VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Implementace protokolu IPv6 v bezdrátové síti

2004 Stanislav Michalec

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

…….. …..…..……

Děkuji Ing. Petru Grygárkovi, Ph. D. za odborné vedení při psaní této diplomové práce.

Abstrakt:

V současné době se téměř v celém Internetu používá protokol IPv4. Avšak pomalu se začíná rozšiřovat také nová verze protokolu IP, a to IPv6. Diplomová práce obsahuje analýzu implementace IPv6 protokolu v síti používající pouze IPv4. Je zde popisováno nasazení protokolu IPv6 na počítačích s operačním systémem Linux, které plní funkce směrovačů. Byly vyhledány programy s podporou protokolu IPv6 dostupné pro systém Linux. V práci je uveden přehled těchto programů. Jsou také popsány praktické zkušenosti získané při jejich konfiguraci. Byl rovněž vytvořen seznam programů, které je doporučeno použít při nasazení protokolu IPv6. Část práce je věnována možnosti použití přechodového mechanismu mezi IPv4 a IPv6.

Klíčová slova:

IPv6, RIPng, OSPFv3, DHCPv6, ICMPv6, NAT-PT, TRT, Zebra, Quagga, RADVD, BIND

Abstract:

An IPv4 protocol is currently used almost in the whole Internet. However, a new version of the IP protocol is starting to be used, the IPv6 protocol. Diploma thesis contains the analysis of implementation the IPv6 protocol in the network using only IPv4 protocol. The setting of the IPv6 protocol is described on the computers with Linux operating system that work as routers. The programs with support of the IPv6 protocol available for Linux systems were searched. There is a list of these programs in the diploma thesis. The practical experiences gained from their configuration are described. A list of the programs which are recommended to be used when setting the IPv6 protocol was also created. A part of the work presents a possibility of using the transition mechanism between IPv4 and IPv6.

Keywords

IPv6, RIPng, OSPFv3, DHCPv6, ICMPv6, NAT-PT, TRT, Zebra, Quagga, RADVD, BIND

Seznam použitých zkratek:

ALG	Application Level Gateway
ARP	Address Resolution Protocol
BGP4	Border Gateway Protocol version 4
DHCP	Dynamic Host Configuration Protocol
DHCPv6	Dynamic Host Configuration Protocol version 6
DNS	Domain Name System
DNS-ALG	Domain Name System – Application Level Gateway
DUID	DHCPv6 Unique Identifier
EUI-64	Extended Unique Identifier – 64 bit
FTP	File Transfer Protocol
FTP-ALG	File Transfer Protocol – Application Level Gateway
HTTP	Hypertext Transfer Protocol
IA	Identity Association
IANA	Internet Assigned Numbers Authority
IGP	Interior Gateway Protocol
ICMP	Internet Control Message Protocol
ICMPv6	Internet Control Message Protocol version 6
IOS	Internetwork Operating System
IP	Internet Protocol
IPv4	Internet Protocol version 4
IPv6	Internet Protocol version 6
IPng	Internet Protocol Next Generation
IPsec	Internet Protocol Security
IPX	Internetwork Packet Exchange
ISP	Internet Service Provider
MAC	Media Access Control
NAT	Network Address Translation
NAPT-PT	Network Address Port Translation – Protocol Translation
NAT-PT	Network Address Translation – Protocol Translation

NIX	Neutral Internet eXchange
OSI	Open System Interconnection
OSPFv2	Open Shortest Path First version 2
OSPFv3	Open Shortest Path First version 3
RAM	Random Access Memory
RFC	Request for Comment
RIPv1	Routing Information Protocol version 1
RIPv2	Routing Information Protocol version 2
RIPng	Routing Information Protocol Next Generation
RSVP	Reservation Protocol
SP	Service Pack
TCP	Transmission Control Protocol
TRT	Transport Relay Translator
TTL	Time to Live
UDP	User Datagram Protocol
USAGI	UniverSAl playGround for Ipv6

Obsah:

1. Úvod

Cílem mé diplomové práce bylo navrhnout implementaci protokolu IPv6 v bezdrátové síti Netopýr provozované firmou Teltech. Tato bezdrátová síť má nyní implementován pouze protokol IPv4 a jako směrovače používá počítače s nainstalovaným operačním systémem Linux. Z tohoto důvodu jsem se ve své diplomové práci zaměřil na testování programů v systému Linux, které by bylo možno použít při implementaci protokolu IPv6.

V průběhu tvorby diplomové práce však došlo k situaci, že firma Teltech nebyla schopna poskytnout potřebné informace, jako např. základní topologii své sítě. Po poradě s vedoucím diplomové práce jsem se rozhodl, že budu svou práci psát obecně. To znamená, že jsem prozkoumal možnosti nasazení protokolu IPv6 v libovolné síti.

Z výše uvedeného důvodu jsem neměl možnost vyzkoušet protokol IPv6 v síti Netopýr. Proto jsem programy podporující IPv6 testoval na své domácí síti. Provedl jsem také případovou studii ve školní laboratoři, kde jsem sestavil síť z pěti směrovačů a pro konfiguraci jsem použil programy, které bych doporučil použít při implementaci IPv6 v reálné síti.

Diplomová práce je rozdělena do devíti kapitol. Kapitoly 2, 3 a 4 jsou věnovány teoretickému popisu protokolu IPv6. V kapitole 2 jsou popsány základní vlastnosti protokolu IPv6. Za ní následuje kapitola 3, ve které je popis pokročilých vlastností IPv6 a také principy fungování přechodových mechanismů mezi IPv4 a IPv6. Kapitola 4 se zabývá specifiky implementace protokolu IPv6 v bezdrátové síti.

Zbývající kapitoly jsou zaměřeny na testování programů, které podporují protokol IPv6. V kapitole 5 je uveden základní popis programů, které jsem při psaní diplomové práce otestoval. Popisu zkušeností s konfigurací protokolu IPv6 je věnována kapitola 6. V kapitole 7 jsou pak popsány možností připojení k primárnímu poskytovateli pomocí IPv6. Kapitola 8 se zabývá výše zmíněnou případovou studií. V přílohách se pak nachází konfigurační soubory, ukázky IPv6 komunikace zachycené programem Ethereal a také popisy základních konfiguračních příkazů pro konfigurování protokolu IPv6.

V zadání diplomové práce je také uvedeno, že mám prozkoumat možnosti návaznosti protokolu IPv6 na autentizační systém bezdrátové sítě Netopýr. Protože v době psaní mé diplomové práce nebyl k dispozici ani návrh, jakým způsobem bude tento autentizační systém fungovat, tak jsme se s vedoucím diplomové práce dohodli, že řešení tohoto problému zde popisovat nebudu.

2. Základní rysy protokolu IPv6

V této kapitole jsou popsány základní charakteristiky protokolu IPv6 a v některých případech také rozdíly oproti IPv4, jako např. změny v zápise adresy nebo změny ve struktuře paketu.

2.1 Vývoj protokolu IPv6

Na začátku devadesátých let začalo být zřejmé, že IP adresy dostupné v rámci protokolu IPv4 budou brzy vyčerpány. Vznikly proto snahy zadefinovat nový protokol, který by obsahoval větší adresní prostor. Z tohoto důvodu byl navržen protokol IPv6, původně označovaný jako IPng (IP Next Generation). Protože byl dostatek času pro vývoj protokolu IPv6, bylo rozhodnuto, že do něj budou zahrnuty i jiné vlastnosti.

Hlavní požadavky při návrhu protokolu IPv6 byly následující:

- rozsáhlý adresní prostor
- tři druhy adres: individuální, skupinové a výběrové
- mechanismy pro autentizaci a šifrování z důvodu zvýšení bezpečnosti
- automatická konfigurace
- podpora mobility
- plynulý přechod z IPv4 na IPv6

V důsledku těchto požadavků vznikla první specifikace protokolu IPv6 v roce 1995, kterou vytvořili Steven Deering a Robert Hinden. Jednalo se o RFC 1883: Internet Protocol, Version 6 (IPv6) Specifikation [2].

Základní specifikace nového protokolu byla vytvořena a bylo možné začít nový protokol implementovat. Místo toho ale byly zavedeny mechanismy pro šetření IPv4 adres, jako např. beztřídní adresování nebo NAT. Tím byla dočasně eliminována hlavní výhoda implementace IPv6. Začaly se masivně šířit nástroje pro NAT, čímž byly odhaleny nedostatky použití NATu. Nemožné je například navázat spojení mezi dvěma počítači, když každý je v jiné síti, která je za NATem. Začalo se tedy konečně s implementací IPv6, ať už v systémech pro koncové uživatele (např. Windows XP), tak ve směrovačích (např. Cisco).

2.2 Adresy v IPv6

V této kapitole je uveden základní popis IPv6 adres, např. v jakém formátu se IPv6 adresy zapisují, jejich rozdělení, jakým způsobem jsou přidělovány atd.

2.2.1 Tvar a zápis IPv6 adres

Aby se předešlo tomu, že by mohl někdy být nedostatek adres, mají IPv6 adresy délku 128 bitů. Adresy se zapisují vždy po čtyřech číslicích šestnáctkové soustavy oddělených dvojtečkou, tedy např.:

3ffe:0123:0000:0000:fedc:baff:fe54:3210

Přebytečné nuly můžeme vynechat a zápis adresy zkrátit na:

3ffe:123:0:0:fedc:baff:fe54:3210

Když se v někde v adrese vyskytuje více nulových skupin za sebou, můžeme je nahradit zkráceným zápisem ::, tedy uvedenou adresu můžeme zkrátit na:

3ffe:123::fedc:baff:fe54:3210

Zkráceny zápis pomocí :: můžeme v adrese použít samozřejmě pouze jednou. Kdybychom ho použili víckrát, nebylo by možné zjistit původní IPv6 adresu. Např. bychom chtěli zkrátit zápis adresy:

3ffe:44:0:0:11:22:0:33

na:

3ffe:44::11:22::33

Pak bychom nevěděli, zda původní adresa byla:

3ffe:44:0:11:22:0:0:33

nebo

3ffe:44:0:0:11:22:0:33

Stejně jako v případě IPv4 adres je IPv6 adresa rozdělena na adresu sítě a adresu uzlu. Adresu sítě pak zapisujeme následujícím způsobem:

3ffe:34:56:78:0:0:0:0/64

nebo zkráceně jako:

3ffe:34:56:78::/64

kde /64 označuje délku prefixu, tzn. že adresu sítě tvoří prvních 64 bitů zapsané adresy.

Celou adresu pak můžeme zapsat například takto:

3ffe:34:56:78:9abc:bcff:fedc:def0/64

ze které můžeme jednoduše zjistit adresu sítě a adresu uzlu.

2.2.2 Rozdělení IPv6 adres

IPv6 adresy můžeme rozdělit na tři skupiny:

- individuální (unicast)
- skupinové (multicast)
- výběrové (anycast)

Individuální adresy jsou obyčejné adresy známé z IPv4. Každá označuje jedno síťové rozhraní, kterému mají být data doručena.

Skupinové adresy jsou rovněž známé z IPv4. Když je nějaký paket zaslán na skupinovou adresu, obdrží ho všechna rozhraní, která jsou členem dané skupiny.

Výběrové adresy jsou novinkou. Myšlenka pro vytvoření těchto adres byla taková, že bude existovat více počítačů se stejnou výběrovou adresou. Pokud bude potřeba odeslat paket na výběrovou adresu, pak bude doručen na rozhraní s danou výběrovou adresou, které je nejblíže odesílateli. Výběrovým adresám nebyl přiřazen žádný speciální adresní prostor, jsou přidělovány z adresního prostoru individuálních adres.

Adresní prostor IPv6 zatím není rozdělen celý, byly přiřazeny pouze některé prefixy, zbytek adres zůstává jako rezerva pro budoucí použití. Prefixy, které již byly přiděleny, jsou uvedeny v tabulce 2.1.

Prefix binárně	Určení
0000 0000	nepřiřazeno, kromě 1)
0000 001	rezervováno pro NSAP
001	globální individuální adresy
1111 1110 10	individuální lokální linkové
1111 1110 11	individulání lokální místní
1111 1111	skupinové
ostatní	nepřiřazeno

Tabulka 2.1: Přidělené IPv6 prefixy

2.2.3 Přidělování adres na rozhraní

Stejně jako v případě IPv4 se v IPv6 přidělují adresy jednotlivým rozhraním. Ovšem na rozdíl od IPv4, kde každé rozhraní mělo obvykle jednu adresu, je v IPv6 běžné, aby každé rozhraní mělo adres více. Dokonce je nadefinováno několik adres (skupinových), které musí mít rozhraní přidělené, jako například:

- lokální linková adresa s prefixem fe80::/10
- adresa pro všechny uzly v rámci linky: ff02::1
- adresa pro vyzývaný uzel s prefixem ff02::1:ff00:0/104 (používá se při objevování sousedů)

2.2.4 Struktura globálních individuálních adres a identifikátory rozhraní

Prozatím jsou přidělovány pouze adresy s prefixem 2000::/3. Struktura těchto adres je na obrázku 2.1.

3 bity	45 bitů	16 bitů	64 bitů
001	globální směrovací prefix	ID podsítě	ID rozhraní

Obrázek 2.1: Struktura globálních individuálních adres

Dále je v definici uvedeno, že ID rozhraní bude vytvářeno pomocí modifikovaného EUI-64. Vytváření identifikátorů rozhraní pomocí EUI-64 je popsáno v [3] a take v příloze dokumentu [4]. Pro sériové linky je nutno ID rozhraní nastavit administrativně. Z tohoto důvodu je zde uveden způsob vytváření ID rozhraní na základě MAC adresy. Např. MAC adresa rozhraní je 00:0c:6e:55:c6:00. Z ní vytvoříme ID rozhraní tak, že mezi třetí a čtvrtý bajt vložíme hodnotu fffe a předposlední bit v nejvyšším bajtu MAC adresy změníme na jedničku. Výsledný identifikátor rozhraní je tedy 020c:6eff:fe55:c600.
Předposlední bit v nejvyšším bajtu MAC adresy je vlastně příznakem globality. V modifikovaném EUI-64 to znamená, že pokud je v tomto bitu jednička, pak je tento identifikátor rozhraní globální, pokud je v něm nula, pak je identifikátor lokální. V našem příkladu je uvedený identifikátor globální a tedy bylo třeba hodnotu tohoto bitu nastavit

na jedničku. Identifikátor rozhraní se používá nejenom při vytváření globálních individuálních adres, ale také např. pro určení vlastní lokální linkové adresy.

2.2.5 Dosahy adres

Každá adresa má určen rozsah platnosti, tzn. oblast, kde je tato adresa jednoznačná. Zatímco pro skupinové adresy je definováno pět stupňů dosahu, pro individuální a výběrové jsou stupně jenom tři.

Dosahy pro skupinové adresy jsou:
- rozhraní – je vlastně použitelné pouze pro zpětnovazební smyčku (ff01::/16)
- linka – dosah je omezen na jednu fyzickou síť (ff02::/16)
- místo – dosah pro jednu organizaci v jedné geografické lokalitě (ff05::/16)
- organizace – několik lokalit v rámci jedné organizace (ff08::/16)
- globální – celosvětový dosah (ff0e::/16)

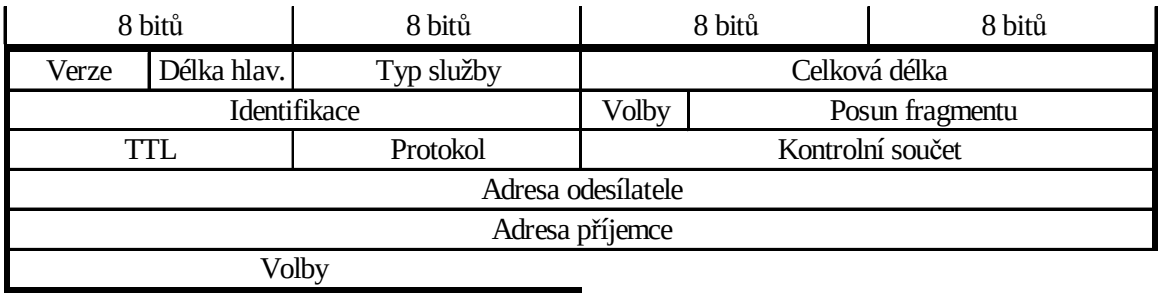
Dosahy pro individuální a výběrové adresy jsou:
- lokální pro linku (fe80::/10)
- lokální pro místo (fec0::/10)
- globální (2000::/3)

2.3 Formát paketu IPv6

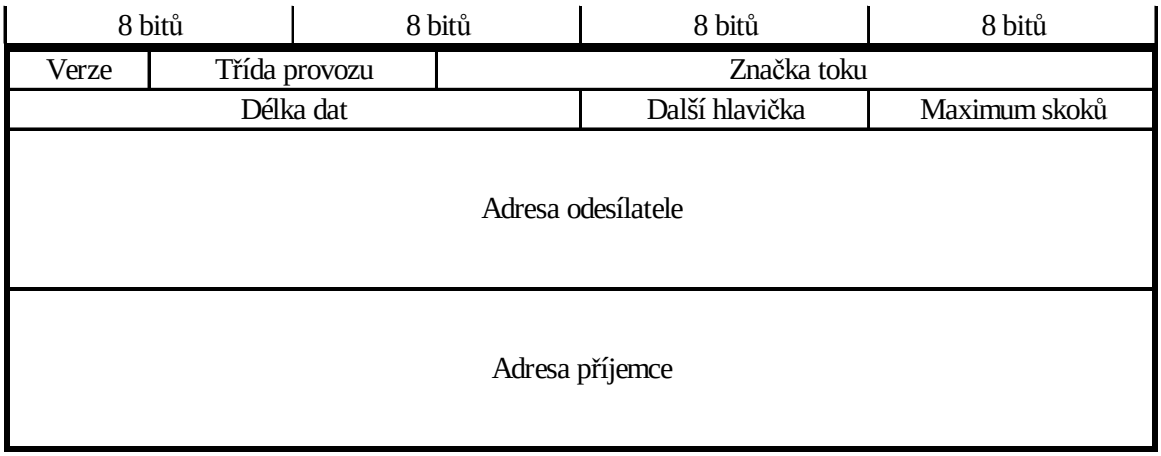
Tato kapitola se zabývá formátem IPv6 paketu. Nejprve je uvedeno, jakým způsobem se změnily hlavičky v IPv6 oproti IPv4. V další části je pak popsána zajímavá novinka v IPv6 paketu, a to rozšiřující hlavičky.

2.3.1 Porovnání hlaviček IPv6 a IPv4

IPv6 paket se stejně jako IPv4 paket skládá z hlaviček a nesených dat. Hlavním rozdílem je, že v IPv6 paketech nejsou obsaženy volby, ty byly nahrazeny rozšiřujícími hlavičkami. To umožnilo dosáhnout minimalizace základní hlavičky paketu, protože všechny méně důležité položky byly přesunuty do již zmíněných rozšiřujících hlaviček.
Pro názorné porovnání obou hlaviček se musíme podívat na jejich strukturu. Formát IPv4 hlavičky je na obrázku 2.2 a formát IPv6 hlavičky je na obrázku 2.3.



Obrázek 2.2: Formát IPv4 hlavičky



Obrázek 2.3: Formát IPv6 hlavičky

Nyní popíši jednotlivé položky IPv6 hlavičky:

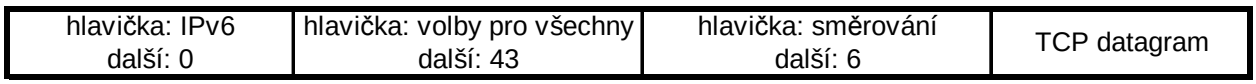
- verze – tato položka je stejná v IPv4 i IPv6, označuje verzi protokolu; v IPv6 je to číslo 6, v IPv4 je to číslo 4
- třída provozu – váže se s kvalitou služeb, je podobná položce Typ služby v IPv4

- značka toku – nová položka, která v IPv4 nebyla, jejímž účelem je urychlit směrování. Předpokládá se, že trojice <adresa odesílatele, adresa příjemce, značka toku> má jednoznačně identifikovat data, která by měla být směrována stejnou cestou
- délka dat – obsahuje informaci o počtu bajtů, které následují za standardní hlavičkou
- další hlavička – obsahuje informaci o typu dat, které následují za standardní hlavičkou. Nahrazuje IPv4 položky Volby a Protokol
- maximální počet skoků je náhradou položky TTL z IPv4.

Ve srovnání obou hlaviček je v paketu IPv6 nápadná absence položek pro fragmentaci a kontrolní součet. Podpora pro fragmentaci byla přesunuta do rozšiřujících voleb a kontrolní součet byl vyřazen bez náhrady, protože se předpokládá, že tuto službu zajišťuje nižší vrstva síťové architektury.

2.3.2 Rozšiřující hlavičky

Zajímavou novinkou IPv6 je zřetězení hlaviček. Pokud potřebujeme zahrnout do hlaviček nějaké nepovinné informace, uděláme to pomoci rozšiřujících hlaviček. Každá rozšiřující hlavička začíná položkou Další hlavička, kde je určeno, jakého typu je následující hlavička, případně data jakého protokolu následují za ní.
Příklad, jak může vypadat paket se zřetězenými hlavičkami, je na obrázku 2.4. Některé typy rozšiřujících hlaviček jsou uvedeny v tabulce 2.2 a typy nesených dat jsou v tabulce 2.3.



Obrázek 2.4: Příklad zřetězení hlaviček

0	volby pro všechny
43	směrování
44	fragmentace
50	šifrování obsahu
51	autentizace
60	volby pro cíl

Tabulka 2.2: Typy rozšiřujících hlaviček

6	TCP
9	IGP
17	UDP
58	ICMPv6

Tabulka 2.3: Typy nesených dat

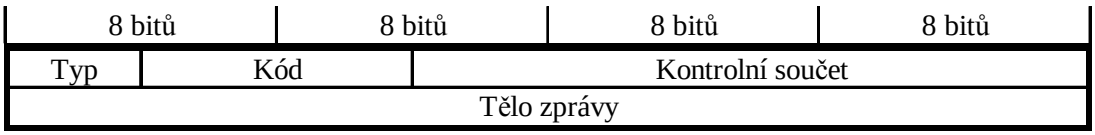
Nevýhodou zřetězení hlaviček by mohlo být, že by každý směrovač musel procházet všechny hlavičky, což by nutně znamenalo snížení výkonu. Proto bylo určeno pořadí, v jakém musí být hlavičky v paketu uspořádány:

1. standardní IPv6 hlavička
2. volby pro všechny (využití např. pro rezervační protokol RSVP)
3. volby pro cíl – pro cílové adresy určené v hlavičce směrování
4. směrování – adresy, přes které paket musí procházet
5. fragmentace
6. autentizace
7. šifrování obsahu
8. volby pro cíl – pro koncového příjemce paketu

Tímto způsobem je zaručeno, že volby důležité pro průchozí směrovače jsou na začátku paketu. Pokud průchozí směrovač nenajde hlavičku s kódem 0 (volby pro všechny), pak ví, že už v paketu nejsou pro něj žádné zajímavé informace a může ho poslat dále.
Kompletní popis rozšiřujících hlaviček včetně jejich formátu je v [1] a také v [2].

2.4 ICMPv6

Stejně jako byl v IPv4 definován pomocný protokol ICMP, tak bylo potřeba zadefinovat podobný protokol pro IPv6. Z tohoto důvodu vznikl protokol ICMPv6. Při návrhu byl rozšířen ještě o další možnosti, které se v původním ICMP nevyskytovaly, jako například objevování sousedů nebo podpora mobility.
Informaci o tom, že v paketu je obsažena ICMPv6 zpráva, určuje hodnota 58 v položce Další hlavička. Formát ICMPv6 zprávy je na obrázku 2.5.



Obrázek 2.5: Formát ICMPv6 zprávy

První položkou ve zprávě je Typ, který určuje základní druh zprávy. Nejvyšší bit v Typu pak určuje, zda se jedná o zprávu chybovou (hodnoty 0-127) nebo informační (128-255). Položka Kód obsahuje informaci o podtypu zprávy.

Tělo zprávy buď zůstává nevyužito nebo obsahuje doplňující informace, např. pro zprávu Typ č. 4 (problém s parametry) je v Těle zprávy čtyřbajtová položka, která nese informaci o počtu bajtů od začátku paketu, kde je parametr, kterému příjemce nerozumí. Některé typy ICMPv6 zpráv jsou uvedeny v tabulce 2.4.

Kompletní popis protokolu ICMPv6 lze najít v RFC 2463 - Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification [5].

Typ	Kód	Popis
1	0	Nedosažitelný cíl - neznám cestu k cíli
1	3	Nedosažitelný cíl - nedosažitelná adresa
1	4	Nedosažitelný cíl - nedosažitelný port
2	0	Příliš velký paket
3	0	Vypršela životnost paketu - maximum skoků je nulové
3	1	Vypršela životnost paketu - vypršel časovač pro doručení fragmentů
4	0	Problém s parametry - chybná položka v hlavičce
4	1	Problém s parametry - neznámy typ v položce Další hlavička
128	0	Echo - požadavek
129	0	Echo - odpověď
133	0	Výzva směrovači
134	0	Ohlášení směrovače
135	0	Výzva sousedovi
136	0	Ohlášení souseda

Tabulka 2.4: Typy ICMPv6 zpráv

2.5 Objevování sousedů

Objevování sousedů je vlastně nástupcem protokolu ARP. Na rozdíl od ARPu má však definováno více možností. Může sloužit nejenom ke zjišťování fyzických adres, nýbrž také ke hledání směrovačů, ověřování dosažitelnosti sousedů, zjišťování prefixu místní sítě apod. Ke své funkci používá ICMPv6 zprávy uvedené v předchozí kapitole.

Popis všech možností Objevování sousedů je v RFC 2461 - Neighbor Discovery for IP Version 6 (IPv6) [6]. Já se v této kapitole se zaměřím pouze na jednu součást a to na hledání fyzických adres. Tato funkce je velmi podobná ARPu. Rozdíl je v tom, že se dotazy posílají na tzv. adresu pro vyzývaný uzel. Tato adresa se vytvoří tak, že se vezme skupinová adresa s prefixem

ff02::1:ff00:0/104

a posledních 24 bitů se použije z IPv6 adresy, jejíž fyzickou adresu chceme zjistit. Tak tedy pro IPv6 adresu

3ffe::1234:56ff:fe78:9abc

bude adresa pro vyzývaný uzel

ff02::1:ff78:9abc.

Pokud počítač chce zjistit fyzickou adresu jiného počítače, vytvoří z cílové IPv6 adresy adresu pro vyzývaný uzel a na tuto adresu pošle ICMPv6 zprávu Výzva sousedovi. Pokud cílový počítač zprávu obdrží, odpoví na ni ICMPv6 zprávou Ohlášení souseda, ve které je uvedena jeho fyzická adresa.

Podmínkou správného fungování tohoto postupu je, aby každý počítač vstoupil do všech skupin určených adresami pro vyzývané uzly pro všechny síťové adresy na všech rozhraních.

3. Pokročilejší mechanismy IPv6

Pro pochopení testovaných programů, které jsou popsány v páté a šesté kapitole, jsou potřeba teoretické znalosti, které jsou uvedeny v této kapitole. Jedná se o směrování, DNS, automatickou konfiguraci a také o přechodové mechanismy mezi IPv4 a IPv6.

3.1 Směrování a směrovací protokoly

Princip směrování na IPv6 je stejný jako na IPv4. Malé rozdíly jsou jen ve směrovací tabulce, kde je třeba ukládat místo 32bitových IPv4 adres 128bitové IPv6 adresy a místo masky podsítě je třeba uložit délku prefixu. Změna nastala také v zápisu implicitní cesty, která se nyní označuje jako 0::/0 místo dřívějšího 0.0.0.0.
Pro směrování na IPv6 byly vytvořeny nové definice směrovacích protokolů jako např. RIPng, OSPFv3 a BGP4+.

3.1.1 RIPng

Při definování protokolu RIPng [7] se vycházelo z definic protokolů RIPv1 [8] a RIPv2 [9] takovým způsobem, aby v novém protokolu bylo minimum změn. Takže nakonec jediná podstatná změna je ve formátu paketu, který je předáván při výměně směrovacích tabulek. Místo 32-bitové adresy IPv4 se posílá 128-bitová adresa IPv6 a místo 32-bitové maky sítě se posílá 8-bitová hodnota, ve které je uložena délka prefixu.

3.1.2 OSPFv3

OSPFv3 [10] vychází z definice OSPFv2 [11] a přidává k němu pouze pár změn tak, aby byl tento protokol použitelný pro IPv6. Mezi nejdůležitější rozdíly patří tyto:

- vynechání autentifikace
- změna označení identifikátoru směrovače
- úprava protokolu pro použití 128bitových adres.

V protokolu OSPFv3 byla vynechána autentifikace, protože tuto službu zajišťuje IPsec, který je definován přímo v protokolu IPv6.

Jako označení směrovače (routerID) byla v OSPFv2 používána IP adresa. To už v nové verzi neplatí, protože by tento identifikátor musel mít 128 bitů. Místo toho bylo určeno, že se pro označení směrovače bude používat 32bitový identifikátor, který se bude zapisovat stejně jako IPv4 adresa.

Dále bylo ještě potřeba upravit protokol OSPFv3, aby správně rozlišoval, které počítače mohou spolu komunikovat na spojové vrstvě. Tento problém vznikl z toho důvodu, že v IPv4 byla na každé lince jedna podsíť. Naproti tomu v IPv6 může mít jedna linka prefixů více a tedy je možné, aby na spojové vrstvě komunikovaly dva počítače a každý z nich má adresu s jiným prefixem.

Další popis protokolu OSPFv3 je v RFC 2740: OSPF for IPv6 [10].

3.1.3 BGP4+

Protokol BGP4+ vychází opět ze starší verze protokolu BGP4 [13]. Nebyl dokonce definován zvlášť nový protokol, ale byl vydán pouze dokument RFC 2283: Multiprotocol Extensions for BGP-4 [12], který definuje směrování pomocí BGP4 libovolného směrovatelného protokolu síťové vrstvy (např. IPv6, IPX). V tomto dokumentu je také kladen důraz na zpětnou kompatibilitu mezi směrovačem podporujícím pouze IPv4 a směrovačem s podporou víceprotokolového rozšíření. V definici byly tedy přidány dva nové volitelné netranzitivní atributy, které směrovač bez podpory rozšíření ignoruje. Ale směrovače s podporou rozšíření si pomocí těchto atributů posílají informace o sítích, které používají jiný protokol než IPv4.

3.2 DNS

Při specifikaci nových záznamů pro DNS došlo k několika problémům. Nejprve bylo navrženo RFC 1886: DNS Extensions to support IP version 6 [14], ve kterém byl pro IPv6 určen typ záznamu AAAA. Později bylo vytvořeno RFC 2874: DNS Extensions to Support IPv6 Address Aggregation and Renumbering [15], kde bylo předchozí RFC označeno jako zastaralé a byly navrženy nové typy záznamů A6. Následovaly debaty, které RFC je tedy správné. Výsledkem byl další dokument RFC 3596: DNS Extensions to Support IP Version 6 [16], ve kterém je určeno, že pro dopředné dotazy se budou používat záznamy typu AAAA a pro zpětné dotazy byla definována doména ip6.arpa.

Při dopředných dotazech se tedy používají záznamy typu AAAA. Tyto záznamy jsou velmi podobné záznamům typu A, které se používají v IPv4. Jediným rozdílem je vlastně délka a zápis IP adresy.

Příklady AAAA záznamů:

pc1	AAAA	2002:1::2cc:ddff:fe11:2345
pc2	AAAA	2002:1::213:abff:fe99:7894

Pro zpětné dotazy byla zavedena doména ip6.arpa. Adresa IPv6 se v těchto záznamech opět zapisuje v obráceném pořadí, jednotlivé hexadecimální číslice IPv6 adresy se oddělují tečkami a nuly se samozřejmě nevynechávají. Tedy dotaz na IPv6 adresu výše uvedeného počítače pc2, by měl tvar:

4.9.8.7.9.9.E.F.F.F.B.A.3.1.2.0.0.0.0.0.0.0.0.0.1.0.0.0.2.0.0.2.ip6.arpa

Díky tomu, že číslice jsou zapsány v obráceném pořadí, je možné jednoduše spravovat reverzní domény. Předpokládejme, že budeme mít doménu mojedomena.cz a síť 2002:1::/64 a v této doméně budou počítače pc1.mojedomena.cz a pc2.mojedomena.cz. Pak můžeme použít v konfiguračním souboru pro doménu mojedomena.cz následující řádek:

$ORIGIN 0.0.0.0.0.0.0.1.0.0.0.2.0.0.2.ip6.arpa

a pak pro jednotlivé počítače pouze řádky:

5.4.3.2.1.1.E.F.F.F.D.D.C.C.2.0	PTR	pc1.mojedomena.cz
4.9.8.7.9.9.E.F.F.F.B.A.3.1.2.0	PTR	pc2.mojedomena.cz

3.3 Automatická konfigurace

Při návrhu protokolu IPv6 byly vytvořeny dvě možnosti automatické konfigurace, a to stavová a bezstavová. Stavová konfigurace vychází z protokolu DHCP a je pojmenována zcela logicky jako DHCPv6. Její kompletní definice lze najít v RFC 3315: Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [18]. Naproti tomu bezstavová konfigurace je zcela nový mechanismus a je definována v RFC 1971: IPv6 Stateless Address Autoconfiguration [17].

3.3.1 Bezstavová konfigurace

Bezstavová konfigurace je velmi jednoduchá, není u ní potřeba manuálně konfigurovat klienty a nejsou potřeba žádné další servery. Princip práce této konfigurace je takový, že si klienti sami vygenerují adresu tak, že si vytvoří identifikátor rozhraní a od směrovače zjistí prefix sítě. Spojením prefixu a identifikátoru rozhraní vznikne jejich IPv6 adresa.

Nyní bezstavovou konfiguraci popíši trochu podrobněji. Klient si po startu vytvoří identifikátor rozhraní a čeká na ICMPv6 zprávu Ohlášení směrovače, kterou směrovače posílají v náhodných časových intervalech na skupinovou adresu pro všechny uzly na dané lince. Pokud je tento interval delší, než by byl klient ochoten čekat, může poslat ICMPv6 zprávu Výzvu směrovači na skupinovou adresu pro všechny směrovače na lince. Na Výzvu směrovači je směrovač povinen odpovědět zprávou Ohlášení směrovače.

Z Ohlášení směrovače se klient dozví podstatné parametry sítě, ke které je připojen. Jsou v něm uvedeny například tyto informace:

- zda se v této síti používá také stavová konfiguraci pro nastavení adresy
- zda klient může použít stavovou konfiguraci pro další parametry sítě (kromě adresy)
- jaké jé MTU zdejší sítě
- prefixy sítě.

U každého prefixu, který se nachází v Ohlášení směrovače, jsou uvedeny ještě další informace:

- délka prefixu
- zda lze prefix použít pro určení vlastní adresy, tedy pro bezstavovou konfiguraci
- doba platnosti a doba preferování prefixu.

V době preferování je adresa označená jako preferována a počítač ji může používat bez omezení. Po vypršení doby preferování je adresa označena jako odmítaná a počítač ji může použít pouze pro komunikaci, která byla zahájena před skončením doby preferování. Po vypršení doby platnosti je adresa označená jako neplatná a počítač ji nesmí dále používat.

3.3.2 Stavová konfigurace – DHCPv6

Pro stavovou konfiguraci je v prostředí IPv6 používáno DHCPv6. Popis DHCPv6 je velmi obsáhlý, RFC 3315: Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [18], které jej popisuje, má asi 100 stran. Já se zde zaměřím jenom na základní popis fungování DHCPv6.

V prostředí DHCPv6 se používají tři typy zařízení: klient, server a zprostředkovatel (relay-agent). Protože klient komunikuje se serverem pomocí lokální linkové adresy, tak by musel být na každé lince server. Tento problém řeší zprostředkovatel, který předává informace mezi klientem a serverem a který je na každé lince, na které bude použito DHCPv6. Navíc se ještě používá pojem agent, který označuje buď server nebo zprostředkovatele, tzn. nějaké zařízení, které je na lokální lince a je schopno poskytnou DHCPv6 odpověď (svoji nebo zprostředkovanou).

Součástí návrhu DHCPv6 jsou skupinové adresy, na které bude klient posílat své dotazy. Pro tyto účely byly určeny následující adresy:

- ff02::1:2 – adresa s dosahem v rámci linky a označuje všechny DHCPv6 agenty na lince
- ff05::1:3 – adresa s dosahem v rámci místa a označuje všechny DHCPv6 servery v místě

Při komunikaci pak klient posílá dotazy na adresu ff02::1:2. Dotaz přijme agent, pokud je to server, tak přímo odpoví. Pokud je to zprostředkovatel, tak předá dotaz DHCPv6 serverům, které má nakonfigurovány (může mít nakonfigurovanou i IPv6 adresu pro všechny servery v daném místě, tzn. ff05::1:3).

Identifikace serverů a klientů

Důležitou součástí DHCPv6 je také identifikace serverů a klientů. Řeší se pomocí DHCPv6 Unique Identifier (DUID). Je to jednoznačný identifikátor počítače, který používá DHCPv6. Servery využívají tento identifikátor k jednoznačné identifikaci klienta při hledání

konfiguračních parametrů a klienti využívají DUID k identifikaci serveru. Z toho vyplývá, že je nutné, aby byl DUID neměnný a pokud je to možné, aby se nezměnil ani při výměně síťové karty.

Tvůrci standardu DHCPv6 nevytvořili univerzální identifikátor, ale rozdělili DUID do několika typů. Každý typ je použitelný v jiné specifické situaci. Zatím byly definovány tři typy vytváření tohoto identifikátoru a je určeno, že v budoucnu bude možno definovat další typy.

První typ předpokládá, že zařízení má trvanlivou zapisovatelnou paměť. Identifikátor je tvořen určením typu síťového hardware (definováno organizací IANA), linkovou adresou a časem vytvoření. Po vygenerování je tento identifikátor uložen do trvanlivé paměti. Zařízení musí uchovávat tento identifikátor, i když síťová karta, která byla použita k jeho generování, bude z počítače odstraněna. Proto zařízení bez trvanlivé paměti nemohou tento typ identifikátoru použít.

Druhý typ je tvořen pomocí identifikátoru výrobce (určuje IANA) a identifikátoru zařízení (určuje výrobce).

Třetí typ se skládá z typu síťového hardware (definuje IANA) a linkové adresy, což je podobné systému používanému v DHCPv4. Při výměně síťové karty se tento identifikátor pochopitelně změní.

Další identifikační konstrukcí je IA (Identity Association). Klient musí mít minimálně jedno IA pro každé rozhraní, pro které chce získat konfigurační informace. IA se skládá z konfiguračních informací, které jsou přiřazeny jednomu rozhraní, a IAID. IAID musí být unikátní v rámci klienta. IAID musí zůstat stejné v čase, tzn. musí být uloženo v trvanlivé paměti nebo musí být pro jeho generování použit vždy stejný algoritmus.

Klient je tedy jednoznačně identifikován pomocí DUID, rozhraní v rámci klienta je jednoznačně identifikováno pomocí IAID.

Typy DHCPv6 zpráv

Při DHCPv6 komunikaci se používají tyto typy zpráv:

- Výzva (solicit) – klient posílá tento typ zprávy, když chce zjistit seznam dostupných DHCPv6 servery
- Ohlášení (advertise) – posílá server, že je schopen přidělit klientovi požadované informace
- Žádost (request) – posílá klient, když požaduje od serveru konfigurační parametry

- Potvrzení (confirm) – posílá klient, když chce zjistit, zda adresy, které mu byly přiděleny, jsou ještě platné
- Obnovení (renew) – posílá klient, když chce prodloužit platnost konfiguračních parametrů
- Převázání (rebind) – posílá klient, když nedostal odpověď na zprávu Obnovení. Dotazuje se libovolného DHCPv6 serveru, zda mu není ochoten prodloužit platnost konfiguračních parametrů
- Odpověď (reply) – posílá server jako odpověď na zprávy Výzva, Žádost, Obnovení, Převázání, Žádost o informace, Potvrzení, Uvolnění nebo Odmítnutí
- Uvolnění (release) – posílá klient, když chce informovat server, že nebude nadále používat určitou adresu
- Odmítnutí (decline) – posílá klient, když chce informovat server, že určitou adresu už používá někdo jiný
- Rekonfigurace (reconfigure) – posílá server, když potřebuje informovat klienty, že nastala změna adres, např. při přečíslování sítě
- Žádost o informace (information-request) – klient žádá, aby mu server poslal konfigurační parametry kromě IPv6 adresy
- Zprostředkování dotazu (relay-forward) – posílá zprostředkovatel, když potřebuje předat zprávu serveru od klienta
- Zprostředkování odpovědi (relay-reply) – posílá server zprostředkovateli, aby obsaženou zprávu předal klientu.

Průběh DHCPv6 komunikace

Nyní už máme všechny potřebné informace, abych mohl popsat, jak probíhá DHCPv6 komunikace. Klient začne tím, že si vytvoří IA pro svá rozhraní včetně IAID. Pak vytvoří Výzvu a pošle ji na adresu všech DHCPv6 agentů. Tuto Výzvu obdrží server (buď přímo nebo zprostředkovaně) a odpoví pomocí zprávy Ohlášení. V této zprávě jsou uvedeny konfigurační parametry, které by klientovi přiřadil v případě, že by klient o to zažádal. Je tam rovněž uvedena preference s jakou je ochoten server přidělit danému klientovi konfigurační parametry. Tato odpověď je předána klientovi (opět buď přímo nebo zprostředkovaně).

Klient si z došlých odpovědí vybere tu s největší preferencí. Pokud obdrží více Ohlášení se stejnou preferencí, může se rozhodnout podle poskytnutých informací (např. podle nabídnutých

adres). Po výběru vytvoří zprávu Žádost, do které zapíše DUID serveru, od kterého požaduje konfigurační parametry. Protože ještě nemá informace o síti, pošle tuto zprávu opět na adresu všech DHCP agentů. Servery, jímž Žádost nepatří, ji ignorují. Vybraný server Žádost příjme a pošle zpátky zprávu Odpověď, ve které uvede konfigurační parametry. Mohlo by se také stát, že by Žádosti nevyhověl, ale protože klient vybíral nejpreferovanější server, tak je tato možnost velice nepravděpodobná. Po přijetí Odpovědi si klient ověří, zda danou adresu již někdo nepoužívá. Pokud zjistí, že už je používaná jiným počítačem, může adresu odmítnout pomocí zprávy typu Odmítnutí.

Adresy jsou přidělovány na určitý čas. Pokud dojde k jeho vypršení, klient žádá o prodloužení platnosti adresy pomocí zprávy Obnovení, kterou posílá na adresu serveru, který mu adresu přidělil. Pokud neobdrží od daného serveru odpověď, pak pošle zprávu Převázání na všechny dostupné servery s dotazem, zda není některý z nich ochoten mu danou adresu prodloužit.

Při návratu klienta do sítě (např. po fyzickém odpojení od sítě) klient musí zjistit, zda je jeho adresa ještě platná. Pro tento případ použije zprávu Potvrzení.

Na všechny předchozí zprávy (tzn. Odmítnutí, Obnovení, Převázání a Potvrzení) reaguje server zprávou Odpověď, ve která uvádí, zda požadavku klienta vyhověl nebo ne a pokud nevyhověl, tak z jakého důvodu.

Když klient končí svou práci v síti (např. před vypnutím), měl by informovat server, že přiřazenou adresu už nepotřebuje pomocí zprávy Uvolnění.

Komunikaci vždy zahajuje klient, výjimku tvoří situace, když dojde ke změně síťových parametrů (např. přečíslování sítě). V tomto případě zahajuje komunikaci server a posílá na adresy všech klientů zprávu Rekonfigurace. Klienti reagují odeslání zprávy Obnovení a server pak pošle nové informace v Odpovědi.

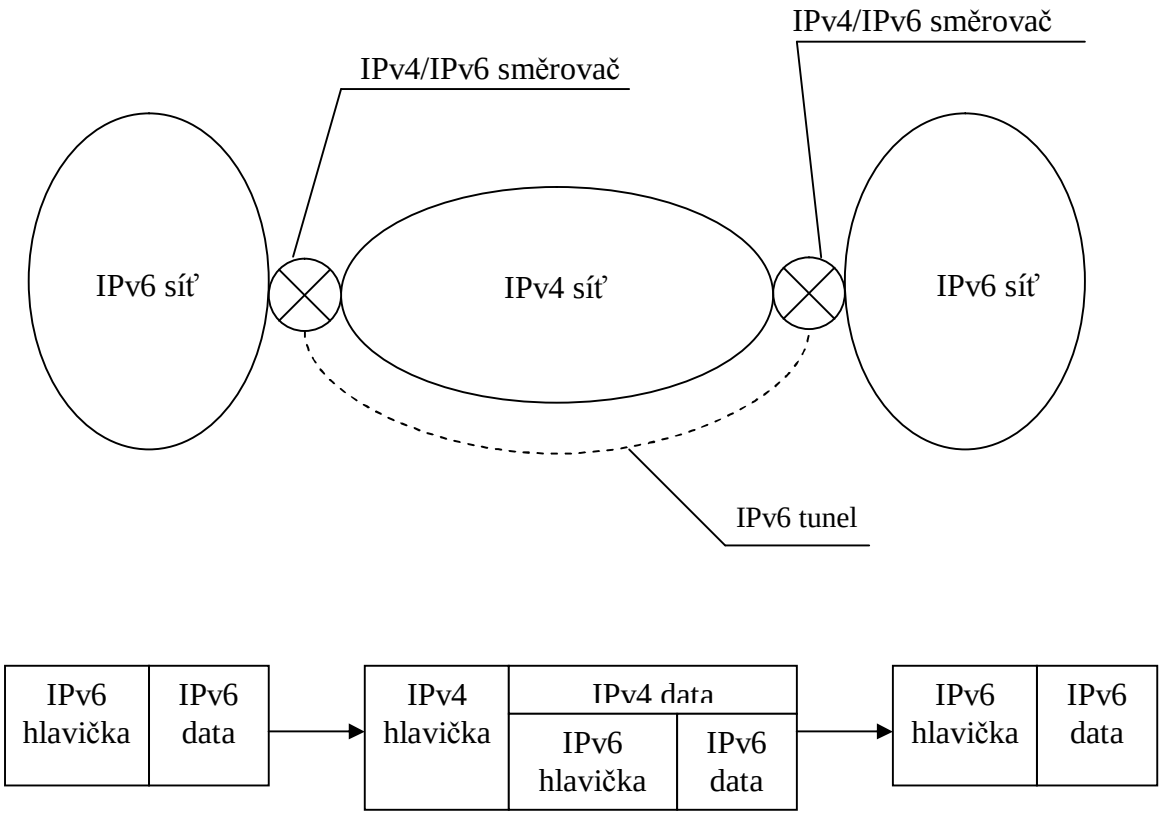
3.4 Přechodové mechanismy

Velmi důležitou roli při nasazování nového protokolu IPv6 hrají přechodové mechanismy, které se dělí na tunelovací a překladové. Tunelovací mechanismy umožňují, aby počítače v různých IPv6 sítích mohly komunikovat skrz IPv4 síť. Naproti tomu překladové mechanismy umožňují, aby počítače používající protokol IPv6 mohly komunikovat s počítači používajícími IPv4.

Důležitým prvkem přechodových mechanismů jsou síťová zařízení, která mají dvojí protokolový zásobník. To znamená, že tyto zařízení umí komunikovat pomocí IPv4 i IPv6 a mohou tedy provádět překlady protokolů a tunelování.

3.4.1 Tunely

Představme si situaci, kdy existují dvě IPv6 sítě propojené pomocí sítě IPv4 a potřebujeme, aby spolu komunikovaly dva počítače, z nichž každý je v jiné IPv6 síti (obrázek 3.1). Řešením této situace je použití tunelu.



Obrázek 3.1: Tunel

Princip tunelování je velice jednoduchý: vytvoříme tunel, jehož konce jsou na hraničních směrovačích obou sítí. Když je na hraniční směrovač doručen IPv6 paket, jehož cílová adresa je na druhém konci tunelu, pak směrovač celý IPv6 paket zabalí do datové části nového IPv4 paketu a pošle na IPv4 adresu druhého konce tunelu. Když hraniční směrovač ve druhé síti obdrží takovýto paket, vybalí z něj původní IPv6 paket a ten pošle na cílovou IPv6 adresu. Názorná ukázka, v jaké situaci se používá tunel a jaký formát mají tunelovaná data je na obrázku 3.1.

Tunely můžeme vytvářet manuálně, poloautomaticky (pomocí tunel serverů) nebo automaticky (pomocí IPv4 kompatibilních adres nebo IPv4 překládaných adres). Podrobnější popis těchto technik lze najít v [1].

3.4.2 Překladové mechanismy

V průběhu nasazování protokolu IPv6 může dojít a většinou taky dojde k situaci, že počítač, který používá IPv6, bude potřebovat komunikovat s počítačem používajícím IPv4. K řešení této situace je potřeba použít nějaký překladový mechanismus. Já se zde zaměřím na dva z nich, ke kterým existuje v současné době implementace, a to NAT-PT (NAPT-PT) a TRT. Dále jsou definovány i jiné přechodové mechanismy, jako např. Socks64 nebo IPv64, ale tyto mechanismy zde popisovat nebudu, protože jejich implementace je už buď hodně zastaralá a momentálně se nevyvíjí nebo implementace ještě nebyla vůbec vytvořena. Jejich popis je uveden v [1].

NAT-PT

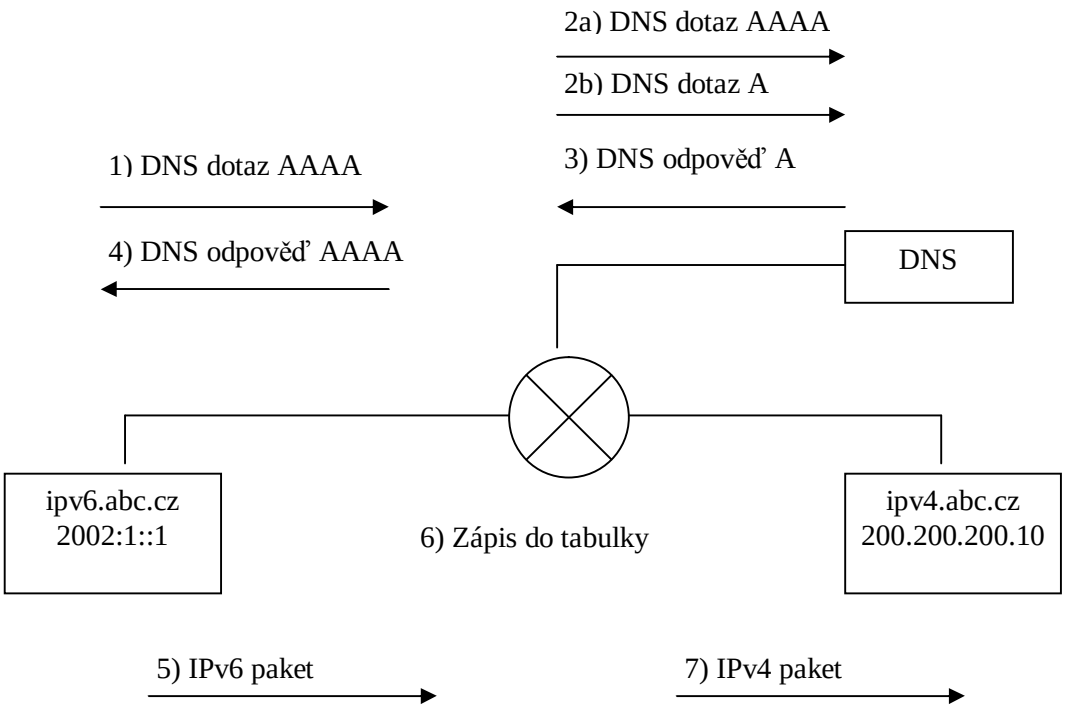
Prvním popisovaným překladovým mechanismem je NAT-PT, který vznikl rozšířením současně používaného NATu. U NATu dochází pouze k překladu IPv4 adresy na jinou IPv4 adresu, zatímco NAT-PT je o něco složitější, protože v něm dochází zároveň k překladu protokolu. Je to v současné době jediný použitelný mechanismus, který umožňuje, aby komunikaci zahájil jak počítač v IPv4 sítí, tak i počítač v IPv6 síti. To je umožněno díky definici aplikační brány (Aplication Level Gateway) pro DNS (DNS-ALG).

Proces překladu protokolů probíhá na NAT-PT překladači. Proto je nezbytné, aby všechny pakety, které vyžadují překlad protokolu, byly směrovány přes NAT-PT překladač. Z tohoto

důvodu je výhodné, aby překladačem byl hraniční směrovač sítě. K mapování adres bude překladač potřebovat skupinu IPv4 adres a IPv6 prefix, aby mohl rozpoznat, u kterých paketů musí provést překlad a které má směrovat bez úprav.

Princip práce překladače ukážu na dvou příkladech pro oba způsoby navazování komunikace. V obou případech bude mít IPv6 počítač jméno ipv6.abc.cz a adresu 2002:1::1 a IPv4 počítač jméno ipv4.abc.cz a adresu 200.200.200.10. Překladač bude mít pro mapování k dispozici prefix 2002:2::/96 a adresy 200.200.200.128/28.

Jak vypadá komunikace při navazování spojení z IPv6 do IPv4 je ukázáno na obrázku 3.2.



Obrázek 3.2: NAT-PT - komunikaci navazuje IPv6 klient

Popis průběhu komunikace je následující:

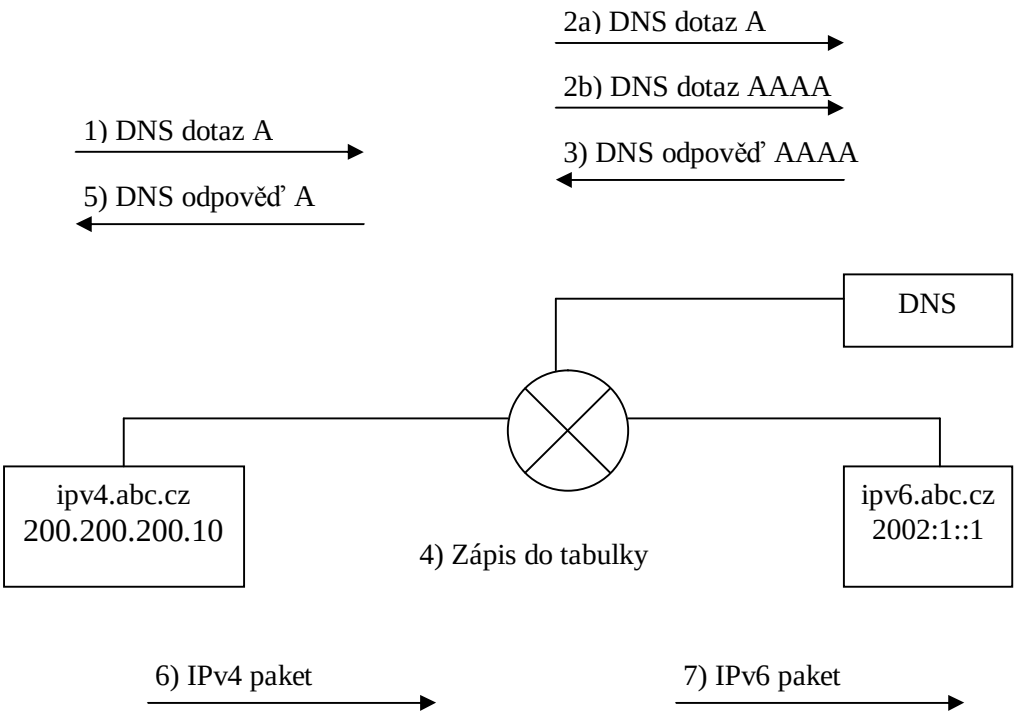
1) ipv6.abc.cz posílá DNS dotaz na záznam typu AAAA počítače ipv4.abc.cz
2) Tento dotaz prochází přes NAT-PT směrovač, který neví, zda ipv4.abc.cz má IPv4 nebo IPv6 adresu. Proto uplatní DNS-ALG a:
   a) ponechá původní dotaz beze změny a pošle dál
   b) vytvoří druhý dotaz na záznam typu A

3) Oba dotazy jsou doručeny na DNS server a protože ipv4.abc.cz nemá IPv6 adresu, bude zpět odeslána pouze DNS odpověď se záznamem typu A (ipv4.abc.cz má adresu 200.200.200.10)
4) DNS odpověď bude procházet přes NAT-PT a protože se ipv6.abc.cz ptal na záznam typu AAAA a odpověď je typu A, uplatní se znovu DNS-ALG a změní DNS odpověď na záznam typu AAAA s obsahem ipv4.abc.cz má adresu 2002:2::200.200.200.10
5) ipv6.abc.cz obdržel IPv6 adresu počítače ipv4.abc.cz a proto může zahájit komunikaci. Odešle tedy IPv6 paket se zdrojovou adresou 2002:1::1 a cílovou adresou 2002:2::200.200.200.10
6) Tento paket prochází přes NAT-PT, který podle prefixu 2002:2::/64 cílové adresy pozná, že má tento paket přeložit. Vybere tedy volnou IPv4 adresu, např. 200.200.200.129 a vytvoří IPv4 paket se zdrojovou adresou 200.200.200.129 a cílovou adresou 200.200.200.10, kterou zjistí z cílové adresy IPv6 paketu. Protože je to paket zahajující komunikaci, tak si zároveň do své překladové tabulky uloží dvojici adres 200.200.200.129 a 2002:1::1 a bude tedy vědět, že jestliže přijde nějaký paket s cílovou adresou 200.200.200.129, má ho poslat na adresu 2002:1::1.
7) Přeložený paket se zdrojovou adresou 200.200.200.129 a cílovou adresou 200.200.200.10 je poté odeslán počítači ipv4.abc.cz

Jak vypadá komunikace při navazování spojení z IPv4 do IPv6 je ukázáno na obrázku 3.3. Popis průběhu komunikace je následující:

1) ipv4.abc.cz posílá DNS dotaz na záznam typu A počítače ipv6.abc.cz
2) Tento dotaz prochází přes NAT-PT směrovač který neví, zda ipv6.abc.cz má IPv4 nebo IPv6 adresu. Proto uplatní DNS-ALG a:
   a) ponechá původní dotaz beze změny a pošle dál
   b) vytvoří druhý dotaz na záznam typu AAAA
3) Oba dotazy jsou doručeny na DNS server a protože ipv6.abc.cz nemá IPv4 adresu, bude zpět odeslána pouze DNS odpověď se záznamem typu AAAA (ipv6.abc.cz má adresu 2002:1::1)
4) DNS odpověď bude procházet přes NAT-PT a protože se ipv4.abc.cz ptal na záznam typu A a odpověď je typu AAAA, uplatní se znovu DNS-ALG a změní DNS odpověď na záznam typu A.

5) Vybere volnou adresu ze skupiny IPv4 adres, např. 200.200.200.130. Do překládací tabulky si uloží dvojici adres 200.200.200.130 a 2002:1::1 a bude vědět, že pokud obdrží paket z cílovou adresou 200.200.200.130, pak ho má odeslat na adresu 2002:1::1.
6) Změněná DNS odpověď s obsahem ipv6.abc.cz má adresu 200.200.200.130 je doručena počítači ipv4.abc.cz.
7) ipv4.abc.cz obdržel IPv4 adresu vzdáleného počítače a proto může zahájit komunikaci. Odešle tedy IPv4 paket se zdrojovou adresou 200.200.200.10 a cílovou adresou 200.200.200.130
8) Přeložený paket se zdrojovou adresou 2002:2::200.200.200.10 a cílovou adresou 2002:1::1 je poté odeslán počítači ipv6.abc.cz



Obrázek 3.3: NAT-PT - komunikaci navazuje IPv4 klient

Z těchto dvou příkladu je vidět, jaký je hlavní rozdíl mezi oběma směry navazování komunikace. Zatímco při navazování komunikace z IPv6 do IPv4 se adresy do překladové tabulky ukládají při příchodu paketu zahajujícího spojení, tak při navazování komunikace z IPv4 do IPv6 se adresy do tabulky ukládají při úpravě DNS odpovědi.

Z tohoto důvodu je také rozdíl v cacheování DNS odpovědí. Při navazování spojení z IPv6 do IPv4 je doporučováno ukládat DNS odpovědi do paměti cache, aby nedocházelo ke zbytečnému zatěžování překladače při úpravách DNS záznamů. V opačném směru je ale třeba nastavit životnost změněné DNS odpovědi na nulu a tak cacheování zakázat, protože jinak bychom mohli obdržet DNS odpověď z vyrovnávací paměti a příslušný záznam v překladové tabulce by už mohl být zrušen.

Používání NAT-PT má ovšem určitou nevýhodu. Protože IPv6 adres je dostatek, tak při mapování IPv4 na IPv6 žádný problém nenastane. Horší je ale situace při mapování IPv6 na IPv4, protože nám brzo mohou chybět volné IPv4 adresy. Řešením této situace je NAPT-PT, který nemapuje jenom síťové adresy, ale také porty. Tedy umožňuje, aby na jednu IPv4 adresu bylo namapováno kolem 65 tisíc IPv6 adres.

Velký problém působí NAT-PT aplikačním protokolům, které ve své datové části přenášejí přímo síťovou adresu a číslo portu. Příkladem takového protokolu je FTP. Pro každý takový protokol by NAT-PT musel obsahovat zvláštní podporu, jinak by tento protokol nemohl fungovat. Pro FTP je určena FTP-ALG, která umožňuje používat protokol FTP i při NAT-PT, ovšem jiné protokoly, které obsahují v datové části adresu a číslo portu, fungovat nebudou. Pro protokol FTP je navíc doporučováno nahradit původní příkazy PASV a PORT, které obsahují adresu a port počítače, novějšími příkazy EPSV a EPRT.

Kompletní popis NAT-PT se nachází v RFC 2766: Network Address Translation - Protocol Translation (NAT-PT) [19].

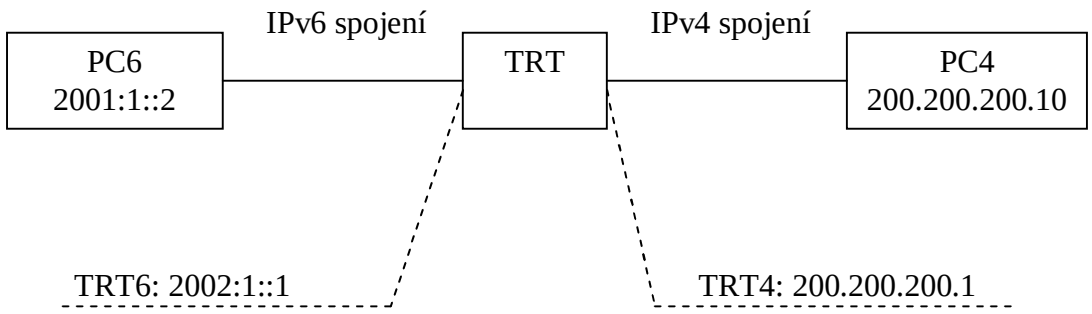
TRT

Druhým používaným přechodovým mechanismem je TRT (Transport Relay Translator). Ten je ale možno použít pouze při navazování spojení z IPv6 do IPv4.

Princip práce TRT je odvozen z dnes používaných firewallů a používá jednoduchou metodu. Místo jednoho TCP spojení jsou vytvořena spojení dvě, jedno na IPv6 a druhé na IPv4. TRT pak předává data mezi těmito spojeními. Podrobněji popíši práci TRT opět na příkladu.

V síti je určen prefix, který je směrován na TRT zařízení. V tomto příkladu bude prefix 2002:2::/64. Komunikaci zahajuje PC6 (viz obrázek 3.4), který chce komunikovat s PC4, který má IPv4 adresu 200.200.200.10. Pošle tedy paket na cílovou adresu 2002:2::200.200.200.10. Komunikace je směrována na zařízení s podporou TRT. Na rozhraní TRT6 (viz obrázek 3.4) je toto spojení zachyceno a TRT při komunikaci s PC6 předstírá, že je PC4. TRT z IPv6 adresy

2002:2::200.200.200.10 zjistí, že PC4 má adresu 200.200.200.10 a vytvoří druhé spojení, tentokrát na IPv4, mezi TRT4 a PC4. Při komunikaci s PC4 pak TRT předstírá, že je PC6. Takže místo jednoho TCP spojení jsou vytvořena spojení dvě, jedno mezi PC6 a TRT6 a druhé mezi TRT4 a PC4. Pak už TRT předává data mezi oběma spojeními. Názorná ukázka, jak vypadá komunikace pomocí TRT je na obrázku 3.4.



Obrázek 3.4: Komunikace pomocí TRT

Komunikace pomocí protokolu UDP funguje podobně, do překladové tabulky se ukládají kombinace IP adres a portů a tyto záznamy jsou z překladové tabulky vyřazovány na základě časovače.

Jak vidíme z příkladu, je třeba pro zahájení komunikace použít speciální tvar adresy. Specifikace TRT neuvádí přesně, odkud se mají tyto adresy zjišťovat, pouze nabízí základní návrhy, jako např. statické mapování (/etc/hosts v Unixu) nebo speciální implementace DNS serveru.

Úplný popis mechanismu TRT je v RFC 3142: An IPv6-to-IPv4 Transport Relay Translator [20].

4. Specifika implementace protokolu IPv6 v bezdrátové síti

Při implementaci protokolu IPv6 v bezdrátové síti je využito výhod, které poskytuje vrstvený model. Protože standardy pro bezdrátové sítě jsou definovány na prvních dvou vrstvách (fyzické a spojové) OSI modelu a protokol IPv6 je definován na třetí (síťové) vrstvě, nemělo by být nasazení tohoto protokolu žádným problémem.
To, že by v tomto případě neměly nastat žádné problémy, dokazuje i dokument Transmission of IPv6 Packets over WiFi Networks [21], který je sice zatím k dispozici pouze v podobě draftu, ale ve většině bodů se odkazuje na RFC 2464: Transmission of IPv6 Packets over Ethernet Networks [22].

Abych ověřil, jestli je opravdu komunikace IPv6 na bezdrátové sítí bezproblémová, tak jsem se rozhodl, že to otestuji. Sestavil jsem jednoduchou bezdrátovou síť podle schématu, které je na obrázku 4.1. Poté jsem vyzkoušel komunikaci mezi počítači PC1 a PC2 na IPv6. Síťový provoz jsem analyzoval programem Ethereal a výsledkem bylo, že komunikace fungovala bezproblémově.

Obrázek 4.1: IPv6 komunikace pomocí WiFi

S ohledem na výše uvedené skutečnosti a také proto, že jsem neměl k dispozici větší bezdrátovou síť, jsem další mechanismy IPv6 testoval na sítích typu Ethernet a předpokládal jsem, že na bezdrátových sítích bude komunikace probíhat stejným způsobem.

5. Dostupný software podporující IPv6

Návrh implementace protokolu IPv6 na aktivních prvcích sítě má být zaměřen na operační systém Linux. Z tohoto důvodu popíši softwarové požadavky, které jsou potřebné pro použití protokolu IPv6 v tomto operačním systému. Ovšem předpokládá se, že na klientských počítačích může být nainstalován operační systém Windows. Z tohoto důvodu také popíši, jaký je stav implementace protokolu IPv6 v těchto operačních systémech.
Nejprve je tedy uveden stav implementací protokolu IPv6 v operačních systémech. Dále jsou pak uvedeny programy podporující protokol IPv6 v operačním systému Linux.

5.1 Implementace IPv6 v operačních systémech

V této podkapitole je popsáno, v jakém stavu jsou implementace protokolu IPv6 v operačních systémech Linux a Windows.

5.1.1 Linuxové jádro a projekt USAGI

Základní implementace protokolu IPv6 byla již v jádrech verze 2.2.x. Ale tato implementace byla prakticky nepoužitelná, což dokládají testy, které jsou k dispozici na stránkách projektu USAGI (http://www.linux-ipv6.org). Proto byl založen projekt USAGI, jehož cílem bylo vytvořit implementaci IPv6, kterou by bylo možno bezproblémově použít.
Pro použití protokolu IPv6 je tedy požadováno jádro verze 2.4.x nebo 2.6.x. Ve standardních jádrech verze 2.4.x je začleněna pouze základní podpora protokolu IPv6. Pro používání pokročilejších vlastností IPv6 (např. IPsec) nebo při nasazení protokolu IPv6 v produkčním prostředí je doporučováno aplikování patche, který najdeme na serveru projektu USAGI.

Při svých testech jsem používal linuxová jádra verze 2.4.20 až 2.4.25 s aplikovaným patchem projektu USAGI pro příslušnou verzi jádra. Při kompilaci jádra s podporou IPv6 ani při aplikaci USAGI patche jsem neměl žádný problém.
Při dalších popisech práce s IPv6 se předpokládá, že je použito jádro se zakompilovanou podporou protokolu IPv6 a pokud byla podpora IPv6 zkompilována jako modul, předpokládám, že je tento modul zaveden.

5.1.2 Implementace IPv6 v systémech Windows

V systémech Windows 95, 98, ME a NT není protokol IPv6 implementován. Pro systém Windows 2000 je podpora pro protokol IPv6 k dispozici ve formě patche, a to pro verzi systému se SP1. Na Internetu lze nalézt návod, jak tento patch jednoduše upravit, aby se dal aplikovat na systémy Windows 2000 se SP2 až SP4. Já jsem zkoušel nainstalovat podporu pro IPv6 na dvou různých počítačích se systémem Windows 2000, na jednom z nich byl SP3, na druhém SP4. Bohužel ani jednou mi protokol IPv6 nefungoval. Pokus o spuštění protokolu IPv6 vždy skončil hlášením, že není možné inicializovat IPv6 protokolový zásobník.
Nejlépe je na tom z pohledu IPv6 systém Windows XP, protože má podporu pro IPv6 přímo v systému. IPv6 stačí spustit příkazem

ipv6 install

a po chvilce je možno IPv6 používat. Zkušenosti s používáním IPv6 v systému Windows XP budou popsány v následujících kapitolách.
Chtěl bych ovšem upozornit na jednu nepříjemnou vlastnost systému Windows XP. Pokud máme spuštěnou podporu IPv6 a nemáme IPv6 konektivitu, pak se nebudeme moci připojit k počítačům, které mají IPv4 i IPv6 adresu. Komunikace ve Windows XP totiž probíhá tak, že se nejdříve vyšle DNS dotaz na záznam typu AAAA a pokud počítač obdrží odpověď s IPv6 adresou, tak se okamžitě pokusí na danou adresu připojit. Když však nemá IPv6 konektivitu, tak tento pokus o spojení nebude úspěšný a počítač se už nepokouší vyslat DNS dotaz na záznam typu A, ale vypíše chybové hlášení.

5.2 Programy podporující IPv6 v systému Linux

Některé programy umožňují konfigurovat více vlastností protokolu IPv6, jako např. Zebra obsahuje podporu pro manuální přiřazení adresy na rozhraní, ohlášení směrovače a také pro směrování. Z tohoto důvodu jsou obecné vlastnosti tohoto programu popsány v této kapitole. Naproti tomu jiné programy, jako např. program pTRTd implementuje pouze jednu část IPv6 a proto je v této kapitole pouze zmíněn a popis zkušeností při používání tohoto programu je v šesté kapitole.

5.2.1 RADVD

Zkratka RADVD vznikla ze slov Router ADVertisiment Daemon. Je to tedy program, který se používá pro posílání Ohlášení směrovače, když chceme například poslat všem počítačům na dané lince informaci, že se na ní používá určitý prefix.
V průběhu vývoje programu RADVD došlo k tomu, že původní autor přestal program dále vyvíjet. Po něm tuto práci převzal jiný programátor, který ovšem pracoval na úpravách tohoto programu pouze jeden rok. Proto je možné program RADVD najít na dvou odlišných serverech. Původní program je na adrese ftp://ftp.cityline.net/pub/systems/linux/network/ipv6/radvd, ovšem pouze do verze 0.6.2, která je ze začátku roku 2001. Novější verze jsou pak na serveru http://v6web.litech.org/radvd, na kterém je nejnovější verze 0.7.2 ze začátku roku 2002.
Zkušenosti s používáním programu RADVD jsou popsány v kapitole 6.2.1.

5.2.2 Zebra, Quagga a ZebOS

Zebra (http://www.zebra.org) je volně šiřitelný software, který umožňuje použít počítač s operačním systémem Linux jako plnohodnotný směrovač. Je vyvíjena už od roku 1996, ale v současné době se už nové aktualizace objevují pouze zřídka. Je to zřejmě způsobené skutečností, že vývoj Zebry je podporován firmou IP Infusion (http://www.ipinfusion.com). Avšak tato firma vyvíjí svůj vlastní směrovací software, který pojmenovala ZebOS. ZebOS je

ale na rozdíl od Zebry komerční produkt, tedy podle mého názoru nemá firma IP Infusion zájem, aby se Zebra dále vyvíjela a byla tak vlastně konkurencí k jejich komerčnímu produktu.

V průběhu vývoje Zebry došlo k rozdvojení. V srpnu 2003 byl na základě Zebry vytvořen projekt Quagga (http://www.quagga.net), jehož cílem je dále vylepšovat tento směrovací software. Nicméně ani v tomto projektu se poslední dobou neobjevují nové verze programu příliš často.

Chtěl jsem rovněž vyzkoušet výše zmiňovaný ZebOS. Firma IP Infusion sice na svých stránkách nabízí možnost po zaregistrování si vyzkoušet tento produkt po dobu 30 dnů, avšak po e-mailovém dotazu mi sdělili, že vývoj v binární podobě (binary evaluation) byl ukončen a tedy vyzkoušet ZebOS nemohu. Jedinou možností je zakoupit si licenci na zdrojové kódy, jejíž cena je minimálně 50 000 USD.

Struktura Zebry a Quaggy je stejná, takže ji popíši společně. Když budu dále mluvit o Zebře, myslím tím zároveň Zebru i Quaggu. Základem je démon zebra, který spolupracuje s jádrem systému a shromažďuje informace od ostatních směrovacích démonů (např. ripngd, ospf6d). Ti ovládají jednotlivé směrovací protokoly a předávají informace démonu zebra. Pro každý podporovaný směrovací protokol je tedy k dispozici jeden démon.

Ovládat Zebru je možné různými způsoby:

1) můžeme upravovat přímo konfigurační soubory jednotlivých démonů,
2) pracovat s jednotlivými démony pomocí příkazového řádku. Toto ovládání je velmi podobné jako u směrovačů firmy Cisco s OS IOS
3) nastavovat konfiguraci pomocí vtysh, což je vlastně centrální konfigurační program celé Zebry. Pracuje se v něm také pomocí příkazového řádku podobného Cisco směrovačům, ale můžeme v něm měnit konfiguraci všech démonů v rámci Zebry.

Práce se Zebrou byla většinou bezproblémová. Jediné připomínky bych měl k ovládání pomocí jednotného rozhraní vtysh. Ve starších verzích bylo možné pomocí rozhraní vtysh konfigurovat všechny démony Zebry, avšak bylo potřeba napsat příkaz pro uložení konfigurace různých démonů vždy v jiném konfiguračním módu. To bylo ve verzi 0.94 vylepšeno a je možné konfiguraci všech démonů uložit najednou. Nepříjemné ale stále je, že při spuštění vtysh se konfigurace nenačte automaticky, ale je třeba to specifikovat parametrem.

Další nepříjemnou vlastností vtysh je, že pokud používáme příkazy, kterými konfigurujeme démona, který není spuštěn nebo byl spuštěn později než vtysh, tak se po zadání příkazu neobjeví žádná chybová hláška, ale uvedený příkaz se nezapíše do konfigurace. Objevil jsem

ještě jednu chybu programu vtysh a to, že po ukončení programu vtysh a návratu do terminálu se v terminálu nezobrazují znaky zapsané z klávesnice.

5.2.3 Ostatní programy

Zbývající programy podporují pouze jednu součást protokolu IPv6. Z tohoto důvodu je v této kapitole pouze seznam testovaných programů a zkušenosti s používáním těchto programů jsou popsány v kapitole 6.

Pro konfiguraci DHCPv6 jsem otestoval tyto tři projekty:

- DHCPv6 implementation on Linux (http://sourceforge.net/projects/dhcpv6-linux/)
- DHCPv6 (http://sourceforge.net/projects/dhcpv6/)
- Dibbler (http://klub.com.pl/dhcpv6/)

Pro použití DNS jsem otestoval program BIND, jehož domovská stránka je http://www.isc.org/sw/bind/.

Pro nasazení přechodového mechanismu TRT jsem otestoval spolupráci programů pTRTd (http://www.litech.org/ptrtd/) a ToTd (http://www.vermicelli.pasta.cs.uit.no/) a také v operačním systému FreeBSD spolupráci programů FAITH a ToTd.

6. Zkušenosti s konfigurací IPv6

Pro zprovoznění IPv6 komunikace na síti je nutné provést následující kroky:

1) přiřadit na rozhraní směrovačů IPv6 adresy
2) zprovoznit automatickou konfigurace IPv6 adres pro klienty
3) nakonfigurovat směrování
4) nakonfigurovat DNS server
5) nakonfigurovat přechodový mechanismus mezi IPv4 a IPv6 (tento krok je volitelný)

Protože většinu těchto kroků lze nakonfigurovat pomocí Zebry, tak se na tuto možnost vždy zaměřím. Nicméně pokud bude i jiná možnost, jak danou vlastnost nakonfigurovat, tak se o ní přinejmenším zmíním.

6.1 Ruční přiřazení adresy IPv6 na rozhraní

V této kapitole je přiřazení adresy IPv6 na rozhraní myšleno jako ruční konfigurace IPv6 adresy. Možnosti, jak nakonfigurovat klientský počítač pomocí automatické konfigurace, jsou popsány v kapitole 6.2.

6.1.1 Konfigurace adres v Linuxu

Pokud chceme ručně nastavit adresu na klientském počítači, můžeme tak učinit pomocí příkazu *ifconfig*. V případě nastavení adresy na počítači, který pracuje jako směrovač, můžeme opět použít příkaz *ifconfig*. Máme zde ale také možnost nakonfigurovat IPv6 adresu v Zebře.

Při testování obou možností nastavení IPv6 adresy nevznikly žádné problémy. Mezi počítači, na kterých byly nastaveny IPv6 adresy, okamžitě fungovala IPv6 komunikace, jejíž funkčnost jsem ověřoval pomocí příkazu *ping6*.

Přesná syntaxe příkazu pro přidání a odebrání IPv6 adresy v Linuxu je uvedena v Příloze 1.

6.1.2 Konfigurace adres ve Windows XP

V případě systému Windows XP, který bude vystupovat jako klientský, je možno nastavit IPv6 adresu pomocí příkazového řádku. Toto je možné provést pomocí příkazu *ipv6* s parametrem *adu*.
Rovněž v tomto případě bylo přiřazování adres na rozhraní bezproblémové a po přiřazení jsem testoval funkčnost spojení pomocí příkazu *ping6*. Výsledkem bylo, že ihned po přiřazení IPv6 adres bylo spojení funkční.
Zde bych chtěl upozornit, že neexistuje zvláštní parametr příkazu *ipv6* pro odebrání IPv6 adresy z rozhraní, je tak třeba učinit opět příkazem *ipv6* s parametrem *adu* a je třeba zadat volitelný parametr *life* s hodnotou 0. Tento parametr označuje, že daná adresa bude mít nulovou životnost a tímto způsobem dojde k jejímu odebrání z rozhraní.

6.2 Nastavení automatické konfigurace klientů

Automatickou konfiguraci klientů je možné v IPv6 provádět dvěma způsoby, pomocí Ohlášení směrovače nebo DHCPv6.

6.2.1 Bezstavová konfigurace – Ohlášení směrovače

Pro bezstavovou konfiguraci můžeme použít buď Zebru nebo RADVD. Já jsem vyzkoušel oba programy a neobjevil jsem žádné problémy.
Oba programy umožňují nakonfigurovat základní parametry Ohlášení směrovače, jako např. prefix, dobu platnosti adresy, dobu preferování adresy, zda bude v sítí použita i stavová konfigurace atd. Skutečnost, že se tyto nastavované parametry opravdu objevily v Ohlášení směrovače, jsem ověřil pomocí programu Ethereal (viz Příloha 2).
Výhodou Zebry je to, že ji můžeme ovládat pomocí příkazového řádku. To nám umožňuje nechat si vypsat všechny dostupné příkazy a jejich parametry a nemusíme je hledat

v manuálových stránkách. Příkazový řádek také umožňuje, aby se nakonfigurované parametry okamžitě projevily v Ohlášení směrovače, aniž bychom museli Zebru restartovat.
Naproti tomu, při použití programu RADVD, je třeba podle manuálových stránek ručně napsat konfigurační soubor a teprve potom spustit RADVD. Pokud potřebujeme provést změnu konfigurace, pak musíme změnit konfigurační soubor, ukončit RADVD a znovu ho spustit.
Výhodou programu RADVD ale může být možnost nakonfigurovat podporu mobility, kterou tento program obsahuje. Ale v současné době, kdy ještě nejsou téměř žádné IPv6 sítě, je podle mého názoru mobilita nepoužitelná. A protože se RADVD momentálně nevyvíjí a u mobility může ještě dojít k nějakým změnám, je opravdu otázkou, jestli bude možné v budoucnu tuto funkci v RADVD použít.
Ukázky konfiguračních souborů pro Zebru a RADVD spolu s popisem základních příkazu lze najít v Příloze 2.

Po zkušenostech, které jsem získal při práci s oběma programy bych doporučil použít Zebru a to hlavně z následujících důvodů:

- poslední verze 0.7.2 programu RADVD je 2 roky stará
- Zebru lze použít i pro konfiguraci jiných vlastností
- konfiguraci Zebry je možno měnit, aniž bychom ji museli restartovat
- podpora pro konfiguraci z příkazového řádku Zebry.

6.2.2 Stavová konfigurace – DHCPv6

Na rozdíl od bezstavové konfigurace, která představuje jednoduchý mechanismus automatické konfigurace, je stavová konfigurace, tzn. DHCPv6, o poznání složitější. Už samotný fakt, že RFC definující DHCPv6 [18] bylo dokončeno teprve v létě roku 2003 napovídá, že implementace nebudou ještě plně funkční. Na druhou stranu ovšem některé implementace byly vyvíjeny ještě před vydáním RFC a byly vytvářeny podle neúplných návrhů.
Na Internetu lze v současné době najít tři projekty, které se pokoušejí o implementaci DHCPv6, ale pouze jeden z nich je možné použít.

DHCPv6 implementation on Linux

Tento projekt lze nalézt na adrese http://sourceforge.net/projects/dhcpv6-linux/. Autoři zřejmě pokládají tento projekt za ukončený, protože poslední aktualizace byla v srpnu roku 2003, kdy byla vydána verze 1.0. U tohoto projektu je velmi citelná absence jakékoliv nápovědy, ať už ve formě webových stránek nebo manuálových stránek. Mezi zdrojovými soubory lze najít pouze soubor s ukázkovou konfigurací a pak soubor, ve kterém je uvedena syntaxe, jakým způsobem se konfigurační soubor píše, ovšem bez vysvětlení jednotlivých příkazu.

Tento projekt se mi nepovedlo téměř vůbec otestovat, protože po spuštění DHCPv6 klient skončil hlášením: „Enter solicitation function“ a pak už neprovedl nic. Pomocí síťového analyzátoru jsem zjistil, že klient nevyslal do sítě ani první zprávu typu Výzva (solicit).

DHCPv6

Projekt DHCPv6 je na adrese http://sourceforge.net/projects/dhcpv6/. Tento projekt byl založen počátkem roku 2003 a je ještě vyvíjen. Momentálně je k dispozici ve verzi 0.10. Tento projekt už obsahuje alespoň základní nápovědu ve formě manuálových stránek, kde lze najít také základní popis konfigurace.

Ačkoliv si klient a server vyměňovaly DHCPv6 zprávy, opět se mi nepovedlo docílit, aby byla klientovi přiřazena IPv6 adresa. Ať už jsem v konfiguračním souboru klienta nastavil, že má žádat o jakoukoli adresu nebo o pevně danou adresu, nikdy ve zprávě zachycené pomocí síťového analyzátoru žádost o IPv6 adresu nebyla.

Zde bych chtěl také poznamenat, že jsem původně používal pro analyzování síťového provozu Ethereal verze 0.9.x, později jsem však zjistil, že nezná všechny DHCPv6 volby a tak jsem nainstaloval Ethereal verze 0.10.3. Při použití novější verze však nebylo možné zachytávat síťový provoz u tohoto projektu. Vždy, když DHCPv6 server vyslal odpověď klientu, přestal Ethereal verze 0.10.3 reagovat. Jelikož Ethereal u jiných testů takovéto problémy neměl, usuzuji, že tento DHCPv6 server zřejmě posílá nějaké nesrozumitelné kombinace voleb, které Ethereal není schopen zpracovat.

Dibbler

Třetím projektem je Dibbler, který byl založen v červenci roku 2003. Lze ho najít na adrese http://klub.com.pl/dhcpv6/, momentálně ve verzi 0.1.1. Ačkoliv je tato implementace vyvíjena ze všech tří nejkratší dobu, má z nich v současné době nejlepší funkčnost a má tedy

nepochybně nejlepší předpoklady, aby ji bylo možné používat v praxi. Další výhodou tohoto projektu je, že je plánováno použití i na jiných platformách. Nyní je Dibbler k dispozici pro Linux a Windows XP, což umožňuje například použít Linux jako DHCPv6 server a Windows XP jako DHCPv6 klienta. Další obrovskou výhodou je dobře propracovaná nápověda ve formátu pdf. Ačkoliv k ní autor napsal, že je velmi strohá, tak oproti zbývajícím dvěma projektům je opravdu vynikající.

Protože je k dispozici pouze raná verze, vyskytují se v ní občas určité problémy. Příkladem může být to, že jsem stáhnul verzi pro Linux v binárním tvaru a DHCPv6 server končil často hlášením: „Neoprávněný přístup do paměti“. Při kompilování došlo taky k problémům, protože podle autora je potřeba mít nainstalován gcc3.3, g++3.3, flex a bison++. Mně se ovšem program pomocí gcc3.3 zkompilovat nepovedlo, ale podařilo se mi to pomocí gcc2.96. Po úspěšném zkompilování už jsem při testování neobjevil žádné problémy.

Pomocí tohoto projektu jsem tedy konečně mohl otestovat fungující DHCPv6. Funkčnost byla na velmi dobré úrovni. Pomocí nápovědy se mi povedlo nakonfigurovat např. i statické přiřazování adresy danému počítači nebo situaci, kdy klient požádá o dvě adresy (jednu statickou a jednu dynamickou) a server mu je podle možností přidělí.

Rovněž jsem vyzkoušel implementaci pro Windows XP a také kombinace: Linux-server s Windows XP-klientem a Windows XP-server s Linux-klientem. Ani při těchto kombinacích nevznikly žádné problémy. Jediným nedostatkem, který jsem zjistil, je, že běžící DHCPv6 server na Windows XP zatěžuje procesor na 100%, takže zřejmě je v programu použita nekonečná smyčka. Ale věřím, že tento malý nedostatek autor do příští verze odstraní.

Jediné, co ještě citelně v této implementaci chybí, je zprostředkovatel (relay-agent). Ale jak jsem zjistil pomocí emailové komunikace s autorem, tak implementace zprostředkovatele je první položkou v seznamu dlouhodobých úkolů.

6.3 Konfigurace směrování

Stejně jako při směrování na IPv4, je možné směrovat na IPv6 staticky nebo dynamicky. Pro statické směrování je možné použít buď linuxový příkaz *route* nebo Zebru. Pro dynamické směrování je pak vždy nezbytné použít Zebru, protože jiná implementace směrování na Linuxu neexistuje.

Já se zde zaměřím na popis zkušeností při používání dynamického směrování. V Zebře jsou pro IPv6 implementovány směrovací protokoly RIPng, OSPFv3 a BGP4+. Já jsem vyzkoušel první dva z nich, tzn. RIPng a OSPFv3.

Celkově bych zhodnotil výsledky směrování pomocí Zebry jako velmi dobré. Jedině při použití RIPng se vyskytly určité problémy. Na některých počítačích RIPng propagoval do sítě vždy pouze jeden prefix s nejnižší adresou, i když bylo k počítači připojeno více síti s více prefixy. Tyto problémy se ale vyskytovaly pouze při použití starší verze Zebry 0.93a. Při použití nejnovější verze 0.94 jsem tyto problémy již neměl.

Dále jsem také otestoval, jak budou tyto protokoly reagovat na rozpojení linky. Měl jsem počítače propojené sítí Ehternet pomocí kříženého kabelu. Když došlo k rozpojení kabelu, sledoval jsem reakci směrovacích protokolů. Cesty k sítím, které byly za rozpojenou linkou, byly samozřejmě po čase odstraněny ze směrovacích tabulek.

Nepříjemné u Zebry byly dost matoucí informace o podpoře oblastí v OSPFv3. Na webových stránkách je napsáno, že podpora oblastí v OSPFv3 ještě není implementována. Ale při zadávání rozhraní v Zebře, na kterých se má používat OSPFv3 je možnost zadat oblast, do které toto rozhraní patří. Nicméně, když jsem vyzkoušel nakonfigurovat několik oblastí, tak propagace cest mezi oblastmi byla velmi podivná. Některé cesty byly do jiných oblastí propagovány, zatímco jiné nikoliv. Takže zatím nedoporučuji oblasti v Zebře používat.

Ukázky konfiguračních souborů pro použití směrovacích protokolů v Zebře jsou uvedeny u konfigurace případové studie v Příloze 5.

6.4 Konfigurace DNS serveru

Situace v implementaci DNS serveru je velice dobrá. Je tomu tak proto, že nejčastěji používaná implementace DNS serveru na Linuxu – BIND obsahuje plnou podporu pro IPv6. Už verze 8.x obsahovala některé části IPv6, jako např. záznamy typu AAAA. Neumožňovala ale přijímat DNS požadavky pomocí protokolu IPv6. Teprve verze 9.x má úplnou podporu IPv6, tzn. umožňuje komunikovat s klienty pomocí IPv6.

Já jsem vyzkoušel verzi 9.2.1 a musím říci, že dojem z této implementace je výborný. Bezproblémově fungovaly jak dopředné, tak i reverzní dotazy. Během testu jsem neobjevil žádný nedostatek, který by mohl bránit, aby byl BIND nasazen v produkčním prostředí.

Jednoduché konfigurační soubory pro program BIND jsou v Příloze 3.

6.5 Konfigurace přechodového mechanismu

V této podkapitole je popsán stav implementace přechodových mechanismů. Protože pro systém Linux neexistují plně funkční implementace, tak jsou zde uvedeny také alternativy, které je možno použít při nasazení přechodových mechanismů v síti.

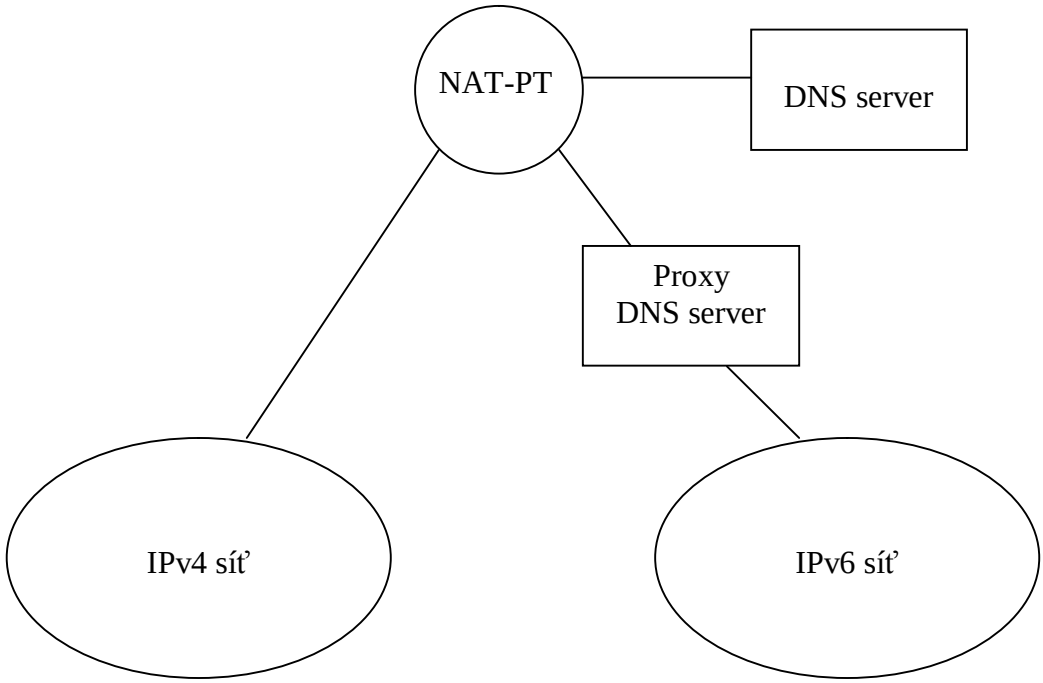
6.5.1 NAT-PT

Nasazení přechodového mechanismu NAT-PT je dosti velkým problémem. Momentálně neexistuje žádná solidní implementace ani pro systém Linux, ani pro systém *BSD. Na těchto systémech sice původně NAT-PT vyvíjen byl, ale vývoj byl ukončen v počátečních fázích implementace, takže není možné je v praxi použít.
Jedinou možností použití NAT-PT je v současné době nákup dostatečně výkonného směrovače firmy CISCO. Ta jako jediná uvádí na svých stránkách, že má implementován NAT-PT a nyní testuje také implementaci NAPT-PT.
Chtěl jsem tedy implementaci NAT-PT vyzkoušet ve školní laboratoři. Bohužel jsme zjistili, že NAT-PT je ale k dispozici jen pro novější řady směrovačů a je k jejímu použití potřeba větší množství paměti flash a RAM, než je k dispozici v laboratoři. Z tohoto důvodu se mi vyzkoušet uvedený přechodový mechanismu nepodařilo.
Pokud je tedy nezbytné, aby spolu komunikovali klienti IPv4 a IPv6 takovým způsobem, aby mohl komunikaci navázat kterýkoli z nich, pak mým jediným doporučením je nákup nového směrovače firmy CISCO. Způsob, jakým lze nakonfigurovat NAT-PT na jejich směrovačích, lze najít na webových stránkách http://www.cisco.com.

Protože NAT-PT není zdaleka jednoduchý mechanismus, o čemž svědčí i neexistující implementace na Linuxu a *BSD, přemýšlel jsem, jakým způsobem je nejlépe navrhnout umístění DNS serverů v lokální síti. Po prozkoumání jiných možností, jako např. použití dvou DNS serverů (jeden pro IPv4 a jeden pro IPv6), jsem došel k závěru, že nejlepší možnost umístění DNS serverů je zobrazena na obrázku 6.1.
Na obrázku 6.1 jsou pro názornost IPv4 síť, IPv6 síť a DNS server nakresleny odděleně, upozorňuji ovšem, že IPv4 klienti mohou být s IPv6 klienty libovolně promíchaní v jedné

fyzické síti. Jediným požadavkem je, aby určité komunikace procházely přes NAT-PT směrovač a to tyto:

- mezi IPv4 klientem a IPv6 klientem
- mezi DNS-proxy serverem a DNS serverem
- mezi IPv4 klientem a DNS serverem



Obrázek 6.1: Doporučené umístění DNS serverů při komunikaci pomocí NAT-PT

Tento návrh je také v souladu s doporučením v [19], kde je uvedeno, že přeložené IPv6 dotazy je vhodné ukládat do paměti cache a přeložené IPv4 záznamy se ukládat do paměti cache nemají.

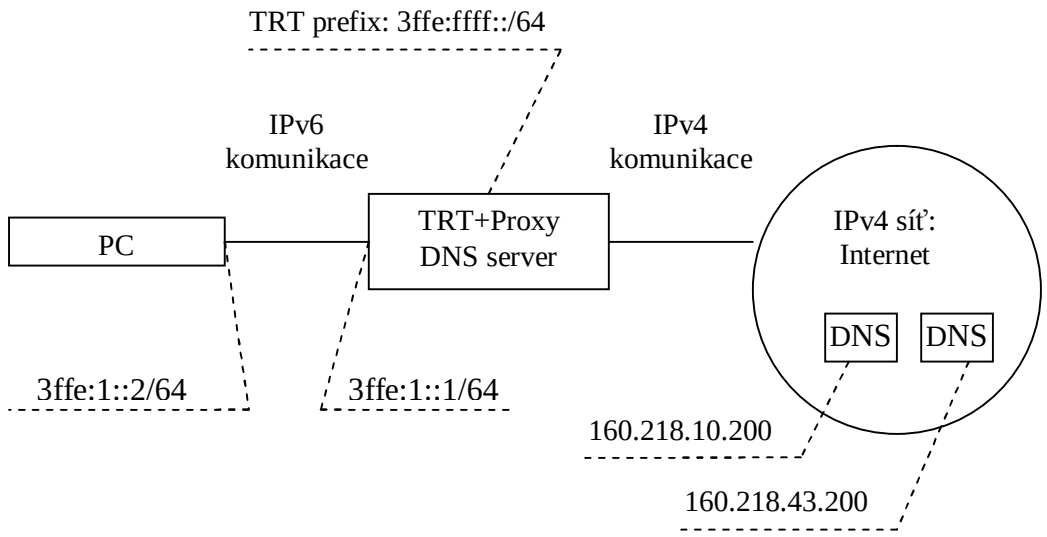
6.5.2 TRT

Na rozdíl od NAT-PT je situace u implementace TRT o poznání lepší, ale není v žádném případě ideální. Protože není v normě obsahující TRT uvedeno, jakým způsobem se mají upravovat DNS zprávy, tak samotná implementace TRT úpravu DNS taky neobsahuje. Je k tomu potřeba použít jiný program.

Implementace mechanismu TRT pro Linux se jmenuje pTRTd a lze ji najít na adrese http://www.litech.org/ptrtd/. Bohužel poslední aktualizace tohoto programu je z roku 2002 a zdá se, že se v současné době nevyvíjí.

Pro úpravu DNS záznamů lze použít program ToTd, který se nachází na adrese http://www.vermicelli.pasta.cs.uit.no/. Tento program se stále vyvíjí a momentálně je ve verzi 1.4.

Já jsem uvedený mechanismus vyzkoušel na sítí, jejíž schéma je na obrázku 6.2.



Obrázek 6.2: Schéma sítě pro testování TRT

S kompilací obou programů jsem neměl vůbec žádné problémy. Při jejich spuštění je nutné zadat IPv6 prefix, který bude používán ve vnitřní sítí k překladům IPv4 adres. Program pTRTd při spuštění vytváří virtuální síťové rozhraní tap0, na které jsou směrovány pakety, které mají být zpracovány tímto programem a právě zde jsem objevil jedinou, ale zato závažnou chybu.

Když jsem totiž program pTRTd ukončil a chtěl jsem ho znovu spustit, tak přestal reagovat a pouze občas vypsal chybové hlášení, že čeká na uvolnění rozhraní tap0. Nebylo pak možné ukončit program pTRTd ani pomocí programu *kill* a bylo nutné restartovat operační systém. Je ovšem zajímavé, že se tato chyba projevila pouze na počítači s operačním systémem Red Hat 8. Na druhém počítači, na kterém byl nainstalován systém Red Hat 9, sice program pTRTd vypsal při opětovném spuštění chybové hlášky, ale bylo ho možné dále používat.
Jinak tato implementace fungovala bez problémů. Vyzkoušel jsem její funkčnost s protokoly HTTP a FTP. Protokol HTTP fungoval bezchybně, u protokolu FTP se při navazování spojení s některými servery objevilo chybové hlášení, že server nezná příkaz EPSV. Ale protože jiné FTP servery fungovaly správně, tak je zřejmě tato chyba způsobena starší implementací FTP serveru, který nepodporuje novější FTP příkazy, které se zavádí u přechodových mechanismů.
Pokud je tedy v síti vyžadován nějaký přechodový mechanismus a postačuje, aby spojení navazovali IPv6 klienti, pak můžeme tuto implementaci použít. Je opravdu škoda, že se pTRTd už nevyvíjí, protože má velmi dobrou funkčnost.

Druhou možností nasazení mechanismu TRT je použití operačního systému *BSD. V tomto systému je TRT implementován pomocí programu FAITH. TRT je označován jako hlavní přechodový mechanismus na platformě *BSD. Jelikož FAITH implementuje také pouze TRT, je pro úpravu DNS zpráv potřeba použít externí program, jako např. výše zmiňovaný ToTd, který byl původně vyvíjen pro použití v systému *BSD.
Já jsem program FAITH vyzkoušel v operačním systému FreeBSD a neměl jsem s jeho používáním žádné problémy, a to ani u protokolu FTP. Jelikož FAITH nepoužívá při IPv4 komunikaci nový příkaz EPSV, ale starší PASV, nevyskytl se ani problém, který jsem měl při použití pTRTd na Linuxu, kdy některé FTP servery vracely chybové hlášení, že neznají příkaz EPSV.

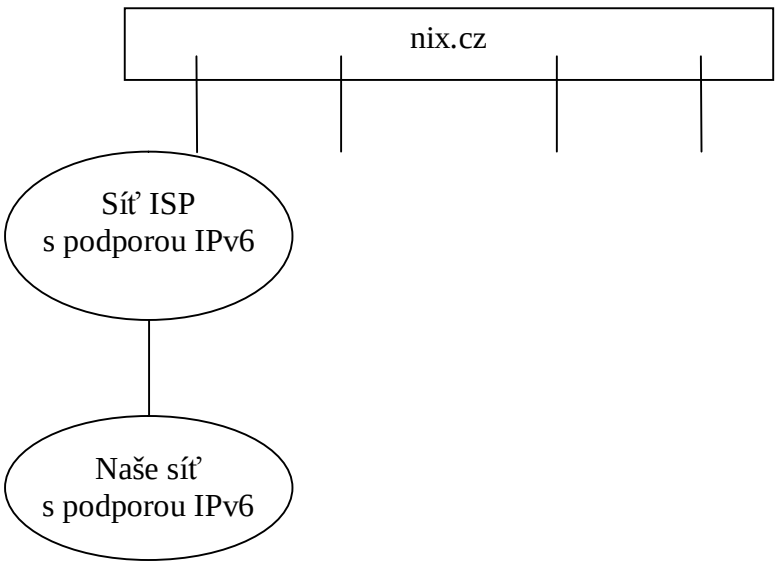
Konfigurační soubor pro program ToTd spolu s názornou ukázkou komunikace probíhající pomocí mechanismu TRT je uveden v Příloze 4.

7. Možnosti připojení k primárnímu poskytovateli

Primárním poskytovatelem (ISP) je poskytoval připojení k Internetu, který je připojen přímo k nix.cz. Možnosti připojení k primárnímu poskytovateli můžeme rozdělit podle toho, zda má náš primární poskytoval implementován protokol IPv6 ve své síti. Pokud má IPv6 implementován, pak můžeme využít nativního připojení, v opačném případě se nevyhneme použití tunelu.

7.1 Nativní připojení k primárnímu poskytovateli

Tato situace je ideální. Nastává v případě, že i primární poskytovatel (ISP) má zprovozněnou IPv6 síť. Tato situace je znázorněna na obrázku 7.1. V tomto případě stačí požádat primárního poskytovatele, aby nám přidělil IPv6 prefix, který budeme používat v naší síti. Pak je pouze potřeba, aby primární poskytovatel směroval pakety s tímto prefixem do naší sítě a my můžeme mít nastavenou implicitní cestu k poskytovateli.



Obrázek 7.1: Nativní připojení k ISP

7.2 Připojení k primárnímu poskytovateli pomocí tunelu

V této kapitole je pod pojmem náš poskytoval myšlen poskytovatel, ke kterému jsme připojeni pomocí protokolu IPv4. Cizí poskytoval je pak libovolný poskytovatel kromě našeho.
Tato možnost připojení se používá v případě, že náš primární poskytovatel nepoužívá ve své síti protokol IPv6. Mohou nastat opět dvě situace:

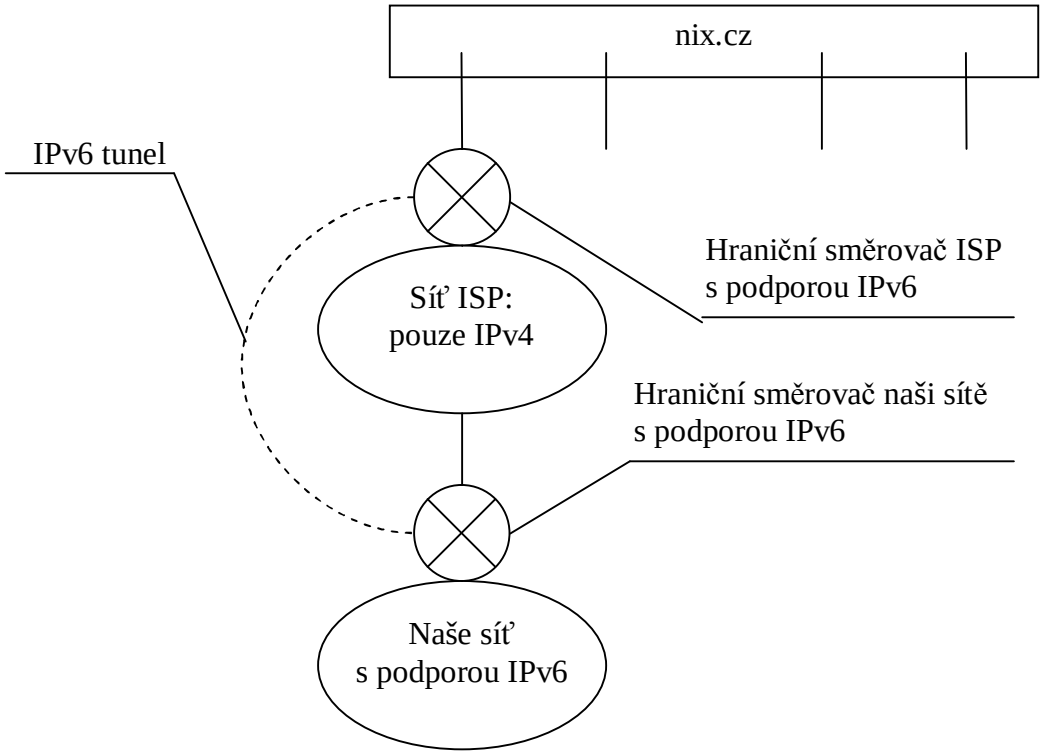
- náš ISP má přidělen IPv6 prefix a má na svém hraničním směrovači směrem k NIXu podporu IPv6
- náš ISP nemá vůbec žádnou podporu IPv6 a ani nemá IPv6 prefix

7.2.1 Připojení tunelem k našemu ISP

Tato situace nastává v případě, kdy náš primární poskytoval ve své síti nemá implementovaný protokol IPv6, ale má alespoň přidělený IPv6 prefix a na svém hraničním směrovači směrem k NIXu má podporu protokolu IPv6. Tato situace je znázorněna na obrázku 7.2.
V tomto případě musíme opět ISP požádat o přidělení IPv6 prefixu. Dále se musíme dohodnout s ISP, aby nám povolil vytvořit tunel IPv6 mezi naším hraničním směrovačem a hraničním směrovačem ISP, který je směrem k NIXu. Potom je potřeba poskytovatele požádat, aby pakety určené pro naši síť směroval do tohoto tunelu. Samozřejmě také pakety z naší sítě, které mají cílovou adresu mimo naši síť, musíme pomocí tunelu směrovat k ISP.

7.2.2 Připojení tunelem k cizímu ISP

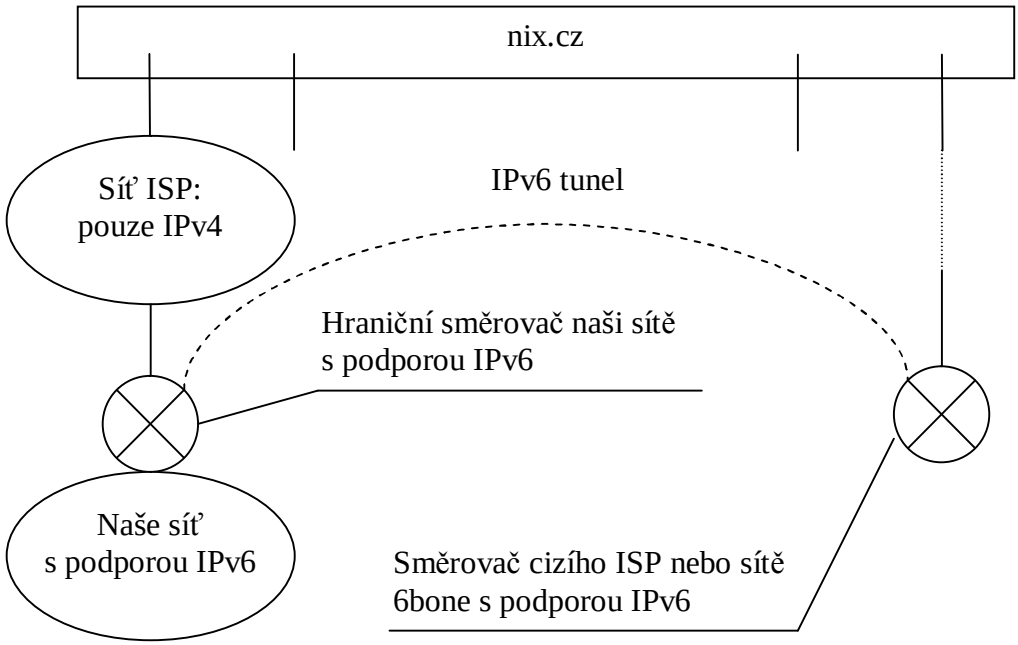
Nejhorší možností je, když náš ISP nemá přidělený IPv6 prefix a tedy nemá ani podporu IPv6 na svém hraničním směrovači směrem k NIXu. V tomto případě máme tři možnosti, z nichž ani jedna není vyhovující:
- požádat cizího ISP s podporou IPv6, aby nám umožnil k němu vytvořit tunel
- připojit se přes testovací síť 6bone
- počkat, až náš ISP zprovozní podporu IPv6



Obrázek 7.2: Připojení k našemu ISP pomocí tunelu

Situace, kdy se tunelem připojujeme k cizímu ISP je na obrázku 7.3. Nevýhody této metody jsou celkem zřejmé. Je totiž velmi malá šance, aby se nám podařilo přesvědčit některého z ISP, aby nám toto řešení umožnil. Mnohem více pravděpodobné je, že nám navrhne, abychom si jako našeho ISP zvolili právě jeho. Další nevýhodou je, že bychom měli přiřazený IPv6 prefix z adres cizího ISP a pokud by někdy v budoucnu začal podporovat IPv6 i náš ISP, pak bychom museli přiřadit nové adresy v celé naší síti.

Druhou možností je připojení pomocí tunelu k síti 6bone. Tato situace je opět znázorněna na obrázku 7.3. 6bone je testovací síť, která je většinou tvořena tunely. Jejím cílem je testovat IPv6 protokol a získávat tak důležité poznatky, které pak budou využity při reálném provozu. Připojení k této síti má podobné nevýhody jako připojení přes cizího ISP. Síť 6bone má totiž přidělen prefix 3ffe::/16 a my můžeme obdržet adresy pouze z tohoto rozsahu. Takže při pozdějším nasazení nativního připojení budeme muset opět znovu přiřazovat nové adresy v naší síti. Informace, jakým způsobem se lze připojit do této sítě, lze nalézt na http://www.6bone.net nebo http://www.xs26.net.



Obrázek 7.3: Připojení k cizímu ISP pomocí tunelu

V případě, že nepotřebujeme IPv6 okamžitě, můžeme počkat, dokud náš ISP nebude mít implementován protokol IPv6 ve své síti. V této situaci se můžeme snažit přesvědčit našeho ISP, aby zprovoznil podporu IPv6 alespoň na svém směrovači směrem k nix.cz co nejdříve. Až bude mít náš ISP podporu IPv6, tak můžeme využít výše uvedených možností nativního připojení IPv6 nebo tunelování k IPv6 směrovači našeho ISP.

7.2.3 Konfigurace tunelů v Linuxu

Vytvoření tunelu v linuxu je možné pomocí příkazu */sbin/ip* s příslušnými parametry. Tento příkaz vytváří virtuální rozhraní sitX, kde X je celé číslo větší než 0. Rozhraní sit0 je obvykle již v systému vytvořeno a jeho použití je rezervováno, takže není v současné době doporučeno rozhraní sit0 používat k námi vytvořeným tunelům.

Po vytvoření rozhraní sitX můžeme s tímto rozhraním pracovat pomocí programu *ifconfig* nebo můžeme přiřadit adresu na toto rozhraní pomocí Zebry.

Já jsem vyzkoušel funkčnost manuálních tunelů a také jsem se zkusil připojit k síti 6bone pomocí serveru http://www.xs26.net, kdy vzdálený konec tunelu byl vytvořen poloautomaticky pomocí tunel serveru. Ani v jednom případě jsem neobjevil žádné problémy a pakety byly v pořádku doručovány.

Přesná syntaxe příkazu pro vytvoření tunelu v Linuxu je uvedena v Příloze 1.

8. Případová studie

Na závěr své diplomové práce jsem sestavil počítačovou síť skládající se z pěti počítačů. Dále jsem provedl její nakonfigurování pomocí programů, které bych použil, pokud by bylo mým úkolem zprovoznit v síti protokol IPv6. Schéma sítě, kterou jsem nakonfiguroval, je na obrázku 8.1. Toto schéma jsem zvolil proto, že jsem chtěl, aby sestavená síť nebyla triviální a aby v zapojené síti byla smyčka. Musel jsem brát ohled také na počet dostupných počítačů a síťových karet ve školní laboratoři.

Pro konfiguraci jsem použil následující nástroje:

- Zebra verze 0.94
- BIND verze 9.2.1
- pTRTd verze 0.7.2
- ToTd verze 1.4.

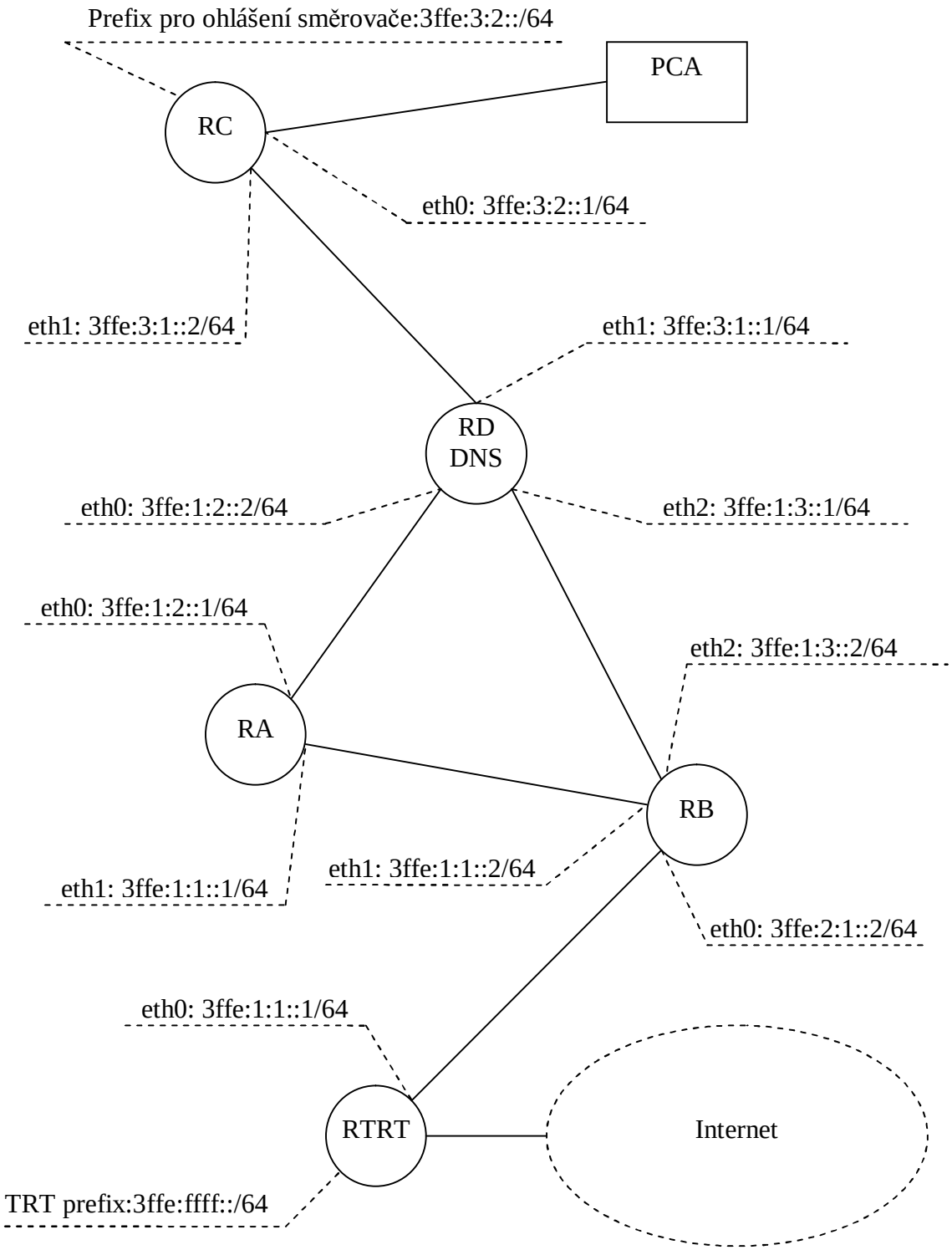
Provedl jsem konfiguraci následujících části protokolu IPv6:

- přiřazení adres na rozhraní pomocí Zebry
- konfigurace směrovacího protokolu OSPFv3 pomocí Zebry, po restartu všech počítačů jsem následně provedl také konfiguraci RIPng taktéž pomocí Zebry
- konfigurace Ohlášení směrovače na směrovači RC pomocí Zebry (prefix odesílaný do sítě byl 3ffe:2:1::/64)
- konfigurace DNS na směrovači RB pomocí BINDu
- konfigurace TRT na směrovači Rtrt (TRT prefix byl 3ffe:ffff::/64)

Konfigurační soubory a výpisy směrovacích tabulek ze směrovače C jsou umístěny v Příloze 5.

Výsledek testu byl pozitivní. Všechny směrovače znaly dostupné sítě, počítače byly dostupné pomocí doménových jmen, PCA podle obdrženého prefixu provedlo autokonfiguraci své adresy. Fungovalo také připojení k internetu, který byl zpřístupněn pomocí mechanismu TRT.

Problémy jsem měl pouze na začátku testu, kdy jsem chtěl definitivně ověřit, zda OSPFv3 v Zebře opravdu nepodporuje oblasti. Výsledkem bylo, že směrovače neznaly cestu k některým sítím, takže jsem ověřil, že oblasti v Zebře plně funkční opravdu nejsou. Poté jsem na všech rozhraních nastavil oblast 0.0.0.0 a vše už fungovalo správně.



Obr. 8.1: Schéma sítě pro případovou studii

9. Závěr

Při psaní své diplomové práce jsem získal mnoho poznatků jak o specifikaci protokolu IPv6, tak o jeho implementaci na různých platformách. Vyzkoušel jsem všechny volně šiřitelé programy pro Linux implementující IPv6, které se mi podařilo najít na Internetu.

Při analýze specifik použití protokolu IPv6 v bezdrátové síti jsem došel k závěru, že se použití protokolu IPv6 v bezdrátové síti téměř neliší od použití v jiných sítích. Proto jsou mé závěry aplikovatelné na libovolnou síť používající směrovače s operačním systémem Linux, ve které se používá protokol IPv4 a ve které je potřeba zprovoznit protokol IPv6.

Můj dojem z implementace IPv6 byl velmi dobrý. Myslím si, že nasazení protokolu IPv6 v sítích je s určitými omezeními už v současné době možné. Jsou ale samozřejmě ještě některé části IPv6, jejichž použití bych zatím nedoporučoval. Jedná se zejména o DHCPv6 a taky o přechodové mechanismy, jejichž implementace na Linuxu buď neexistují nebo nejsou plně funkční. Naštěstí u přechodových mechanismů je možné použít v sítí směrovač na jiné platformě s implementovaným příslušným přechodovým mechanismem.

Já osobně bych doporučil, aby se protokol IPv6 začal v počítačových sítích nasazovat. Jinak bychom se mohli dostat do neřešitelné situace. Programátoři by totiž mohli říci, že nebudou vyvíjet implementace IPv6, protože je stejně skoro nikdo nepoužívá. A zároveň síťoví správci by se obhajovali slovy, že nejsou žádné plně funkční implementace IPv6 a tedy nemá smysl provozovat protokol IPv6 v jejich síti. Myslím, že právě z důvodu malého zájmu o protokol IPv6 přestaly být vyvíjeny některé nadějně vyhlížející implementace, jako např. RADVD a pTRTd. Ale věřím, že když se začne IPv6 protokol používat více, tak se k nedokončeným implementacím programátoři vrátí a začnou je opět vyvíjet.

Kdybych byl síťovým správcem, pak bych ve své síti pro implementaci IPv6 použil následující konfiguraci:

- směrovací protokol RIPng nebo OSPFv3 nakonfigurovaný Zebrou
- DNS server implementovaný BINDem
- bezstavovou konfiguraci implementovanou pomocí Zebry
- přechodový mechanismus TRT při použití programů pTRTd a ToTd (případně nasazení systému *BSD a démona FAITH ve spolupráci s programem ToTd)

Do budoucna bych také plánoval použití DHCPv6 a NAPT-PT.

Seznam použité literatury:

[1] Satrapa, P.: IP verze 6. Neocortex, Praha 2002

[2] Deering, S., Hinden R.: Internet Protocol, Version 6 (IPv6) Specification. RFC 1883, http://www.ietf.org/rfc/rfc1883.txt, 1995

[3] Guidelines For 64-bit Global Identifier (EUI-64). http://standards.ieee.org/regauth/oui/tutorials/EUI64.html

[4] Hinden, R., Deering, S.: IP Version 6 Addressing Architecture. RFC 2373, http://www.ietf.org/rfc/rfc2373.txt, 1998

[5] Conta, A., Deering, S.: Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification. RFC 2463, http://www.ietf.org/rfc/rfc2463.txt, 1998

[6] Narten, T., Nordmark, E., Simpson, W.: Neighbor Discovery for IP Version 6 (IPv6). RFC 2461, http://www.ietf.org/rfc/rfc2461.txt, 1998

[7] Malkin, G., Minnear, R.: RIPng for IPv6. RFC 2080, http://www.ietf.org/rfc/rfc2080.txt, 1997

[8] Hedrick, C.: Routing Information Protocol. RFC 1058, http://www.ietf.org/rfc/rfc1058.txt, 1988

[9] Malkin, G.: RIP version 2 – Carrying Additional Information. RFC 1723, http://www.ietf.org/rfc/rfc1723.txt, 1994

[10] Coltun, R., Ferguson, D., Moy, J.: OSPF for IPv6. RFC 2740, http://www.ietf.org/rfc/rfc2740.txt, 1999

[11] Moy, J.: OSPF Version 2. RFC 2328, http://www.ietf.org/rfc/rfc2328.txt, 1998

[12] Bates, T., Chandra, R., Katz, D., Rekhter, Y.: Multiprotocol Extensions for BGP-4. RFC 2283, http://www.ietf.org/rfc/rfc2283.txt, 1998

[13] Rekhter, Y., Li, T.: A Border Gateway Protocol 4 (BGP-4). RFC 1771, http://www.ietf.org/rfc/rfc1771.txt, 1995

[14] Thomson, S., Huitema, C.: DNS Extensions to support IP version 6. RFC 1886, http://www.ietf.org/rfc/rfc1886.txt, 1995

[15] Crawford, M., Huitema, C.: DNS Extensions to Support IPv6 Address Aggregation and Renumbering. RFC 2874, http://www.ietf.org/rfc/rfc2874.txt, 2000

[16] Thomson, S., Huitema, C., Ksinant, V., Souissi, M.: DNS Extensions to Support IP Version 6. RFC 3596, http://www.ietf.org/rfc/rfc3596.txt, 2003.

[17] Thomson, S., Narten, T.: IPv6 Stateless Address Autoconfiguration. RFC 1971, http://www.ietf.org/rfc/rfc1971.txt, 1996

[18] Droms, R., Bound, J., Volz, B., Lemon, T., Perlina, C., Carney, M., Dynamic Host Configuration Protocol for IPv6 (DHCPv6). RFC 3315, http://www.ietf.org/rfc/rfc3315.txt, 2003

[19] Tsirtsis, G., Srisuresh, P.: Network Address Translation - Protocol Translation (NAT-PT). RFC 2766, http://www.ietf.org/rfc/rfc2766.txt, 2000

[20] Hagino, J., Yamamoto, K.: An IPv6-to-IPv4 Transport Relay Translator. RFC 3142, http://www.ietf.org/rfc/rfc3142.txt, 2001

[21] Park, S., D., Madanapalli, S.: Transmission of IPv6 Packets over WiFi Network. http://www.ietf.org/internet-drafts/draft-syam-ipv6-over-wifi-00.txt, 2004

[22] Crawford, M.: Transmission of IPv6 Packets over Ethernet Network. RFC 2464, http://www.ietf.org/rfc/rfc2464.txt, 1998

Seznam příloh:

1. Konfigurace IPv6 v Linuxu

V této příloze jsou popsány základní konfigurační příkazy pro nastavení protokolu IPv6 v Linuxu. U příkazů, ve kterých je třeba zadat nějakou proměnnou (např. IPv6 adresu) jsou uvedeny také konkrétní příklady.

Přidání IPv6 adresy na rozhraní

/sbin/ifconfig inet6 add IPv6ADRESA/DELKAPREFIXU

například:

/sbin/ifconfig eth0 inet6 add 3ffe:ffff:0:f101::1/64

Odebrání IPv6 adresy z rozhraní

/sbin/ifconfig inet6 del IPv6ADRESA/DELKAPREFIXU

například:

/sbin/ifconfig eth0 inet6 del 3ffe:ffff:0:f101::1/64

Zobrazení aktualních cest

/sbin/route -A inet6

Přidání statické cesty přes bránu

/sbin/route -A inet6 add PREFIX/DELKAPREFIXU gw ADRESABRANY

například:

/sbin/route -A inet6 add 2000::/3 gw 3ffe:ffff:0:f101::1

Odebrání statické cesty přes bránu

/sbin/route -A inet6 del PREFIX/DELKAPREFIXU gw ADRESABRANY

například:

/sbin/route -A inet6 del 2000::/3 gw 3ffe:ffff:0:f101::1

Přidání statické cesty přes rozhraní

/sbin/route -A inet6 add PREFIX/DELKAPREFIXU dev ROZHRANI

například:

/sbin/route -A inet6 add 2000::/3 dev eth0

Odebrání statické cesty přes rozhraní

/sbin/route -A inet6 del PREFIX/DELKAPREFIXU dev ROZHRANI

například:

/sbin/route -A inet6 del 2000::/3 dev eth0

Zobrazení tunelů

/sbin/ip -6 tunnel show

Přidání tunelu

Je třeba vytvořit virtuální rozhraní sitX, kde X je celé číslo větší než 0, rozhraní sit0 je v systému Linux již vytvořeno a má speciální význam, takže ho není možno použít. Tunel je třeba nakonfigurovat na obou koncích tunelu. Vždy je nutné zadat IPv4 adresu lokálního konce tunelu a IPv4 adresu vzdáleného konce tunelu.

/sbin/ip tunnel add sitX mode sit ttl TTL remote VZDALENAIPv4 local LOKALNIPv4

/sbin/ip link set dev sitX up

například:

/sbin/ip tunnel add sit1 mode sit ttl 64 remote 192.168.0.2 local 192.168.0.1

/sbin/ip link set dev sit1 up

Odebrání tunelu

/sbin/ip -6 route del dev sitX

/sbin/ip link set sitX down

/sbin/ip tunnel del sitX

například:

/sbin/ip -6 route del dev sit1

/sbin/ip link set sit1 down

/sbin/ip tunnel del sit1

2. Konfigurace Ohlášení směrovače

V této příloze je ukázka dvou konfiguračních souborů pro Ohlášení směrovače. První z nich je pro konfiguraci Ohlášení směrovače pomocí Zebry, druhý pak pro konfiguraci pomocí programu RADVD. U každého konfiguračního souboru je příslušné Ohlášení směrovače, které jsem zachytil programem Ethereal.

2.1 Konfigurace Ohlášení směrovače pomocí Zebry

Příslušný konfigurační soubor se zachyceným Ohlášením směrovače lze najít na obrázku P2.1. Uvedená konfigurace ukazuje volby, pomocí kterých můžeme nakonfigurovat Ohlášení směrovače v Zebře. Pokud chceme v Zebře používat Ohlášení směrovače, pak musíme pro každé rozhraní zadat příkaz:

ipv6 nd send-ra,

který určuje, že se na daném rozhraní má používat Ohlášení směrovače. Tento příkaz se ale v konfiguračním souboru neobjeví, Zebra ho totiž nahrazuje příkazem:

no ipv6 nd suppress-ra,

který má stejný význam. Dále je potřeba pro každý prefix, který se má pro toto rozhraní použít v Ohlášení směrovače, zapsat příkaz:

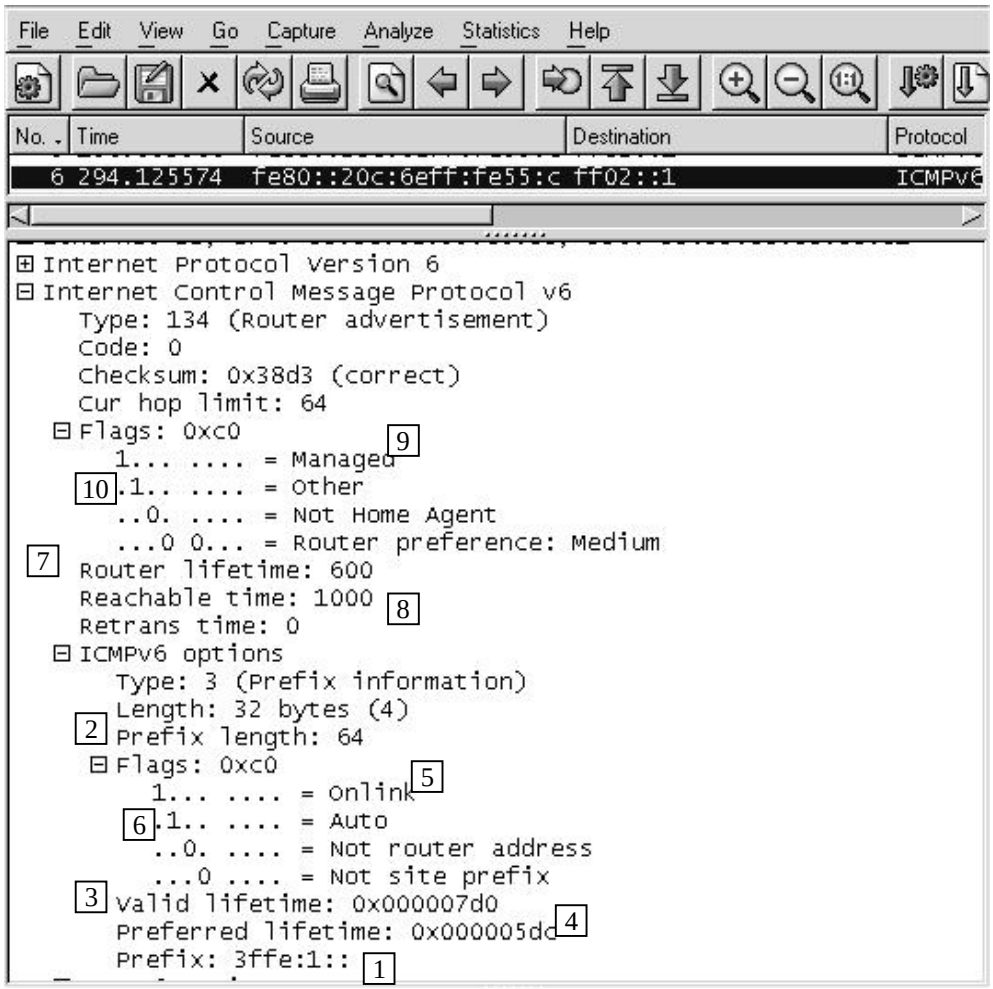
ipv6 nd prefix-advertisement PREFIX

Další parametry tohoto příkazu jsou nepovinné a pokud je nezadáme, tak je Zebra doplní výchozími hodnotami.

V konfiguračním souboru na obrázku P2.1 jsou uvedeny i další volitelné příkazy. Význam těchto příkazů lze najít v nápovědě k programu Zebra. Na obrázku P2.1 je také ukázáno, že se nakonfigurované volby opravdu objevily v Ohlášení směrovače.

zebra.conf:

```
password zebra
enable password zebra
interface eth0
 ipv6 address 3ffe:1::1/64
 no ipv6 nd suppress-ra
 ipv6 nd ra-lifetime 600 [7]
 ipv6 nd reachable-time 1000 [8]
 ipv6 nd managed-config-flag [9]
 ipv6 nd other-config-flag [10]
 ipv6 nd prefix-advertisement 3ffe:1::/64 2000 1500 onlink [1] [2] [3] [4] [5]
autoconfig [6]
```



Obrázek P2.1: Ohlášení směrovače pomocí Zebry

2.2 Konfigurace Ohlášení směrovače pomocí RADVD:

Konfigurační soubor programu RADVD s několika pokročilejšími volbami je na obrázku P2.2. Pro použití Ohlášení směrovače na daném rozhraní je třeba v konfiguračním příkazu zadat pro toto rozhraní příkaz:

AdvSendAdvert on;

Pak je třeba zadat ještě seznam prefixů, které se budou používat v Ohlášení směrovače pro toto rozhraní. Prefix se zadává příkazem:

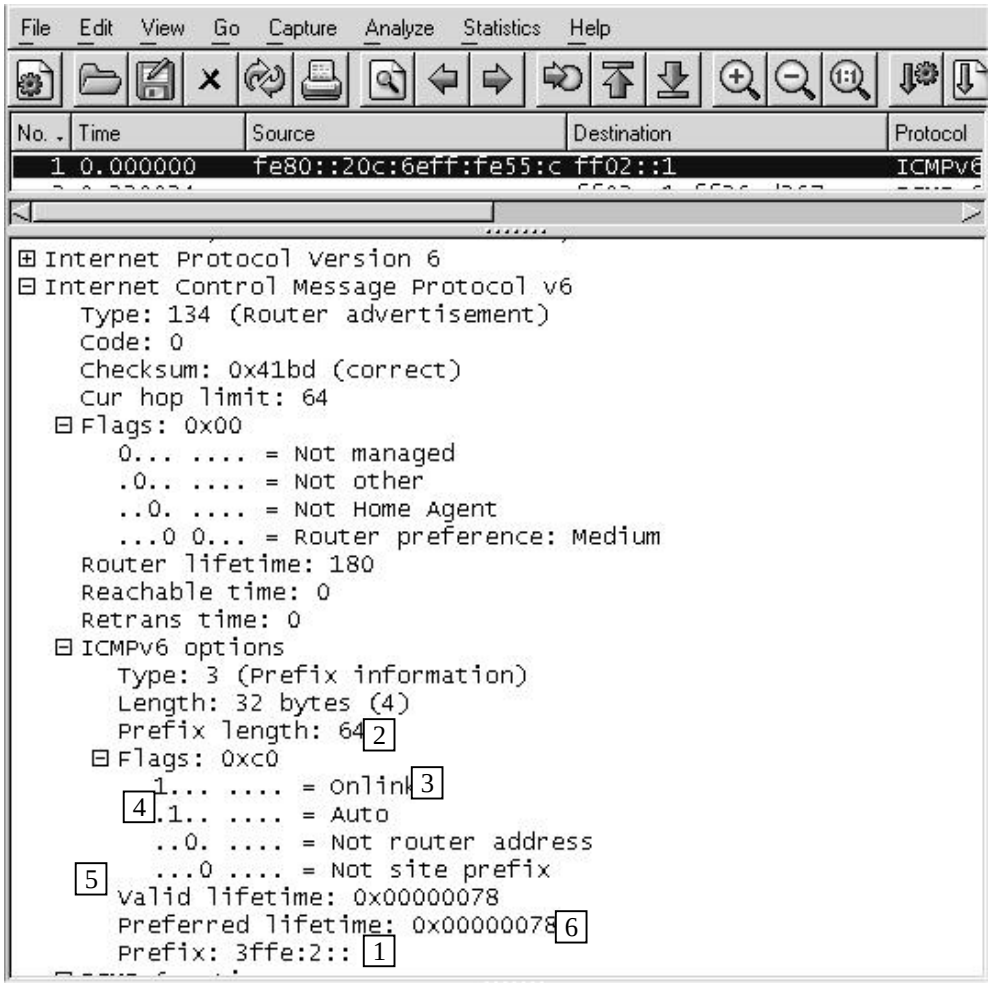
prefix PREFIX/DÉLKAPREFIXU;

Ke každému prefixu je možno zadat ještě další parametry, jako např. dobu platnosti prefixu nebo dobu preferování prefixu. Další volby lze najít v manuálových stránkách programu RADVD.

Na obrázku P2.2 je ukázán zachycený paket s Ohlášením směrovače a opět je na tomto obrázku znázorněno, že se tyto volby objevily ve skutečném Ohlášení směrovače.

radvd.conf:

```
interface eth0 {
    AdvSendAdvert on;
    MaxRtrAdvInterval 60;
                 1     2
    prefix 3ffe:2::/64 {
      AdvOnLink on; 3
      AdvAutonomous on; 4
      AdvValidLifetime 120; 5
      AdvPreferredLifetime 120; 6
    };
};
```



Obrázek P2.2: Ohlášení směrovače pomocí RADVD

3. Konfigurace DNS

Konfiguraci DNS jsem provedl pomocí programu BIND. Tato konfigurace je velice jednoduchá. Skládá se ze tří souborů:

- named.conf
- pokus.org.conf
- 3ffe000100000000.ip6.arpa.conf

Hlavním konfiguračním souborem je named.conf, který se odkazuje na zbývající dva soubory, které obsahují záznamy pro DNS. V souboru pokus.org.conf je definice pro doménu pokus.org a v souboru 3ffe000100000000.ip6.arpa.conf jsou pak uvedeny záznamy pro příslušnou reverzní doménu.

Pomocí příkazů:

ping6 pc1.pokus.org

a

ping6 pc2.pokus.org

jsem otestoval funkčnost konfigurace programu BIND. Ukázka této komunikace zachycené programem Ethereal je na obrázku P3.1.

named.conf:

```
options {
     directory "/etc/bind";
     listen-on-v6 port 53 {any;};
};
zone "pokus.org" {
     type master;
     file "pokus.org.conf";
};
zone "0.0.0.0.0.0.0.0.1.0.0.0.e.f.f.3.ip6.arpa" {
     type master;
     file "3ffe000100000000.ip6.arpa.conf";
};
```

pokus.org.conf:

```
$TTL 86400
@    IN   SOA  pokus.org.  root.pokus.org. (
               9 ; serial
               28800 ; refresh
               7200 ; retry
               604800 ; expire
               86400 ; ttl
               )

     IN   NS   ns.pokus.org.
ns.pokus.org.        IN   AAAA      3ffe:1::1
pc1.pokus.org.       IN   AAAA      3ffe:1::1
pc2.pokus.org.       IN   AAAA      3ffe:1::2
```

3ffe000100000000.ip6.arpa.conf:

```
$TTL 86400
@    IN   SOA  pokus.org. root.pokus.org. (
               7 ; serial
               28800 ; refresh
               7200 ; retry
               604800 ; expire
               86400 ; ttl
               )

     IN   NS   pc1.pokus.org.
$ORIGIN 0.0.0.0.0.0.0.0.1.0.0.0.e.f.f.3.ip6.arpa.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0      IN   PTR  pc1.pokus.org.
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0      IN   PTR  pc2.pokus.org.
```

Obrázek P3.1: Komunikace s DNS serverem

4. TRT

Konfigurační soubor pro mechanismus TRT je pouze jeden, a to pro program ToTd. V tomto souboru stačí uvést dva příkazy. Prvním z nich je:

forwarder IPADRESA

kde IPADRESA může být buď adresa IPv4 nebo adresa IPv6 DNS serveru, na který má ToTd předávat DNS dotazy. Těchto DNS serverů může být samozřejmě uvedeno víc. Druhým příkazem je:

prefix IPv6PREFIX

který určuje IPv6 prefix, který se v sítí používá pro potřeby TRT mechanismu (v našem příkladě je to prefix 3ffe:ffff::).

Program pTRTd nemá konfigurační soubor, parametry se programu předávají zapsáním do příkazového řádku. Program pTRTd se tedy spouští příkazem:

ptrtd –p IPv6PREFIX –l DELKAPREFIXU

a tedy v našem případě je to příkaz:

ptrtd –p 3ffe:ffff:: -l 64

totd.conf:

```
forwarder 160.218.10.200
forwarder 160.218.43.200
prefix 3ffe:ffff::
```

Názorná ukázka, jakým způsobem probíhá zahájení komunikace pomocí mechanismu TRT, je na obrázku P4.1. Při pohledu na tento obrázek může být nápadné to, že některé pakety se v něm vyskytují dvakrát za sebou. Toto není chyba komunikace pomocí IPv6, ale chyba v Etherealu, který při volbě zachytávání paketů na všech rozhraních počítače zobrazuje některé pakety dvakrát.

Na obrázku P4.1 je tedy vidět, jakým způsobem TRT pracuje. Já zde popíši ty pakety, které jsou pro fungování mechanismu TRT podstatné:

- paket 58: klient posílá na proxy-DNS server dotaz, ve kterém se ptá, jaká je adresa IPv6 serveru www.atlas.cz
- paket 59: proxy-DNS předává původní dotaz DNS serveru

- paket 60: DNS server vrací odpověď, ve které uvádí, že www.atlas.cz nemá adresu IPv6
- paket 61: proxy-DNS vytváří nový dotaz a ptá se DNS serveru na adresu IPv4 serveru www.atlas.cz
- paket 62: DNS server posílá informaci, že www.atlas.cz má adresu IPv4 212.47.13.117
- paket 63: proxy-DNS změní adresu IPv4 na IPv6 pomocí prefixu (v našem případě 3ffe:ffff::) a pošle klientovi informaci, že www.atlas.cz má adresu IPv6 3ffe:ffff::d42f:d75)
- paket 68: klient zahajuje TCP spojení pomocí IPv6 tak, že pošle paket na adresu 3ffe:ffff::d42f:d75
- paket 70: TRT zachytí tento paket, z cílové adresy IPv6 zjistí cílovou adresu IPv4 a zahajuje druhé TCP spojení, v tomto případě na IPv4
- paket 90: na TRT je doručena odpověď od www.atlas.cz
- paket 92: odpověď od www.atlas.cz je předána klientu komunikujícímu pomocí IPv6

-

Obrázek P4.1: Ukázka komunikace pomocí TRT

5. Konfigurace případové studie

V této kapitole jsou uvedeny konfigurační soubory použité u případové studie, která je popsána v kapitole 8. Nejdříve jsou uvedeny konfigurační soubory pro Zebru, dále pak konfigurační soubory pro DNS a TRT a nakonec jsou v této příloze uvedeny výpisy směrovacích tabulek ze směrovače RC. Konfigurační soubory pro DNS a TRT už znovu popisovat nebudu, protože tomu jsou věnovány přílohy 3 a 4.

Pro každý směrovač jsou pro konfiguraci Zebry uvedeny tyto tři soubory:

- zebra.conf – pro přiřazení IPv6 adres na rozhraní
- ripngd.conf – pro konfiguraci směrovacího protokolu RIPng
- ospf6d.conf – pro konfiguraci směrovacího protokolu OSPFv3

Přiřazení adresy na rozhraní pomocí Zebry je jednoduché. Pro každé rozhraní, na které chceme přiřadit IPv6 adresu, zadáme příkaz:

ipv6 address IPv6ADRESA/DÉLKAPREFIXU

Pro konfiguraci protokolu RIPng je třeba v konfiguračním módu zadat příkaz:

router ripng

a pak pro každé rozhraní, na kterém se bude RIPng používat, potřeba zadat příkaz:

network ROZHRANÍ

a nakonec je nutné zadat příkaz:

redistribute connected

aby byly pomocí protokolu RIPng rozšiřovány informace o přímo připojených sítích.

Pro konfiguraci OSPFv3 musíme v konfiguračním módu zadat příkaz:

router ospf6

Následně je třeba zadat ID směrovače, které se obvykle odvozuje z jeho IPv6 adresy, ve tvaru:

router-id IDSMĚROVAČE

Pro každé rozhraní, na kterém se bude OSPFv3 používat, je třeba zadat příkaz:

interface ROZHRANÍ area OBLAST

Jelikož podpora oblastí pro OSPFv3 není v Zebře plně funkční, doporučuji jako OBLAST zadat 0.0.0.0. No a nakonec je opět třeba zadat příkaz:

redistribute connected

5.1 Konfigurační soubory směrovače RA

zebra.conf:

```
hostname RA
password zebra
enable password zebra
interface eth0
 ipv6 address 3ffe:1:2::1/64
 ipv6 nd suppress-ra
interface eth1
 ipv6 address 3ffe:1:1::1/64
 ipv6 nd suppress-ra
```

ripngd.conf:

```
password zebra
enable password zebra
router ripng
 network eth0
 network eth1
 redistribute connected
```

ospf6d.conf:

```
password zebra
enable password zebra
router ospf6
 router-id 158.196.135.16
 redistribute connected
 interface eth0 area 0.0.0.0
 interface eth1 area 0.0.0.0
```

5.2 Konfigurační soubory směrovače RB

zebra.conf:

```
hostname RB
password zebra
enable password zebra
interface eth0
 ipv6 address 3ffe:2:1::2/64
 ipv6 nd suppress-ra
interface eth1
 ipv6 address 3ffe:1:1::2/64
 ipv6 nd suppress-ra
interface eth2
 ipv6 address 3ffe:1:3::2/64
 ipv6 nd suppress-ra
```

ripngd.conf:

```
password zebra
enable password zebra
router ripng
 network eth0
 network eth1
 network eth2
 redistribute connected
```

ospf6d.conf:

```
password zebra
enable password zebra
router ospf6
 router-id 158.196.135.17
 redistribute connected
 interface eth0 area 0.0.0.0
 interface eth1 area 0.0.0.0
 interface eth2 area 0.0.0.0
```

5.3 Konfigurační soubory směrovače RC

zebra.conf:

```
hostname RC
password zebra
enable password zebra
interface eth0
 ipv6 address 3ffe:3:2::1/64
 no ipv6 nd suppress-ra
 ipv6 nd prefix-advertisement 3ffe:3:2::/64 2592000 604800
onlink autoconfig
interface eth1
 ipv6 address 3ffe:3:1::2/64
 ipv6 nd suppress-ra
```

ripngd.conf:

```
password zebra
enable password zebra
router ripng
 network eth0
 network eth1
 redistribute connected
```

ospf6d.conf:

```
password zebra
enable password zebra
router ospf6
 router-id 158.196.135.18
 redistribute connected
 interface eth0 area 0.0.0.0
 interface eth1 area 0.0.0.0
```

5.4 Konfigurační soubory směrovače RD

zebra.conf:

```
hostname RD
password zebra
enable password zebra
interface eth0
 ipv6 address 3ffe:1:2::2/64
 ipv6 nd suppress-ra
interface eth1
 ipv6 address 3ffe:3:1::1/64
 ipv6 nd suppress-ra
interface eth2
 ipv6 address 3ffe:1:3::1/64
 ipv6 nd suppress-ra
```

ripngd.conf:

```
password zebra
enable password zebra
router ripng
 network eth0
 network eth1
 network eth2
 redistribute connected
```

ospf6d.conf:

```
password zebra
enable password zebra
router ospf6
 router-id 158.196.135.19
 redistribute connected
 interface eth0 area 0.0.0.0
 interface eth2 area 0.0.0.0
 interface eth1 area 0.0.0.0
```

5.5 Konfigurační soubory směrovače Rtrt

zebra.conf:

```
hostname Rtrt
password zebra
enable password zebra
interface eth0
 ipv6 address 3ffe:2:1::1/64
 ipv6 nd suppress-ra
```

ripngd.conf:

```
password zebra
enable password zebra
router ripng
 network eth0
 redistribute connected
 redistribute kernel
```

ospf6d.conf:

```
password zebra
enable password zebra
router ospf6
 router-id 158.196.135.50
 redistribute connected
 redistribute kernel
 interface eth0 area 0.0.0.0
```

5.6 Konfigurační soubory TRT

totd.conf:

```
forwarder 158.196.149.9
forwarder 158.196.158.11
prefix 3ffe:ffff::
```

5.7 Konfigurační soubory DNS serveru

named.conf:

```
options {
      directory "/var/named";

      listen-on-v6 port 53 {any;};
};
zone "pokus.org.conf" {
       type master;
      file "pokus.org";
};
zone "e.f.f.3.ip6.arpa" {
      type master;
      file "3ffe.arpa.conf";
};
```

pokus.org.conf:

```
$TTL 86400
@     IN    SOA   pokus.org.  root.pokus.org. (
                  9 ; serial
                  28800 ; refresh
                  7200 ; retry
                  604800 ; expire
                  86400 ; ttl
                  )

    IN     NS    rc.pokus.org.

rca.pokus.org.  IN    AAAA 3ffe:1:2::1
rcb.pokus.org.  IN    AAAA 3ffe:1:3::2
rcc.pokus.org.  IN    AAAA 3ffe:3:1::2
rcd.pokus.org.  IN    AAAA 3ffe:1:3::1
pca.pokus.org.  IN    AAAA 3ffe:3:2::2C:6eff:fe55:c600
```

3ffe.arpa.conf:

```
$TTL 86400
@     IN    SOA   pokus.org. root.pokus.org. (
                  7 ; serial
                  28800 ; refresh
                  7200 ; retry
                  604800 ; expire
                  86400 ; ttl
                  )
      IN    NS    rcc.pokus.org.

$ORIGIN e.f.f.3.ip6.arpa.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.2.0.0.0.1.0.0.0    IN
      PTR   rca.pokus.org.
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.3.0.0.0.1.0.0.0    IN
      PTR   rcb.pokus.org.
2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.1.0.0.0.3.0.0.0    IN
      PTR   rcc.pokus.org.
1.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.3.0.0.0.1.0.0.0    IN
      PTR   rcd.pokus.org.
0.0.6.c.5.5.e.f.f.f.e.6.c.2.0.0.0.0.0.0.3.0.0.0.1.0.0.0    IN
      PTR   pca.pokus.org.
```

5.8 Výpis směrovacích tabulek ze směrovače RC:

Protokol RIPng:

```
RC# show ipv6 route
Codes: K - kernel route, C - connected, S - static, R - RIPng, O -
OSPFv3, B - BGP, * - FIB route.
C>* ::1/128 is directly connected, lo
R>* 3ffe:1:1::/64 [120/0] via fe80::200:c0ff:fea6:c868, eth1, 00:03:40
R>* 3ffe:1:2::/64 [120/0] via fe80::200:c0ff:fea6:c868, eth1, 00:03:40
R>* 3ffe:1:3::/64 [120/0] via fe80::200:c0ff:fea6:c868, eth1, 00:03:40
R>* 3ffe:2:1::/64 [120/0] via fe80::200:c0ff:fea6:c868, eth1, 00:03:40
K * 3ffe:3:1::/64 is directly connected, eth1
C>* 3ffe:3:1::/64 is directly connected, eth1
K * 3ffe:3:2::/64 is directly connected, eth0
C>* 3ffe:3:2::/64 is directly connected, eth0
R>* 3ffe:ffff::/64 [120/0] via fe80::200:c0ff:fea6:c868, eth1,
00:03:40
C * fe80::/64 is directly connected, eth1
K * fe80::/64 is directly connected, eth1
C>* fe80::/64 is directly connected, eth0
K>* ff00::/8 is directly connected, eth1
```

Protokol OSPFv3:

```
RC# show ipv6 route
Codes: K - kernel route, C - connected, S - static, R - RIPng, O -
OSPFv3, B - BGP, * - FIB route.
C>* ::1/128 is directly connected, lo
O>* 3ffe:1:1::/64 [110/0] via fe80::200:c0ff:fea6:c868, eth1, 00:19:43
O>* 3ffe:1:2::/64 [110/0] via fe80::200:c0ff:fea6:c868, eth1, 00:20:09
O>* 3ffe:1:3::/64 [110/0] via fe80::200:c0ff:fea6:c868, eth1, 00:20:09
O>* 3ffe:2:1::/64 [110/0] via fe80::200:c0ff:fea6:c868, eth1, 00:20:09
0   3ffe:3:1::/64 [110/0] is directly connected, eth1, 00:20:14
K * 3ffe:3:1::/64 is directly connected, eth1
C>* 3ffe:3:1::/64 is directly connected, eth1
O   3ffe:3:2::/64 [110/0] is directly connected, eth0, 00:20:14
K * 3ffe:3:2::/64 is directly connected, eth0
C>* 3ffe:3:2::/64 is directly connected, eth0
O>* 3ffe:ffff::/64 [110/0] via fe80::200:c0ff:fea6:c868, eth1,
00:20:09
C * fe80::/64 is directly connected, eth1
K * fe80::/64 is directly connected, eth1
C>* fe80::/64 is directly connected, eth0
K>* ff00::/8 is directly connected, eth1
```