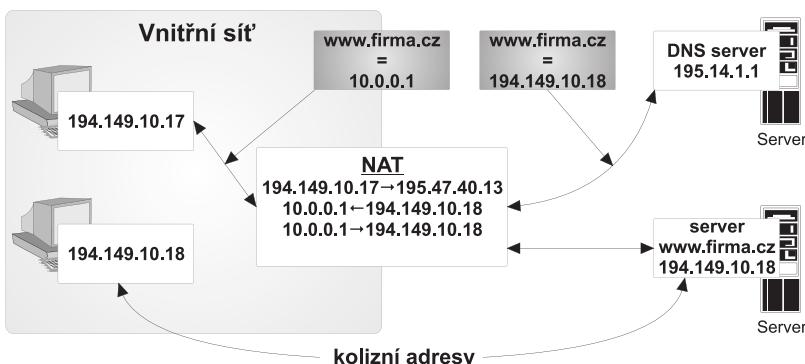
```
ip nat pool pytel 195.47.40.16 195.47.40.31 netmask 255.255.255.240
access-list 1 permit 10.0.0.0 0.255.255.255
ip nat inside source list 1 pool pytel
ip nat outside
```

```
interface serial 1 ...
ip nat inside
```

Rozšířený NAT je oblíbený na Linuxu, kde se označuje jako maškaráda.

### Dvojitý NAT

Dvojitý NAT slouží k řešení kolizních IP adres, tj. situace, kdy ve vnitřní síti používáme IP adresy, které jsou v Internetu přiděleny někomu jinému, tzv. „Overlapping Addresses“. V tomto případě již obecně nelze ve vnitřní síti přepravovat libovolné IP adresy Internetu. Někdy se tento typ překladu označuje INAT podle „Illegal NAT“.



Obrázek 18.25: Dvojitý NAT

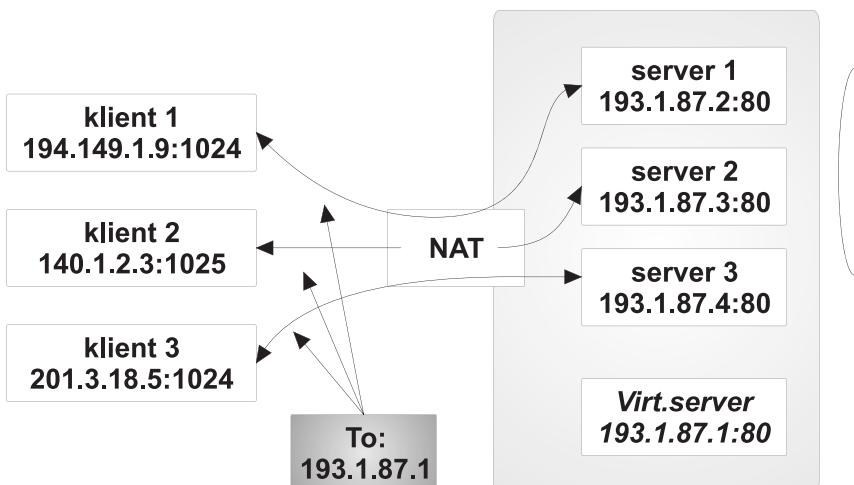
Ve vnitřní síti používáme interval IP adres 194.149.10.0/24 přidělený jiné firmě. Shodou okolností chceme komunikovat s webovým serverem *www.firma.cz*, který má IP adresu 194.149.10.18, která je v kolizi s IP adresami vnitřní sítě.

Prvním krokem je vždy překlad jména (*www.firma.cz*) na IP adresu pomocí DNS serveru. Klient vygeneruje příslušný DNS dotaz. IP adresa odesilatele (194.149.10.17) musí být v dotazu přeložena směrovačem na adresu 195.47.40.13, která je z bloku adres oficiálně přidělených poskytovatelem Internetu. Nyní DNS dotaz dorazí na DNS server, který vrací odpověď. V DNS odpovědi je ale uvedena IP adresa 194.149.10.18, která je v kolizi s IP adresami vnitřní sítě. Proto směrovač vezme z bloku nesměrovatelných adres (pytel2) adresu 10.0.0.1, na kterou přeloží IP adresu serveru *www.firma.cz*. Nyní již může klient začít navazovat spojení se serverem *www.firma.cz*. Ten má z pohledu klienta adresu 10.0.0.1, která je směrovačem přeložena na oficiální adresu 194.149.10.18.

## Rozložení výkonu

Nyní budeme řešit zcela odlišný problém. Máme server, na který přistupuje extrémně mnoho klientů, takže to jeden počítač nemůže zvládnout. Nasadíme místo jednoho počítače tři, ale problém vznikne v okamžiku, kdy každý z nich bude mít jinou IP adresu, a tedy i jiné jméno. Klienti jsou však zvyklí používat stále jedno jméno. Jak to udělat, aby se zátěž rovnoměrně rozložila na všechny počítače?

Původní jméno dáme pomyslnému virtuálnímu serveru. Před tento pomyslný server umístíme směrovač s aktivovanou komponentou NAT pro rozložení výkonu (*TCP Load Distribution*). Přijde-li první požadavek na spojení s naším virtuálním serverem, NAT změní IP adresu virtuálního serveru na IP adresu prvního reálného serveru. Druhý požadavek přesměruje na druhý reálný server atd. Až vyčerpá všechny reálné servery, přesměrovává další požadavek opět na první reálný server atd.



Obrázek 18.26: Rozložení výkonu

## ALG

Zatím se zdá, že NAT pracuje dobře, protože jsme se nezmínili o některých problémech, které se musí řešit tzv. aplikačními bránami (*Application Layer Gateway* – ALG). ALG modifikují datový tok mezi klientem a serverem. Pro korektní činnost některých aplikačních protokolů musí být mnohdy aktivovány příslušné ALG. Nyní se podíváme na ALG pro protokoly FTP a DNS.

## FTP

Problém s protokolem FTP a překladem adres spočívá v tom, že v příkazu PORT a odpovědi na příkaz PASV se přenáší IP adresa a port, na který má druhá strana navázat TCP spojení pro datový kanál.

Dojde-li k překladu IP adresy přenášené v příkazu PORT (resp. odpovědi na příkaz PASV), pak IP -adresa nesená v datovém toku musí být rovněž přeložena. To znamená, že směrovač musí v případě komunikace na portu 21/tcp určeném pro FTP sledovat, nedochází-li k uvedeným situacím, případně provést odpovídající překlad adresy a portu.

Potíž je také v tom, že v protokolu FTP je IP adresa přenášena znakově, tj. při překladu může mít nová adresa jinou délku než adresa původní. To musí být ošetřeno při překladu.

## DNS

Na obr. 18.25 jsme znázornili, že v některých případech je nutné, aby směrovač sledoval DNS odpověď a rovněž i v nich opravoval IP adresy.

## Domácí cvičení

Procvičíme si nejenom kap. 18, ale i kapitolu 16:

- ◆ Pomocí programu Telnet si stáhněte webovou stránku (obr. 16.2 str. 362).
- ◆ Tuto operaci provedte v síti, která je za proxy, tj. např. v intranetu vašeho zaměstnavatele (např. do metody GET se zadává absolutní URI).

V operačních systémech Microsoft je velice užitečný nástroj netsh, kterým lze z příkazového řádku spravovat síťové záležitosti. Tímto nástrojem lze dokonce vytvářet i skripty. Nás bude zajímat generická proxy, kterou si tímto nástrojem budeme moci spouštět na svém počítači.

- ◆ Zjistěte si IP adresu (IP adresa) nějakého serveru dostupného z vašeho počítače bez proxy.
- ◆ Nyní spusťte generickou proxy na vašem počítači – např.:  
`netsh interface portproxy add v4tov4 listenport=2000 IP-adresa`
- ◆ Jaké jsou spuštěné generické proxy, lze vypsat příkazem:  
`netsh interface portproxy show all`
- ◆ Nyní pomocí programu Telnet vypište obsah webové stránky serveru IP-adresa tak, že navážete spojení s portem 2000 vašeho počítače:  
`telent  
telent> open localhost 2000  
GET /`
- ◆ Proč jste zadávali relativní URI? (Protože se jednalo o generickou proxy – nikoliv HTTP proxy.)

## Kapitola 19

# Firewall

Firewall je systém tvořený jedním nebo více počítači, které jsou vyhrazeny k „bezpečnému“ oddělení vnitřní sítě od Internetu tak, aby mohli uživatelé vnitřní sítě přistupovat k informacím dostupným na Internetu. Firewall využívá všech mechanismů, které jsme popsali v předchozích kapitolách, tj. filtrace, proxy, brány či SOCKS.

Co přináší firewall kromě toho, že je realizován na vyhrazených počítačích? Firewall ukládá informace o provozu do auditních záznamů (logů), ze kterých lze nejenom vyčíst dílčí akce, ale i vytvářet sumární přehledy (reporty). Původní firewalls používaly běžný, neupravený operační systém, kdy správce firewallu musel periodicky ručně procházet logy a vyhodnocovat, zda nedošlo k nějakému bezpečnostnímu incidentu. Nevýhodou těchto firewallů bylo, že se na bezpečnostní incidenty příšlo až s určitým časovým odstupem.

Aktivní firewalls nezapisují vzniklé události pouze do logu, ale je na nich možné stanovit jistá pravidla, kdy vzniklá událost může způsobit jistou akci (*alert*). Akcí je pochopitelně spuštění či ukončení nějakého programu. Ve většině případů ale nejde o obecné spuštění nějakého obecného programu, ale nejčastěji o:

- ◆ Odeslání elektronické pošty, SMS apod. s informací o události na předem zadanou adresu.
- ◆ Uzavření proxy či filtru, na kterém k události došlo.
- ◆ Zapsání IP adresy potenciálního útočníka na černou listinu.
- ◆ Ukončení práce celého systému (*shutdown*).
- ◆ Spuštění obecného programu.

Standardní knihovny TCP/IP nejsou schopny předat aplikační vrstvě zaručenou informaci, z jakého síťového rozhraní (z jaké sítě) datový rámec přišel. Takže při použití klasické proxy máme vždy obavu, nepodvrhuje-li nám náhodou někdo z Internetu paket, jako by přišel z vnitřní sítě. Tomu se však umí většina filtrů bránit dostatečně efektivními metodami, aniž by se proxy musela o tuto věc starat.

V závislosti na typu firewallu je snaha, aby firewall řešil i zabezpečení na linkové vrstvě, tj. aby firewall ošetřil i stavky, kdy:

- ◆ IP datagramy s adresou odesilatele z vnitřní sítě přicházejí z Internetu.
- ◆ IP datagramy s adresou odesilatele z Internetu přicházejí z vnitřní sítě.
- ◆ Uživatelé vnitřní sítě se pokouší přes firewall napadnout servery ve vnitřní sítě, na které běžně nemají přístup, tj. útočníci z vnitřní sítě se chtějí od firewallu „odrazit“ na servery ve vnitřní sítě.

U firewallu je každé síťové rozhraní označeno jako vnitřní, jako vnější či jako rozhraní pro DMZ. Toto označení je u některých firewallů jen formální pro účely konfigurace; u jiných firewallů se pak

opravdu na linkové vrstvě kontroluje, přicházejí-li IP datagramy na rozhraní, na které přicházet mají. Všechny ostatní situace se pak označují za potenciální bezpečnostní incidenty a generují příslušné události.

Některé služby, jako je filtrace, proxy či NAT, provádí firewall sám. Jiné služby, jako je zjišťování virů v datech procházejících firewallem či autentizace klientů, může vyžadovat od aplikací běžících i na jiných počítačích specializovaných pouze pro tyto účely. Takové počítače jsou zpravidla umístěny ve vnitřní síti. Firewall pak pracuje jako klient těchto serverů.

Základem každého firewallu je vždy buď proxy (např. MS IIS), nebo filtrace (např. CISCO). Mnohé firewalls pak podporují jak proxy, tak i filtrace. V takovém případě můžeme pro některé aplikace využít filtrace a pro jiné proxy.

Pojem firewall není přesně vymezen. Existuje i norma RFC-2979 zabývající se firewalls. Jedná se však o ne příliš přesnou a výstižnou normu, která by problematiku obšírně vymezila. Tato norma specifikuje firewall jako agenta, který sleduje datový provoz mezi vnitřní sítí a Internetem, a zjistí-li provoz, který považuje za neodpovídající či dokonce nebezpečný, tak jej zablokuje. I tato norma říká, že firewall pracuje buď jako filtr, nebo cílový bod z hlediska protokolu TCP, nebo je firewall kombinací obou technik.

## Architektura firewallů

Design firewallů je až překvapivě různorodý. Před tím, než se rozhodneme pro konkrétní řešení, tak bychom měli provést analýzu rizik, která by nám měla napovědět nejenom samotnou architekturu firewallu, ale i:

- ◆ Nejhodnější metody ochrany jednotlivých protokolů (filtrace, proxy, NAT).
- ◆ Design DNS (viz níže).
- ◆ Způsob autentizace klientů, kterým se blíže zabýváme v publikaci „Velký průvodce PKI a technologií elektronického podpisu“.
- ◆ Způsob ochrany proti zlomyslným kódům a nevyžádané poště.
- ◆ Způsob zaznamenávání auditních záznamů („logů“).
- ◆ Případně i způsob automatizovaného vyhledávání bezpečnostních incidentů v souborech auditních záznamů.

Dalšími neméně důležitými parametry firewallu jsou bezesporu jeho výkon (propustnost) a jeho dostupnost. Dostupnost se zpravidla řeší využitím clusterů firewallů, které z uživatelského hlediska tvoří jeden celek.



**Poznámka:** Propojení intranetu s Internetem více nezávislými cestami (byť ochráněnými firewally) je vždy problematické, proto se takovému řešení snažíme vyhnout. Je to řešení spíše pro globální firmy, které jsou pak v některých zemích připojeny lokálními firewalls. To se ale neobejde bez zavedení přísných bezpečnostních pravidel.

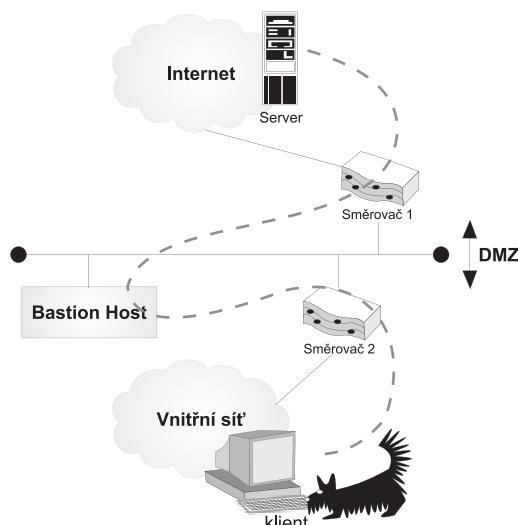
Nyní si uděláme exkurzi do historie firewallů.

## TIS Firewall Toolkit

Legendárním řešením firewallu je TIS Firewall Toolkit. Jedná se o soubor programů, které je možné zkompilovat na operačním systému typu UNIX. Tento soubor programů obsahuje jednotlivé funkce potřebné pro činnost firewallu. Firewall si pak můžeme postavit podle našich potřeb „na míru“.

TIS Firewall Toolkit vychází z představy použít firewall na klasickém operačním systému typu UNIX. Filozoficky nám může vzniknout jedno z následujících zapojení:

- ◆ Firewall se dvěma síťovými rozhraními, jedním do Internetu a druhým do vnitřní sítě. Toto řešení se dříve nepovažovalo za bezpečné, protože je použit neupravený operační systém UNIX, který by zajišťoval i bezpečnost na linkové vrstvě. Dnešní unixové systémy mají velmi sofistikované IP filtry, které na síťové i linkové vrstvě zajišťují bezpečnost velmi kvalitně. Např. Linux má svoje filtry ipchains a iptables, ve FreeBSD pak najdete velmi kvalitní IPFILTER a IPFIREWALL.
- ◆ Firewall s jedním síťovým rozhraním (obr. 19.1), který je tvořen dvěma směrovači s filtrací, mezi kterými je umístěn systém UNIX, označovaný jako ochranná bašta (*bastion host*). Filtr na směrovači 1 umožní komunikaci pouze mezi baštou a Internetem a filtr na směrovači 2 zase umožní komunikaci pouze mezi vnitřní sítí a baštou. Na baště pak běží proxy, poštovní server apod. V případě potřeby zvýšit výkon je možné baštu realizovat i více počítači. Na jednom pak mohou běžet proxy, na druhém poštovní server atp.



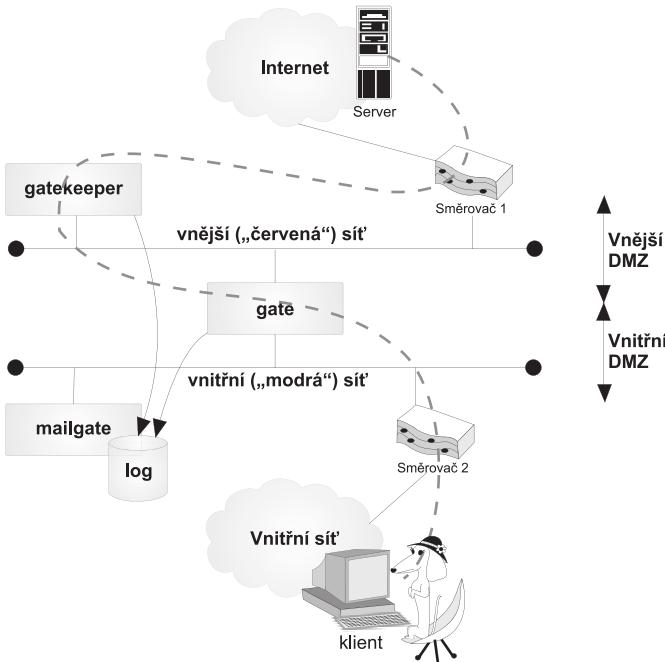
Obrázek 19.1: TIS Firewall Toolkit v zapojení s jedním počítačem osazeným jedním síťovým rozhraním

## SEAL

Jiným řešením firewallu byl firewall SEAL, který byl řešen třemi počítači s klasickým operačním systémem UNIX bez jakýchkoliv úprav (obr. 19.2). Těmi třemi UNIXovými počítači byly:

- ◆ Počítač gatekeeper, na kterém běžely proxy. Dále zde mohl běžet webový server firmy, FTP server apod. V případě velkých firem je možné místo jednoho počítače gatekeeper použít i více systémů: jeden pro proxy, jiný pro webový server apod.

- ◆ Počítač gate, který sloužil jako filtr na úrovni protokolů IP, TCP a UDP.
- ◆ Počítač mailgate sloužící jako poštovní server vnitřní sítě. Jelikož je tento počítač v podstatě ve vnitřní síti, odesílájí se sem logy z počítačů gatekeeper a gate.



Obrázek 19.2: Firewall SEAL

Použití tří systémů má mnohé výhody:

- ◆ Nedochází zde ke komplikacím s konfigurací DNS. Pro doménu *firma.cz* ve vnitřní síti slouží jako jmenný server počítač mailgate a pro doménu *firma.cz* v Internetu slouží počítač gatekeeper (mailgate je pak slave serverem systému gatekeeper).
- ◆ Problémy s elektronickou poštou jsou také eliminovány. Pošta pro doménu *firma.cz* je v Internetu směrována na gatekeeper, který ji mechanicky předává na mailgate. Obdobně pošta z vnitřní sítě je předávána ze systému mailgate na gatekeeper, který ji odešle dále do Internetu. Pro elektronickou poštu se povolí pouze komunikace mezi systémy gatekeeper a mailgate. Toho lze dosáhnout filtrování na systému gate.

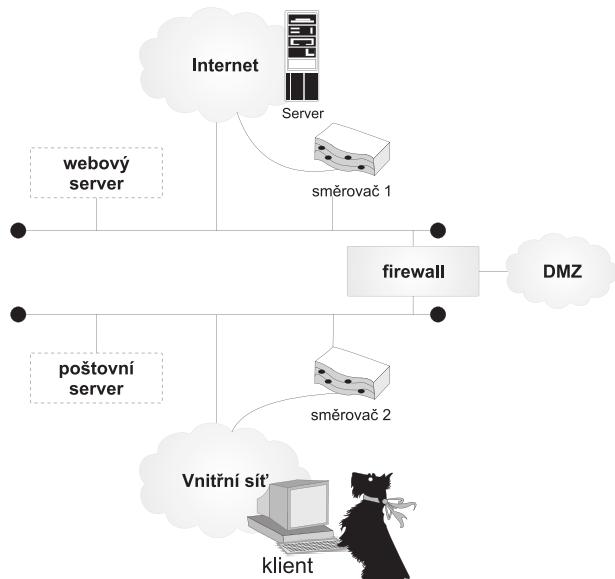
Prohlédneme-li si obrázek 19.2, pak LAN, na které je směrovač 1, systém gatekeeper a vnější rozhraní systému gate označené jako „červená síť“ tvoří DMZ v případě, že na směrovači 1 aktivujeme filtrování. Na tuto DMZ je možno umísťovat např. webové veřejné servery. Obdobně „modrá síť“ tvoří také DMZ. Každá z těchto DMZ má ale jiné bezpečnostní charakteristiky. Na vnitřní DMZ je např. poštovní server. Vnitřní DMZ vznikne aktivací filtrů na směrovači 2. Filtry na směrovači 2 chrání firewall před útoky z vnitřní sítě.

Pokud chce klient přistupovat např. protokolem HTTP z vnitřní sítě do Internetu, kontaktuje proxy běžící na systému gatekeeper. Tato proxy pak jménem klienta zprostředkuje spojení do Internetu.

Jinými slovy, červená síť musí mít adresy použitelné v Internetu, tj. přidělené poskytovatelem Internetu, protože klientská část proxy bude používat tuto adresu. Avšak červená síť musí být adresovatelná i ve vnitřní síti, protože klient bude kontaktovat z vnitřní sítě proxy ležící na červené síti.

### **Jednopočítačové firewalls s dvěma síťovými rozhraními**

Použití tří počítačů se ale jevilo jako plýtvání. Cílem bylo zredukovat firewall na jeden počítač oddělující vnitřní a vnější síť (obr. 19.3). Na takto vytvořený systém vznikly extrémní nároky na bezpečnost. Výsledkem je „jednopočítačový“ firewall zabezpečující veškeré funkce a teoreticky nevyžadující ani nějakou bezpečnostní součinnost směrovačů 1 a 2. Tento firewall zpracovává procházející informace od linkové vrstvy až po aplikaci vrstvu. Avšak z bezpečnostních důvodů by na „jednopočítačovém“ firewallu neměly být spouštěny jakékoli jiné aplikace než aplikace realizující samotný firewall, tj. neměl by zde např. běžet podnikový webový či poštovní server atd. (o použití firewallu k editaci textů ani nemluvě, protože to by znamenalo navíc ještě přístup nepovolaných osob k systému). To má za následek, že většina firem si stejně zakoupí další dva systémy: jeden pro webový server a druhý pro poštovní server.



**Obrázek 19.3:** Klasické zapojení jednopočítačového firewallu

### **Firewalling**

Někteří výrobci směrovačů (např. CISCO) nabízí jako součást svých operačních systémů pro směrovače tzv. firewalling. Firewalling umožňuje přímo ve směrovači využívat nástroje jako filtrace, NAT nebo i proxy, tunel a bránu. Tato řešení jsou ale vesměs postavena na filtrace a NAT. Využití proxy je zde omezené např. i tím, že směrovače standardně nemívají disk, který by bylo možné využívat pro cache nebo pro ukládání auditních záznamů.

Jedná se o řešení, které ocení zejména menší firmy.

## Personální firewall

Tak jsme si firewallem zabezpečili intranet, ale co když se svým PC přihlásím přímo na Internet? Dobre je si takové PC zabezpečit tzv. personálním firewallem, jehož cílem není bránit síť, ale pouze jen daný počítač. To znamená aktivovat si na PC software, který zamezuje útočníkům přístup během připojení PC do Internetu.

Jedná se o aktivaci filtrů, které umožní přistupovat z jiných počítačů jen na vyjmenované služby běžící na mému počítači. Sofistikovanější personální firewallsy od jiných výrobců pak mohou obsahovat i databáze znalostí o známých typech útoků a veškeré kontakty cizích počítačů vyhodnocovat proti této databázi (tzv. intrusion detection). Při detekci takového útoku mohou být personální firewallsy obdobně aktivní, jako jsou aktivní klasické firewallsy.

Personální firewall je součástí dnešních operačních systémů Windows. Na Linuxu si jej sami můžeme vybudovat např. pomocí příkazu `iptables`. Nebo si můžeme kupit i sofistikovanější personální firewallsy, které mají i řadu dalších funkčností. U nás je znám např. personální firewall Kerio.

## Demilitarizované zóny (DMZ)

Už v předchozích kapitolách se vyskytl termín DMZ. Co to tedy je ta demilitarizovaná zóna (DMZ)? DMZ je síť, jež je omezeně dostupná jak z Internetu, tak i z vnitřní sítě. Na obr. 19.4 vidíme, že taková síť vzniká např. na třetím síťovém rozhraní firewallu.



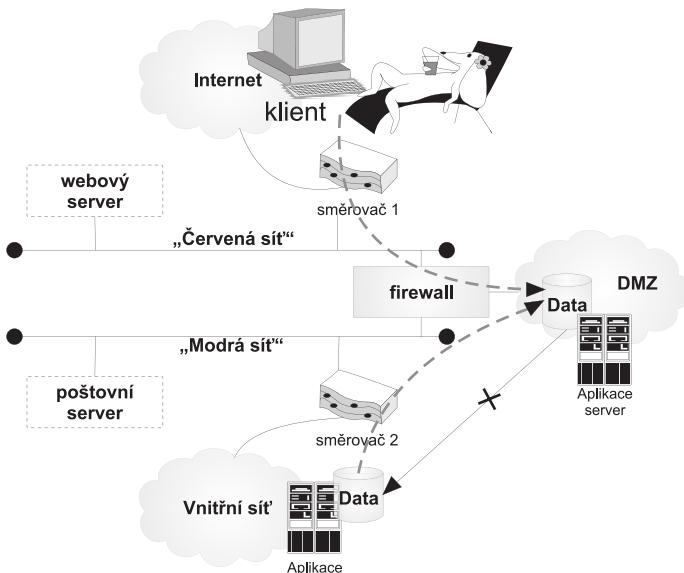
**Poznámka:** DMZ je síť, jež je omezeně dostupná jak z Internetu, tak i z vnitřní sítě.

Prohlédněme si ale obr. 19.4 důkladněji. V případě, že použijeme i filtrace na směrovači 1, pak demilitarizovanou zónou je i „Červená síť“ s webovým serverem. Avšak každá z těchto DMZ má jiné bezpečnostní charakteristiky. Zatímco DMZ za třetím rozhraním firewallu je vhodná pro umístění Portálu firmy dotazujícího se na data do intranetu, tak DMZ za směrovačem 1 je vhodná pro umístění webového serveru se statickými informacemi o firmě.

Někdy se jako interní DMZ označuje „Modrá síť“ (obr. 19.4) s poštovním serverem. Tato síť již však není dostupná z Internetu. Tuto síť lze využít nejenom k umístění poštovního serveru firmy, ale i serverů určených pro ukládání auditních záznamů či serverů ověřujících identitu apod.

Na DMZ zpravidla umísťujeme aplikační servery, abychom do Internetu zpřístupnili aplikace provozované ve vnitřní síti. Postupujeme přitom následovně: Aplikace provozovaná ve vnitřní síti má informace uloženy v databázi. Tu část databáze, kterou potřebuje aplikační server na DMZ, v pravidelných intervalech replikujeme do DMZ.

Nabízí se možnost, aby se v DMZ neudržovala replika databáze, ale aby se aplikace běžící na aplikačním serveru přímo dotazovaly databáze aplikace ve vnitřní síti. To však není z bezpečnostního hlediska ideální, protože internetový útočník by se po prolomení aplikace běžící na aplikačním serveru v DMZ mohl odrazit od aplikačního serveru a útočit přímo proti aplikaci ve vnitřní síti.



Obrázek 19.4: DMZ

## Firewall on Firewall

Na cestě mezi klientem a cílovým serverem nemusí být jeden firewall, ale může jich být i více. Podobně může být více filtrů i proxy.

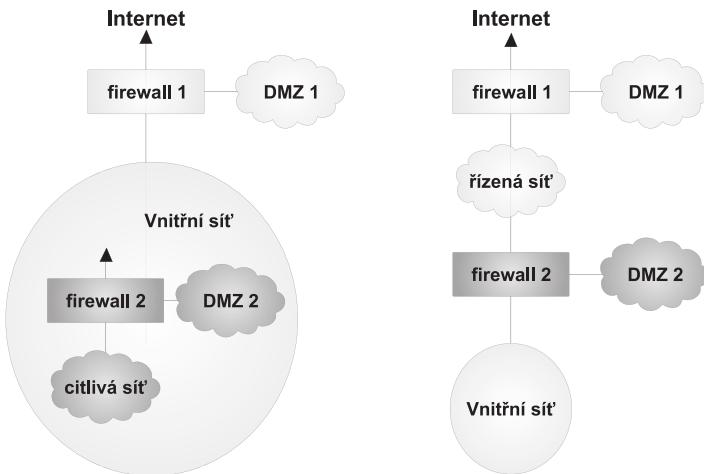
Použití dvou (resp. více) firewallů za sebou může mít dva různé důvody (obr. 19.5):

1. Vnitřní síť v sobě obsahuje nějakou ještě citlivější síť (např. síť s centrálními servery). Pak je z bezpečnostního hlediska žádoucí, aby citlivá síť byla oddělena od zbytku vnitřní sítě firewallem zamezujícím běžným zaměstnancům přístup na citlivou síť. Celá vnitřní síť je pak oddělena od Internetu dalším firewallem.
2. Informací udržovaných ve vnitřní síti si zvláště ceníme, proto jejich ochranu chceme zvýšit zdvojením firewallu. V takovém případě většinou volíme dva různé firewally od dvou různých výrobců. Např. firewall 1 volíme na bázi proxy a firewall 2 na bázi filtrace (či opačně). Vznikne nám pak soustava DMZ. Pro jednotlivé aplikace pak můžeme projektovat tu nejhodnější DMZ. Je třeba si uvědomit, že přímo adresovatelná z Internetu by měla být pouze DMZ 1.

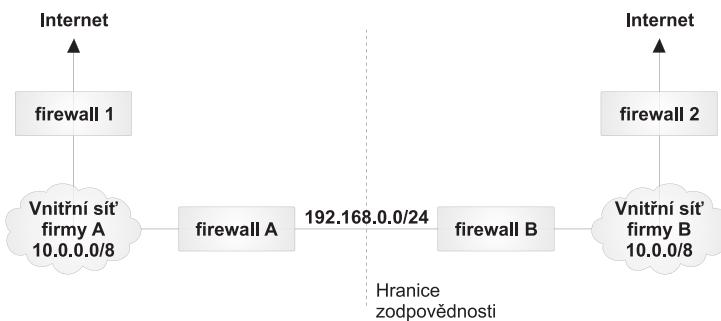
## Extranet

Zatím jsme se zabývali pouze oddělením vnitřní sítě od Internetu. Mnohdy je však potřebné propojit mezi sebou vnitřní sítě jednotlivých firem, tj. vytvořit mezi vnitřními sítěmi tzv. extranet.

Na obr. 19.6 je znázorněno řešení extranetu pomocí dvou firewallů. V každém případě je třeba mít oddělující prvek na straně každé firmy, tj. extranet neřešíme jedním firewallem společným pro obě firmy. Jsou pro to nejméně dva důvody:

**Obrázek 19.5:** Firewall on firewall

- ◆ Je třeba přesně definovat hranici, na které se láme zodpovědnost za komunikaci a její bezpečnost. Tuto hranici nelze objektivně určit fiktivně uvnitř nějakého boxu. Nejlépe je dohodnout se, kdo je zodpovědný za linku mezi oběma firewally, a zodpovědnost pak určit na rozhraní (zásuvce), kterým tato linka vstupuje do boxu druhé firmy.
- ◆ Firewall i ostatní bezpečnostní prvky jsou určeny pro ochranu vnitřní sítě před útoky z Internetu. Je vždy určeno, co je vnitřní a co vnější síť. V případě extranetu by pak nebylo jasné, která firma by byla vnitřní síť (tou chráněnou) a vnitřní síť které firmy by pak byla vnější – tou nebezpečnou. Proto je logické použít dvou boxů, jež jsou propojeny vnějšími rozhraními.

**Obrázek 19.6:** Extranet

Prvotně se snažíme použít ochranu pouze na úrovni směrovačů. Pouze v případě, že existují pádné důvody použít proxy, volíme firewally. Je možná i eventualita, že jedna firma použije ochranu filtrování ve směrovači a druhá firewall.

Problémem je také to, že jednotlivé vnitřní sítě mohou používat stejné IP adresy (síť 10.0.0.0/8). To ovšem nemusí vadit, protože komunikace ze sítě 10.0.0.0/8 firmy A se firewallem A převede na

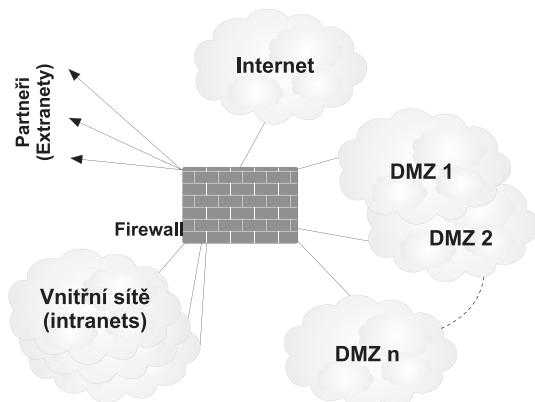
komunikaci na společnou síť např. 192.168.0.0/24. A komunikace až z této sítě se převede na komunikaci s vnitřní sítí 10.0.0.0/8 firmy B. Toho lze dosáhnout použitím jak proxy, tak i NAT v obou firewallech.

Zásadním pravidlem pro nastavení oddělujících prvků tvořících extranet je, že povolíme pouze tu komunikaci, která je bezpodmínečně nutná – vše ostatní zakážeme. Pokud mají spolu komunikovat např. pouze dva servery, pak striktně nastavíme, že spolu budou komunikovat pouze dvě IP adresy na konkrétních portech a víc nic.

V současné době si ale firmy spíše budují mohutnější firewalls s mnoha síťovými rozhraními. Na ně pak postupně připojují (obr. 19.7):

- ◆ Internet.
- ◆ Jednotlivé DMZ, přitom každá může splňovat jiná bezpečnostní pravidla.
- ◆ Extranety ke svým partnerům, tj. pro každý extranet nainstalují samostatný firewall, jak bylo znázorněno na obr. 19.6.
- ◆ Vnitřní sítě. Přitom jedna organizace může mít i více samostatných vnitřních sítí, které jsou pak vzájemně dostupné jen přes jeden centrální bod – firewall.

Samotný firewall přitom může být realizovan clusterem serverů.



Obrázek 19.7: Obecný design firewallu

## Viruswall a antispamový filter

Celá tato publikace se věnuje výhradně počítačové komunikaci – nikoliv obsahu přenášených dat. V případě firewallu však musíme udělat výjimku. Firewall je totiž bezpečnostní prvek. A bohužel nebezpečí číhá i v obsahu přenášených dat. Ta nebezpečí jsou v zásadě dvě:

- ◆ Zlomyslné kódy (viry, červy, spyware,...)
- ◆ Nevyžádané informace (spam)

Proti obojímu je možné postupovat tak, že se procházející data testují, neobsahují-li jednu z uvedených eventualit. Rozdíl mezi oběma eventualitami je zásadní v tom, že:

- ◆ V případě zlomyslného kódu se data testují proti databázi známých zlomyslných kódů. Výsledkem je pak jasné rozhodnutí, že data nějaký zlomyslný kód obsahují, nebo jsou čistá (rozhodnutí může být i nesprávné).
- ◆ V případě spamu je situace jiná. Antispamové filtry pouze ohodnocují procházející data. Výsledkem je číslo, které nám říká s jakou pravděpodobností se jedná o spam. Závisí pak na konkrétní instalaci, jestli např. e-mailsy s vysokým hodnocením zahazuje poštovní server nebo rozhodnutí provede až klient sám.



**Poznámka:** Je jasné, že pokud firewallem data prochází šifrována (např. SSL/TLS tunely), tak firewall nemá šanci zjistit cokoliv, co se týče obsahu přenášených dat.

Ochrana proti škodlivému obsahu je velice významná u elektronické pošty. Zajímavé je, že samotná tato ochrana začíná být náročnější na poštovní servery než samotná poštovní komunikace. Výsledkem je, že tato ochrana se umísťuje na samostatné servery. Samotný poštovní server se pak zabývá jen poštovní komunikací a lustraci obsahu předává specializovaným serverům.

Vývoj jde ale dále. Konfigurace a údržba systémů pro lustraci obsahu je stále náročnější a vyžaduje specializované odborníky. Výsledkem je, že vznikají firmy specializující se nejenom na instalaci a konfiguraci takových systémů, ale na samotné poskytování ochrany proti škodlivému obsahu na serverech.

Řešení je velice prosté. Svou poštu si v Internetu pomocí MX-záznamů přesměrujete na server specifikované firmy. Ten pak pracuje nejenom jako klasická poštovní relay, ale mezitím provede i lustraci obsahu.

Bohužel se celá situace ohledně lustrace obsahu zvrhla. Mnozí globální poskytovatelé poštovních služeb totiž také lustrují procházející data. Někteří z komerčních důvodů, aby vám mohli zaslat cílenou reklamu. Bohužel v jiných zemích se to používá i k politické cenzuře následované represí. Ideálním řešením je tak opět mít vlastní poštovní server pod vlastní kontrolou.



**Poznámka:** Rovněž bychom neměli zapomínat na šifrování zpráv (S/MIME).

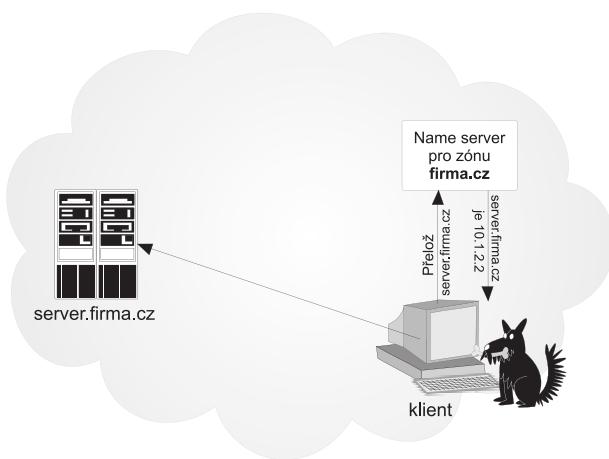
## DNS

Absolutní základem správné činnosti firewallu je pochopení DNS. Nejprve si ukážeme úskalí designu DNS v uzavřených sítích, pak přikročíme k designu DNS na firewallu.

### DNS v uzavřených podnikových sítích

Uzavřenou podnikovou síť rozumíme síť, která nemá spojení do Internetu. Na první pohled se může zdát, že konfigurace DNS pro uzavřené podnikové sítě musí být úplně bezproblémová. V čem tedy tkví problém?

Ve vaší firmě je používána doména *firma.cz*. Nakonfigurujete vcelku bez problémů jmenný server pro doménu *firma.cz*. Jste dokonce pečliví a nakonfigurujete primární jmenný server a na jiném počítači sekundární jmenný server pro doménu *firma.cz*.

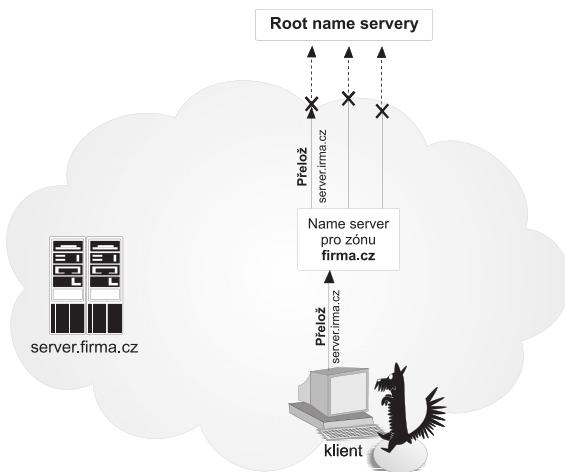


**Obrázek 19.8:** Překlad jména *server.firma.cz* na IP adresu

Všem klientům nasměrujete resolver na tyto jmenné servery. Na obr. 19.8 je znázorněno, jak klient žádá vámi nakonfigurovaný jmenný server o překlad jména *server.firma.cz* na IP adresu.

Vše pracuje korektně až do okamžiku, kdy se klient splete a místo *server.firma.cz* napiše *server.irma.cz* (splete se v jednom znaku).

Taková vcelku banální chyba ve jméně počítače způsobí, že nám aplikace zmrzne na několik desítek sekund. To mnohé uživatele přivádí přímo k šílenství. Co se stalo? Na obr. 19.9 je zobrazena podstata problému. Nakonfigurovaný podnikový jmenný server je autoritou pro doménu *firma.cz*. Není však autoritou pro doménu *irma.cz*. Jelikož není autoritou pro doménu *irma.cz*, pak o překlad musí požádat kořenové jmenné servery, které mu pomohou nalézt autoritativní jmenný server, jenž jediný může sdělit, že počítač server v doméně *irma.cz* neexistuje (nebo existuje a je to úplně jiný počítač).



**Obrázek 19.9:** Místní jmenný server se pokouší neúspěšně kontaktovat kořenové jmenné servery v Internetu

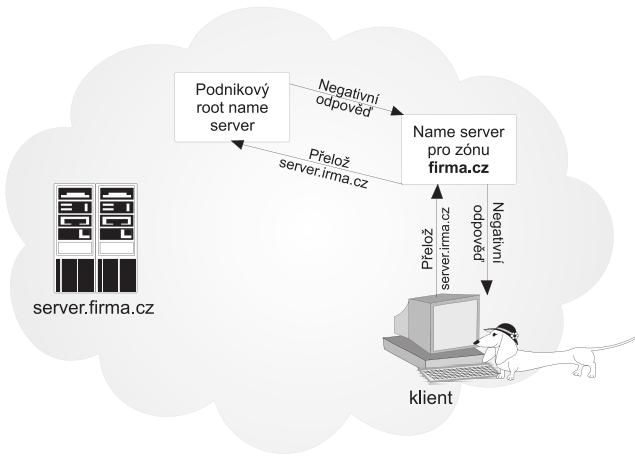
Jenže jsme v uzavřené podnikové síti, která nemá spojení do Internetu. Takže datagramy nesoucí dotazy na kořenové jmenné servery jsou nejpozději na hranici sítě zahodeny. Podnikový jmenný server nemůže dostat odpověď a nechá klienta na holičkách.

Když resolver po několika desítkách sekund nedostane odpověď, domyslí si, že někde musí být nějaká chyba, a uživateli vrátí chybovou hlášku. Ovšem chybová hláška se uživateli zobrazí jen v případě, že uživatel byl dostatečně trpělivý a mezi tím např. nezavedl znova operační systém nebo...

První reakcí správce je, že pochopí, že kořenové jmenné servery z uzavřené podnikové sítě kontaktovat nelze. Vzpomene si, že při startu jmenného serveru se do paměti jmenného serveru nahrávají data o kořenových jmenných serverech. Správce se domnívá, že přičinou jeho problémů je tento soubor, a proto jej zruší. Je však nemile překvapen, že se vůbec nic nezměnilo. Vysvětlení je prosté: pokud jmenný server nenajde vůbec žádné informace o kořenových jmenných serverech, tak přímo v kódu programu jmenného serveru jsou pro takový případ uvedeny autorem programu IP adresy některých jmenných serverů.

Na obr. 19.10 je nakresleno, jak je třeba správně postupovat. Vytvoří se podnikový kořenový jmenný server (může jich být i více) a soubor s informacemi o kořenových jmenných serverech se nesmaže, ale opraví se tak, aby vše bylo nasměrováno na nás podnikový kořenový jmenný server.

Pro kořenový jmenný server ani nemusí být nějaký speciální počítač. Můžeme jej nakonfigurovat i na stávajícím jmenném serveru tím, že zde vytvoříme primární jmenný server pro kořenovou doménu.



**Obrázek 19.10:** Podnikový kořenový jmenný server vrací negativní odpověď: „V našem podnikovém Internetu neexistuje žádná doména `irma.cz`“

## DNS a firewall

Firewall odděluje vnitřní podnikovou síť (intranet) od Internetu. Umožňuje klientům vnitřní sítě čerpat informace z Internetu a znemožňuje útočníkům z Internetu útočit proti počítačům ve vnitřní síti.

Firma má v Internetu přiděleno doménu *firma.cz* a s největší pravděpodobností ji bude chtít používat jak v Internetu, tak i ve vnitřní síti. V Internetu bude v doméně *firma.cz* nejspíš pouze: *www.firma.cz*, *mail.firma.cz* a několik málo dalších záznamů (MX-záznamy pro *firma.cz* ukazující na *mail.firma.cz* atd.). V intranetu však mohou být v doméně *firma.cz* desítky či dokonce tisíce počítačů.

Jinými slovy: budou dvě domény *firma.cz*, každá bude obsahovat jiné záznamy, ale potíž je v tom, že se jmenní stejně – *firma.cz*. V Internetu nemohou existovat dvě různé domény stejného jména. Avšak tato situace nenastává v samotném Internetu, ale jedna doména je v Internetu a druhá v intranetu.

Problém je s firewallem jako takovým. Na firewallu běží aplikace (např. proxy), které potřebují pracovat s doménou *firma.cz* vnitřní sítě, ale i s ostatními domény z Internetu. Navíc firewall jako jmenný server se z hlediska klientů v Internetu musí tvářit, že pracuje s doménou *firma.cz* z Internetu.

Aplikace běžící na firewallu (např. proxy) využívají resolver, kdežto firewall jako jmenný server bude poskytovat informace jako server. Jako nástroj se využívá skutečnost, že resolver nemusí být nasměrován na jmenný server běžící na místním počítači (na 127.0.0.1).

Jedním problémem je vlastní zapojení firewallu a přidělení IP adres. Jiným problémem je konfigurace firewallu z hlediska DNS. Oba problémy jsou na sobě v podstatě nezávislé.

Pokud na firewallu používáme jako jmenný server BIND verze 9.2 nebo novější, pak je možné celý problém elegantně vyřešit využitím pohledů (view). Pokud nechceme využít pohledy nebo nemáme k dispozici požadovanou verzi Bindu, pak z hlediska konfigurace DNS na firewallu může nastat celá řada eventualit. Probereme několik modelových situací. V praxi však nejspíš použijete jejich kombinaci těchto situací.

## Společné DNS pro Internet i intranet

Nejjednodušším řešením je udělat společné DNS pro Internet i intranet. Toto řešení je považováno za nedokonalé zejména ze dvou příčin:

1. Na Internetu se zveřejňují překlady počítačů, které mají interní adresy (např. ze sítí 10/8, 172.16/12 či 192.168/16).
2. Zveřejňují se informace o struktuře firmy (IP adresy počítačů vnitřní sítě). Tyto informace jsou zpravidla utajovány.

Kardinální otázkou při konfiguraci DNS na firewallu vždy je, zdali se mají překládat všechna jména Internetu ve vnitřní síti, nebo zdali klienti vnitřní sítě mají možnost překládat pouze jména z domény *firma.cz* ve vnitřní síti.

## Ve vnitřní síti se překládá celý Internet

Pokud se má překládat celý Internet ve vnitřní síti (nevýživá se ani NAT!), pak se i ve vnitřní síti musí směrovat (dopravovat) IP adresy celého Internetu. To má opět dva nepříjemné důsledky:

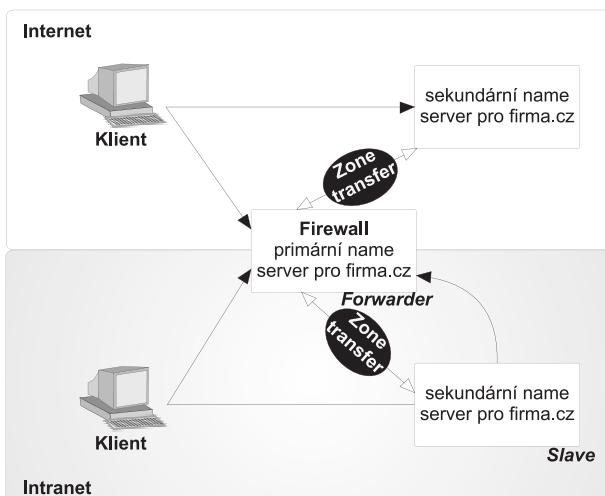
1. Směrování vnitřní sítě musí být na tuto skutečnost připraveno. To znamená, že veškeré IP adresy, které nejsou z vnitřní sítě, musí směrování dopravit na firewall. Zpravidla se to provádí pomocí položky *default* ve směrovacích tabulkách. Zejména v případě rozsáhlých intranetů nemusí být triviálním problémem takové směrování stabilně udržet.

2. Bezpečnostní manažeři sledují vnitřní síť, zdali v ní nedochází k útokům z jiných sítí. Pokud se v síti dopravují pouze IP datagramy adresované v sítích 10/8, 172.16/12 či 192.168/16, pak při výskytu jiné adresy se snadno může signalizovat možnost bezpečnostního incidentu. Pokud se však mohou ve vnitřní síti vyskytovat legálně zcela libovolné adresy Internetu, pak bezpečnostním manažerům bereme tento nástroj.

Překlad celého Internetu ve vnitřní síti se používá zejména ve dvou případech:

- ◆ Pokud jsou na firewallu provozovány transparentní proxy. Transparentní proxy je příjemná pro uživatele používajícího POP3, telnet, ftp atd. Pokud se chce uživatel přihlásit programem telnet např. na server *www.playboy.com*, pak se nemusí hlásit nejprve k proxy (firewallu) a teprve z něj dále na cílový server. Prostě ve vnitřní síti napiše jen „telnet *www.playboy.com*“. Transparentní proxy na firewallu akceptuje spojení, jako by byla sama cílovým serverem, a předá požadavek dále jménem klienta na cílový server do Internetu. Jenže na *www.playboy.com* se většinou uživatelé nepřihlašují programem Telnet, ale pomocí webového prohlížeče. A webový prohlížeč žádné transparentní proxy nepotřebuje. Závěr je takový, že běžní uživatelé Telnet a ftp nepoužívají a správcům a vývojářům použití klasické proxy pro programy Telnet a ftp příliš nevadí.
- ◆ Pokud se nepoužívá klasický firewall s proxy, ale pouze ochrana vnitřní sítě pomocí filtrace (bez NAT).

Firewall zpravidla pracuje jako primární jmenný server pro doménu *firma.cz* (viz obr. 19.11), která je společná jak pro intranet, tak i pro Internet.



**Obrázek 19.11:** Firewall je primárním jmenným serverem pro Internet i intranet – ve vnitřní síti se překládá celý Internet

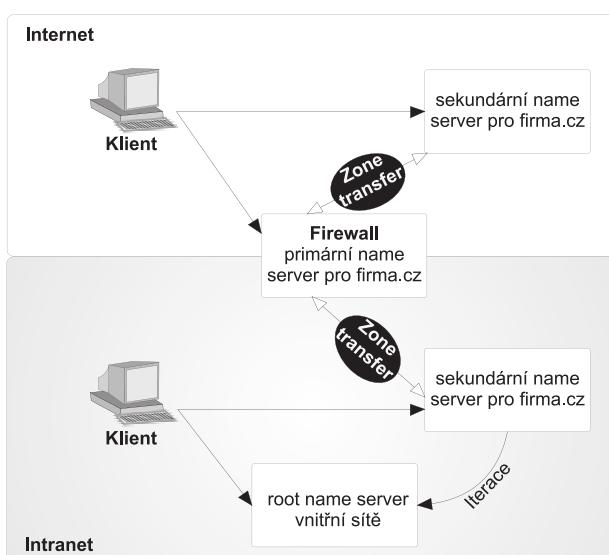
Je rozumné mít alespoň dva jmenné servery pro doménu (primární a sekundární). Avšak je třeba mít dva jmenné servery dostupné v Internetu a dva v intranetu. Jelikož firewall je dostupný z obou sítí, tak stačí nakonfigurovat jeden sekundární jmenný server pro Internet a druhý pro intranet.

Sekundární jmenný server v Internetu pro naši doménu budeme nejpravděpodobněji provozovat u našeho poskytovatele Internetu.

Sekundární jmenný server v intranetu zřídíme na nějakém dalším počítači. Pokud bude klient intranetu požadovat překlad jména z jiné domény přímo na firewallu, pak to není potíž. Firewall může o pomoc požádat kořenové jmenné servery Internetu a klienta uspokojí. Pokud však klient požádá o překlad jména z jiné domény sekundární jmenný server v intranetu, pak je problém v tom, že tento sekundární jmenný server nemá spojení do Internetu a nemohl by proto o takový překlad požádat kořenové jmenné servery. Aby i takové překlady provádět mohl, tak se sekundární jmenný server intranetu nakonfiguruje jako podřízený server vůči firewallu jako forwarderovi. Firewall překlad provede a předá jej podřízenému serveru, který jej obratem předá klientovi.

**Ve vnitřní síti se Internet nepřekládá**

Překlad adres Internetu ve vnitřní síti většinou není vůbec nutný. Ve vnitřní síti je nutné umět přeložit pouze jméno firewallu (proxy), protože klient navazuje spojení s proxy a ta teprve navazuje jménem klienta spojení s cílovým serverem v Internetu. Tj. až proxy musí umět přeložit jméno cílového serveru v Internetu na IP adresu.



**Obrázek 19.12:** Firewall je primárním jmenným serverem pro Internet i intranet – ve vnitřní síti se překládá pouze doména *firma.cz*

Na obr. 19.12 je znázorněno, jak je nakonfigurováno DNS ve vnitřní síti, aby neprovádělo překlady Internetu. Ve vnitřní síti se zřídí kořenový jmenný server. Sekundární jmenný server pro *firma.cz* se nasměruje na tento kořenový jmenný server. Pokud je požadavek na jinou doménu než na doménu *firma.cz*, pak kořenový jmenný server odpoví negativně.

Jelikož je praktické mít ve vnitřní síti dva jmenné servery, tak se prakticky zřizují dva počítače. Na obou běží sekundární jmenné servery pro `firma.cz` a současně na nich běží kořenové jmenné servery.

Je třeba ještě upozornit, že pokud klient vnitřní sítě nasměruje svůj resolver přímo na firewall, pak mu firewall přeloží libovolnou adresu Internetu. Klientův resolver proto musí být nasměrován na jmenný servery ve vnitřní síti.

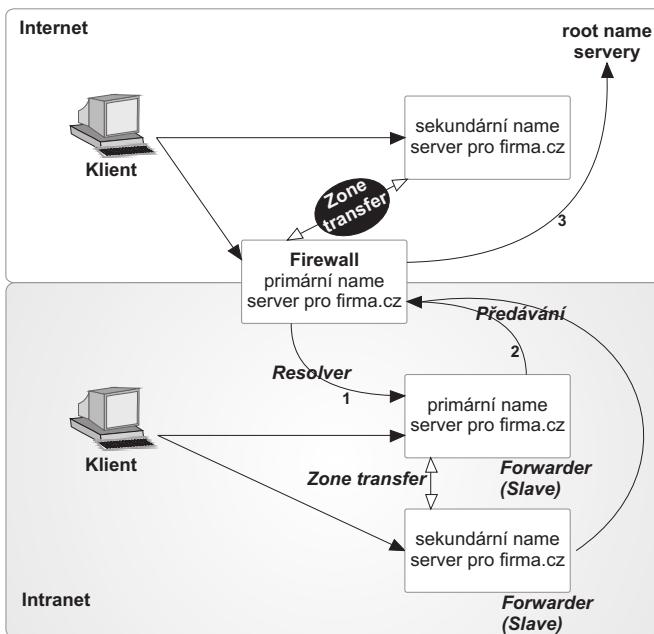
## Na firewallu je jen jmenný server pro Internet, a nikoliv pro intranet

Pokud chceme mít dvě oddělené zóny pro doménu *firma.cz*, pak zpravidla bývá primární jmenný server pro Internet na firewallu a sekundární jmenný server pro Internet na počítači poskytovatele Internetu. Pro vnitřní síť se zřídí samostatná dvojice primární/sekundární server na serverech ve vnitřní síti.

Opět jsou tu dvě možnosti. První možnost dovoluje překládat ve vnitřní síti celý Internet a druhá možnost překládat ve vnitřní síti pouze zónu vnitřní sítě.

### Ve vnitřní síti se překládá celý Internet

Máme dvě samostatné dvojice jmenných serverů (viz obr. 19.13). Jedna dvojice je ve vnitřní síti a druhá dvojice v Internetu.



Obrázek 19.13: Samostatná zóna pro Internet a intranet – ve vnitřní síti se překládá celý Internet

Prvotním problémem je, že aplikace běžící na firewallu (např. proxy) potřebuje informace o zóně *firma.cz* vnitřní sítě, avšak potřebuje také informace o všech ostatních doménách Internetu. Proveďte se to tak, že resolver firewallu není nasměrován na jmenný server firewallu, ale na jmenný server vnitřní sítě, který má k dispozici zónu vnitřní sítě.

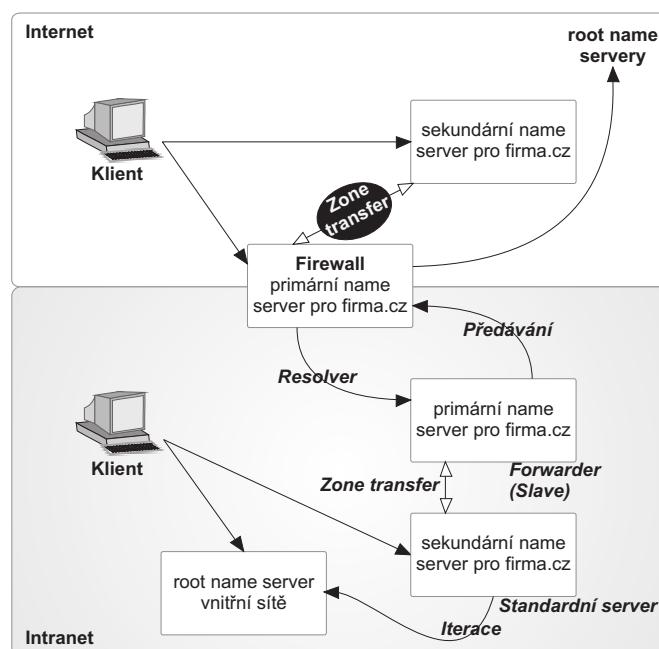
Avšak v případě, že aplikace na firewallu potřebuje přeložit např. [www.playboy.com](http://www.playboy.com), pak o tento překlad také požádá jmenný server vnitřní sítě (šipka 1 na obrázku 19.13). Jmenný server vnitřní sítě je forwarder (slave), který všechny požadavky, jež neumí vyřešit sám, předá předávači – jmennému serveru firewallu (šipka 2). Jmenný server firewallu pak již má přístup ke kořenovým jmenným ser-

verům (šipka 3), a tak může provést překlad. Výsledek předá zpět podřízenému serveru, který obrazem vše předá klientu na firewallu.

Klient vnitřní sítě požaduje překlady od jmenných serverů vnitřní sítě. Pokud se jedná o překlad místní domény, pak mu vrátí odpověď; pokud se však jedná o překlad domény z Internetu, pak takový požadavek předá předávači, tj. firewallu.

### Ve vnitřní síti se Internet nepřekládá

Varianta, kdy se ve vnitřní síti Internet nepřekládá, znamená vytvořit ve vnitřní síti kořenový jmenný server (viz obr. 19.14). Zajímavé je, že ve vnitřní síti jsou minimálně dva jmenné servery a každý má jinou funkci. První (na obr. 19.14 označený jako primární name server) slouží firewallu. Pokud by si klient vnitřní sítě na něj nasměroval svůj resolver, pak by mu tento jmenný server přeložil cokoliv z Internetu a ještě zónu *firma.cz* ve vnitřní síti. To však nechceme, proto jsou resolvery klientů vnitřní sítě nasměrovány na další jmenné servery ve vnitřní síti, které používají kořenový jmenný server vnitřní sítě. Kořenové jmenné servery vnitřní sítě pak zamezí překladu jiných domén.



Obrázek 19.14: Samostatná zóna pro Internet a intranet – ve vnitřní síti se překládá pouze doména *firma.cz*

### Duální DNS

Pokud chceme mít samostatnou zónu pro vnitřní síť a samostatnou zónu pro Internet, pak kvůli tomu, že se stejně jmenují, musíme mít každou zónu na samostatném počítači. Cílem duálního DNS je provozovat primární jmenný server pro zónu *firma.cz* vnitřní sítě i Internetu na jednom počítači. Důvod je ekonomický. Zatímco u velkých firem se na vnitřní síti provozuje celá řada serverů, na

kterých mohou běžet jmenné servery, tak u menších firem by bylo často nutné instalovat další počítač jen proto, aby na něm mohl běžet jmenný server.

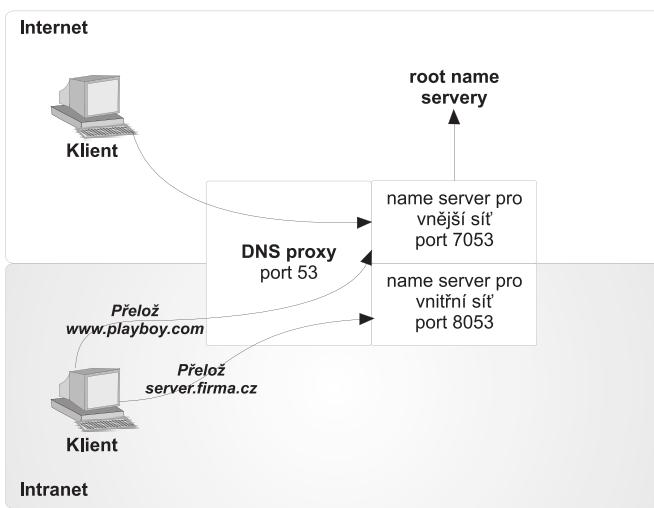
Princip duálního DNS spočívá v tom, že na firewallu poběží dva jmenné servery (dva procesy). Každý však běží na jiném portu. Na obr. 19.15 jmenný server pro Internet běží na portu 7053 a jmenný server pro intranet běží na portu 8053.

Běžní klienti mohou těžko využít tyto jmenné servery na jiném portu než na portu 53, protože o portech 7053 a 8053 se nedoví.

Na firewallu běží na standardním portu pro jmenný server tzv. DNS proxy. DNS proxy zjišťuje, odkud požadavek přichází. Podle toho, odkud požadavek přichází, jej buď zamítne, předá jmennému serveru na portu 7053 nebo jmennému serveru na portu 8053.

Požadavek může přicházet od:

- ◆ Klienta Internetu, pak jej předá jmennému serveru pro Internet (na obrázku port 7053).
- ◆ Klienta intranetu, pak se jedná o dva případy:
  1. Jedná se o požadavek na překlad z domény *firma.cz*, pak jej předá jmennému serveru pro vnitřní síť (na obrázku port 8053).
  2. Jedná se o požadavek na překlad jiné domény Internetu. Pak se DNS proxy rozhoduje:  
Chceme-li ve vnitřní síti překládat Internet, pak požadavek předá jmennému serveru pro Internet (port 7053).  
Nechceme-li ve vnitřní síti překládat jiné domény Internetu, pak odpoví negativně. Je zajímavé, že pokud ve vnitřní síti nemáme další (např. sekundární) jmenné servery, pak nepotřebujeme na vnitřní síti kořenový jmenný server. Negativní odpověď vydá přímo DNS proxy.
- ◆ Aplikace běží na firewallu (např. proxy) pak se zkoumá, zdali se jedná o požadavek na doménu *firma.cz* – ten je předán jmennému serveru pro vnitřní síť (port 8053), nebo jestli se jedná o jiný požadavek – ten je předán jmennému serveru pro Internet (port 5073).



**Obrázek 19.15:** Duální DNS

## Ostatní aplikační protokoly a firewall

Na firewallu konfigurujeme průchody pro jednotlivé aplikační protokoly. Mnohé aplikační protokoly se vzhledem k firewallu chovají podobně, takže v principu řešíme tuto problematiku pro jednotlivé aplikační protokoly.

### HTTP a FTP

Součástí protokolu HTTP je podpora proxy. Výhodou použití proxy je paměť cache, která může zrychlit komunikaci mezi klientem a serverem.

U protokolu HTTP se snažíme vyhnout filtraci nejenom proto, že filtrace nám nenabídne cache, ale zejména také proto, že mnohé servery používají nestandardní čísla portů, což je pro filtraci značně nepříjemné.

Protokol FTP se zase špatně filtrouje. Pro protokol FTP je brána bezproblémová, kdežto filtrace obtížná. Navíc mnohé proxy pro HTTP jsou kombinovány s bránami pro FTP. Brána FTP/HTTP umožňuje pouze některé funkce protokolu FTP a prakticky nám neumožňuje např. odesílání souborů na server pomocí prohlížeče.

### SSL/TLS

U protokolu SSL (resp. HTTPS) se většinou podřizujeme řešení pro protokol HTTP, protože tunel pro SSL bývá součástí softwaru zajišťující proxy pro HTTP.

### Telnet či SSH

Tyto protokoly jsou klasické protokoly na bázi klient-server, pro které se nepoužívá cache. Lze je proto snadno filtrovat i snadno pro ně spustit proxy. Obě řešení mají výhody i nevýhody.

### Elektronická pošta

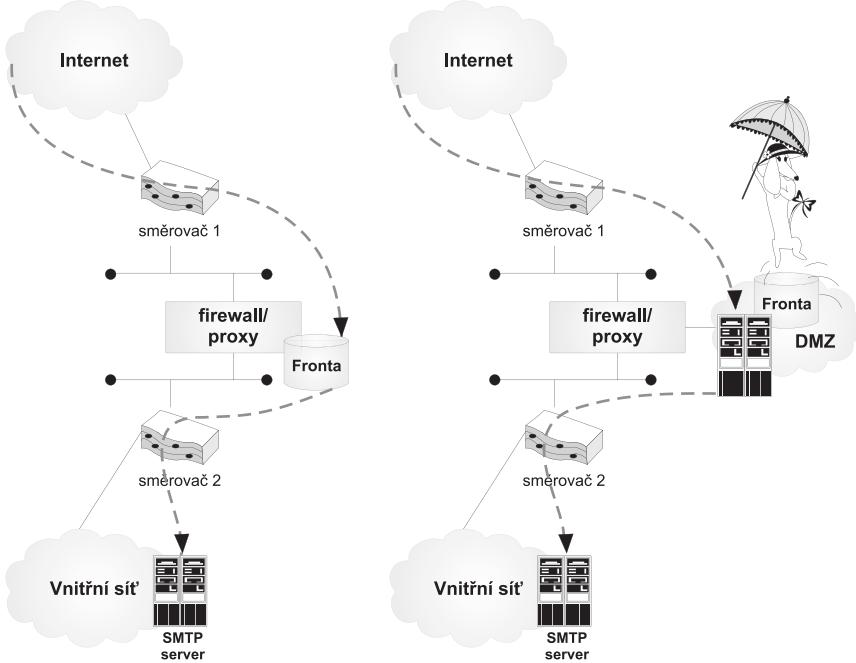
Elektronická pošta spolu s DNS je vždy jádrem problému každého návrhu firewallu. Architekturu elektronické pošty ve firmě je vhodné vyřešit ještě před volbou firewallu.

Základní otázkou pro řešení elektronické pošty ve firmě je, kde bude umístěn poštovní server firmy:

1. Na Internetu, tj. elektronická pošta zaměstnanců bude např. na poštovním serveru poskytovatele Internetu či na samostatném serveru v Internetu.
2. V DMZ. To má výhodu, že elektronickou poštu je možno vybírat jak z vnitřní sítě, tak i z Internetu. Toto řešení volí i velké firmy. Avšak ty si vybudují ve vnitřní síti poštovní servery a jeden poštovní server umístí na DMZ. V případě, že zaměstnanec vyjede na cesty, přesměruje si elektronickou poštu na poštovní server na DMZ.
3. Poštovní server bude ve vnitřní síti.

První dva případy lze řešit jednoduše, protože ve vnitřní síti jsou klienti protokolů SMTP, POP3 či IMAP4 a v Internetu (resp. na DMZ) je poštovní server. Všechny tyto protokoly lze snadno filtrovat či pro ně spustit generické proxy. Nebo ještě efektivněji – spustit transparentní proxy.

Problémem je, jak pouštět příchozí poštu z Internetu na poštovní server umístěný ve vnitřní síti (obr. 19.16). V případě ochrany vnitřní sítě filtrací je nutné zúžení na filtrace mezi dvěma SMTP servery. Např. jeden umístíme na DMZ a druhý do vnitřní sítě. Pak povolíme komunikaci na portu 25/tcp mezi těmito dvěma servery a jakoukoliv jinou komunikaci na portu 25/tcp zakážeme.



**Obrázek 19.16:** Vlevo SMTP proxy, vpravo ochrana filtrací se dvěma SMTP servery

Trochu odlišná je situace, kdy je firewall založený na proxy. V takovém případě spouštíme nějaký zjednodušený poštovní server přímo na firewallu (tzv. SMTP proxy). Tento zjednodušený server přijímá poštu z Internetu určenou pro vnitřní síť a ukládá ji do fronty. Před uložením do fronty mohou být aktivovány i antivirové programy, které v poště vyhledávají viry.

V pravidelných intervalech pak SMTP proxy spouští SMTP klient, který z fronty předává došlu poštu na poštovní server ve vnitřní síti či odesílá poštu z vnitřní sítě do Internetu. SMTP server přijímá jednak poštu z Internetu určenou pro vnitřní síť a jednak poštu z vnitřní sítě určenou pro Internet.

## NTP (Network Time Protocol)

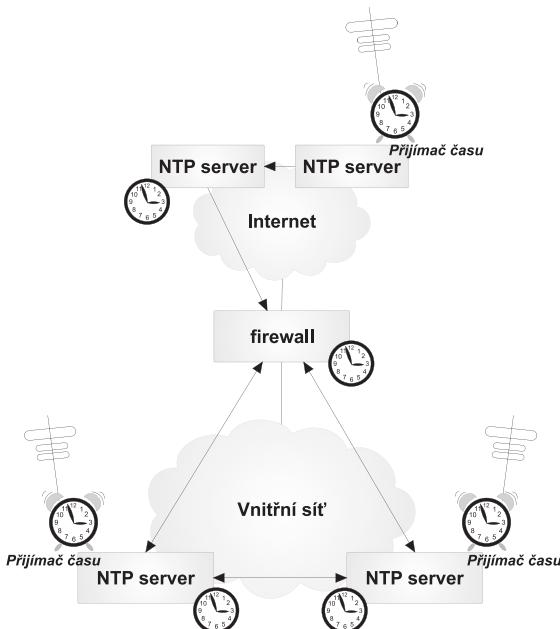
Synchronizace času je velmi důležitá nejen pro správný chod systémů ve vnitřní síti, ale je důležitá i v případě prokazování času útoku zejména vlastními zaměstnanci. Může se totiž stát, že by zaměstnanec posunul systémový čas na dobu, kdy byl prokazatelně mimo pracoviště, provedl útok a pak vrátil čas zpět. V takovém případě se pak marně hledá útočník z Internetu. Při podezření na manipulaci s časem je dobré si prohlédnout log úplně jiných služeb, než na které byl proveden útok. Pokud totiž útočník nebyl důsledný, můžeme v záznamech najít nesrovonalosti v posloupnosti zápisu do logu.

Zásada je, abychom ve vnitřní síti měli tři nezávislé zdroje času. Protože zpravidla máme jeden firewall, může být jedním ze zdrojů času pouze čas získaný protokolem NTP z Internetu.

Jelikož synchronizace času protokolem NTP běží nad protokolem UDP, nemá smysl komplikovat propojení mezi vnitřní sítí a Internetem např. reflexivními filtry. Stačí synchronizovat čas na firewalu z Internetu a firewall pak prohlásit za jeden ze tří časových serverů vnitřní sítě.

Při komunikaci protokolem NTP s časovými servery v Internetu a pomocí filtrace na přístupovém směrovači nastavíme, abychom komunikovali pouze s jedním až třemi vybranými servery v Internetu. Ostatní komunikaci protokolem NTP zakážeme.

Pro synchronizaci času v malých firmách většinou postačí synchronizace času z Internetu. Pak je dobré ji brát ze tří NTP-serverů v Internetu.



Obrázek 19.17: Synchronizace času ve vnitřní síti



# Rejstřík

## #

1000Base 68  
100Base 68  
10Base 68  
10GBase 68

## A

A záznam 276  
A6 záznam 283  
AAAA záznam 283  
ABM (Asynchronous balanced mode) 76  
ACK 233  
active close 232  
Ad-hoc sítě 71  
adresa sítě 168  
adresní plán 183  
adresný oběžník 167  
ADSL (Asymmetric Digital Subscriber Line) 57  
AfriNIC 313  
AH 215  
AirPcapp 72  
aktivní FTP 348  
algoritmus Nejkratší cesty 199  
analogová linka 52  
anonimní FTO 358  
antispamový filter 469  
anycast 224  
APNIC 313  
ARIN 312  
ARM (Asynchronous response mode) 76  
ARP (Address Resolution Protocol) 153  
arytmický přenos 47  
AS 176  
ASO (The Address Supporting Organization) 312  
Asynchronní přenos 26, 47  
AT-přiaz 53

autonomní systém 176, 194

AXFR 305  
asynchronní přenos 47

## B

bandwidth interval 103  
BAP 98  
beacon frame 119  
BECN (Backward Explicit Congestion Notification) 107  
Bellman-Fordův algoritmus 197  
brána 367  
BRI (Basic Rate) 55  
bridge 67  
broadcast 167, 225  
BSS (Basic Service Set) 114

## C

cache zóna 260  
caching only server 268  
CBCP 95  
ccNSO (The Country Code Names Supporting Organisation) 312  
ccTLD 256  
cHDLC (CISCO HDLC) 81  
CIR (Committed Information Rate) 103  
CNAME záznam 276  
Compression Control Protocol (CCP) 99  
Congestion Window 247  
cookie 387  
CSMA/CA (Carrier Sense Multiple Access/Collision Avoidance) 70  
CSMA/CD (Carrier Sense Multiple Access/Collision Detection) 66  
CWND 247  
CWR 233  
CZ.NIC 313

**Č**

černá listina 440

**D**

další hlavičky IPv6 209

databáze DNS 273

datový okruh 18

datový paket 20

DE (Discard Eligibility) 107

demilitarizovaná zóna (DMZ) 466

Differentiated services 163

Dig 291

Dijkstrův algoritmus 199

distribuční systém (DS) 115

DLCI (Data Link Connection Identifier) 105

DMZ 466

DNAME věta 285

DNS (Domain Name System) 255

DNS a firewall 472

DNS duální 477

DNS NCACHE 307

DNS Notify 303

DNS QUERY 293

DNS UPDATE 299

DNS v uzavřených podnikových sítích 470

dobře známé porty 230

domény 256

Downstream 57

DSAP (Destination service access point address) 124

DSN 406

Dumpcap 39

dynamicky přidělované adresy 182

**E**

EAP 91, 93

EAP-TLS 94

EC (Echo Cancellation) 58

ECE 233

ECN (Explicit Congestion Notification) 164

Editcap 39

EGP (Exterior Gateway Protocols) 194

EIA (Electronic Industries Aliance) 2

elektronická pošta 391

ESMTP (Extensions SMTP) 403

ESP 215

ESS (Extended Services Set) 115

Ethereal 29

Ethernet 64, 111

Ethernet II 111, 112

euroISDN 55

extranet 467

EXTRN 405

**F**

Fast Ethernet 63, 2

FDM (Frequency Division Multiplexing) 58

FECN (Forward Explicit Congestion Notification) 107

ferule 64

filtrace 425

filtrace ARP 156

FIN 233

firewall 461

firewall on firewall 467

firewalling 465

flag 77

forwarding 185, 187

fragmentace 142, 252

Frame Relay 103

FTP 345

– datový kanál 346

– příkazový kanál 346

– aktivní 348

– pasivní 350

– proxy 355

full duplex 68

**G**

gateway 367

generická proxy 448

generické TLD 256

Gigabitový Ethernet 65

global unicast 328

glue records 273

GNSO (Generic Names Supporting Organization) 312

gTLD 256

## H

half close 232

HDLC 76

header 20

HINFO záznam 277

hint zóna 260

hostmaster 269

HTTP (Hypertext Transfer Protocol) 361

– cache 385

– dotaz 373

– odpověď 380

## CH

CHAP (Challenge Handshake Authentication Protocol) 91

## I

IAB (Internet Architecture Board) 311

IAC (Interpret As Command) 232, 334

ICANN (Internet Corporation for Assigned Names and Numbers) 312

ICMP 135

ICMPv6 216

identifikátor (index) síťového rozhraní 228

IEEE 15

– 802 64

– 802.11 114

– 802.11i 124

– 802.1X 124

– 802.2 124

– 802.3 111

IESG (Internet Engineering Steering Group) 311

IETF (Internet Engineering Task Force) 311

IGMP 158

IGP (Interior Gateway Protocols) 194

IMAP (Internet Message Access Protocol) 411

inkrementální zone transfer (IXFR) 305

Integrated Services 163

IP datagram 20, 21, 129, 131

IP přes Frame Relay 109

IP Router Alert 152

IPCP 101

IPv4 127

IPv6 207

IPv6 – adresa 224

IPv6CP 84

I-ráme 78, 79

IRTF (Internet Research Task Force) 311

ISDN 55

ISO (International Organization for Standardization) 15

ISO OSI 16

ISO/IEC 11801 60

Iterace 271

ITU (International Telecommunication Union) 15

IXFR 305

## J

jmenový server 255, 267

## K

kategorie strukturované kabeláže 61

komprese DNS 298

křídlová značka 77

## L

LACNIC 313

LAN (Local Area Network) 45

LCP 85

linkový rámec 20, 129

LLC 124

LMI (Local Management Interface) 110

loopback 167

LSP (Link State Protocols) 194, 198

## M

majákové rámcce 119

master server 268

- Mbone 159  
 Mergecap 39  
 metoda  
   – GET 374  
   – HEAD 378  
   – Options 379  
   – POST 377  
   – TRACE 378  
 mezilehlé uzly 369  
 modem 52  
   – ADSL 58  
 most 67  
 multicast 157, 167, 225, 226  
 Multilink Protocol (MP) 96  
 MX záznam 278
- N**
- Nagleův algoritmus 245  
 Name Server 255  
 národní TLD 256  
 NAT (Network Address Translator) 455  
 navazování TCP spojení 236  
 nedoručitelný IP datagram 138  
 Negativní caching 307  
 Neighbor solicitation 218  
 nmap 40  
 NRM (Normal response mode) 76  
 NS věta 277  
 nslookup 286  
 NTP (Network Time Protocol) 480  
 nulový modem 51  
 NVT 332
- O**
- oběžník 161  
 odmítnutí spojení 242  
 okno 79, 232, 245, 246  
 opakovač 67  
 optické vlákno 62  
   – jednovidové 63  
   – vícevidové 63  
 optický konektor 64
- originator 55  
 OSPF (Open Shortest Path First) 203  
 OUI (Organizationally Unique Identifier)  
   125  
 outdoor 71  
 oznámení o linkové adrese 218
- P**
- P/F bit 76, 124  
 packet driver 30  
 paketový přenos 26  
 PAN (Personal Area Network) 46  
 PAP 92  
 paralelní přenos dat 46  
 pasivní close 232  
 pasivní FTP 350  
 payload 20  
 personální firewall 466  
 PLCP (Physical Layer Convergence Procedure)  
   65  
 PMD (Physical Medium Dependent) 65  
 polouzavřené spojení 240  
 pomalý start 247  
 POP3 (Post Office Protocol verze 3) 408  
 port 230  
 poštovní zpráva 397  
 PPP 82  
 PPP Encryption Control Protocol (ECP) 100  
 prasečí ocásek 63  
 PRI (Primary Rate) 55  
 primární  
   – jmenný server 267  
   – master server 267  
 probe frames 121  
 promiskuitní mód 30  
 proxy 364, 445  
   – ARP 1, 56  
   – FTP 355  
 přenosové  
   – médium 17  
   – rychlosti modemů 59  
 přepínač 20

přístupový bod 115  
PTR záznam 279

**Q**

QoS 162

**R**

Radius 94  
RARP 153, 157  
redistribuce 204  
reflexivní filtr 436  
relativní URI 372  
repeater 67  
resolver 264  
Resource Records 255  
reverzní doména 258  
RFC (Request For Comments) 15  
RFC-822 391  
RIP, RIP2 a RIPng 198  
RIPE NCC (Reseaux IP Europeens Network Coordination Centre) 312  
RIR (Regional Internet Registeries) 312  
RJ-45 61  
roaming 71  
rodina protokolů TCP/IP 16  
router 21  
routing 185  
rozbočovač 67  
RR 255  
RST 233  
RVP (Routing Vector Protocols) 194, 195

**S**

scanování portů 40  
SEAL Firewalal 463  
sekundární jmenný server 268  
sekundární těsná ochrana 64  
seriové rozhraní 46  
schéma 370  
sítě infrastrukturní 71  
sítě nečíslované 181  
síťová maska 168, 170, 171

skryté sítě 453  
slave server 268  
SLIP 74  
směrovací  
– protokoly 194  
– tabulka 189  
směrovač 21  
směrování 185, 187  
směrování explicitní 150  
SMTP (Simple Mail Transfer Protocol) 391, 400  
SNAP (Subnetwork Access Protocol) 125  
sniž rychlosť odesílání 138  
SOA věta 274  
SOCKS 451  
splitter 58  
S-rámc 78, 79  
SRV věta 280  
SSAP (Source service access point address) 124  
SSID (Service Set IDentifier) 119  
SSTHRESH 247  
STA (Stanice) 114  
stealth server 268  
strukturovaná kabeláž 60  
stub zóna 260  
subdoména 256  
subsít 172  
superrámc 25  
supersíf 172  
switch 67  
symetrický signál 47  
SYN 233  
SYNCH 337  
synchronní přenos 25, 47

**Š**  
šíře přenosového pásma 103

**T**  
TCP 229  
TCP segment 20, 23

tcpdump 29  
technika okna 245  
technika zpoždění odpovědi 243  
telefonní okruh 18  
telnet 331  
TIA (Telecommunications Industry Association) 15  
TIA/EIA-568-B 60  
TIS Firewall Toolkit 463  
TLD (Top Level Domains) 256, 315, 322  
tok dat 208  
traceroute 139  
tracert 139  
trailer 20  
transparentní proxy 449  
TShark 39  
tunel 368  
TXT záznam 277

**U**

UDP (User Datagram Protocol) 251  
UDP datagram 18 23  
ukazatel naléhovaých dat 232  
ukončování spojení 240  
unicast 224  
upstream 57  
U-rámc 78, 80  
URC (Uniform Resource Characteristics) 370  
URG 233  
URI (Uniform Resource Identifiers) 370  
URL (Uniform Resource Locators) 370  
URN (Uniform Resource Names) 370

**V**

V.24 48  
V.35 48  
věty RR 271  
virtuální okruh 26  
Virtuální terminál – protokol 332  
viruswall 469  
volitelné položky IP záhlaví 145  
všeobecný oběžník 167  
vyhledávání připojených systémů 40  
vyhýbání se zahlcení 248

**W**

WAN (Wide Area Network) 45  
well known ports 230  
WEP (Wired Equivalent Privacy) 123  
whois 314  
Wi-Fi 69  
wireshark 29  
WLAN 69, 114

**X**

X.21 48

**Z**

záhlaví 20  
zahlcení sítě 247  
zakázané adresy 440  
základní pásmo 57  
zápatí 20  
zaznamenávej směrovače 147  
zenmap GUI 43  
zóna 259  
zone transfer 262, 272  
zpoždění odpovědi 243  
žádost o linkovou adresu 218