Upozornění

Toto je pracovní verze knihy, která není určena k distribuci. Jejím účelem je doladit text a vychytat případné mouchy. Velmi uvítám vaše upozornění na chyby a náměty na případné úpravy, změny či doplňky. Pište, prosím, na adresu Pavel.Satrapa@tul.cz do 4. listopadu 2011. Nedojde-li k něčemu nečekanému, finální podoba knihy vyjde v prosinci 2011, abyste ji měli pod stromeček. Vydá ji opět CZ.NIC s otevřenou licencí, stejně jako v případě předchozího vydání.

Pavel Satrapa

Třetí, aktualizované vydání

Vydal
CZ.NIC, z. s. p. o.
Americká 23, 120 00 Praha 2
http://www.nic.cz/

Vydání této publikace podpořil
CESNET, z. s. p. o.
Zikova 4, 160 00 Praha 6
http://www.cesnet.cz/

ISBN 978-80-904248-4-5

Pavel Satrapa

Internetový protokol IPv6

IPv6

Edice CZ.NIC

http://knihy.nic.cz/

Pokud není výslovně uvedeno jinak, jsou všechny domény a IP adresy v této knize smyšlené. Jakákoli podobnost se skutečnými IP adresami či doménovými jmény je čistě náhodná.

Předmluva vydavatele

Vážení čtenáři,

tady bude předmluva vydavatele...

Ondřej Filip

Praha, listopad 2011

Předmluva ke třetímu vydání

Síťové protokoly se dělí na dvě kategorie: ty, které byly za standard oficiálně prohlášeny, a ty, které se jím doopravdy staly. IP, nosný protokol Internetu, nepochybně patří do druhé skupiny. Jednoznačně ovládl pole a představuje dnes standardní cestu ke vzájemné komunikaci počítačů.

Své popularitě však vděčí i za určité problémy, které se objevily při masovém nasazení. Tím nejpalčivějším je nedostatek adres, který pociťují především noví uživatelé (staří mazáci mají nahrabáno). Proto se od první poloviny devadesátých let vyvíjí jeho nástupce – IP verze 6.

Nový protokol si klade za cíl nejen zvětšit adresní prostor, ale i přidat některé pokročilé vlastnosti, které posunou možnosti Internetu zase o kus dál. Ovšem nelze zamlčovat, že se prosazuje pomalu a bolestně. Firmám se příliš nechce investovat do vývoje, protože návratnost je nejistá, zatímco na současném IPv4 se dá vydělat hned. Takže všichni chodí opatrně kolem rybníka, trousí optimistické fráze, tu a tam se někdo osmělí, ale do vody se stále příliš nehrnou.

Cílem této knihy je popsat, jak rybník vypadá a co se v něm děje. Snažil jsem se velmi zevrubně vysvětlit principy a mechanismy, na kterých IPv6 stojí. Najdete zde formát datagramu, adresování, automatickou konfiguraci, směrování i pokročilé prvky, jako je IPsec či podpora mobilních zařízení. Nemalý prostor jsem věnoval také metodám, které mají umožnit hladký přechod od staré verze protokolu k nové a které tak nepěkně drhnou.

Tyto teoretické pasáže jsou shromážděny v první části knihy. Druhá se věnuje praxi – jak nakonfigurovat IPv6 ve vybraných operačních systémech či směrovačích a jak používat některé programy s jeho podporou.

Přestože byl základ IPv6 položen v polovině 90. let, protokol se stále vyvíjí. Přesněji řečeno jeho jádro je stabilní, ale váže se k němu celá řada doprovodných mechanismů vytvářejících košatý strom vzájemně souvisejících protokolů, na němž stále raší nové listy a nahrazují své předchůdce. V posledních letech už se spíše pilují detaily, odstraňují objevené problémy a upřesňují nejasná místa, nicméně občas je přijata i zásadnější specifikace, jako například NAT64 z jara letošního roku.

Nesnažil jsem se popsat vše do posledního detailu. U složitějších protokolů (jako je OSPFv3) by takový přístup vydal na samostatnou knihu. V těchto případech jsem dal přednost popisu základních prvků a principů, na kterých daný mechanismus stojí, abyste pochopili jeho funkci. Zajímají-li vás detaily, jako jsou přesné formáty zpráv, podmínky pro jejich odesílání, přesná definice chování účastníků komunikace a podobně, budete se muset obrátit na RFC a další dokumenty.

Přesto si troufám tvrdit, že zejména u komplikovanějších témat, jako je IPsec, mobilita či některé směrovací protokoly, jde kniha do výrazně větší hloubky, než je v kraji zvykem. Dostupné publikace o IPv6 tyto oblasti zpravidla jen naznačují. Nevím o tom, že by byl (a to v celosvětovém měřítku) k dispozici text s takto uceleným a aktuálním popisem problematiky IPv6.

První vydání této knihy vyšlo v roce 2002 u společnosti Neocortex, s. r. o., druhé vydal o šest let později CZ.NIC jako první publikaci své nově zahájené *Edice CZ.NIC*. Nyní, bezmála deset let od prvního vydání, vychází vydání třetí, jež je opět aktualizováno podle současného stavu světa IPv6. Změny proti jeho předchůdci nejsou dramatické, ve většině kapitol se jedná spíše o údržbu.

Nicméně několik významných událostí a změn se do obsahu promítlo. Nejvýznamnější je nepochybně vyčerpání IPv4 adres, k němuž po létech více či méně úspěšných prognóz skutečně došlo a od letošního února zeje jejich centrální zásoba prázdnotou. Další významnou novinkou byla – opět letošní – standardizace překladače NAT64, který zacelil Macochu v přechodových mechanismech vzniklou likvidací NAT-PT. Třetí změna, jež stojí za zmínku, je zařazení softwarového směrovače BIRD, který se má v posledních letech čile k světu. Společně s desítkami drobných aktualizací a doplňků navýšily tyto úpravy počet stránek přibližně o padesát.

Text předpokládá, že čtenář má jisté základní znalosti o IPv4 a fungování Internetu. Pravděpodobně byste se obešli i bez nich, ale pochopení některých pasáží by se tak o poznání ztížilo.

Děkuji všem, kteří přispěli ke vzniku tohoto textu. V první řadě své ženě Marcele a celé rodině, která mi jako vždy poskytla zázemí pro práci a měla se mnou trpělivost. Dále si speciální poděkování zaslouží kolegové, jejichž poznámky a rady pomohly dovést text do konečné podoby, zejména Petr Adamec, Luboš Pavlíček, Stanislav Petr a Emanuel Petr.

Pavel Satrapa

Liberec, říjen 2011

Obsah

1 Úvod

Internet Protocol verze 6 (IPv6) se má stát následníkem nosného protokolu současného Internetu, kterým je Internet Protocol verze 4 (IPv4). Ve starší literatuře bývá označován též jako *IP Next Generation (IPng)*.

1.1 Vlastnosti a vývoj

cíle a vlastnosti

Jeho kořeny sahají do začátků devadesátých let, kdy začalo být zřejmé, že se adresní prostor dostupný v rámci IPv4 rychle tenčí. Tehdy vypracované studie ukazovaly, že s perspektivou přibližně deseti let dojde k jeho úplnému vyčerpání. Jelikož na řešení problému bylo k dispozici poměrně dost času, rozhodlo se IETF navrhnout zásadnější změnu, která by kromě rozšířeného adresního prostoru přinesla i další nové vlastnosti.

U kolébky IPv6 proto stály následující požadavky:

- rozsáhlý adresní prostor, který vystačí pokud možno navždy
- tři druhy adres: individuální (unicast), skupinové (multicast) a výběrové (anycast)
- jednotné adresní schéma pro Internet i vnitřní sítě
- hierarchické směrování v souladu s hierarchickou adresací
- zvýšení bezpečnosti (zahrnout do IPv6 mechanismy pro šifrování, autentizaci a sledování cesty k odesilateli)
- podpora pro služby se zajištěnou kvalitou
- optimalizace pro vysokorychlostní směrování
- automatická konfigurace (pokud možno plug and play)
- podpora mobility (přenosné počítače apod.)
- hladký a plynulý přechod z IPv4 na IPv6

vývoj

Jak je vidět, cíle nebyly skromné. Chopili se jich především Steven Deering a Robert Hinden, kteří mají největší podíl na vzniku nového protokolu. Jejich snaha vyústila koncem roku 1995 ve vydání sady RFC definujících základ IPv6. Jedná se o RFC 1883: *Internet Protocol, Version 6 (IPv6) Specification* a jeho příbuzné.

Oficiální specifikace protokolu tedy byla na stole a mohlo se začít s implementováním a uváděním do života. Jenže nezačalo. IPv6 bylo příliš ožehavou a nejistou půdou, zatímco na poli IPv4 čekaly zisky *teď hned*. Většina

firem se proto věnovala raději snaze o rozvoj IPv4, než aby se angažovala v IPv6, protože návratnost investic byla v prvním případě rychlejší.

Mimo jiné se podařilo otupit ostří největšího nože na krku IPv4 – nedostatku adres. Začalo se používat beztřídní adresování CIDR, zpřísnila se kritéria pro přidělování síťových adres a byly zavedeny mechanismy pro překlad adres (NAT, viz níže).

Tím IPv6 přišlo o svou hlavní hnací sílu a jeho nasazení se začalo odkládat. Aby se dokázalo prosadit do praxe, musí nabídnout nějaké zásadní výhody. Ovšem všechny jeho lákavé vlastnosti byly mezitím implementovány i pro IPv4. Pravda, ne vždy tak elegantně a zdaleka ne každá implementace je podporuje, ale principiálně jsou k dispozici. A jak již bylo řečeno, většina hráčů na tomto poli preferuje rychlé a velké zisky před vzdálenými a nejistými.

To neznamená, že by se vývoj IPv6 zastavil. Koncem roku 1998 vyšla *revidovaná sada RFC* dokumentů s definicí základních protokolů a služeb a postupně jsou aktualizovány či doplňovány další kousky této velké mozaiky. Poslední verze adresní architektury pochází z roku 2006, podpora mobility byla dokončena v roce 2004 (a revidována v roce 2011), o rok později došlo k revizi bezpečnostních prvků ... Navíc – a to je nejdůležitější – se začaly množit a zlepšovat implementace v nejrůznějších operačních systémech. Také řada aplikací dnes již podporuje nový protokol.

uzavřené	
ipv6	(původně *ipng*) vytvořila většinu základních specifikací
mip6	mobilita
multi6	multihoming
shim6	multihoming
6bone	vytvoření sítě *6bone*
aktivní	
6man	údržba a aktualizace specifikací
v6ops	provoz IPv6 sítí
6renum	přeadresování IPv6 sítí
mext	rozšíření mobility
6LoWPAN	IPv6 v nízkonapěťových osobních sítích

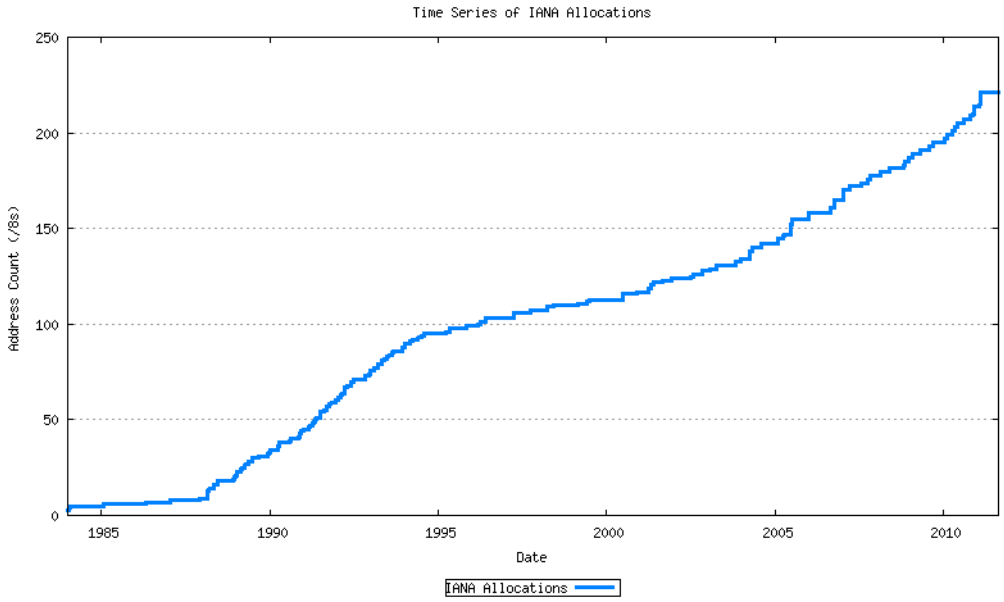
Tabulka 1.1: Pracovní skupiny IETF zapojené do vývoje IPv6

Na vývoji IPv6 a jeho komponent se podílela a podílí celá řada pracovních skupin IETF. Přehled těch nejvýznamnějších uvádí tabulka 1.1. Veškeré jejich dokumenty najdete na adrese

www▸ *http://www.ietf.org/html.charters/wg-dir.html*

Priority pro nasazení se časem měnily. Tlak nedostatku adres sice polevil, zato se do popředí začaly drát jiné přednosti IPv6, zejména podpora mobility. Při rychle rostoucím zájmu o nejrůznější přenosná zařízení a jejich zapojení do Internetu by se právě jejich podpora, která je v IPv6 výrazně lepší než u jeho předchůdce, mohla stát rozhodujícím argumentem.

Ovšem nelze nepřiznat, že trvalo bezmála deset let, než se podařilo dokončit specifikaci mobilního IPv6 – RFC 3775: *Mobility Support in IPv6* vyšlo v roce 2004. Po celou tu dobu byla podpora mobility všude vyhlašována za povinnou součást IPv6 a jeden z důvodů, proč přejít na nový protokol. Právě rozpor mezi slibnými vlastnostmi na papíře a tristním stavem implementací, v nichž pokročilé prvky zpravidla chyběly, odvedl IPv6 medvědí službu.



Obrázek 1.1: Spotřeba IPv4 adres (zdroj: *ipv4.potaroo.net*)

adresy jsou zpět

Jenže rok se s rokem sešel a adresní prostor vrátil úder, a to rovnou KO. Internet si sice našel způsob, jak zpomalit jeho konzumaci, ale i ten má své meze. Obrázek 1.1 ukazuje historický vývoj počtu osmibitových prefixů přidělených jejich centrálním správcem IANA. Je v něm pěkně vidět, jak opatření z poloviny 90. let razantně snížila tempo spotřeby, proč prognózy kolem roku 2000 ukazovaly dostatek adres na 20 let a jak později začala křivka zase ošklivě stoupat.

Aktuálně se nacházíme v situaci, kdy je vyčerpána centrální zásoba IANA a jednotlivé regionální registry (RIR) postupně spotřebovávají své zásoby.

Nejrychleji rostoucí APNIC skončil s adresami velmi brzy po IANA, vyčerpání ostatních registrů je očekáváno během několika let – evropský RIPE NCC na přelomu let 2011/2012, zbývající tři zhruba o rok později.

vyčerpáno	
IANA	3. února 2011
APNIC	19. dubna 2011
prognóza	
RIPE NCC	únor 2012
LACNIC	březen 2014
ARIN	duben 2014
AFRINIC	duben 2014

Tabulka 1.2: Vyčerpání IPv4 adres

Vyčerpání registru neznamená, že v dané oblasti nelze získat IPv4 adresu. Ale místní poskytovatelé Internetu (v roli lokálních registrů, LIR) už nedostanou žádný větší blok. V režimu po vyčerpání regionální registry přidělují jen velmi omezené množství adres – každý lokální registr může získat jen jeden malý blok. Oficiálně jsou tyto adresy určeny pro přechodové mechanismy.

Jak rychle lokální registry vyčerpají své zásoby IPv4 adres závisí na tom, kolik si jich stačily nashromáždit, jakým tempem roste jejich zákaznictvo a která úsporná opatření nasadí. Zároveň se všeobecně očekává rostoucí obchodování s adresami, jehož některé případy již proběhly s nemalou mediální pozorností[1]. IPv4 adresy z pohledu provozovatelů sítí a zákazníků nejsou a hned tak nebudou zcela nedostupné, ale přístup k nim se postupně komplikuje a prodražuje.

NAT

Opatření k úspoře adres navíc porušují nejzákladnější principy Internetu – možnost přímé komunikace libovolných dvou zařízení. Začaly se totiž masivně šířit nástroje pro překlad adres – *Network Address Translation, NAT*. Fungují tak, že přístupový směrovač sítě mění IP adresy paketů, které jím procházejí ze sítě do Internetu a naopak. Díky tomu celá síť vystačí s jednou jedinou veřejnou IP adresou, ale počítače uvnitř nejsou z vnějšího Internetu adresovatelné. To znamená, že komunikace se dá zahájit jen směrem zevnitř sítě ven.

Zavedením NAT se ztrácí přímočarost komunikace. Vstupuje do ní nový prostředník, který představuje citelnou překážku. Zcela protichůdnou tendencí je rostoucí popularita služeb pro přímou komunikaci mezi uživateli (ICQ a podobné komunikátory, IP telefonie, videokonference a další). Potřebují vytvářet přímá spojení mezi komunikujícími zařízeními. Leží-li každý

[1] Na jaře 2011 koupil Microsoft od bankrotujícího Nortelu blok přesahující 600 tisíc IPv4 adres za 7,5 milionu USD.

v jiné NATované síti, není jak je navázat. Vymýšlejí se tedy různé berličky, kontaktní servery s veřejnými adresami, na nichž se mohou neveřejně adresovaní klienti spojit, komunikace přes prostředníky a podobně. Tunelovací mechanismus Teredo popsaný na straně 270 je pěknou ukázkou, jakou lahůdkou je život v síti protkané NATy.

Jako lék nabízí IPv6 svůj olbřímí adresní prostor. Již nikdy nedostatek adres, již nikdy více NAT. Každý počítač, hodinky, lednička či další zařízení bude mít svou vlastní, celosvětově jednoznačnou IP adresu.

zápory

V předchozím textu jsem opakovaně naznačil, že IPv6 nepřináší jen samá pozitiva a sociální jistoty. Podívejme se na nejvýznamnější pihy jeho krásy. Tou největší nepochybně je, že je příliš jiný a především zpětně nekompatibilní s IPv4. To podstatným způsobem komplikuje jeho nasazení – uživatelé s počítači hovořícími pouze novým protokolem se nedostanou ke službám poskytovaným pouze po IPv4. Byla sice vymyšlena celá řada protokolů a mechanismů pro přechod od starého protokolu k novému, včetně překladu datagramů mezi nimi, v praxi ale toto úsilí vyznívá do prázdna.

IPv6 se potácí v bludném kruhu slepice versus vejce. Uživatelé o něj nemají zájem, protože v něm nejsou dostupné služby. A kdo by převáděl služby pod IPv6, když tam nejsou žádní uživatelé? Existují sice snahy postrčit poskytovatele služeb i připojení směrem k novému protokolu, jako byl například Světový den IPv6 v červnu 2011, ale statistiky stále ukazují objem IPv6 provozu v desetinách procenta vůči IPv4.

V poslední době je zřetelná snaha přispět k rozetnutí tohoto kruhu politicky. Vlády vydávají prohlášení a výzvy podporující přechod na IPv6, financují se projekty rozvíjející infrastrukturu a služby. Podařilo se dosáhnout mírného pokroku v mezích zákona, ale nástup IPv6 je stále velmi pomalý.

Své nepochybně vykonal i pomalý vývoj některých specifikací. O nejkřiklavějším případu mobility jsem se již zmínil. Bohužel není sama, DHCPv6 bylo definováno jen o rok dříve, přestože se jedná o protokol ve světě IPv4 dobře známý a hojně používaný. Standardizace jednotlivých součástí světa IPv6 stále probíhá, i když nyní už se spíše jen dolaďují detaily. Nejisté výnosy[2] v kombinaci s nestabilními specifikacemi jsou silně odrazující pro všechny, kteří by chtěli nový protokol implementovat. Proto jim to šlo jako psovi pastva, počáteční implementace byly značně nedokonalé a zlepšovaly se jen velmi zvolna.

1.2 Současný stav

Sečteno a podtrženo: IPv6 je zajímavý a nadějný protokol, který je mnohými považován za jedinou možnost pro budoucnost Internetu. Přesto míra jeho nasazení dlouhodobě pokulhává za vizemi a plány. Ještě pořád se nedá

[2] respektive spíše jisté nevýnosy

vyloučit, že se stane stejně slepou vývojovou větví, jako svého času ISO OSI, ale pravděpodobnost takového vývoje klesá. IETF nevyvíjí žádnou alternativu a největším konkurentem nového protokolu je stávající IPv4, od nějž se nikomu moc nechce ustupovat. Jenže Internet s NATem na každém rohu či obchodování s adresami, což jsou nejčastěji citované scénáře dalšího vývoje IPv4 při vyčerpání adresního prostoru, prodraží provozování sítí a bude motivovat k přechodu na nový protokol.

Geoff Huston, autor grafů spotřeby IPv4 adres a matematických modelů jejich vyčerpání, vytvořil několik pěkných prezentací na téma současného i budoucího nasazení IPv6.

www▸ *http://www.potaroo.net/presentations/*

Jeho *Measuring IPv6 Day* ze září 2011 odhaduje počet strojů používajících IPv6 na 0,4 %. Číslo nikterak oslňující, které ale zahrnuje jen ty počítače, které při přístupu otevřeném oběma protokoly dávají přednost IPv6. K obsahu, který je vystaven jen protokolem IPv6 se dostane zhruba desetinásobek strojů. A pokud není třeba použít DNS, je takový obsah dostupný bezmála 30 % uživatelských strojů. Situace se pozvolna zlepšuje, ale tempo je stále mnohem menší, než by bylo potřeba.

Google Jedním z velmi viditelných subjektů na poli IPv6 je Google. Protokolu se soustavně věnuje od roku 2008 a řeší dilema, zda své servery zpřístupnit nativně po IPv6 za cenu ztráty malého počtu zákazníků. Jenže i desetina procenta je v případě světové vyhledávací jedničky obrovský počet uživatelů. Výsledkem je šalamounské řešení, kdy se k serverům Google sice dá dostat nativním IPv6, ovšem uživatel musí aktivně chtít. Nejjednodušší variantou je použít doménové jméno

www▸ *http://ipv6.google.com/*

Koncepčnější cesta je k dispozici pro organizace, které mohou získat plnohodnotný IPv6 přístup ke službám Google. Je postaven na proměnlivém chování DNS. Pokud se průměrný stroj dotáže DNS, zda existuje IPv6 adresa pro *www.google.com*, skončí neúspěšně. Jestliže se však zeptá ze sítě, jíž Google povolil IPv6 přístup, dostane kladnou odpověď (a pravděpodobně se připojí novějším protokolem, protože operační systémy mu obvykle dávají přednost).

Google tedy eviduje seznam sítí, jimž má poskytovat IPv6 služby, a pro ně se jeho DNS servery chovají jinak. Chcete-li se mezi ně zařadit, je třeba o to požádat a mimo jiné se zavázat, že budete řešit případné problémy v IPv6 komunikaci, které by se vyskytly. Seznam sítí zapojených do IPv6 programu není veřejný, v České republice vím o několik univerzitách, jež jsou do něj zapojeny. Podrobnější informace najdete na adrese

www▸ *http://www.google.com/ipv6/*

světový den IPv6

Jednou z novějších aktivit byl *Světový den IPv6*, který se konal 8. června 2011. Cílem bylo vyzkoušet si IPv6 v ostrém provozu, protože řada významných zdrojů[3] během tohoto dne poskytovala své služby nativně oběma protokoly. Vše se pečlivě sledovalo a měřilo. Pokud je mi známo, nedošlo k žádným dramatickým problémům na straně uživatelů, ale kupodivu ani k dramatickému nárůstu IPv6 provozu. Některé ze zúčastněných serverů už své servery ponechaly přístupné nativním IPv6. Celkově lze den IPv6 považovat za úspěšný a pravděpodobně se bude dříve či později v rámci všeobecného osmělování opakovat.

Internetová komunita rozhodně netrpí nezájmem o nový protokol. Konference na toto téma se těší notoricky dobré účasti, čile se diskutuje a v současnosti i experimentuje a trochu nasazuje. Jen těm provozním grafům se pořád nechce odlepit se výrazněji ode dna.

1.3 Základní principy

Na začátku kapitoly jsem popsal úkoly, které mělo IPv6 vyřešit. Zde se budu ve stručnosti zabývat některými nosnými principy, na kterých je postaveno.

Požadavek na větší rozsah *adresního prostoru* vedl k nemalým debatám o optimální délce adresy. Nakonec byla stanovena na 128 bitů, tedy čtyřnásobek délky použité v IPv4. To znamená, že k dispozici je $3{,}4 \cdot 10^{38}$ adres. To je jen těžko představitelné číslo, zkusme je uvést do souvislostí. Povrch zeměkoule činí přibližně půl miliardy kilometrů čtverečních. To znamená, že na jeden čtvereční milimetr zemského povrchu připadá $667 \cdot 10^{15}$ adres. Ano, řeč je o milionech miliard. V kapitole o adresování uvidíte, že IPv6 velmi plýtvá. Celých 64 bitů věnuje na identifikátor rozhraní, což znamená, že v jedné podsíti lze rozlišit miliardy miliard počítačů. Každá síť má prostor na adresaci 65 tisíc podsítí. A takovýchto sítí je k dispozici bezmála 30 tisíc na každého obyvatele zeměkoule[4]. IPv6 adres je v každém ohledu dost a dost, jak se přesvědčíte v kapitole 3 na straně 53.

Formát datagramu byl podroben zásadní revizi. Stručně řečeno: počet položek byl minimalizován a jejich složení upraveno tak, aby základní hlavička datagramu měla konstantní délku. Dřívější volitelné položky byly přesunuty do samostatných hlaviček, které mohou být přidávány k pevnému základu. Pořadí přidávaných hlaviček je zvoleno tak, aby směrovač co nejrychleji mohl zpracovat ty, které jsou určeny pro něj, a zbývající ignorovat.

Popsané změny v záhlaví datagramu mají za cíl usnadnit jeho zpracování a umožnit tak směrování paketů vysokou rychlostí. Dalším aspektem z téže oblasti je zavedení koncepce toku (proud souvisejících datagramů se spo-

[3] Mezi jinými servery Google, Facebook, Yahoo! a distribuční sítě Akamai a Limelight Networks.

[4] Počítáno pro deset miliard pozemšťanů.

lečnými parametry), který má opět usnadnit vysokorychlostní směrování. Formát datagramu popisuje kapitola 2 na straně 35.

Z hlediska *automatické konfigurace* se autoři IPv6 snažili, aby byla pokud možno zcela bezpracná. Zavedli dvě alternativy: Stavová konfigurace je staré známé DHCP, ovšem upravené pro IPv6. Jeho podpora je nyní povinná. Bezstavová konfigurace představuje nový princip, kdy si počítač dokáže sám stanovit svou adresu a naučí se směrovat, aniž by jeho správce kdekoli cokoli konfiguroval. Automatickou konfigurací se zabývá kapitola 6 na straně 119.

S bezstavovou konfigurací je poměrně těsně svázáno i *objevování sousedů*. Jeho primárním cílem je nahradit dřívější protokol ARP při hledání fyzických adres sousedních počítačů. Ovšem objevování sousedů má poněkud širší záběr a zahrnuje i mechanismy pro automatickou konfiguraci (objevování směrovačů a parametrů sítě) či testování jednoznačnosti adresy. Vše se dočtete v kapitole 5 na straně 103.

Požadavek na *služby se zaručenou kvalitou* se projevil zavedením priorit. Ovšem jejich význam zatím nebyl přesně definován, takže zde zůstává ještě pole otevřené.

Pro zajištění *bezpečnosti* slouží dvě rozšiřující hlavičky: autentizační a šifrovací. Autentizační umožňuje ověřit, zda odesilatelem dat je skutečně ten, kdo to o sobě tvrdí, a zda během přepravy nedošlo ke změně dat. Hlavička pro šifrování dokáže totéž a navíc lze její pomocí zašifrovat celý obsah datagramu. Způsob zabezpečení IPv6 popisuje kapitola 10 na straně 201.

Podpora *mobilních uzlů* staví na domácích agentech. Jedná se o směrovač, který je umístěn v domácí síti mobilního uzlu a „zastupuje jej“ v době nepřítomnosti. Mobilní uzel svému agentovi hlásí aktuální polohu a pokud mu do domácí sítě dorazí nějaká data, domácí agent je přepošle. Následně mobilní uzel oznámí odesilateli, že dočasně změnil svou IP adresu a další komunikace s ním již bude probíhat přímo. Více najdete v kapitole 11 na straně 223.

Pro usnadnění *společné existence IPv6 a IPv4* byla vymyšlena řada nástrojů. Nejjednodušší možností je klasické tunelování, které ponechává oba světy víceméně oddělené a pouze využívá infrastrukturu jednoho k přenosu dat druhého. Kromě něj jsou však k dispozici i rafinovanější metody nabízející překlad datagramů a podobné věci. Zabývá se jimi kapitola 12 na straně 255.

1.4 Implementace

Podpora IPv6 ve směrovačích, operačních systémech a aplikacích se začala objevovat poměrně záhy po vydání první sady RFC. V listopadu 1996 se objevilo IPv6 jako experimentální vlastnost jádra Linuxu verze 2.1.8, další systémy na sebe nenechaly dlouho čekat.

Druhou polovinu 90. let lze označit jako experimentální období plné velkých nadějí, většinou nenaplněných. Zavedení producenti operačních systémů a síťových krabic pozorovali novinku s odstupem, jen tu a tam lehce ochutnali. Několik mladých firem a startupů zkusilo rychlou implementací nového protokolu získat dobrou pozici na trhu „Internetu budoucnosti". Podobně se asijské firmy snažily touto cestou prosadit proti tradičním výrobcům.

Zřejmě i v reakci na tyto snahy začala kolem roku 2000 implementační vlna, kterou bych označil jako marketingovou. Bylo třeba mít v produktovém letáku zaškrtnutou kolonku „podpora IPv6", na kvalitě skutečné podpory příliš nezáleželo. Typická implementace IPv6 z počátku nového tisíciletí měla jen ty nejzákladnější schopnosti a také výkonem často zaostávala za svým předchůdcem[5].

Postupem času se ale situace lepšila. Pozitivní roli rozhodně sehrálo *IPv6 Forum* a jeho program *IPv6 Ready*, k nimž se co nevidět dostanu podrobněji. Přestalo stačit napsat „podporujeme IPv6". Bylo třeba opatřit si certifikát, čili projít příslušnými testy. Výsledkem je, že nejvýznamnější platformy – operační systémy i hardwarové směrovače – se v současnosti mohou pochlubit podporou IPv6 na velmi slušné úrovni. Chcete-li experimentovat či uvažovat o seriózním nasazení nového protokolu, nemělo by vám z této strany nic zásadního stát v cestě.

Pravda, některé pokročilé prvky – jako je mobilita či zabezpečení – ještě mají své mouchy, obecně ale implementace za posledních několik let udělaly velký krok dopředu a dále se zlepšují. Testy kompatibility a schopností vzájemné spolupráce přispívají k tomu, aby vznikalo reálně použitelné prostředí.

Naopak příliš nedrží krok weby věnované této problematice. Obvykle vzniknou ve velkém nadšení, ale následně nebývají aktualizovány a jejich obsah se postupně rozchází s realitou. Většina výrobců programů a zařízení ale dodržuje konvenci vytvořit na svém webu stránku věnovanou IPv6. Obvykle mívá adresu

www▸ *http://web_výrobce/ipv6/*

a najdete na ní informace o podpoře protokolu v produktech dané společnosti a často i vize dalšího vývoje.

1.5 IPv6 Forum a program IPv6 Ready

Stalo se již zvykem, že na podporu nových síťových technologií vznikají společenství organizací a osob usilujících o prosazení novinky do reálného

[5] V počáteční fázi hadwarové směrovače často implementovaly IPv6 softwarově, tedy s výkonem řádově nižším proti IPv4.

života. Jistě nejznámějším příkladem je *Wi-Fi Alliance*, jejíž pozice na poli bezdrátových lokálních sítí je taková, že oficiální název těchto technologií IEEE 802.11 znají jen lidé zasvěcení, zatímco pojem Wi-Fi zlidověl.

IPv6 Forum

Analogickým sdružením pro podporu nové verze IP je *IPv6 Forum* založené v roce 1999. Jeho cíle sahají od propagace nového protokolu přes sdílení a šíření znalostí a zkušeností až po vývoj technických specifikací a řešení problémů při praktickém nasazení. *IPv6 Forum* původně vzniklo jako centralistická organizace, později ovšem začalo zakládat své národní a regionální pobočky. Na podzim roku 2011 jich existuje kolem šedesáti, jejich seznam najdete na webu

www▸ *http://www.ipv6forum.com/*

Ten se bohužel nachází ve velmi neutěšeném stavu a s výjimkou titulní stránky nestojí za návštěvu. Jednotlivé sekce jsou buď prázdné (dokumenty), nebo nebyly několik let aktualizovány. Na titulní stránce ovšem najdete odkazy na významné konference s tématikou IPv6 a další zajímavé odkazy.

IPv6 Ready

Nejvýznamnější aktivitou fóra jsou rozhodně certifikační programy, mezi nimiž má prominentní roli nejstarší *IPv6 Ready*. Motivací jeho vzniku byly rané implementace IPv6, jež vykazovaly celou řadu více či méně závažných problémů.

Již v roce 1998 vznikl japonský program *TAHI*, který testoval dodržování specifikací v implementacích IPv6 a jejich vzájemnou interoperabilitu. Rychle získal technické znalosti a zkušenosti i dobré jméno mezi implementátory, neměl však žádný oficiální statut. Po založení IPv6 Fóra se nabízelo spojit síly a vytvořit certifikační program, za nímž budou stát jak odborné kompetence, tak oficiálně respektované jméno. Výsledkem je *IPv6 Ready*:

www▸ *http://www.ipv6ready.org/*

V jeho rámci si každý autor programu či zařízení podporujícího IPv6 může nechat otestovat jeho kompatibilitu se standardy. Pokud uspěje, získá oficiální certifikát a může používat stříbrné či zlaté logo *IPv6 Ready*. Míra kompatibility má totiž různé úrovně, v oficiální terminologii nazývané fáze.

Fáze 1 (stříbrné logo) ověřuje nejzákladnější kompatibilitu se specifikacemi IPv6. Konkrétně se testuje, zda zařízení podporuje

- IPv6 adresy
- ICMPv6
- objevování sousedů
- bezstavovou automatickou konfiguraci

Obrázek 1.2: Logo *IPv6 Ready*: vlevo fáze 1, vpravo fáze 2

Testuje se pouze povinné chování (v RFC označené jako „must“). Od roku 2003 bylo vydáno bezmála 500 certifikátů. V současné době je fáze 1 považována za překonanou a *IPv6 Forum* doporučuje zaměřit se na pokročilejší fázi 2.

Fáze 2 (zlaté logo) je všeobecně komplikovanější. Kromě povinných ověřuje i prvky důrazně doporučené (v RFC označené jako „should“). Především se ale rozpadá do různých kategorií. Povinný je základní test, který představuje rozvinutou fázi 1 doplněnou navíc o objevování MTU cesty. Při testech se zároveň rozlišuje, zda je produkt certifikován jako koncový stroj (hostitel) nebo jako směrovač. K povinné základní certifikaci může získat ještě specializovaný certifikát v některé z následujících kategorií:

- bezpečnost (IPsec)
- IKEv2
- mobilní IPv6
- mobilní sítě (NEMO)
- DHCPv6
- SIP
- SNMP
- multicast aneb MLDv2
- uživatelský agent IMS pro mobilní sítě (testováno)

- přechodové mechanismy (připravováno)

Na podzim 2011 překročil počet udělených certifikátů fáze 2 číslo 600. Vybrané nejvýznamnější držitele shrnuje v chronologickém pořadí tabulka 1.3. Jejich aktuální přehled i podrobné informace o testovacích procedurách najdete samozřejmě na stránkách programu *IPv6 Ready*.

platforma	*kategorie*	*získáno*
FreeBSD (KAME)	hostitel	3/2006
	směrovač	3/2006
Cisco IOS 12.4(9)T	směrovač	4/2006
Linux 2.6.15	hostitel	5/2006
	IPsec konec	5/2006
Linux 2.6.20	směrovač	9/2007
	IPsec brána	10/2007
MS Windows Vista	hostitel	10/2007
	IPsec konec	1/2008
MS Windows Server 2008	hostitel	1/2008
	IPsec konec	3/2008
MS Windows 7	hostitel	10/2010

Tabulka 1.3: Vybraní držitelé certifikátů *IPv6 Ready* fáze 2

IPv6 Enabled

Postupem času začalo *IPv6 Forum* svůj certifikační program rozšiřovat. Vzhledem k tomu, že v posledních letech již není pes nejhlouběji zakopán v technice, ale spíše v ochotě nový protokol nasadit, nabízí se myšlenka certifikovat služby. Jejím ztělesněním je program *IPv6 Enabled* zahrnující dva podprogramy: pro WWW servery a poskytovatele Internetu.

Webový certifikát *IPv6 Enabled WWW* je dost jednoduchý. Garantuje, že dotyčný web server má v DNS registrovánu IPv6 adresu a je tímto protokolem dosažitelný. Čili klientovi používajícímu IPv6 nebude stát nic v cestě k jeho využívání. Ve veřejně dostupné databázi držitelů certifikátu najdete více než 1300 položek. Do domény *cz* patří 19 z nich, za nejvýznamnější lze považovat *www.nic.cz* a *www.regzone.cz*.

Poskytovatel Internetu získá certifikát *IPv6 Enabled ISP*, jestliže disponuje IPv6 adresami a přiděluje je svým zákazníkům, je dosažitelný z hlediska směrování a trvale nabízí IPv6 služby zákazníkům. V září 2011 počet certifikovaných subjektů převyšoval stovku. Z České republiky mezi nimi najdete šest regionálních poskytovatelů Internetu a jednu housingovou firmu. Velká jména byste mezi nimi hledali marně[6].

IPv6 Education

Vedle techniky a nabídky služeb jsou důležité také znalosti. *IPv6 Forum* se proto pustilo i do této oblasti a zahájilo certifikační program *IPv6 Education*.

[6] Jedno je ale zastoupeno nepřímo, protože jedním z držitelů je Losan internet, který dnes patří pod Telefónica O2 Business Solutions.

Opět se člení do několika větví, v nichž lze ověřit vzdělávací kursy nebo osoby, a to jak pro pozici IPv6 odborníků (Engineer), tak jeho šiřitelů (Trainer). Asi nejkurióznější složkou programu jsou metacertifikáty, kdy *IPv6 Forum* certifikuje jiné certifikační programy, jimiž vydávané certifikáty tak získávají na váze.

1.6 6bone

Když se začalo experimentovat s prvními implementacemi, vznikla potřeba rozlehlé IPv6 sítě, která by posloužila k testování a získávání praktických zkušeností. Tak v roce 1996 vznikla síť *6bone*. Původně propojila jen tři instituce – G6 ve Francii, UNI-C v Dánsku a WIDE v Japonsku. Svého maxima dosáhla v roce 2003, kdy bylo do *6bone* zapojeno kolem tisíce institucí z 50 zemí.

6bone byla takzvanou virtuální sítí. To znamená, že neměla vlastní vyhrazenou infrastrukturu, ale využívala existující sítě. Skládala se z lokálních IPv6 sítí, navzájem propojených tunely. To znamená, že IPv6 datagramy se balily jako data do běžného IPv4 a přenášely se standardním Internetem až do cílové sítě. Bylo to jednoduché, levné a dala se vytvořit topologie, jaká byla potřeba.

Hlavním cílem *6bone* bylo „hrát si na opravdický IPv6 Internet“ a získat tak praktické zkušenosti s jeho provozem. Proto byla v rámci sítě definována směrovací politika, vypracovány procedury na přidělování adres a další potřebné operace. Řadu let byla jedinou IPv6 sítí s globálním dosahem.

Síť měla vyhrazeny vlastní adresy, jež začínaly čtveřicí 3ffe (čili prefixem 3ffe::/16, jak se dočtete později). Organizace, které poskytovaly připojení k *6bone*, dostaly k dispozici určitý rozsah adres, vyjádřený jejich společným prefixem (označovaným jako pTLA). Z něj pak poskytovatel přidělovал části připojeným sítím. Směrovače poskytovatelů disponujících pTLA zároveň tvořily páteř sítě *6bone*.

Když po roce 2000 začaly být IPv6 adresy přidělovány standardní cestou a IPv6 začalo postupně pronikat do Internetu, začal klesat i zájem o *6bone*. Svůj účel síť splnila, pomohla získat praktické zkušenosti s provozem IPv6 a doladit řadu jeho prvků. Od samotného počátku byla deklarována jako síť dočasná, což se naplnilo po deseti letech existence.

Síť *6bone* skončila stylově 6. 6. 06 a její prefix 3ffe se vrátil k pozdějšímu využití pro běžné adresy. Odvedla cenné služby a má zajištěno čestné místo v historii IPv6.

1.7 Politická podpora a projekty

IPv6 se během své existence dočkalo oficiální podpory z řady míst, včetně těch nejvyšších. Velmi aktivní je Asie, která do kolotoče IPv4 Internetu vstoupila pozdě. V důsledku toho zdejší výrobci hrají spíše druhé housle a některé země (v první řadě Čína) mají citelný nedostatek IPv4 adres.

Nepřekvapí, že japonská vláda již v roce 2000 vyhlásila oficiální podporu IPv6 a následně ji uplatňovala v podobě různých projektů, ale i daňových úlev. V roce 2005 vyhlásila směr IPv6 vláda USA – nejprve ministerstvo obrany, později se přidala celá federální administrativa. V roce 2008 měly všechny vládní sítě v USA podporovat IPv6, následovat měl postupný přechod aplikací.

Nepodařilo se, nicméně vláda USA to nevzdává. V září 2010 vydala memorandum, které požaduje po vedoucích IT oddělení všech orgánů vlády:

- Do konce září 2012 zpřístupnit všechny služby po IPv6[7].
- Do konce září 2014 plošně nasadit nativní IPv6 ve svých sítích.
- Jmenovat všude manažery pro přechod k IPv6.
- Pořizovat pouze IT vybavení s kvalitní podporou IPv6.

Ke splnění posledního bodu vytvořil NIST testovací program označovaný jako USGv6, který definuje požadavky a způsoby jejich ověřování. Jeho web rozhodně stojí za návštěvu:

www ▸ *http://w3.antd.nist.gov/usgv6/*

Aktivní je také Evropská komise. Z února 2002 pochází její *Next Generation Internet – priorities for action in migrating to the new Internet protocol IPv6*. Tento dokument stál v pozadí financování několika velkých projektů orientovaných na IPv6 z prostředků evropských rámcových programů. Výzvy ke členským státům v něm obsažené však na příliš úrodnou půdu nepadly.

Z května 2008 pochází akční plán Evropské komise k nasazení IPv6 – *Action Plan for the deployment of Internet Protocol version 6 (IPv6) in Europe*. Jedná se o dokument místy rozumný, místy bezzubý a místy zcela neuvěřitelný[8]. Mimo jiné požaduje, aby projekty financované ze 7. rámcového projektu používaly ke komunikaci IPv6, pokud to je možné. Také ohlašuje, že při

[7] Aktuálně využívá řada vládních webů distribuční síť Akamai, která s IPv6 sice experimentuje, ale zatím je nepodporuje. Bude zajímavé pozorovat, jestli se podaří Akamai dotlačit k podpoře IPv6, nebo zda vládní weby změní dodavatele.

[8] Obávám se, že zpřístupnění webů „Europa“ a „CORDIS“ po IPv6 v roce 2010 (které se navíc podařilo jen napůl, *codis.europa.eu* není ani v roce 2011 dostupný po IPv6!) nebyla taková bomba, jak se domnívají autoři dokumentu, když tento bod zařadili jako první akci stimulující dostupnost obsahu a služeb po IPv6.

inovaci technického vybavení evropských struktur bude požadována podpora IPv6 a k podobnému kroku vyzývá i vlády členských států.

Evropská komise už v rámci 6. rámcového programu podpořila několik významných projektů rozvíjejících novou verzi IP. Některé z nich byly zaměřeny na vytvoření reálných IPv6 sítí, získání a dokumentaci zkušeností s jejich provozem. Sem patří například *6NET* či *Euro6IX*. Další mířily do oblasti vzdělávání a šíření informací, jako například *6DISS* a jeho nástupce *6DEPLOY*. Mezi podporovanými projekty najdete i tématicky úzce zaměřené výzkumy dílčích oblastí souvisejících s IPv6, třeba projekt *ENABLE* zabývající se mobilitou ve velkých heterogenních IP sítích.

Ani Vláda České republiky nezůstala k IPv6 lhostejná. 8. června 2009 přijala usnesení číslo 727, ve kterém uložila ministrům a vedoucím ústředních orgánů státní správy, aby od poloviny roku 2009 při obnově síťových prvků požadovali podporu IPv6 a do konce roku 2010 zajistili přístup ke službám eGovernmentu novým protokolem. Usnesení zároveň doporučuje hejtmanům a pražskému primátorovi postupovat obdobně.

Jak už to s usneseními bývá, v plnění jsou značné rezervy. Na podzim 2011 je dostupná po IPv6 necelá polovina ministerských webů. Nejsmutnější je jeho absence na ministerstvu vnitra, které má v gesci informatiku. eGovernment a jeho Portál veřejné správy jsou k mání stále jen po IPv4.

Mnohé státy se zkrátka snaží různými metodami posouvat rozvoj IPv6 vpřed, protože vnímají blížící se vyčerpání IPv4 adres a další problémy stávajícího protokolu jako ohrožení svého dalšího rozvoje. Bohužel zatím IPv6 pořád není v pozici, kdy by se prosazoval „samospádem“ a kdy by se do něj hrnuli uživatelé i poskytovatelé z toho prostého důvodu, že se jim to okamžitě vyplatí. Snad k jeho prosazení alespoň částečně přispěje i tato kniha.

1.8 Webové zdroje

Na webu pochopitelně najdete nepřeberné množství stránek věnovaných IPv6. Podívejme se na ty, které stojí za pozornost. „Oficiálním“ webem odkazovaným ze stránek IPv6 Fóra je *gogoNET*

www▸ *http://gogonet.gogo6.net/*

Je prezentován jako sociální síť a služby, jež mají profesionálům usnadnit cestu k IPv6. Jedná se o směs informací (dovedně skrytých pod položku *Interact*) a praktických nástrojů, protože pod hlavičku *gogo6* se přestěhovala služba *Freenet6* nabízející volné tunelované připojení k IPv6. Podrobněji se jí budu věnovat v kapitole 13 na straně 295. Z informačního obsahu stojí určitě za pozornost instruktážní videa, prezentace a občas se objeví cenný text v blogu.

Významným a často odkazovaným zdrojem je také *The IPv6 Portal* na adrese

www▸ *http://www.ipv6tf.org/*

Obsahuje řadu informací, orientovaných zejména na evropské aktivity. Jeho jednoznačným kladem, že zdejší novinky ze světa IPv6 jsou udržovány v aktuálním stavu již řadu let. Jeho autoři bohužel pravděpodobně nikdy neslyšeli o použitelnosti. Některé stránky se jeví jako prázdné, než si všimnete, že vpravo nahoře sídlí decentní podmenu, které vám zpřístupní jednotlivé zdejší sekce.

Trudnomyslným mohu doporučit web specializovaný na vyčerpání IPv4 adres

www▸ *http://www.ipv4depletion.com/*

Řadu materiálů (včetně multimediálních) najdete ve výstupech projektu *6DISS*. Postupně zastarávají, protože projekt skončil v září 2007, stále však představují použitelný a ucelený zdroj informací.

www▸ *http://www.6diss.org/*

Na domácí půdě to s relevantními informacemi také není nijak oslňující. Pravděpodobně nejlepším informačním zdrojem je web

www▸ *http://www.ipv6.cz/*

o jehož obsah se stará několik autorů, pocházejících zejména z pracovní skupiny IPv6 při sdružení CESNET. Pokud máte k dané problematice co říci, rádi Vás uvítáme mezi autory.

I

Jak funguje IPv6

2 Formát datagramu

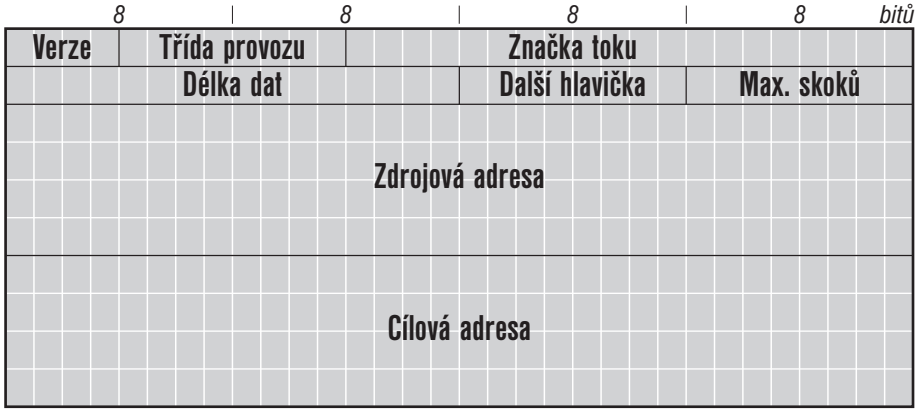
Základním kamenem IPv6 je dokument RFC 2460: *Internet Protocol, Version 6 (IPv6) Specification*, který obsahuje především formát datagramu. Ostatním mechanismům a datovým formátům, které souvisejí s IPv6, jsou věnovány další RFC specifikace.

2.1 Datagram

Datagram má v IPv6 obvyklý základní tvar: začíná hlavičkami, za kterými pak následují nesená data. V porovnání s IPv4 však došlo v hlavičkách ke koncepční změně. Dříve byla jejich délka proměnlivá a jednotliví účastníci komunikace mohli připojovat další nepovinné volby podle potřeby. Hlavička obsahovala kontrolní součet, který bylo třeba znovu vypočítat na každém směrovači, jímž datagram prošel (protože se změnila přinejmenším položka TTL).

IPv6 naproti tomu standardní hlavičku minimalizovalo a omezilo její prvky jen na ty nejnutnější. Tato základní hlavička má konstantní velikost. Veškeré doplňující, nepovinné či příležitostně užívané údaje byly přesunuty do rozšiřujících hlaviček, které v datagramu mohou a nemusí být přítomny. Jejich podobu a zpracování popíši v následující části.

Tvar základní hlavičky vidíte na obrázku 2.1. Přestože se adresy odesilatele a příjemce prodloužily čtyřikrát, celková délka základní hlavičky datagramu vzrostla ve srovnání s IPv4 jen dvojnásobně (z 20 B na 40 B, z toho 32 B zabírají adresy). Minimalismus je patrný na první pohled.



Obrázek 2.1: Základní hlavička datagramu

verze Položka *Verze (Version)* je obvyklým zahájením IP datagramu, které identifikuje verzi protokolu. Zde obsahuje hodnotu 6.

třída provozu Za ní následuje osmibitová *Třída provozu (Traffic class)*, která vyjadřuje prioritu datagramu či jeho zařazení do určité přepravní třídy. Cílem je, aby tato položka umožnila IP poskytovat služby se zaručenou kvalitou. V praxi ale tak daleko nejsme a v nejbližší době ani nebudeme. IP, a to ani ve verzi 6, neumí zaručit dopravní parametry, jako jsou přenosová rychlost, zpoždění či jeho rozptyl. Dovede však poskytovat tak zvané *diferencované služby (differentiated services, diffserv)*. Jejich prostřednictvím mohou mít datagramy různé priority a odlišné způsoby zacházení, které vedou k jejich přednostnímu zpracování či naopak odkládání až po ostatních. Právě diferencované služby využívají pro přenos svých informací položku *Třída provozu*. Ve vlastní definici IPv6 není nijak blíže upřesněna, pouze se zde požaduje, aby implicitní hodnotou byla nula.

značka toku Dalších 20 bitů je věnováno *Značce toku (Flow label)*. Koncepce toku je novinkou v IPv6 a stejně jako třída provozu zatím není přesně definována. V zásadě by jako tok měl být označován proud datagramů se společnými vlastnostmi (odesilatel, adresát, požadavky na vlastnosti spojení). Prostřednictvím identifikátoru (značky) směrovač rychle rozpozná, že datagram je součástí určitého toku, což mu usnadní rozhodování o jeho dalším osudu (bude s ním naloženo stejně, jako s předchozími členy téhož toku). Jak již bylo řečeno, jedná se stále o experimentální půdu a nic moc konkrétního zatím nebylo stanoveno. K tématu se vrátím v části 2.9 na straně 50.

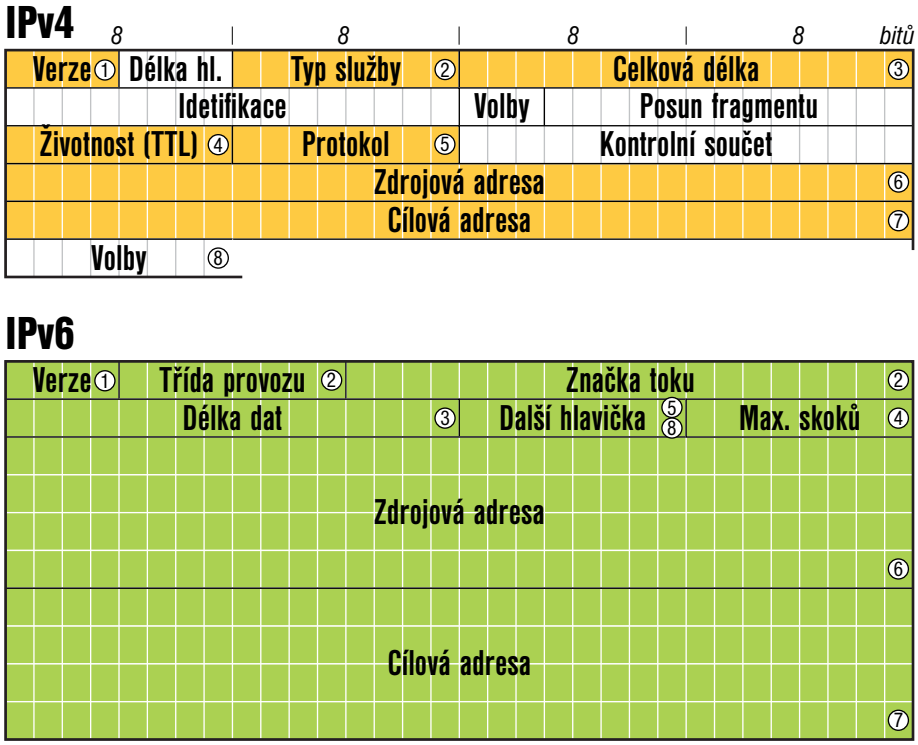
délka dat *Délka dat (Payload length)* nese údaj o délce datagramu. Přesně řečeno počet bajtů následujících za standardní hlavičkou. Z toho plyne, že základní hlavička se do této délky nepočítá, zatímco případné rozšiřující hlavičky ano. Jelikož je položka dvoubajtová, je maximální délkou 64 KB. Pokud je třeba vytvořit delší datagram, lze použít rozšiřující hlavičku *Jumbo obsah* popsanou v části 2.7 na straně 49.

další hlavička *Další hlavička (Next header)* obsahuje identifikaci, jaká hlavička či jaký druh dat následuje za standardní hlavičkou. Podrobněji se jí budu věnovat zanedlouho v části 2.2.

dosah *Maximální počet skoků (Hop limit)* je náhradníkem dřívější životnosti datagramu (TTL). Průchod datagramu jedním směrovačem je považován za jeden skok. Odesilatel v této položce uvede, kolik takových skoků smí datagram maximálně absolvovat. Každý směrovač po cestě pak sníží hodnotu o jedničku. Dojde-li tím k vynulování položky, datagram bude zlikvidován a odesilateli se pošle ICMP zpráva o vypršení maximálního počtu skoků. Smyslem omezení je ochrana proti cyklům při směrování (zacyklený datagram nebude v síti strašit do nekonečna).

adresy Závěrečnými dvěma položkami je dvojice IPv6 adres: *Zdrojová adresa (Source address)* a *Cílová adresa (Destination address)*. Vzhledem k délce

adresy v IPv6 zabírají tyto dvě položky 80 % rozsahu celé hlavičky. Podrobnosti o adresování se dočtete v kapitole 3 na straně 53.



Obrázek 2.2: Porovnání hlaviček IPv4 a IPv6

Při srovnání s IPv4 je nejnápadnější absence tří informací: rozšiřujících voleb, kontrolního součtu a fragmentace. Rozšiřující volby byly nahrazeny obecnějším principem zřetězení doplňkových hlaviček. Obdobně údaje související s fragmentací byly přesunuty do těchto rozšiřujících hlaviček. Zdaleka ne každý paket je totiž fragmentován a lze očekávat, že v IPv6 bude fragmentace ještě vzácnější než v současnosti. IPv6 totiž požaduje, aby infrastruktura pro jeho přenos dovedla přenášet pakety minimálně o délce 1280 B (MTU). Vzhledem k tomu, že drtivá většina koncových zařízení je dnes připojena prostřednictvím různých variant Ethernetu s MTU 1500 B, lze očekávat, že tato maximální velikost paketů se usídlí téměř všude a fragmentace prakticky zmizí ze světa.

Kontrolní součet zmizel bez náhrady. Tuto službu typicky vykonává nižší vrstva síťové architektury (např. zmiňovaný Ethernet), takže na úrovni IP by se jen opakovalo její snažení. Vzhledem k tomu, že hlavička se mění

v každém směrovači (klesá dosah datagramu), znamenalo by to zbytečné zpomalování.

Porovnání hlaviček IPv4 a IPv6 názorně představuje obrázek 2.2. V IPv4 datagramu jsou vybarveny položky, které byly (zpravidla v poněkud pozměněné podobě) převzaty do IPv6. Stejná čísla označují položky, které si navzájem odpovídají.

2.2 Zřetězení hlaviček

IP verze 6 používá odlišný způsob reprezentace rozšiřujících hlaviček než jeho předchůdce. Každá hlavička je nyní samostatným blokem a k jejich vzájemnému propojení slouží položka *Další hlavička (Next header)*. Kód v ní obsažený identifikuje, jakého typu je hlavička, která následuje za tou stávající. Každá rozšiřující hlavička začíná položkou *Další hlavička*. Prostřednictvím těchto hodnot lze za sebe zřetězit hlaviček, co hrdlo ráčí.

Poslední z nich obsahuje v položce *Další hlavička* typ dat, která datagram nese. Hodnota položky *Další hlavička* tak zároveň zastupuje dřívější položku *Protokol*. Nejvýznamnější hodnoty shrnuje tabulka 2.1. Aktuální a kompletní specifikaci hodnot pro typy přenášených dat najdete na adrese

www▸ *http://www.iana.org/assignments/protocol-numbers*

.

Rozšiřující hlavičky	
0	volby pro všechny (hop-by-hop options)
43	směrování (routing)
44	fragmentace (fragment)
50	šifrování obsahu (ESP)
51	autentizace (AH)
59	poslední hlavička (no next header)
60	volby pro cíl (destination options)
135	mobilita (mobility)
Typ nesených dat	
6	TCP
8	EGP
9	IGP
17	UDP
46	RSVP
47	GRE
58	ICMP

Tabulka 2.1: Vybrané hodnoty položky *Další hlavička*

Pokud tedy datagram neobsahuje žádné rozšířené hlavičky, bude přímo jeho základní IPv6 hlavička obsahovat jako *Další hlavičku* identifikátor typu nesených dat. Tuto situaci ilustruje obrázek 2.3a. Na obrázcích 2.3b a 2.3c můžete sledovat, jak se změní obsah položek *Další hlavička*, když datagramu přidáme rozšiřující hlavičky *Směrování* a *Fragmentace*.

a) bez rozšiřujících hlaviček

b) s hlavičkou *Směrování*

c) s hlavičkami *Směrování* a *Fragmentace*

Obrázek 2.3: Zřetězení hlaviček datagramu

Hlavními devizami koncepce hlaviček v IPv6 je pružnost a úspornost. Součástí datagramu jsou jen ty průvodní informace, které skutečně potřebuje. Rubem mince je, že zpracování kompletních hlaviček může představovat průchod relativně dlouhým řetězcem. Pokud by se mělo odehrávat v každém směrovači na cestě mezi odesilatelem a příjemcem, mohlo by to vést k nezanedbatelné degradaci výkonu.

pořadí hlaviček

Tento problém řeší IPv6 velmi jednoduše – rozšiřující hlavičky mají předepsáno následující pořadí:

1. základní hlavička IPv6
2. volby pro všechny (hop-by-hop opions)
3. volby pro cíl (destination options) – pro první cílovou adresu datagramu a případné další uvedené v hlavičce *Směrování*
4. směrování (routing)
5. fragmentace (fragment)
6. autentizace (authentication)
7. šifrování obsahu (encapsulating security payload)

8. volby pro cíl (destination options) – pro konečného příjemce datagramu
9. mobilita (mobility)

Jeho cílem je, aby se informace zajímavé pro uzly, kterými datagram prochází, ocitly vpředu a hlavičky určené až pro koncového příjemce následovaly teprve za nimi. Pro průchozí směrovač jsou potenciálně zajímavé jen *Volby pro všechny*, které se smí vyskytnout jen bezprostředně za základní hlavičkou. Ničeho jiného si nemusí všímat. Jakmile vidí v *Další hlavičce* jiný kód než 0 (*Volby pro všechny*), ví, že může s analýzou datagramu skončit.

Ostatní rozšiřující hlavičky jsou zajímavé jen pro adresáta datagramu – ať už průběžného (pocházejícího z hlavičky *Směrování*) či koncového. Průběžného adresáta zajímají jen první tři (volby pro všechny, volby pro cíl a směrování), zatímco koncového se týkají všechny. Podle RFC 2460 adresát musí být schopen se vyrovnat s libovolným pořadím hlaviček, nicméně důrazně se doporučuje dodržovat výše uvedené.

Každá z rozšiřujících hlaviček by se měla objevit nanejvýš jednou. Výjimkou jsou volby pro cíl, které se mohou vyskytnout dvakrát – jednou před *Směrováním* a podruhé před *Mobilitou*.

Speciální význam má, pokud položka *Další hlavička* obsahuje hodnotu 59 (no next header). Ta signalizuje, že se jedná o poslední hlavičku, za kterou již nenásleduje vůbec nic. Pokud datagram podle své délky obsahuje ještě nějaká data, musí být ignorována. Je-li datagram přeposílán dále, musí do něj předávající tato data zkopírovat beze změny.

Podívejme se nyní podrobněji na tvar a význam jednotlivých rozšiřujících hlaviček.

2.3 Volby

Stávající IPv6 zavádí dvě hlavičky obsahující volby: *Volby pro všechny* (*hop-by-hop options*, *Další hlavička* před nimi má hodnotu 0) a *Volby pro cíl* (*destination options*, předcházející *Další hlavička* má hodnotu 60).

Obě hlavičky mají společný tvar, který najdete na obrázku 2.4. Význam položky *Další hlavička* jsem již vysvětlil. *Délka dat* obsahuje délku hlavičky v osmicích bajtů. Do délky se nepočítá prvních 8 bajtů, takže pokud má hodnotu 1, znamená to, že celá hlavička s volbami měří 16 B.

Položka *Volby* pak obsahuje vlastní volby. Ty mohou být zavedeny jako součást jednotlivých konkrétních mechanismů. Například v rámci podpory mobilních počítačů se objevila volba (*Domácí adresa*). Samotná definice IPv6 obsahuje jen dvě: *Pad1* a *PadN*. Slouží ke vkládání „vaty“ – volného místa, které má sloužit k lepšímu zarovnání ostatních prvků s přihlédnutím

Obrázek 2.4: Rozšiřující hlavičky *volby pro všechny* a *volby pro cíl*

k hranicím čtyřbajtových slov. Jedná se o vycpávky, které nenesou žádnou aktivní informaci. Přehled doposud definovaných voleb najdete v tabulkách 2.2 a 2.3.

Typ	*Význam*	*Strana*
0	Pad1	41
1	PadN	41
5	Upozornění směrovače	42
38	Rychlý start	50
194	Jumbo obsah	49

Tabulka 2.2: Volby pro všechny

Typ	*Význam*	*Strana*
0	Pad1	41
1	PadN	41
201	Domácí adresa	239

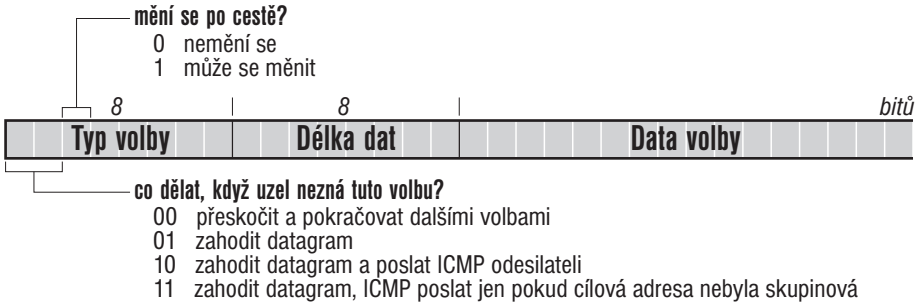
Tabulka 2.3: Volby pro příjemce

Pad1 *Pad1* vynechává 1 bajt. Tvar této volby je triviální: jedná se o jeden bajt s hodnotou 0, která identifikuje typ volby a zároveň říká, že to je vše.

PadN *PadN* umožňuje vynechat dva a více bajtů. První bajt opět určuje typ volby a má hodnotu 1. Za ním následuje jeden bajt obsahující délku volby, do níž se první dva bajty nepočítají. Následují data uvedené délky, jejichž hodnoty jsou nulové. Chcete-li tedy vynechat celkem 6 bajtů, bude mít *Délka dat* hodnotu 4 a za ní budou následovat čtyři nulové bajty „dat“.

formát voleb Všechny volby musí dodržovat jednotný tvar. Odpovídá tomu, který jste viděli u volby *PadN*. První bajt identifikuje, o jakou volbu se jedná. Za ním pak následuje *Délka dat* (do níž se nepočítají první dva bajty) a po ní data. Jejich strukturu musí definovat dokument, který zavede danou volbu.

V rámci *Typu volby* byl pevně předepsán význam nejvyšších tří bitů. První dva určují, co se stane s datagramem, pokud zpracovávající uzel dotyčnou volbu nezná. Za nimi následuje bit, který indikuje, zda se volba může měnit během průchodu sítí. Konkrétní hodnoty najdete v obrázku 2.5.



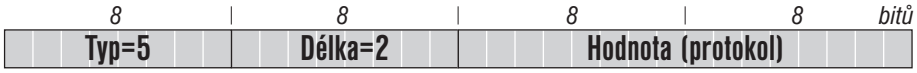
Obrázek 2.5: Tvar voleb pro rozšiřující hlavičky

upozornění směrovače

Jednou z „opravdových“ voleb je tak zvané *Upozornění směrovače (router alert)* definované v RFC 2711. Jedná se o volbu pro všechny, která má za cíl upozornit každý směrovač po cestě, že tento paket nese data, která by jej mohla zajímat.

Volba najde uplatnění například v rezervačním protokolu RSVP, který posílá řídicí pakety pro alokaci kapacit po cestě. Tyto pakety jsou určeny všem směrovačům. Právě *Upozornění směrovače* může napovědět, že paket nese zajímavou informaci. Bez něj by směrovač musel prohlížet všechny datagramy a zkoumat, jakému protokolu vyšší vrstvy patří. Když by narazil na RSVP paket, zabýval by se jím podrobněji. V opačném případě by jej poslal dále po cestě k cíli.

Díky *Upozornění směrovače* lze rychle odlišit datagramy potenciálně zajímavé od těch, které se mají prostě předávat dál. Formát volby najdete na obrázku 2.6. Obsahuje vlastně jedinou položku, která slouží k identifikaci protokolu, jehož data nese. Dosud definované hodnoty shrnuje tabulka 2.4.



Obrázek 2.6: Volba *Upozornění směrovače*

Aby tato volba přinášela nějaký efekt, musí odpovídající protokol nařizovat její použití. Směrovač má právo ignorovat obsah všech datagramů, které nejsou adresovány jemu a neobsahují *Upozornění směrovače*. Chce-li určitý protokol získat jeho pozornost, musí k datagramu přihodit tuto volbu.

2.4 Směrování

Standardně je datagram směrován podle své cílové adresy. Hlavička *Směrování (Routing)* umožňuje do tohoto procesu zasáhnout a předepsat jeden

Hodnota	*Význam*
0	obsahuje MLD zprávu
1	obsahuje RSVP zprávu
2	obsahuje zprávu *Aktivní sítě*
3	rezervováno
4–35	úroveň vnoření agregovaných rezervací (RFC 3175)
36–67	úroveň agregací QoS NSLP (RFC 5974)
68	NSIS NATFW NSLP (RFC 5973)

Tabulka 2.4: Definované *Hodnoty* pro volbu *Upozornění směrovače*

či několik bodů (IPv6 adres), jimiž musí datagram projít před doručením adresátovi. Motivace pro takové chování jsou různé, jak zanedlouho uvidíte.

IPv6 ponechává prostor pro zavedení různých typů směrovacích hlaviček. K jejich rozlišení slouží hodnota položky *Typ směrování*. Zatím byly definovány dva přesně popsané typy (0 a 2) a dva volné typy (hodnoty 253 a 254) určené pro experimentování se směrovacími mechanismy. Další informace o experimentálních typech najdete v RFC 4727, zde si jimi nebudu zabývat.

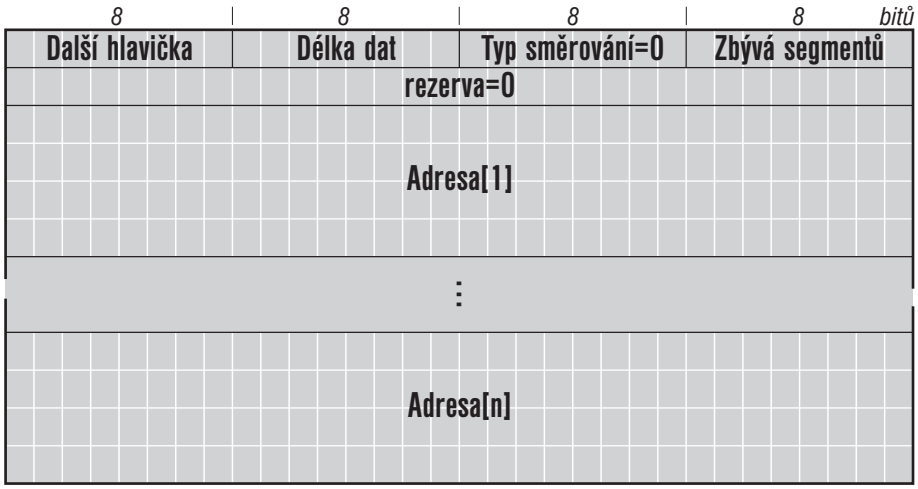
směrování typu 0

Typ 0 je starší, byl zaveden přímo v RFC 2460 jako součást definice jádra IPv6. Umožňuje předepsat datagramu určité body, kterými musí v daném pořadí projít. Zároveň slouží jako záznam, kterými z nich již prošel. Tyto „průchozí body“ nemusí následovat bezprostředně za sebou, mezi každými dvěma může datagram projít libovolným počtem směrovačů. Podobnou rozšiřující volbu nabízí i IPv4.

Formát hlavičky *Směrování* typu 0 představuje obrázek 2.7. Pokud chce odesilatel, aby jeho datagram po cestě k cíli prošel určitými uzly, uvede jako jeho cílovou adresu IP adresu prvního z těchto průchozích uzlů. Do hlavičky *Směrování* pak zapíše postupně adresy zbývajících a na závěr konečný cíl datagramu. V položce *Zbývá segmentů* uvede jejich počet.

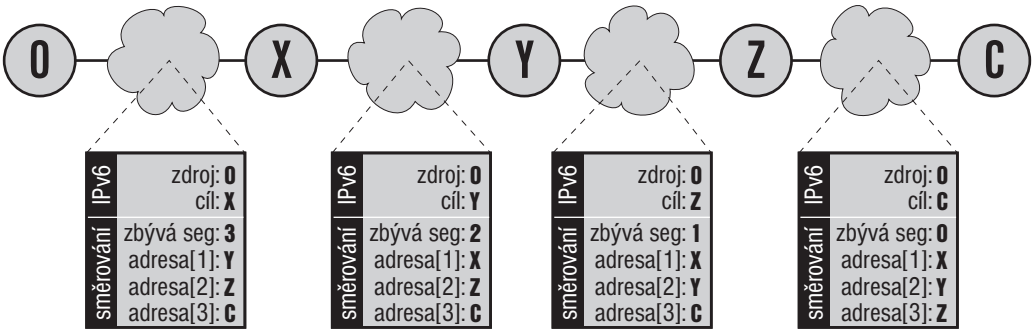
Když datagram dorazí na cílovou adresu uvedenou v základní IPv6 hlavičce a obsahuje hlavičku *Směrování* s nenulovým počtem zbývajících segmentů, vezme počet zbývajících segmentů jako index do tabulky průchozích adres. Určuje, kolikátá od konce je adresa následujícího průchozího bodu. Tuto adresu zapíše jako cílovou do základní hlavičky a dosavadní cílovou (tedy svoji) uloží místo ní do směrovací hlavičky. Následně zmenší počet zbývajících segmentů o jedničku a pošle datagram novému cíli (dalšímu na vytyčené cestě).

Položka *Zbývá segmentů* tedy odděluje, které z uvedených adres již byly navštíveny a kterými datagram ještě musí projít. Je-li nulová, znamená to, že datagram dorazil do svého cíle.



Obrázek 2.7: Rozšiřující hlavička *Směrování* typu 0

Obrázek 2.8 ilustruje vývoj hodnot zajímavých položek při průchodu datagramu sítí. Jeho odesilatelem je *O* a koncovým adresátem *C*. Odesilatel předepsal, že datagram má projít uzly *X*, *Y* a *Z*.



Obrázek 2.8: Změny v hlavičkách datagramu

Směrovací hlavička typu 0 byla zavedena především pro testování dosažitelnosti mezi libovolnými adresami. Můžete v ní předepsat, odkud kam se mají datagramy dopravit a tedy ověřit funkčnost spojení. Mimo jiné jí to dává schopnost projít NATem na neveřejnou adresu – jednoduše lze uvést jako koncový cíl neveřejnou adresu uzlu a jako mezilehlou veřejnou adresu jeho NATu. To může být přínosné pro některé aplikace, zároveň je však

tato vlastnost vnímána jako nebezpečná[1]. Podobně může přispět k ošizení firewallu a proniknutí jeho ochranou.

V roce 2007 byl široce diskutován jiný problém, který nakonec vedl k odmítnutí směrovací hlavičky typu 0. Může totiž posloužit k útokům usilujícím o zahlcení přenosových tras. Útočník díky ní může nechat přepravovat datagramy sítí sem a tam. A když navíc použije několik směrovacích hlaviček napěchovaných po okraj, může se datagram potulovat sítí velmi dlouho. Série takových datagramů dokáže v síti vytvořit datové toky s objemem mnohonásobně převyšujícím kapacitu útočníkova připojení[2], navíc i na velmi dlouhé trase.

Důsledkem bylo RFC 5095: *Deprecation of Type 0 Routing Headers in IPv6*. Podle něj je cílový IPv6 uzel, který obdržel datagram s hlavičkou *Směrování* typu 0, povinen ji ignorovat, pokud je počet zbývajících segmentů nulový. Je-li nenulový, musí datagram zahodit a ohlásit jej odesilateli jako chybný. Kromě toho se zde doporučuje datagramy s tímto typem hlavičky *Směrování* filtrovat na aktivních prvcích.

směrování typu 2

Typ 2 byl definován speciálně pro mobilitu. De facto se jedná o silné zjednodušení obecnějšího typu 0. Když je mobilní uzel na cestách, má kromě své původní pevné adresy i adresu dočasnou, jež se mění podle sítě, ve které se právě nachází. Pokud přechází mezi buňkami, může se dočasná adresa během komunikace měnit. Aby nebyla narušena komunikace běžících programů, používá pro ni svou trvalou, tak zvanou domácí adresu.

Jeho partner pomocí směrovací hlavičky typu 2 stanoví, že koncovou adresou je pevná adresa mobilního uzlu, ale má se nejprve dopravit na jeho dočasnou adresu. Čili datagram je dopraven na aktuální dočasnou adresu, tam se postupem podobným směrování typu 0 nahradí cílová adresa hodnotou ze směrovací hlavičky a vyšším komunikačním vrstvám se data doručí, jako by přišla na trvalou adresu.

Směrovací hlavička typu 2 proto umožňuje uložit jen jedinou adresu (domácí adresu mobilního uzlu, jemuž je datagram určen). To výrazně omezuje její zneužitelnost. Formát této směrovací hlavičky najdete na obrázku 11.16 na straně 239 v kapitole o mobilitě, kde se dočtete i podrobnější informace o jejím fungování.

2.5 Fragmentace

Každá z podřízených technologií, které IPv6 používá pro přepravu svých datagramů, má jistou maximální velikost paketů, které dokáže přenášet. Tato konstanta se označuje zkratkou MTU (Maximum Transmission Unit). Například nejpopulárnější Ethernet má MTU = 1500 B.

[1] Stále se sice zdůrazňuje, že NAT *není* bezpečnostní prvek, ale stejně je tak částečně vnímán.
[2] Prakticky byl předveden 88násobek.

Cílem fragmentace je umožnit IPv6 přepravovat datagramy větší, než je MTU používaných technologií. Základní myšlenka je prostá: odesilatel rozloží datagram do několika dostatečně malých částí a příjemce z nich poskládá původní datagram.

odlišnosti IPv4

Analogickou techniku používal i protokol IPv4, lišil se však v několika důležitých detailech. Zatímco v IPv4 mohl datagram fragmentovat libovolný směrovač po cestě (kdykoli měl být odeslán linkou, jejíž MTU bylo menší než velikost datagramu), *v IPv6 fragmentuje výlučně odesilatel*. Pokud má některý ze směrovačů odeslat datagram linkou s nedostačujícím MTU, zahodí jej a pošle odesilateli ICMP zprávu „příliš velký paket“, jejíž součástí je i MTU, které tento stav způsobilo. Druhou odlišností je, že zatímco IPv4 má všechny podklady pro fragmentaci zařazeny již do standardní hlavičky, IPv6 pro ni používá hlavičku rozšiřující a spíše se snaží, aby k fragmentaci vůbec nedocházelo.

formát hlavičky

Rozšiřující hlavička *Fragmentace (Fragment)* je identifikována kódem 44 v položce *Další hlavička* svého bezprostředního předchůdce. Její tvar vidíte na obrázku 2.9. Velikost je konstantní a kromě obvyklé *Další hlavičky* obsahuje tři informační položky.

Obrázek 2.9: Rozšiřující hlavička *Fragmentace*

Identifikace (Identification) slouží k rozpoznání, které fragmenty patří k sobě. Jedná se o 32bitové celé číslo, které je v rámci dané dvojice odesilatel–příjemce pokud možno jednoznačné (každý další fragmentovaný datagram má číslo o jedničku vyšší než předchozí, po naplnění kapacity čítače se začne znovu od nuly). *Posun fragmentu (Fragment offset)* říká, kam tento fragment patří. Jednotkou jsou osmice bajtů od začátku fragmentovatelné části původního datagramu (viz níže). A konečně příznak *M (More fragments)* signalizuje, zda je tento fragment poslední (hodnota 0) nebo za ním následuje další (hodnota 1).

fragmentovatelná a nefragmentovatelná část

Má-li dojít k fragmentaci, vymezí se v původním datagramu dvě části: na začátku je tak zvaná *nefragmentovatelná část*, kterou tvoří standardní IPv6 hlavička a všechny po ní následující rozšiřující hlavičky až po *Směrování* (včetně). Tedy vše, co v pořadí rozšiřujících hlaviček předchází před fragmentací. Zbytek datagramu je považován za *fragmentovatelnou část* a pouze on je předmětem fragmentace.

postup fragmentace

Tato fragmentovatelná část se rozdělí na úseky, jejichž velikost je násobkem osmi bajtů a je dostatečně malá na to, aby celková velikost výsledných

datagramů nepřekročila požadované MTU. Tím z původního datagramu vznikne několik fragmentů – nových datagramů. Jejich hlavičky jsou sestaveny takto:

- Převezme se nefragmentovatelná část původního datagramu. Jedinými změnami, které se v ní pro jednotlivé fragmenty provedou, je úprava velikosti v základní hlavičce, aby odpovídala skutečné velikosti fragmentu, a změna hodnoty poslední *Další hlavičky* na 44.
- Za ni se přidá rozšiřující hlavička *Fragmentace*, jejíž hodnoty se naplní následovně:
  - vygeneruje se nový *Identifikátor* paketu a tato hodnota se přidělí všem jeho fragmentům
  - hodnota *Další hlavičky* se převezme z poslední *Další hlavičky* nefragmentovatelné části původního datagramu
  - *Posun* každého fragmentu se určí jako počet osmic bajtů, o které je jeho začátek vzdálen od začátku fragmentovatelné části původního datagramu; jelikož všechny fragmenty (kromě posledního) budou mít stejnou délku x, bude mít první fragment *Posun* nulový, druhý fragment ponese *Posun*=x, třetí *Posun*=$2x$ atd.
  - poslednímu fragmentu se příznak *M* nastaví na 0, ostatním na 1
- Na konec se připojí dotyčný fragment (úsek fragmentovatelné části původního datagramu).

Vzniklé fragmenty jsou jako samostatné datagramy odeslány adresátovi. Ten je posbírá a z údajů ve fragmentační hlavičce dokáže složit původní datagram: podle *Identifikátoru* pozná, které fragmenty patří k sobě, pomocí *Posunutí* určí správné pořadí a v kombinaci s *Délkou dat* zjistí případné chybějící části a konečně příznak *M* mu prozradí, zda má k dispozici všechny kousky.

Na základě těchto údajů příjemce poskládá původní datagram do podoby, kterou měl před fragmentací (tím zaniknou hlavičky *Fragmentace* jednotlivých částí) a ten pak dále zpracovává bez ohledu na to, že mu přišel po kouskách.

2.6 Velikost datagramů

Fragmentace těsně souvisí s velikostí odesílaných datagramů. Každý datagram navíc přináší určitou (byť malou) zátěž – musí mít své hlavičky, směrovače po cestě se musí rozhodovat, kudy jej poslat, a podobně. Ideálem je, aby datagramy byly pokud možno co největší, aby jich bylo co nejméně a snižovala se tak nadbytečná zátěž. Na druhé straně však datagramy musí

být natolik malé, aby nikde po své cestě nepřekročily MTU a nedocházelo tudíž k fragmentaci.

objevování MTU cesty

O dosažení tohoto kompromisu se snaží algoritmus nazvaný objevování MTU cesty. Definuje jej RFC 1981: *Path MTU Discovery for IP version 6*.

Z pohledu teoretika nemá vůbec smysl mluvit o nějakých cestách v souvislosti s protokolem IP. Nabízí službu bez spojení, kdy je každý datagram směrován samostatně a nezávisle na ostatních. To znamená, že každý ze skupiny datagramů tvořících jeden soubor může dorazit k cíli jinou cestou. V praxi se však směrovací tabulky nemění příliš rychle a je vysoce pravděpodobné, že datagramy odeslané v krátkém časovém intervalu ke stejnému cíli budou putovat stejnou trasou. Na tomto pozorování ostatně stojí již letitý program *traceroute*.

Objevování MTU cesty má za cíl najít maximální velikost paketu, který lze poslat danému cíli. Postupuje jednoduše: nejprve pošle datagram, jehož velikost je rovna MTU rozhraní, kterým datagram odesílá. Celkové MTU jistě nemůže být větší. Pokud datagram úspěšně dojde, máme nalezeno MTU cesty.

Jestliže někde narazí na úsek s menším MTU, směrovač na jeho začátku datagram zahodí a pošle odesilateli ICMP zprávu „příliš velký datagram“. Její součástí je i hodnota MTU dotyčné linky. Odesilatel si příslušně zmenší svůj odhad MTU cesty a zkusí štěstí znovu s datagramem této velikosti. Celý proces se opakuje tak dlouho, dokud se datagramy nedostanou až k cíli.

Pokud komunikace trvá delší dobu, s vysokou pravděpodobností dojde ke změně cesty, případně i několikanásobné. Hledání MTU se snaží s touto skutečností vyrovnat. Pokud MTU cesty poklesne, odesilatel na to přijde hned – obdrží ICMP zprávu o příliš velkém datagramu. O případném zvětšení se však touto cestou nedozví. Proto by měl čas od času zopakovat celý algoritmus hledání MTU, aby zjistil, zda aktuální hodnota není vyšší, než se domnívá. V RFC se požaduje, aby interval mezi těmito zkouškami byl minimálně 5 minut, doporučená hodnota je 10 minut.

Ostatně vzhledem k tomu, že MTU na linkách podporujících IPv6 má být alespoň 1280 B a doporučuje se používat 1500 B nebo více, lze očekávat, že MTU cesty bude zpravidla 1500 B a prakticky se nebude měnit. Klient nesmí zmenšit MTU cesty pod 1280 B. Pokud mu někdo ohlásí nižší hodnotu, musí datagramy fragmentovat.

Objevování MTU cesty lze používat i pro skupinové adresy. V tomto případě může dostat na jeden datagram celou řadu ICMP zpráv. Bude se chovat podle očekávání – použije nejmenší ohlášenou hodnotu.

Implementace popsaného algoritmu je autory IPv6 důrazně doporučena, není však povinná. Jedná-li se o minimalistickou implementaci IPv6 (např.

v ROM přenosného zařízení), může používat hodnotu 1280 B, aniž by se pokoušela zjistit, zda skutečné MTU cesty není vyšší.

2.7 Jumbogramy

Jelikož je délka nesených dat v IPv6 datagramu ukládána do 16bitové položky, je maximální dosažitelnou hodnotou 65 535 bajtů. Troufám si tvrdit, že případy, kdy by tento horní limit byl pociťován jako omezení, budou opravdu velmi velmi vzácné. Nicméně i pro ně nabízí IPv6 řešení. Jedná se o volbu *Jumbo obsah (Jumbo payload)*, která umožňuje vytvářet datagramy o délce 65 536 až 4 294 967 295 B. Patří mezi *Volby pro všechny*, takže se jí bude zabývat každý směrovač po trase.

Použití je prosté: *Délka dat* v základní hlavičce se vynuluje a přidá se rozšiřující hlavička s volbami pro všechny obsahující *Jumbo obsah*. Nese položku *Délka jumbo dat (Jumbo payload length)*, která měří 32 bitů a umožňuje proto výše uvedený rozsah přípustných hodnot. Takto velké datagramy jsou označovány jako jumbogramy.

Obrázek 2.10: Volba *Jumbo obsah*

Použití jumbogramů má pochopitelně smysl jen v případě, kdy linková technologie umožňuje přenos takto velkých paketů. Jinými slovy pokud MTU dotyčné linky přesahuje 65 575 (maximální velikost nesených dat plus IPv6 hlavička). Uzly, které nemají tak velké MTU, nemusí jumbogramy podporovat a ani této volbě rozumět.

UDP Příliš velké datagramy ale vadí i protokolům vyšší vrstvy. Například UDP má svou vlastní položku pro délku dat, která je také 16bitová. RFC 2675, které definuje jumbogramy, proto doporučuje, aby na strojích s jejich podporou byl pozměněn kód i ve vyšších vrstvách. Konkrétně pro UDP doporučuje, aby se u jumbogramů uváděla na úrovni UDP nulová délka a aby si kód pro UDP nechal sdělit skutečnou délku od IP vrstvy.

TCP TCP sice nemá ve svých hlavičkách délku, ale definuje volbu *Maximální délka segmentu (Maximum Segment Size, MSS)*, která – jak jinak – používá 16bitovou hodnotu. Doporučenou strategií je prohlásit 65 535 za nekonečno. Pokud jeden z partnerů dostane MSS s touto hodnotou, určí si skutečnou maximální délku segmentu z nalezeného MTU cesty (odečtením 60 B na IPv6 a TCP hlavičky).

Druhým délkovým údajem v TCP je délka urgentních dat. Autoři RFC 2675 považují za nepravděpodobné, že by se urgentní data používala v kombi-

naci s jumbogramy. Kdyby k tomu však přece jen došlo, doporučují i zde prohlásit 65 535 za nekonečno. Tato hodnota v položce *Urgent pointer* TCP hlavičky znamená „všechna data v tomto datagramu jsou urgentní“. Při odesílání TCP paketu s dlouhou urgentní částí je třeba jej rozdělit na dva tak, aby ve druhém byla délka urgentní části menší než 65 535.

Upřímně řečeno považuji jumbogramy spíše za zajímavou teoretickou konstrukci než za prakticky použitelný nástroj. MTU tak velká, aby ji umožňovala použít, se v současném Internetu nevyskytují.

2.8 Rychlý start

Rozšiřující hlavička *Rychlý start (Quickstart)* byla přidána experimentálním RFC 4782: *Quick-Start for TCP and IP*. Jeho cílem je zvýšit propustnost transportních protokolů, především TCP. Stroj zahajující komunikaci přidá do žádosti o navázání TCP spojení tuto hlavičku, v níž vyznačí přenosovou rychlost, jakou by rád používal.

Jedná se o volbu pro všechny, hlavičkou se tedy zabývají všechny směrovače po cestě a pokud některý z nich považuje navrženou přenosovou rychlost za příliš vysokou, sníží hodnotu na akceptovatelnou úroveň. Při příchodu do cílového stroje tedy hlavička obsahuje rychlost přijatelnou pro všechna zařízení na cestě mezi odesilatelem a příjemcem. Během komunikace je pochopitelně tato informace čas od času aktualizována.

Vzhledem k tomu, že dotyčný protokol je experimentální a s vlastním IPv6 souvisí jen volně, nebudu mu zde věnovat větší pozornost.

2.9 Toky

Jedním z nových prvků IPv6 je koncepce toku. Idea je jasná: tok je proud datagramů, které spolu „nějak souvisí“. Často tok odpovídá transportnímu spojení (například TCP spojení mezi WWW klientem a serverem či IP telefonní hovor mohou být dobrými kandidáty pro tok), ale nemusí tomu tak nutně být.

Přestože se termín ve světě IPv4 nepoužívá, analogie toků zde existuje. Obvykle bývají identifikovány pěticí údajů:

- zdrojová IP adresa
- zdrojový port
- cílová IP adresa
- cílový port
- transportní protokol

Pokud jste někdy konfigurovali firewall, jistě vám tahle pětka je důvěrně známá. Typickým příkladem uplatnění de facto toku je stavový firewall, který povolí otevřít TCP spojení jen v jednom směru. Jakmile se tak stane, uloží si pětici uvedených údajů do paměti a po určitou dobu propouští datagramy s těmito hodnotami, protože je považuje za součást otevřeného spojení (čili toku).

Problém je, že tři z pěti údajů patří do transportní vrstvy a nemusí být snadno dostupné. Dojde-li k fragmentaci datagramu, jsou transportní údaje obsaženy jen v prvním fragmentu. Při utajení pomocí hlavičky ESP se k nim směrovače po cestě nedostanou vůbec, protože jsou zašifrovány a z principu věci je dešifrovat umí jen příjemce. Nebo sice jsou dostupné, ale cesta k nim vede dlouhou sekvencí rozšiřujících hlaviček a zbytečně zpracující zařízení zdržuje.

Proto se objevil koncept toků, který má pomoci identifikovat související datagramy snadno a rychle, jen pomocí údajů ze základní IP hlavičky. Výše zmíněnou pětici má nahradit trojice:

- zdrojová IPv6 adresa
- cílová IPv6 adresa
- značka toku

Problematika toků je dosud živá. Původní RFC 2460 ji neřeší vůbec, odkládá definici na později. První krok na cestě k funkčním tokům učinilo RFC 3697: *IPv6 Flow Label Specification*, které definovalo pravidla pro zacházení se značkami toků v datagramech. V době vzniku tohoto textu již stojí ve frontě na standardizaci RFC, které by je mělo nahradit a postrčit specifikaci toků zase o něco dál. Text vychází z jeho návrhu (*draft-ietf-6man-flow-3697bis-07*), který už se pravděpodobně měnit nebude.

Přidělení značky toku má na starosti výlučně odesilatel datagramu, po cestě se značka nesmí měnit[3]. Svou vlastní značku typicky dostane každý datový tok se stejnou pěticí základních identifikačních údajů, již jsem zmínil výše. Nicméně není to předepsáno pevně, rozhodnutí je na odesilateli.

Hodnota značky nemá žádnou strukturu ani význam. Slouží čistě jako identifikátor. Pokud odesilatel nechce své datagramy značkovat, vloží do položky *Značka toku* nulu, která signalizuje, že paket není značkován.

Specifikace požaduje, aby hodnoty značek byly rovnoměrně rozděleny v celém dostupném prostoru a aby se nedaly předem odhadnout. Důvodem těchto požadavků je snaha o jejich snadnou použitelnost při hashování a omezení bezpečnostních rizik. Jako vhodné generátory značek dokument

[3] Jedinou výjimkou jsou bezpečnostní důvody. Pokud by některý ze směrujících strojů dospěl k závěru, že se někdo snaží zneužít značky k vytvoření tajného informačního kanálu, smí do nich zasáhnout.

zmiňuje hashovací funkci nebo generátor pseudonáhodných čísel. Naopak výslovně nedoporučuje sekvenční přiřazování, kdy každá další značka je o jedničku větší, než poslední použitá.

Způsob využití při přepravě není pevně definován. Existují v zásadě dvě cesty: může být bezstavový, kdy si přepravující prvky neukládají žádné informace, jež by při doručování značkovaných datagramů využívaly, či stavový, který se právě o takové informace opírá. Návrh dává přednost bezstavové variantě, zatímco o stavové se zmiňuje jen okrajově.

Podpora toků není povinná. Průchozí zařízení může brát na tok zřetel, nebo nemusí. V tom případě však musí informace související s tokem ignorovat a nijak do nich nezasahovat. Tím je zajištěno, že nic nepokazí strojům, které jsou za ním a věci rozumějí.

Navzdory ubíhajícím létům stále nelze definici značek prohlásit za dokončenou a jejich praktické využívání zůstává v nedohlednu.

3 Adresy v IPv6

Rychle se tenčící adresní prostor byl jedním z hlavních hnacích motorů vzniku IPv6. Přestože navržený protokol má i řadu jiných zajímavých vlastností, dodnes je košatost jeho adresního prostoru považována za klíčovou přednost a s krátící se zásobou IPv4 adres nabývá na naléhavosti. Podívejme se na ni podrobněji.

Základním dokumentem pro definici adres je RFC 4291: *IP Version 6 Addressing Architecture* určující jejich délku a podobu, typy adres a další koncepční prvky. Je doplněn několika dalšími dokumenty popisujícími podrobněji vybrané části adresního prostoru.

3.1 Jak se adresuje

V IPv6 – stejně jako u jeho předchůdce – jsou adresy přiřazovány síťovým rozhraním, nikoli počítačům. Má-li váš počítač dvě síťové karty, bude mít každá z nich svou adresu. Přesněji řečeno své adresy. Později uvidíte, že IPv6 s adresami pro rozhraní nikterak neskrblí.

druhy adres

Existují tři druhy adres s odlišným chováním:

Individuální (unicast) jsou staré známé krotké adresy. Každá z nich identifikuje jedno síťové rozhraní a data mají být dopravena právě jemu.

Skupinové (multicast) slouží pro adresování skupin počítačů či jiných zařízení. Pokud někdo odešle data na tuto adresu, musí být doručena všem členům skupiny.

Výběrové (anycast) představují novinku a nejzajímavější přírůstek v IPv6. Také výběrové adresy označují skupinu, data se však doručí jen jedinému jejímu členovi – tomu, který je nejblíže.

Porovnání s IPv4 ukazuje, že zmizely všesměrové (broadcast) adresy. Nejsou potřeba, protože jejich funkce přebírají adresy skupinové. Jsou definovány speciální skupiny, např. pro všechny uzly na dané lince, které umožňují plošnou distribuci zpráv.

adresy rozhraní

IPv6 umožňuje, aby rozhraní mělo libovolný počet adres různých druhů. Ba dokonce přikazuje několik povinných adres, které musí být přiděleny (viz část 3.9 na straně 77). Stejně jako v IPv4 se předpokládá, že všechny počítače v jedné fyzické síti (např. na jednom Ethernetu) budou náležet do stejné podsítě a budou tudíž mít společný prefix podsítě.

3.2 Podoba a zápis adresy

Při rozhodování o velikosti adresy pro IPv6 se autoři řídili heslem „aby nám už nikdy nedošly“. Frustrace způsobená nedostatkem IPv4 adres byla velmi silná. Proto se rozhodli délku prodloužit na čtyřnásobek, adresa v IPv6 tedy měří 128 bitů.

zápis adres

Standardním způsobem jejího zápisu je osm skupin po čtyřech číslicích šestnáctkové soustavy, které vyjadřují hodnoty 16 bitů dlouhých částí adresy. Navzájem se oddělují dvojtečkami. Příkladem IPv6 adresy je

fedc:ba98:7654:3210:fedc:ba98:7654:3210

Upřímně řečeno se očekává, že uživatelé budou striktně používat DNS a ručního psaní uvedených hrůz budou ušetřeni. Černý Petr zbude v rukou správců sítí, kteří se jim při sebevětším úsilí nevyhnou...

zkracování

Jelikož je poměrně častou hodnotou nula, nabízí se dvě možnosti pro zkrácení zápisu. Jednak v každé čtveřici můžete vynechat počáteční nuly. Místo „0000“ tedy lze psát jen „0“. Někdy se dokonce vyskytuje několik nulových skupin za sebou. Ty můžete nahradit zápisem „::“ (dvě dvojtečky). Například adresu

0123:0000:0000:0000:fedc:ba98:7654:3210

můžete zkrátit na

123:0:0:0:fedc:ba98:7654:3210

nebo dokonce jen na

123::fedc:ba98:7654:3210

Koncovou nulu (v poslední čtveřici) pochopitelně vynechat nelze. Kdybyste napsali jen „321“, znamenalo by to „0321“, nikoli „3210“. Úplný extrém představuje nedefinovaná adresa

0000:0000:0000:0000:0000:0000:0000:0000

kterou lze zkrátit až na samotné

::

Konstrukci „::“ můžete v každé adrese použít jen jednou. Jinak by nebylo jednoznačné, jak se má adresa rozvinout do původní podoby. Například adresu

0123:0000:0000:0000:4567:0000:0000:0000

můžete psát jako

123::4567:0:0:0 nebo 123:0:0:0:4567::

nikoli však

123::4567::

kanonický zápis

Velká variabilita v zápisu adres komplikuje jejich porovnávání. Výše vidíte několik příkladů výrazně odlišných zápisů stejné adresy, navíc mohou situaci ještě komplikovat malá/velká písmena a pro lidského čtenáře v některých písmech potenciálně zaměnitelné znaky „B“ a „8“ či „D“ a „0“.

RFC 5952: *A Recommendation for IPv6 Address Text Representation* proto definovalo kanonický zápis, jehož cílem je učinit psanou podobu adresy jednoznačnou. Dokument zdůrazňuje, že aplikace musí podporovat všechny přípustné podoby adresy, ale ve svých výstupech, jako jsou výpisy či hodnoty v konfiguračních dialozích, by měly používat kanonický tvar. Pravidla pro jeho vytvoření jsou následující:

- Šestnáctkové číslice reprezentované písmeny se píší vždy malými znaky[1].
- Vynechání počátečních nul ve čtveřici je povinné.
- Konstrukce „::“ musí být použita tak, aby měla největší možný efekt. Musí pohltit všechny vzájemně sousedící nulové skupiny (není povoleno „:0::“ ani „::0:“) a musí být použita pro nejdelší sekvenci nulových skupin v adrese. Má-li shodnou maximální délku několik skupin, použije se „::“ pro první z nich. Není povoleno ji použít pro jedinou nulovou skupinu, ta vždy zůstane jako jednoduchá nula.

Kanonický tvar výše uvedené adresy je 123::4567:0:0:0 a software by ji vždy měl vypisovat v této podobě.

adresy s vloženým IPv4

Některé přechodové mechanismy potřebují vyjádřit adresy, které pocházejí ze světa IPv4. Aktuálně se k tomuto účelu používá formát zavedený v RFC 6052: *IPv6 Addressing of IPv4/IPv6 Translators* a nazvaný *adresy s vloženým IPv4 (IPv4-embedded)*. Využívají skutečnosti, že adresní prostor IPv4 je mnohem menší, proto lze vyčlenit část IPv6 prostoru a použít ji pro reprezentaci IPv4. Tato část je identifikována určitým prefixem a mapovaná adresa vznikne jednoduše tak, že se za prefix připojí IPv4 adresa.

Obecnou strukturu IPv6 adresy s vloženou IPv4 adresou znázorňuje obrázek 3.1. Začíná prefixem o délce nanejvýš 96 bitů, za nímž následuje IPv4 adresa a případně přípona. Pomocí přípony lze v rámci jedné mapované

[1] Toto pravidlo bylo nahlášeno jako chybné, protože se tradičně používala velká písmena. O osudu „chyby“ dosud nebylo rozhodnuto, ale nedovedu si představit, že by byla akceptována a pravidlo se změnilo. Argumentace je dost fundamentalistická, opírá se o literaturu starší než 20 let a odporuje současné praxi.

Obrázek 3.1: Obecná struktura IPv6 adresy s vloženou IPv4 adresou

adresy identifikovat jednotlivé části. RFC 6052 však zároveň doporučuje přípony vypustit a používat jen prefixy délky 96 b.

Prefix může být dvou typů – buď se jedná o místní prefix, který přidělí správce sítě z lokálního adresního prostoru. Může například vyčlenit pro tento účel část jedné z podsítí. Druhou, pravděpodobně častější variantou bude použití univerzálního prefixu (well known prefix)

64:ff9b::/96

definovaného pro tyto účely přímo v RFC 6052. Pouze při jeho použití můžete vloženou IPv4 adresu zapsat ve standardním tvaru – v desítkové soustavě s bajty oddělenými tečkami. Adresu s univerzálním prefixem obsahující 10.1.2.3 lze tedy zapsat ve tvaru 64:ff9b::10.1.2.3 nebo 64:ff9b::a01:203. Zatímco při mapování pomocí lokálního prefixu je přípustný jen druhý tvar, například 2001:db8:ff:ee::a01:203.

Obrázek 3.2: IPv6 adresa s vloženou IPv4 s univerzálním prefixem

IPv4-mapované adresy

Adresy s vloženým IPv4 nahradily předchozí formát, tak zvané IPv4-mapované adresy, jejichž počátečních 80 bitů obsahuje samé nuly, následuje 16 bitů jedničkových a v posledních 32 bitech je zapsána IPv4 adresa. Například adresu 147.230.49.73 bychom tímto způsobem vyjádřili jako

::ffff:93e6:3149

Opět je přípustné psát IPv4 adresu v obvyklé podobě, takže tutéž adresu lze psát i komfortněji:

::ffff:147.230.49.73

IPv4-kompatibilní adresy

Ještě starší specifikace definovaly také IPv4-kompatibilní adresy, které měly počátečních 96 bitů nulových a za nimi následovalo 32 bitů s IPv4 adresou. Už v nich byl k dispozici pohodlný zápis, takže adresa 147.230.49.73 zapsaná jako IPv4-kompatibilní IPv6 adresa má podobu:

::147.230.49.73

Poslední dva formáty uvádím pouze proto, že na ně ve starších textech můžete ještě narazit. Nicméně oba už jsou překonány: IPv4-kompatibilní adresy byly odmítnuty v RFC 4291 ve prospěch IPv4-mapovaných, jež pak v RFC 6052 podlehly adresám s vloženým IPv4.

prefixy Příslušnost k určité síti nebo podsíti se vyjadřuje prefixem – všechna rozhraní v jedné síti mají stejný prefix (začátek adresy). Jeho délka může být různá. Záleží na tom, s jakou podrobností se na adresy díváte. Může vás zajímat jen prefix poskytovatele Internetu (který bude poměrně krátký) nebo o poznání delší prefix určité konkrétní podsítě.

Tento přístup se používá již v současném Internetu pod názvem *Classless Inter-Domain Routing (CIDR)*. Z něj je také převzat způsob, kterým se prefixy zapisují:

IPv6_adresa/*délka_prefixu*

Délka_prefixu určuje, kolik bitů od začátku adresy je považováno za prefix. Například 60 bitů dlouhý prefix 12ab 0000 0000 cd3 lze zapsat několika možnými způsoby:

12ab:0:0:cd30:0:0:0:0/60
12ab::cd30:0:0:0:0/60
12ab:0:0:cd30::/60

Nejvhodnější je poslední z nich, protože odpovídá kanonickému tvaru a navíc konstrukcí „::“ logicky nahrazuje závěrečnou část adresy, která je z pohledu prefixu nezajímavá. Povšimněte si, že do prefixu nepatří ani závěrečná nula ve skupině cd30, protože při délce 60 bitů do prefixu z této skupiny patří jen 12 bitů, čili první tři šestnáctkové číslice. Tuto nulu však nelze vynechat. Kdybychom to udělali, byla by příslušná skupina interpretována jako 0cd3 a zápisem 12ab:0:0:cd3::/60 bychom ve skutečnosti vyjádřili prefix 12ab 0000 0000 0cd, což je krajně matoucí.

Prefix pochopitelně nemusí končit na hranici šestnáctkových číslic. Například prefix 2000::/3 požaduje, aby první tři bity adresy obsahovaly hodnotu 001 (binárně). Tomu vyhoví všechny IPv6 adresy, jejichž první číslicí je 2 nebo 3.

Ve zkratce lze použít i zápis, který současně oznamuje jak konkrétní adresu rozhraní, tak délku prefixu (a tudíž adresu podsítě):

12ab:0:0:cd30:123:4567:89ab:cdef/64

3.3 Rozdělení aneb typy adres

Obrovský adresní prostor, který má IPv6 k dispozici, byl rozdělen do několika skupin – typů adres. Každý typ sdružuje adresy se společnou charakteristikou. Příslušnost k jednotlivým typům určuje prefix adresy. Dříve

se pro tyto určující počáteční bity používal termín *prefix formátu (format prefix, FP)*, novější dokumenty však od tohoto pojmu upouští.

prexif	*význam*
::/128	nedefinovaná adresa
::1/128	smyčka (loopback)
fc00::/7	unikátní individuální lokální
fe80::/10	individuální lokální linkové
ff00::/8	skupinové adresy
ostatní	individuální globální

Tabulka 3.1: Základní rozvržení adres

Základní rozdělení uvádí tabulka 3.1. Jak je vidět, drtivou většinu zabírají globální (celosvětově jednoznačné) individuální adresy. Z jejich prostoru je navíc většina prefixů dosud nepřiřazena, zatím se využívá pouze výše zmiňovaný prefix 2000::/3. Ostatní se ponechávají jako rezerva a očekává se, že budoucí RFC jim přiřknou určitý význam a vnitřní strukturu. Aktuální stav jejich přidělení najdete na adrese

www▸ *http://www.iana.org/assignments/ipv6-address-space*

Skupinové adresy jsou snadno identifikovatelné, protože jejich první bajt má v šestnáctkovém zápisu hodnotu ff. Naproti tomu výběrové adresy nemají přiřazeno žádné speciální rozmezí a přidělují se ze stejného prostoru, jako adresy individuální.

Několika menším oblastem adresního prostoru byl přidělen specifický význam. Celý prefix ::/8 byl původně rezervován pro speciální účely. Nyní je deklarován jako nepřiřazený, některé adresy v jeho rámci však přiřazeny byly. Jedná se o individuální adresy ::0 a ::1. První se používá pro nedefinovanou adresu. Říká, že dotyčnému rozhraní dosud nebyla přidělena IPv6 adresa. ::1 je pak adresou lokální smyčky (loopback), kterou počítač-schizofrenik může komunikovat sám se sebou. Spadá sem také prefix přidělený pro IPv4-mapované adresy (::ffff:0:0/96).

Skupinka prefixů identifikuje adresy s omezeným dosahem. Nejčastěji se setkáte s lokálními linkovými adresami, které jsou jednoznačné vždy jen v rámci jedné linky (jednoho Ethernetu, jedné Wi-Fi buňky,...). Poznáte je podle prefixu fe80::/10 a najdete je u každého rozhraní se zapnutým IPv6. Vedle nich dříve existovaly místní individuální lokální adresy s prefixem fec0::/10 jednoznačné v místní síti. Později však byly zrušeny, proto se jejich prefix v tabulce nevyskytuje. Nahradily je unikátní individuální lokální adresy s prefixem fc00::/7.

Podívejme se nyní podrobněji na jednotlivé kategorie.

3.4 Globální individuální adresy

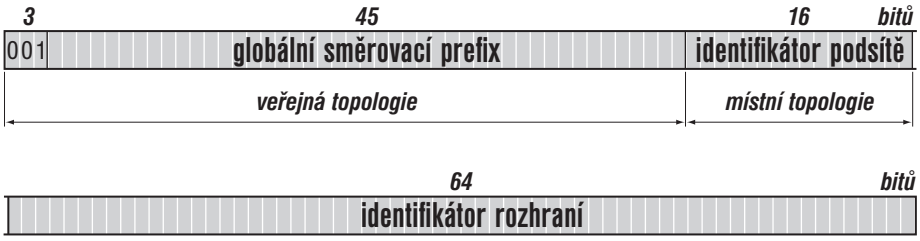
Tento typ adres je nejdůležitější, protože se jedná o „normální“ adresy – protipól adres současného IPv4. Slůvko globální naznačuje, že identifikují svého nositele v rámci celého Internetu a musí tudíž být celosvětově jednoznačné. Zatím byla definována jen část z nich (prefix 001 binárně), jejíž strukturu definuje RFC 3587: *IPv6 Global Unicast Address Format*.

agregace

Globální adresy jsou přidělovány hierarchicky podle pravidel podobných CIDR ze světa IPv4. To znamená, že poskytovatel Internetu (neboli lokální registr, LIR) obdrží určitý prefix, jehož části v podobě delších prefixů se shodným začátkem pak přiděluje svým zákazníkům. Cílem tohoto přístupu je agregace směrovacích údajů – aby bylo možné při pohledu zvenčí celou poskytovatelovu síť i se všemi zákazníky popsat jediným záznamem ve směrovacích tabulkách, obsahujícím onen společný prefix.

Toto shlukování je velmi důležité, protože významným způsobem zmenšuje velikost směrovacích tabulek. Jemnost členění směrovacích informací přirozeně klesá se vzdáleností od místa určení. Původně se koncept agregace promítal i do struktury adresy, která byla složena z identifikátorů několika úrovní. K praktickému naplnění této vize však nedošlo a reálně používané adresy původní koncept nedodržovaly.

Proto byl opuštěn a RFC 3587 zavedlo maximálně zjednodušený model, v podstatě odpovídající struktuře adresy pro IPv4. Ta má tři části: adresu sítě, podsítě a rozhraní v podsíti. Analogické části má i IPv6 adresa, jen adresa sítě byla přejmenována na globální směrovací prefix. Jejich délky jsou definovány zcela obecně, podle současných pravidel přidělování však globální směrovací prefix měří nejčastěji 48 bitů, adresa podsítě 16 bitů a adresa rozhraní v podsíti 64 bitů. Strukturu globální individuální adresy znázorňuje obrázek 3.3.



Obrázek 3.3: Obvyklá struktura globální individuální adresy

globální směrovací prefix

Globální směrovací prefix identifikuje koncovou síť. Je síti přidělen „zvenčí“ lokálním internetovým registrem, čili zpravidla poskytovatelem Internetu. Proto bývá tato část adresy označována jako „veřejná topologie“. Podrob-

něji se k problematice přidělování globálního směrovacího prefixu vrátím v části 3.13 na straně 91.

identifikátor podsítě

Identifikátor podsítě slouží k rozlišení jednotlivých podsítí v rámci dané sítě. Tato část adresy je, společně s identifikátorem rozhraní, záležitostí správy koncové sítě a používá se pro ni označení „místní topologie“. Dva bajty umožňují adresovat báječných 65 536 podsítí, což by mělo dostačovat i pro opravdu velké sítě. Díky tomu není nutné laborovat s počtem bitů věnovaných na adresu podsítě a lze si dovolit konstantní délku této části adresy.

Pouze v ojedinělých případech, jako jsou například propojovací podsítě na linkách spojujících pouhá dvě zařízení, má smysl uvažovat o dlouhých adresách podsítě a ponechání jen minimálního prostoru pro identifikátor rozhraní. Podrobněji se této problematice věnuji na straně 308, kde jsou rozebrány různé varianty adresování dvoubodových sítí, jejich přednosti a nevýhody.

Správce sítě se musí rozhodnout, jakou strategii při přidělování identifikátorů podsítí zvolí. Obecně je třeba najít rozumný kompromis mezi dvěma protichůdnými tendencemi. Adresy lze přidělovat plošně systémem „kdo dřív přijde, ten dřív mele“. Tedy číslovat postupně od jedničky, tak jak se objevují požadavky na nové podsítě. Tento systém je velmi jednoduchý, ale vede k větším směrovacím tabulkám směrovačů v koncové síti, protože podsítě jsou rozloženy náhodně a každá z nich musí mít v tabulce svůj vlastní záznam.

Alternativou je vytvořit si vlastní hierarchii. Například prvním bajtem adresy podsítě identifikovat areál či budovu a druhým pak rozlišit podsítě v ní. Dá to více práce a vyplýtváte některé adresy, ovšem směrovací tabulky budou menší, protože pro celé skupiny identifikátorů postačí vždy jen jedna položka.

Jakou strategii adresace podsítí zvolit závisí především na velikosti sítě, a to včetně výhledu na její rozvoj v nejbližších letech. V malé síti nemá smysl usilovat o budování hierarchie. Pokud však počet vašich podsítí půjde do desítek či stovek, vyplatí se vážně uvažovat o zavedení hierarchie jejich identifikátorů.

Zvolíte-li tento přístup, je rozumné používat protiběžné identifikátory. Tedy aby adresy hierarchicky vyšších celků (budov) narůstaly od levého okraje identifikátoru podsítě, zatímco adresy hierarchicky nižších celků (podsítí v budovách) od pravého okraje. Například pro čtyři budovy je rozumné, aby se jejich identifikátory lišily v nejvyšších bitech – tedy 0000, 4000, 8000 a c0000. Podsítě v poslední z budov pak budou mít identifikátory c000, c001, c002, atd.

Hlavní výhodou tohoto přístupu je, že dává možnost v budoucnosti posunout hranici mezi úrovněmi hierarchie doleva či doprava podle toho, ve

které části vám začne docházet místo. Podrobněji je popsán v RFC 3531: *A Flexible Method for Managing the Assignment of Bits of an IPv6 Address Block*.

Celý tento mechanismus vychází ze základní myšlenky, že směrovače se o nějakou vnitřní strukturu adres vůbec nestarají. Ta slouží pouze pro přidělování a správu adres. Z hlediska směrovače je situace zcela prostá: porovná adresu se svými směrovacími tabulkami a najde v nich položku s nejdelším prefixem, který odpovídá adrese. Tu použije.

identifikátor rozhraní

Závěrečný *identifikátor rozhraní* zabírá celou polovinu adresy, což umožňuje v jedné podsíti rozlišit něco přes $18 \cdot 10^{18}$ různých rozhraní (tedy miliardy miliard). Motivací k takto velkorysému dimenzování podsítě byla snaha o maximální zjednodušení automatické konfigurace počítačů. Nicméně nelze přehlížet, že AppleTalk zvládal automatickou konfiguraci s jediným bajtem[2] a IPv4 stačí čtyři bajty pro celosvětově jednoznačné adresy. Investovat osm bajtů na dosažení jednoznačnosti v jediné podsíti je zkrátka plýtvání.

Přesto RFC 4291 jednoznačně stanoví, že pro všechny individuální adresy (s výjimkou adres s prefixem 0::/3) je vyžadována délka identifikátoru rozhraní 64 bitů a používání identifikátorů ve tvaru modifikovaného EUI-64. Podívejme se na ně podrobněji.

3.5 Identifikátory rozhraní – modifikované EUI-64 a spol.

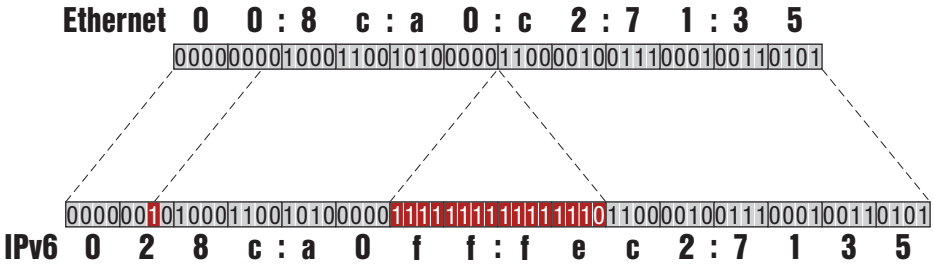
Základní podoba identifikátoru rozhraní v IPv6 je odvozena z IEEE EUI-64. Jedná se o standard zaměřený na přidělování globálních (celosvětově jednoznačných) identifikátorů pro rozhraní v počítačových sítích. Jejich délka je 64 bitů a odpovídá tedy délce místa vyhrazeného v IPv6 adrese.

Nejjednodušším případem je, když dané rozhraní již má přidělen EUI-64 identifikátor. Ten se do IPv6 adresy prostě převezme, ale s jednou změnou. Předposlední (druhý nejméně významný) bit v nejvyšším bajtu EUI-64 identifikátoru slouží jako příznak globality. Ve standardním EUI-64 zde hodnota 0 signalizuje celosvětově jednoznačnou adresu, zatímco 1 označuje adresu lokální.

IPv6 používá *modifikované EUI-64* a hodnotu tohoto bitu invertuje. Čili 0 zde představuje lokální identifikátor a hodnota 1 identifikátor globální. Tato změna usnadňuje vytváření identifikátorů rozhraní „na koleně“ – například pro sériové linky. Zde může správce přiřadit jednoduché lokální identifikátory 1 a 2. Kdyby zůstaly zachovány původní významy, kdy lokální identifikátor nese v příslušném bitu jedničku, měly by analogické identifikátory rozhraní podobu 200:0:0:1 a 200:0:0:2.

[2] Pravda, omezovalo to počet rozhraní v podsíti na 256, což by pro IPv6 jistě bylo neakceptovatelné.

Asi nejčastějším případem budou sítě založené na některé z variant Ethernetu či bezdrátové sítě IEEE 802.11. V takovém případě mají jednotlivá rozhraní výrobcem přidělené celosvětově jednoznačné MAC adresy o délce 48 bitů. Jejich transformace na EUI-64 je velmi snadná a standardní: mezi třetí a čtvrtý bajt MAC adresy se vloží 16 bitů s hodnotou fffe. Kromě toho se obrátí příznak globality. Takže z MAC adresy 00:8c:a0:c2:71:35 se v IPv6 adrese stane identifikátor rozhraní 028c:a0ff:fec2:7135.



Obrázek 3.4: Vytvoření modifikovaného EUI-64 z ethernetové adresy

soukromí

Používání EUI-64 je velmi přímočaré, ale ozývají se proti němu ochránci soukromí uživatelů. Jelikož je EUI-64 odvozeno z MAC adresy, která je celosvětově jednoznačná a mění se jedině s výměnou síťové karty[3], vzniká tak jednoznačná identifikace uživatelova počítače.

Navíc tato část adresy zůstává stejná, i když se počítač pohybuje. Jak se dočtete v kapitolách o automatické konfiguraci a mobilitě, počítač si svou adresu generuje stále stejně: zjistí prefix zdejší podsítě a připojí k němu svůj identifikátor rozhraní. Pokud se někomu podaří odposlouchávat síťový provoz na strategických místech, může krásně sledovat, s kým vším váš počítač komunikuje a jak se pohybuje po světě. Nepomůže ani šifrování, protože se šifruje jen obsah datagramu. Adresy musí zůstat otevřené.

Reakcí na tyto problémy je RFC 4941: *Privacy Extensions for Stateless Address Autoconfiguration in IPv6*. V podstatě navrhuje náhodné generování identifikátorů rozhraní, které budou mít životnost několik hodin až dnů a počítač je bude neustále měnit. Ovšem je potřeba i nějaký pevný bod, aby se s takovýmto počítačem dalo vůbec navázat spojení.

Proto RFC 4941 navrhuje, aby počítač měl jeden pevný identifikátor rozhraní (podle EUI-64), pod nímž bude zaveden v DNS. Hlavním smyslem této adresy je sloužit jako cílový bod pro komunikaci navazovanou zvenčí. Navíc si bude počítač generovat náhodné dočasné identifikátory. Adresám z nich odvozeným bude dávat přednost, když sám navazuje spojení s někým jiným. Tyto identifikátory nebudou zavedeny v DNS (jinak by se celý

[3] tedy velmi zřídka

efekt znehodnotil – počítač by sice střídal adresy, ale dal by se poznat podle shodného jména v DNS).

Díky tomu klienti na daném počítači používají náhodné krátkodobé adresy a nelze dlouhodobě sledovat jejich aktivity.

kryptograficky generované identifikátory

RFC 3972 zavedlo další odrůdu identifikátorů rozhraní, jež umožňují zabezpečit objevování sousedů. Vycházejí z veřejného klíče svého vlastníka a znemožňují nepřátelské stanici vydávat se za někoho jiného. Podrobněji se jim budu věnovat na straně 110 v souvislosti s mechanismy, pro jejichž ochranu jsou určeny.

3.6 Lokální adresy

Koncept adres, které neplatí v celém Internetu, ale pouze v jeho malé části, zavedlo RFC 1918: *Address Allocation for Private Internets*. Malou část adresního prostoru IPv4 vyhradilo pro neveřejné adresy, které lze používat v koncových sítích, ale nejsou podporovány za jejich hranicemi. Tyto adresy nejsou celosvětově jednoznačné, každá koncová síť si s nimi může nakládat, jak se jí zlíbí. Původně byly určeny především pro experimenty či pro sítě, které neměly ambice připojit se k Internetu. V současnosti se používají masově v kombinaci s NATem, který jim přístup k Internetu dokáže zprostředkovat, byť s řadou omezení.

IPv6 posouvá tuto myšlenku ještě o krok dál. Zavádí koncept dosahu adres, k němuž se dostaneme později (viz část 3.10 na straně 78). Ten je ale přínosný především pro skupinové adresy, jejichž součástí je přímo informace o dosahu. Pro individuální adresy jsou možnosti omezené, nicméně i zde existuje několik typů adres s omezeným dosahem. Jejich přehled uvádí obrázek 3.5. Jsou na něm zobrazeny pouze první poloviny adres, protože druhá polovina ve všech třech případech obsahuje standardní identifikátor rozhraní.

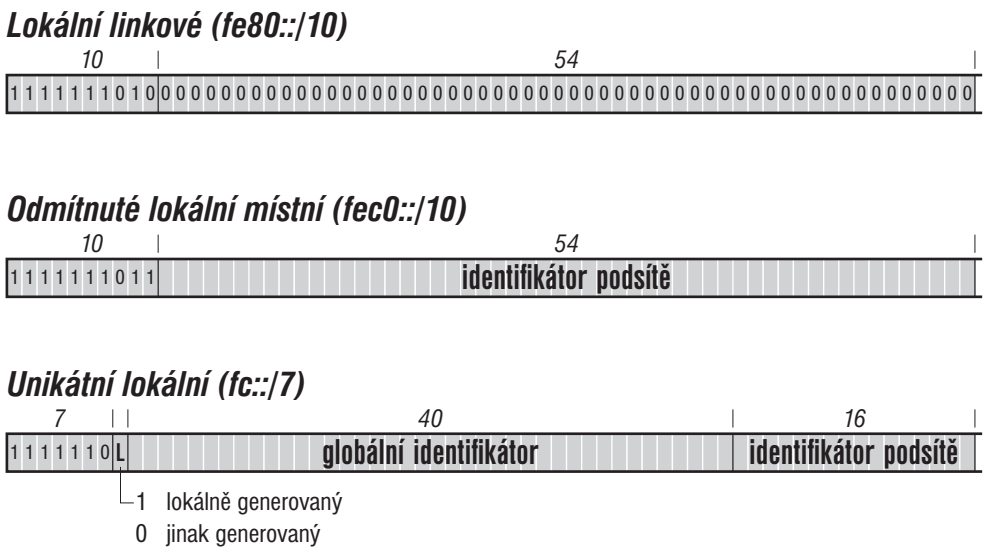
lokální linkové

Největší význam mají *lokální linkové adresy (link local)*. V adresní architektuře mají svou vyhrazenou část – začínají prefixem fe80::/10. Následujících 54 bitů je nulových, za nimi najdete 64bitový identifikátor rozhraní podle modifikovaného EUI-64.

Počítač z příkladu v předchozí části s ethernetovou adresou 00:8c:a0:c2:71:35 by tomuto rozhraní přidělil lokální linkovou adresu:

fe80::28c:a0ff:fec2:7135

Vzhledem k její standardní podobě si tuto adresu vytvoří sám a pomocí nástrojů automatické konfigurace ověří, že je pro danou linku skutečně jednoznačná (podrobnosti se dočtete v kapitole 6 na straně 119).



Obrázek 3.5: Typy lokálních adres

Počátečních 64 bitů je v ní oficiálně interpretováno jako adresa sítě a podsítě, neslouží však ke směrování. Ani nemohou, protože hodnota těchto bitů je pevně dána a je u všech stejná. Nijak to nevadí, protože dosah lokálních linkových adres je omezen na jedinou linku. Tedy na skupinu počítačů vzájemně komunikujících na linkové úrovni, například propojených Ethernetem či bezdrátovou sítí Wi-Fi. Datagramy nesoucí lokální linkovou adresu jako cíl neprojdou žádným směrovačem, protože za ním již leží jiné linky.

identifikátor rozhraní

Ze samotné lokální linkové adresy se nedá ani poznat, ke které lince se vlastně vztahuje. Proto se poměrně často vyskytují v kombinaci s identifikátorem rozhraní, díky němuž lze určit, která konkrétní linka je obsahuje. Od adresy se odděluje znakem procento, takže zápis

fe80::2a4:3bff:fee3:35e8%1

představuje lokální linkovou adresu fe80::2a4:3bff:fee3:35e8 nacházející se na lince připojené k rozhraní 1. Podrobněji se budu těmto otázkám věnovat v části 3.10 na straně 78.

Jejich hlavní výhodou je, že počítač si takovou adresu dokáže vygenerovat sám a nepotřebuje k tomu žádnou infrastrukturu. Díky tomu je lokální linková adresa k dispozici vždy. Stačí propojit počítače ethernetovým přepínačem, nemusíte mít žádný směrovač ani server, a přesto mohou rovnou komunikovat prostřednictvím lokálních linkových adres, které si samy vytvoří. V takovéto provizorní síti nebude DNS server, takže uživatelé budou

muset zadávat nepříliš přítulné adresy ručně, nicméně budou mít k dispozici alespoň nějaké spojení.

Všudypřítomnost lokálních linkových adres využívají i některé interní mechanismy související s IPv6. Například automatická konfigurace pomocí DHCP používá pro výměnu zpráv mezi klientem a serverem tyto adresy.

lokální místní

Roli velmi podobnou adresám z RFC 1918 hrály ve starších definicích adresního prostoru pro IPv6 *lokální místní adresy (site local)*. Byl jim přidělen prefix fec0::/10 a jejich platnost byla omezena na jedno „místo“. Typickým místem je koncová síť organizace připojené k Internetu.

Jenže existují také organizace připojené k Internetu v několika lokalitách téhož města či dokonce v různých městech a státech. Mají být areály MFF UK v na Karlově, v Karlíně, v Tróji a na Malé Straně považovány za čtyři různá místa nebo za jedno místo? Praxe ukázala, že definice místa je vágní a její výklad se velmi liší. Navíc se připojily problémy s konfiguracemi směrovačů a další obtíže při pokusech o reálné použití místních adres.

Výsledkem bylo RFC 3879: *Deprecating Site Local Addresses*, které místní lokální adresy zamítlo a dokonce zakazuje novým implementacím podporovat speciální zpracování adres s prefixem fec0::/10.

unikátní lokální

Jejich nástupcem se staly *unikátní lokální adresy (unique local, ULA)* definované v RFC 4193: *Unique Local IPv6 Unicast Addresses*. Poznají se podle prefixu fc00::/7. Za ním následuje jednobitový příznak *L*, zda byl prefix adresy přiřazen lokálně (L=1) nebo jinak[4]. Vzhledem k tomu, že všechny v současnosti používané adresy tohoto typu jsou generovány lokálně, mají nastaven příznak L na jedničku a začínají proto prefixem fd00::/8.

Dalších 40 bitů obsahuje globální identifikátor, kterým je náhodně vygenerované číslo[5]. RFC 4193 výslovně zakazuje jeho sekvenční či jinak předvídatelné určení a v části 3.2.2 doporučuje postup vycházející z aktuálního času, adresy generující stanice a algoritmu SHA-1. Čtyřicetibitová položka může nabývat více než bilionu různých hodnot. Pravděpodobnost, že dvojice sítí zvolí stejný globální identifikátor je tedy zhruba 10^{-12}. Při milionu koncových sítí je pořád ještě pravděpodobnost, že si alespoň dvě vygenerují stejný globální identifikátor, méně než poloviční.

Prefix společně s globálním identifikátorem dohromady vytvoří obvyklý síťový prefix délky 48 bitů. Za ním následuje v adrese vše podle vyježděných kolejí: 16bitový identifikátor podsítě a 64bitový identifikátor rozhraní podle modifikovaného EUI-64.

[4] V RFC 4193 se píše o „jiné“ metodě přiřazení bez bližšího určení. V pozadí zjevně čeká myšlenka jakési centrální autority, která by u adres s L=0 ručila za celosvětovou jednoznačnost jejich prefixu. Myšlenka globálně koordinovaných lokálních adres má své urputné zastánce i kritiky.

[5] Generátor prefixů je k dispozici na adrese *http://www.sixxs.net/tools/grh/ula/*

Proč se globální jednoznačnost těchto adres považuje za tak podstatnou, když se beztak předpokládá jejich lokální využití a stejně jako v případě místních adres nejsou směrovány v internetové páteři? Vyjděme z výše uvedeného příkladu se čtyřmi pražskými lokalitami MFF UK. Řekněme, že správci sítě je považují za jedno místo a kromě veřejných adres chtějí používat také lokální adresy. Vygenerují si tedy prefix, řekněme fdd6:c246:22a9::/48, který ponesou všechny lokální adresy ve spravované síti. Adresami podsítí pak rozliší jednotlivé podsítě. To vše by se snadno dalo zajistit i místními lokálními adresami.

Jednotlivé lokality jsou ale poměrně vzdáleny a k jejich propojení bude využita některá páteřní síť, v daném případě nepochybně PASNET. Po ní budou zároveň směrovány analogické lokální adresy ostatních fakult a univerzit. Unikátní lokální adresy nezpůsobí problém – různé sítě si vygenerovaly odlišné prefixy a budou mít proto jiné adresy.

V případě místních lokálních adres, které obsahují jen konstantní prefix, identifikátor podsítě a rozhraní je naproti tomu značná pravděpodobnost kolize. Například lze očekávat, že podsíť 1 si vytvoří více institucí. Jejich propojení sdílenou páteřní sítí by vyžadovalo tunely, virtuální privátní síť či podobnou nadstandardní konfiguraci. Navíc by případné „prosáknutí“ směrovacích informací mohlo způsobit zmatek v jiných částech sítě, zatímco unikátní lokální adresy tímto problémem netrpí.

3.7 Skupinové adresy

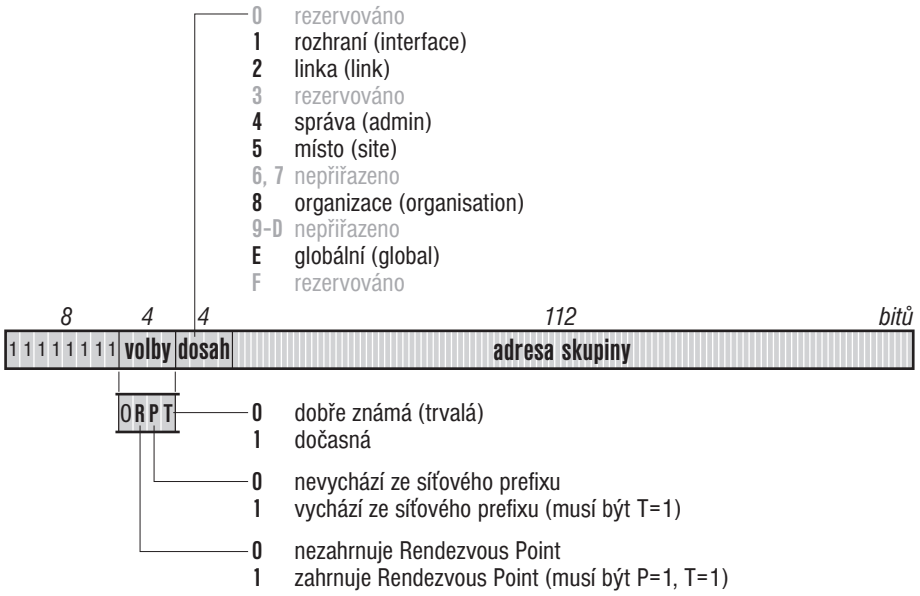
Princip skupin a skupinových adres není žádnou výjimkou již v současném Internetu. Slouží především k distribuci zvukového a obrazového signálu v reálném čase (videokonference, rozhlasové či televizní vysílání a podobně).

Ve skupinách podle IPv6 by nemělo dojít k žádné zásadní revoluci. Strukturu adresy představuje obrázek 3.6. Její největší část slouží k identifikaci skupiny, které mají být data dopravena. K tomu se přidružují dvě krátké podpůrné položky: příznaky a dosah skupiny.

První ze čtyř příznaků je rezervován pro pozdější použití a zatím musí být nulový. Za ním následuje trojice příznaků *R* (rendezvous point), *P* (prefix) a *T* (transient). Vlastní adresní architektura definuje pouze přínzak *T*. Ostatní dva jsou zavedeny v samostatných dokumentech a jejich popis o chvilku odložím, protože využívají některé další koncepty.

příznak T

Čtvrtý bit je označován písmenem *T* (transient) a signalizuje, zda je daný identifikátor skupiny přidělen trvale a jedná se tedy o „dobře známou“ adresu (hodnota 0) nebo zda je přidělen pouze dočasně (*T* má hodnotu 1). Dobře známé adresy přiděluje IANA, zatímco dočasné si mohou genero-



Obrázek 3.6: Struktura skupinové adresy

vat aplikace podle potřeby. Právě jimi se zabývá většina dalších specifikací a návrhů. Zanedlouho se k nim vrátím.

dosah Dosah skupiny sděluje, jak daleko od sebe mohou jednotliví členové být. Jedná se opět o čtyřbitovou položku se šestnácti možnými hodnotami. Nějaký význam byl zatím přidělen zhruba deseti z nich, jak uvádí tabulka 3.2. Podrobnější komentář k dosahům najdete v samostatné části 3.10 na straně 78. Zejména doporučuji vaší pozornosti tabulku 3.7 na straně 80 obsahující stručnou charakteristiku existujících dosahů.

Nepřiřazené dosahy, jejichž význam v tabulce je proškrtnut, jsou volně k použití. Určitý význam jim může přidělit například poskytovatel Internetu či správce části sítě. Mělo by přitom zůstat zachováno, že větší hodnota dosahu v adrese bude znamenat doručování paketů do větší[6] části Internetu, než dosahy menší. Například v síti CESNET2, stejně jako v dalších evropských národních akademických sítích, je definován dosah A pokrývající danou národní síť. Skupinové pakety s dosahem A budou proto u nás doručovány všem zájemcům v rámci sítě CESNET2. Lze očekávat, že podobný přístup zavedou i komerční poskytovatelé Internetu a dosah A bude všeobecně znamenat „poskytovatel a jeho zákazníci“.

identifikátory Jedná-li se o permanentní skupinu (příznak *T* má hodnotu 0), je její identi-

[6] případně stejné, ale rozhodně ne menší

dosah	*význam*
0	rezervováno
1	lokální pro rozhraní
2	lokální pro linku (fyzickou síť)
3	rezervováno
4	lokální pro správu
5	lokální pro místo
6	—
7	—
8	lokální pro organizaci
9	—
A	—
B	—
C	—
D	—
E	globální
F	rezervováno

Tabulka 3.2: Dosah skupinových adres

fikátor stále platný a nezávisí na dosahu. Například skupina adres ff0*x*::101 (kde *x* představuje různé dosahy) byla přidělena NTP serverům. Důsledkem jsou následující významy adres:

ff01::101	NTP servery na tomtéž rozhraní (čili on sám)
ff02::101	NTP servery na stejné lince (např. Ethernetu)
ff05::101	NTP servery v daném místě (lokalitě)
ff0e::101	NTP servery v celém Internetu

Naproti tomu dočasná skupina má význam jen v rámci svého dosahu. Takže například skupina s adresou ff15::101 nemá žádný vztah ke skupině, která má stejnou adresu, ale je vytvořena na jiném místě. Dokonce nemá žádný vztah ani k dočasné skupině se stejným identifikátorem, ale jiným dosahem (např. ff1e::101) ani k trvalým skupinám se stejným identifikátorem. Nemá tedy nic společného s žádnou z výše uvedených skupin NTP serverů.

přidělování adres

Pravidla pro přidělování identifikátorů skupinových adres definuje RFC 3307: *Allocation Guidelines for IPv6 Multicast Addresses*. Teoreticky je pro identifikátor skupiny k dispozici 112 bitů. Za chvilku ale uvidíme, že některé formáty definují určitou strukturu i v této části adresy a pro skutečný identifikátor skupiny ponechávají jen posledních 32 bitů. RFC 3307 toto omezení kodifikuje a navíc rozděluje skupinové identifikátory do tří oblastí, které najdete v tabulce 3.3.

Rozdíl mezi prvními dvěma skupinami se zdá být poněkud esoterický. Do první patří případy, kdy IANA definuje celé skupinové adresy, jako napří-

0–3fff:ffff	skupiny přidělené IANA
4000:0000–7fff:ffff	identifikátory přidělené IANA
8000:0000–ffff:ffff	dynamické, volně k použití

Tabulka 3.3: Rozdělení skupinových identifikátorů podle RFC 3307

klad výše zmíněnou adresu ff0*x*::101 pro NTP servery. Ve druhé skupině jsou identifikátory, kde IANA definuje pouze samotný skupinový identifikátor, zatímco prefix před ním může být libovolný. Předpokládá se jejich použití především pro adresy odvozené z individuálních, k nimž se hned dostanu. Zatím byl definován jediný, 4000:0000 pro proxy sítě, většina definic IANA spadá do první skupiny. Vybrané adresy a identifikátory přidělené IANA najdete v příloze A na straně 389.

Do třetí skupiny spadají identifikátory, které si mohou přidělovat podle potřeby jednotlivé aplikace a služby. Existují dva základní přístupy ke správě tohoto typu identifikátorů. Jedním je alokační server, u nějž si aplikace požádají o přidělení skupinového identifikátoru[7]. Podle druhého si berou identifikátory samostatně prostřednictvím vhodného autokonfiguračního protokolu. V každém případě se však jedná o adresy dočasné, jejich příznak *T* proto musí mít hodnotu 1. Ostatně rozdělení skupinových identifikátorů v tabulce 3.3 je navrženo tak, aby první bit v čísle skupiny kopíroval hodnotu příznaku *T*.

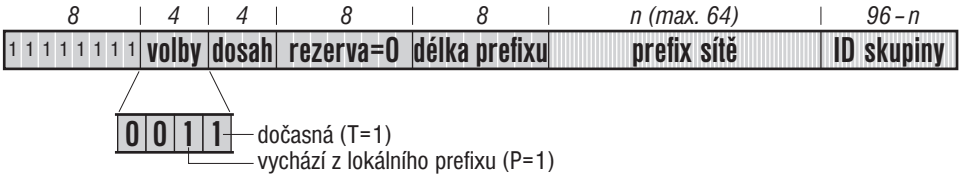
příznak P skupinové adresy vycházející z individuálních

Příznak *P* byl definován v RFC 3306: *Unicast-Prefix-based IPv6 Multicast Addresses*, které zavedlo *skupinové adresy vycházející z individuálních*. Vznikly s cílem usnadnit generování jednoznačných skupinových adres, aniž by generující stroj musel komplikovaně zjišťovat, zda adresa již někde neexistuje. Proto je jako součást adresy zařazen prefix individuálních adres zdejší sítě. Ten je celosvětově jednoznačný, takže stačí zajistit jednoznačnost identifikátoru v rámci sítě a máme vystaráno.

Jedná se vlastně o jeden možný konkrétní formát pro identifikátor skupiny. Jeho uspořádání najdete na obrázku 3.7. Začíná osmibitovou rezervovanou položkou, jejíž hodnotou jsou povinně samé nuly. Následuje délka použitého prefixu, tedy počet významných bitů v něm. Nejčastěji bude obsahovat hodnoty 48 či 64. V dalších bitech je uložen prefix odpovídající části sítě, z níž tato skupinové adresa pochází. Jejich počet odpovídá délce z předchozí položky, nanejvýš jich však může být 64. A konečně závěrečných minimálně 32 bitů obsahuje vlastní identifikátor skupiny.

Příznak *P* s hodnotou 1 oznamuje, že identifikátor skupiny ve skupinové adrese byl vytvořen tímto způsobem. Je-li *P*=1, musí se jednat o dočasnou adresu a proto musí mít i příznak *T* hodnotu 1. Zároveň RFC 3306 požaduje,

[7] Například protokolem Multicast Address Dynamic Client Allocation Protocol (MADCAP) podle RFC 2730.



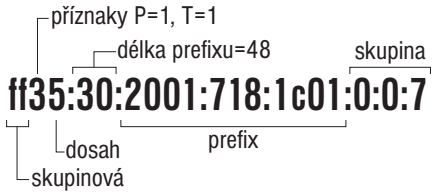
Obrázek 3.7: Skupinová adresa založená na individuální

aby dosah takové adresy nepřesahoval dosah prefixu použitého při jejich vytvoření.

Například Technická univerzita v Liberci má pro své individuální IPv6 adresy přidělen prefix 2001:718:1c01::/48. Řekněme, že z tohoto prefixu chceme odvodit skupinovou adresu s dosahem pro místo (dosah 5) se skupinovým identifikátorem 7. Výsledkem bude skupinová adresa

ff35:30:2001:718:1c01:0:0:7

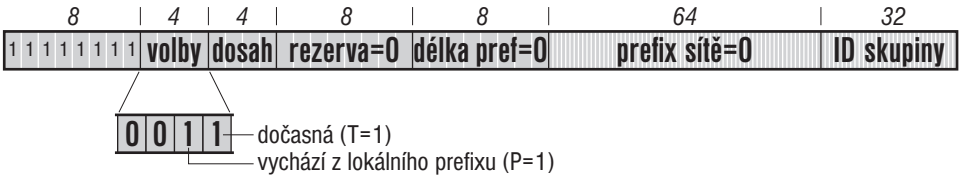
Její strukturu rozebírá obrázek 3.8.



Obrázek 3.8: Příklad skupinové adresy vycházející z individuální

adresy pro SSM

Speciálním případem skupinově adresovaného vysílání je tak zvaný *Source Specific Multicast (SSM)*. Slouží pro přenosy dat z jednoho zdroje skupině příjemců, například pro internetové rádio či televizi. Pro něj byla vyčleněna samostatná část skupinových adres založených na individuálních. Pozná se podle toho, že délka prefixu i prefix sítě jsou nulové. Skupinové adresy pro SSM tedy mají prefix ff3*x*::/96, za nímž následuje 32b identifikátor skupiny. Jejich strukturu představuje obrázek 3.9.

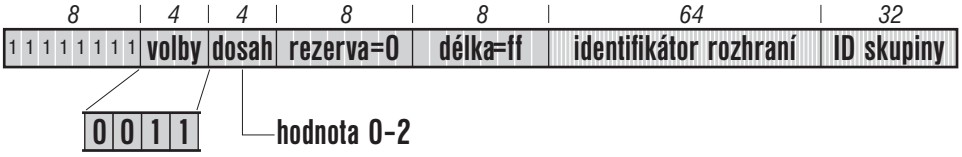


Obrázek 3.9: Skupinová adresa pro SSM

Jednoznačnosti zde není těžké dosáhnout, protože skupiny mají vždy jen jediného odesilatele. Stačí, aby on sám si udržel pořádek v jejich identifikátorech. Skupina je jednoznačně určena dvojicí zdrojové adresy svého jediného odesilatele a skupinové adresy.

skupinové adresy vycházející z rozhraní

Konkurenci adresám založeným na prefixu sítě tvoří *skupinové adresy vycházející z rozhraní* definované v RFC 4489: *A Method for Generating Link-Scoped IPv6 Multicast Addresses*. Jejich dosah nesmí být větší než jediná linka, za to si je ale každý stroj může generovat sám bez rizika konfliktu s adresami generovanými jeho sousedy. Adresa tohoto typu totiž obsahuje jeho identifikátor rozhraní, který je na lince vždy jednoznačný. Stačí tedy, aby si udržoval přehled o přidělených identifikátorech skupin, a může si být jist, že všechny používané adresy tohoto typu jsou jednoznačné.



Obrázek 3.10: Skupinová adresa založená na identifikátoru rozhraní

Strukturu adresy vycházející z identifikátoru rozhraní znázorňuje obrázek 3.10. Volby má nastaveny stejně jako v předchozím případě (*P*=1, *T*=1), od adresy vycházející z prefixu sítě se pozná podle hodnoty pole *Délka prefixu*, jejíž všechny bity obsahují jedničky. Následujících 64 bitů je tvořeno identifikátorem rozhraní, tedy spodní polovinou jeho lokální linkové adresy. Závěrečných 32 bitů nese identifikátor skupiny.

Můj oblíbený příkladný počítač s ethernetovou adresou 00:8c:a0:c2:71:35 si vygeneruje lokální linkovou adresu fe80::28c:a0ff:fec2:7135 a po ověření její jednoznačnosti ji může začít používat pro vytváření skupinových adres. Jejich prefix bude ff32:ff:28c:a0ff:fec2:7135::/96.

Obrázek 3.11: Příklad adresy založené na identifikátoru rozhraní

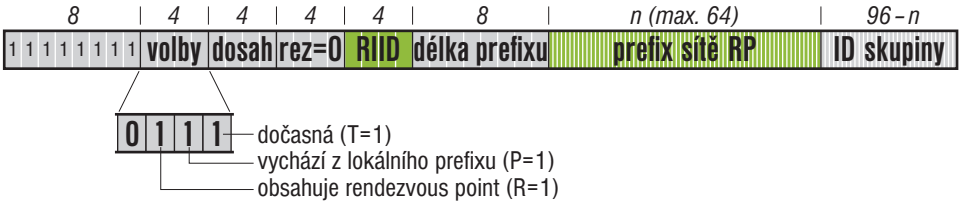
příznak R skupinové adresy obsahující RP

Další možný formát skupinového identifikátoru zavádí RFC 3956: *Embedding the Rendezvous Point (RP) Address in an IPv6 Multicast Address*. Jedná se o skupiny používané ve spojitosti se směrovacím protokolem PIM-SM,

o kterém se dočtete v části 8.3.1 na straně 179. Klíčovou roli v něm hraje tak zvané shromaždiště (rendezvous point, RP), v němž koření distribuční strom skupiny. Tento typ skupinových adres v sobě obsahuje adresu shromaždiště, což přináší dva významné klady. Jednak to usnadňuje vytváření jednoznačných adres – stačí udržet v nich přehled v rámci shromaždiště. Především ale kdokoli v celém Internetu si ze skupinové adresy odvodí adresu jejího shromaždiště a ví, kde se do ní přihlásit.

Skupinové adresy obsahující RP nadále rozvíjejí skupinové adresy s individuálním prefixem definované v RFC 3306 (viz obrázek 3.7). Přidávají k nim další příznak *R*, podle nějž je poznáte. Skupinová adresa obsahující RP musí mít všechny tři příznaky nastaveny na jedničku, začíná tedy prefixem ff70::/12.

Zabudování adresy RP do skupinové adresy je o něco komplikovanější než v předchozích případech. Adresa RP je rozdělena na dvě části: prefix sítě a identifikátor rozhraní. Prefix sítě je do adresy vložen stejně jako v předchozích případech, zatímco identifikátor rozhraní (označen jako RIID) nahradí spodní čtyři bity v původně rezervované osmibitové položce před délkou prefixu. Výsledek vidíte na obrázku 3.12.



Obrázek 3.12: Skupinová adresa obsahující RP

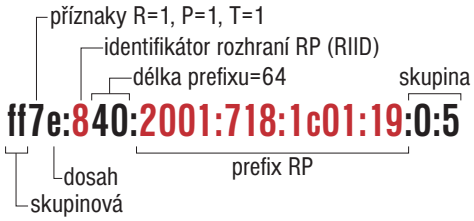
Z takto vytvořené skupinové adresy lze odvodit adresu jejího RP tak, že síťový prefix použijeme jako její začátek, RIID jako konec a bity mezi nimi vynulujeme. Například skupina globálního dosahu s identifikátorem 5 odvozená od shromaždiště s adresou 2001:718:1c01:19::8 bude mít adresu

ff7e:840:2001:718:1c01:19:0:5

Vysvětlení její podoby najdete na obrázku 3.13, části obsahující adresu shromaždiště jsou v ní vyznačeny červeně.

předdefinované adresy

RFC 4291 přiděluje několika skupinovým adresám speciální významy. Jedná se o adresu pro všechny uzly (tedy všechna fungující IPv6 rozhraní) v rámci jednoho rozhraní (ff01::1) či v rámci dané linky (ff02::1). Tyto skupiny nahrazují dřívější všesměrové adresy (broadcast).



Obrázek 3.13: Příklad skupinové adresy obsahující RP

Do další speciální skupiny patří všechny směrovače. Opět je k dispozici v několika dosazích: v rámci rozhraní (ff01::2), linky (ff02::2) či místa (ff05::2).

Poslední ze speciálních skupinových adres je adresa vyzývaného uzlu. Těchto skupin je celá řada a jejich adresy mají jednotný tvar ff02::1:ff*xx*:*xxxx*. Hodnota závěrečné trojice bajtů (zde nahrazená písmeny „*x*“) se vždy převezme z hledané adresy. Využití najde při objevování sousedů. Podrobnosti o ní se dozvíte v části 5.1 na straně 104.

Celou přehršel permanentních (dobře známých) skupinových adres definuje RFC 2375: *IPv6 Multicast Address Assignments*. Zavádí přes 70 adres pro nejrůznější kategorie počítačů a především typy síťových protokolů a služeb. Např. výše zmiňovaná skupina všech NTP serverů (101) pochází právě odtud. Přehled nejvýznamnějších skupinových adres najdete v příloze A na straně 389.

zákazy Skupinová adresa se nesmí vyskytnout na místě odesilatele IPv6 datagramu a nesmí být obsažena ani v jeho směrovací hlavičce. Kromě toho nelze přidělovat adresy ff0*x*:0:0:0:0:0:0:0, které jsou rezervovány.

3.8 Výběrové adresy

Jak již bylo řečeno, výběrové adresy představují asi nejzajímavější novinku v oboru adresování a zároveň velkou výzvu, protože jsou dosud ne zcela probádaným územím. Poskytují velmi zajímavé možnosti. Jejich prostřednictvím lze řešit třeba zdvojování počítačů, směřující ke zvýšení výkonu či spolehlivosti. Mohou se také využít k vyhledání nejbližšího stroje poskytujícího určitou službu.

Například současné nejzatíženější servery bývají ve skutečnosti realizovány skupinou spolupracujících počítačů. Prostřednictvím triků s DNS se dosahuje rozkládání dotazů na jednotlivé členy skupiny. Zbývá však celá řada obtíží (jak rovnoměrně rozkládat zátěž, připojení k Internetu musí mít odpovídající kapacitu apod.).

S výběrovými adresami lze daný problém řešit daleko elegantněji: servery ze skupiny rozmístíte ve vhodných místech Internetu a přidělíte všem stejnou výběrovou adresu. Klient bude posílat pakety na tuto adresu a standardní směrovací mechanismy zajistí, že dorazí vždy k nejbližšímu ze skupiny serverů. Navíc lze složení skupiny průběžně měnit podle potřeby.

Výkladní skříní výběrových adres se staly kořenové DNS servery. Těch by na jedné straně mělo být mnoho, aby docházelo k rozkládání zátěže, služba byla rychle dostupná z libovolné části Internetu a lépe odolávala útokům usilujícím o její zahlcení (DoS, DDoS). Na druhé straně by jich ale mělo být málo, protože jejich adresy musí znát skoro všechny ostatní DNS servery. Seznam adres by proto měl být krátký a velmi konzervativní.

Výběrové adresy právě pro tento případ nabízejí ideální řešení: adres kořenových serverů je třináct, ovšem postupně přecházejí na výběrové. V polovině roku 2011 bylo deset z nich výběrových a za třinácti adresami se díky tomu skrývalo přes 250 serverů. Jejich složení lze navíc průběžně měnit, aniž by se to projevilo na seznamu adres kořenových serverů.

Vzhledem ke své zjevné užitečnosti byl koncept výběrových adres později převzat i pro v IPv4. Základní vlastnosti jsou pochopitelně společné, ovšem rozlehlý adresní prostor IPv6 jejich použití poněkud usnadňuje. Nejčastěji se nasazením výběrové adresy sleduje některý z následujících cílů:

- Přibližné rozkládání zátěže – dotazy z určité části sítě se sejdou vždy na jednom z uzlů poskytujících výběrově adresovanou službu. Dochází k rozdělení sítě na spádové oblasti.
- Zrychlení doby odezvy díky kratší cestě mezi klientem a serverem.
- Lepší odolnost proti útokům typu DoS a DDoS – útočníci jsou schopni „dosáhnout“ jen na servery, v jejichž spádových oblastech se sami nacházejí.
- Zmenšení počtu adres, na nichž je služba poskytována. Představte si, že seznam adres kořenových DNS serverů by měl 250 položek a měnil se několikrát za měsíc...

Všude je samozřejmě chléb o dvou kůrkách a některé další vlastnosti výběrových adres nejsou až tak zářivé. Ale než se k nim dostanu, podívejme se, jak jsou vlastně realizovány.

realizace

Výběrovým adresám nebyla rezervována samostatná část adresního prostoru. Pocházejí ze stejných oblastí jako adresy individuální a mohou se s nimi libovolně míchat. Syntakticky je od sebe nelze rozlišit a ze samotné adresy se nedozvíte, zda je individuální či výběrová. Pokud přidělujete některému rozhraní výběrovou adresu, musí se to příslušným způsobem odrazit v konfiguraci.

Vezmete-li všechna rozhraní, která nesou určitou výběrovou adresu, jistě dokážete najít jistou (co nejmenší) obalovou síť či skupinu sítí, v níž jsou obsažena. Tuto síť lze charakterizovat prefixem *P*. Například pokud se budou všichni členové výběrové skupiny nacházet ve stejné podsíti, bude *P* prefix (adresa) této podsítě.

Uvnitř sítě dané prefixem *P* musí mít výběrová adresa svůj vlastní směrovací záznam, který v jednotlivých směrovačích ukazuje vždy na nejbližšího člena skupiny. Podle těchto záznamů jsou doručovány pakety adresované výběrové skupině. Mimo oblast danou prefixem *P* pak již není třeba s výběrovou adresou zacházet nijak speciálně a může být zahrnuta do agregovaného bloku adres.

Skutečnost, že výběrové adresy lze směrovat obvyklými metodami (de facto se jedná o cesty k individuálním počítačům, které dnešní směrovací algoritmy a protokoly podporují), je rozhodně dobrou zprávou. Stačí, aby počítač, který se zapojil do výběrové skupiny, ohlásil tuto skutečnost některému směrovači. Ten pak zajistí její distribuci ostatním.

V nejhorším případě jsou příslušníci skupiny natolik rozptýleni, že společný prefix má nulovou nebo zanedbatelnou délku (např. tříbitový prefix 001 příliš nepomůže). V takovém případě by se výběrová adresa přidala mezi globální směrovací informace, což potenciálně představuje obrovský nárůst velikosti směrovacích tabulek páteřních směrovačů. Proto je existence globálních výběrových adres velmi silně omezena a ani do budoucna se neočekává, že by si je snadno mohl zřizovat kdokoli.

Vazba na směrování představuje lesk a bídu tohoto způsobu adresování. Na jedné straně poskytuje pohodlí – aplikace adresuje datagram výběrově a směrování se postará o doručení. Na straně druhé představuje omezení, jimž se musí výběrově poskytované služby přizpůsobit.

problémy

Za základní problém lze považovat dynamičnost směrování, které se přizpůsobuje změnám v síti. Pošlete-li sérii datagramů na stejnou výběrovou adresu, mohou být dopraveny různým počítačům. To způsobuje problémy stavovým protokolům, jako je TCP, ale i službám uchovávajícím stav na straně serveru. Možným řešením je rozdělit komunikaci na dvě fáze. V úvodní, která používá výběrové adresy, klient zjistí od serveru jeho individuální adresu a tu pak použije pro vlastní, stavovou fázi přenosu dat.

Ideálem výběrového adresování je však služba, která stavové informace nevyžaduje. Typickým příkladem je právě DNS, kdy dotaz a odpověď představují po jednom datagramu přenášeném protokolem UDP. Čili není co uchovávat, jestliže každý klientův dotaz přistane na jiném serveru, nijak to nevadí. Jen je samozřejmě třeba zajistit konzistenci dat poskytovaných členy výběrové skupiny.

Tím jsme ovšem s problémy způsobovanými směrováním neskončili. Klacky pod nohy výběrového směrování může házet celá řada mechanismů. Směrovací politiky, kdy páteřní internetové směrovače odmítají příliš dlouhé prefixy, a tedy záznamy pro výběrové adresy. Agregace prefixů, která může napáchat na směrování výběrových adres nepěkné škody. Změny ve skupině mohou být vyhodnoceny jako kolísání (flapping) a následně blokovány. Mohou mít také problémy s bezpečnostními RPF testy, které mohou v jejich případě neprávem vyhodnotit zdrojovou adresu jako falšovanou.

Všechny popsané problémy jsou nejožehavější mezi páteřními směrovači Internetu, kde se vyměňují největší objemy směrovacích informací a proto zde panuje největší přísnost na jejich obsah. Navíc jsou různé části páteře řízeny různými subjekty, jejichž směrovací politiku lze jen těžko ovlivnit. To všechno vede k závěru, že výběrové adresování je v globálním měřítku použitelné jen velmi omezeně pro úzký sortiment vybraných služeb (třeba ony kořenové DNS servery).

Naopak uvnitř menší části sítě (v jednom autonomním systému či v koncové zákaznické síti), kde směrování má zcela pod kontrolou jeden provozovatel, může výběrové adresování představovat rozumně a celkem široce použitelný mechanismus. Výše popsané problémy jsou zde řešitelné bez většího úsilí a lze očekávat i řídké změny topologie, takže datagramy budou zpravidla doručovány témuž stroji.

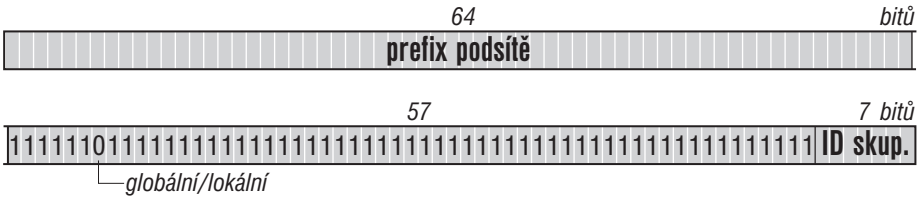
lokální adresy

Určitý extrém představují výběrové adresy v rámci jediné podsítě. U nich pochopitelně nemá smysl mluvit o nejbližším členovi, z pohledu směrování jsou všichni členové stejně daleko. Slouží jako obecné adresy, kdy počítač chce komunikovat se strojem poskytujícím určitý typ služby a nezáleží mu na tom, s kým konkrétně.

Tyto výběrové adresy mají podobu pevně definovaných identifikátorů rozhraní, před nějž se přidá prefix příslušné podsítě. Zatím byly definovány dvě takové adresy: samé nuly v identifikátoru rozhraní znamenají výběrovou adresu pro směrovače v podsíti a adresa *prefix*:fdff:ffff:ffff:fffe identifikuje domácí agenty v podsíti (více se dočtete v kapitole o mobilitě na straně 223). Kromě toho RFC 2526: *Reserved IPv6 Subnet Anycast Addresses* rezervuje horních 128 identifikátorů rozhraní pro různé speciální účely.

Jak konkrétně bude oněch horních 128 adres vypadat závisí na konstrukci adresy (a tedy na prefixu). Pokud identifikátor rozhraní bude mít tvar podle EUI-64 (nejčastější případ), musí bit označující globální/lokální identifikátor obsahovat hodnotu 0 (lokální), protože dotyčná adresa není globálně jednoznačná. Naopak se opakuje v každé podsíti. Strukturu takové adresy vidíte na obrázku 3.14.

Význam těchto adres je jednotný a postupně je přiděluje IANA. Zatím jedinou přidělenou adresou z tohoto balíku je výše zmíněná adresa pro domácí



Obrázek 3.14: Rezervovaná výběrová adresa podle EUI-64

agenty. Tabulka 3.4 poskytuje přehled pevně definovaných výběrových adres pro podsíť.

adresa	*význam*
prefix:0:0:0:0	směrovače v podsíti
prefix:fdff:ffff:ffff:ff80 až *prefix*:fdff:ffff:ffff:fffd	rezervováno
prefix::fdff:ffff:ffff:fffe	domácí agenti
prefix::fdff:ffff:ffff:ffff	rezervováno

Tabulka 3.4: Definované lokální výběrové adresy

Dřívější definice adresní architektury IPv6 zavedla – vzhledem k nedostatku zkušeností – pro výběrové adresy značná omezení. Směly být přiřazeny jen směrovačům a bylo zakázáno uvádět je do zdrojové adresy IPv6 datagramu. V současnosti (počínaje RFC 4291) již tato omezení neplatí.

Pokud by vás zajímal podrobnější rozbor problematiky výběrových adres, určitě si přečtěte RFC 4786: *Operation of Anycast Services*.

3.9 Povinné adresy uzlu

V IPv4 mívalo rozhraní zpravidla právě jednu adresu. Existovaly sice výjimky (např. virtuální WWW servery bývaly svého času realizovány tak, že počítač slyšel na několik IP adres pro totéž rozhraní), ale valná většina uzlů toto pravidlo dodržovala.

IPv6 naproti tomu adresami přímo hýří a nejen umožňuje, že rozhraní bude mít více adres, ale dokonce mu to nařizuje. Existuje totiž jasně definovaná minimální množina adres, ke kterým se každý uzel IPv6 sítě *musí* hlásit.

pro počítač

Pro koncový počítač se jedná o následující adresy:

- lokální linková adresa pro každé rozhraní
- všechny individuální a výběrové adresy, které mu byly přiděleny
- lokální smyčka (loopback)

- skupinové adresy pro všechny uzly
- skupinová adresa pro vyzývaný uzel pro všechny přidělené individuální a výběrové adresy
- všechny skupinové adresy, jejichž je členem

Nejlépe si to demonstrujeme na příkladu. Vezměme počítač s jednou síťovou kartou, která má MAC adresu 00:2a:0f:32:5e:d1. Z ní odvozený identifikátor rozhraní je 22a:fff:fe32:5ed1. Počítač má dvě individuální adresy – je členem dvou podsítí v rámci organizace. Jedna má prefix 2001:db8:a319:15::/64 a druhá 2001:db8:a319:3::/64. Kromě toho je členem skupiny ff15::ac07.

Seznam všech adres, na kterých je povinen přijímat data, shrnuje tabulka 3.5. Všimněte si, že ačkoli má několik individuálních adres, díky shodnému identifikátoru rozhraní vystačí s členstvím v jediné skupině pro vyzývaný uzel.

lokální linková	fe80::22a:fff:fe32:5ed1
přidělená individuální	2001:db8:a319:15:22a:fff:fe32:5ed1
přidělená individuální	2001:db8:a319:3:22a:fff:fe32:5ed1
lokální smyčka	::1
všechny uzly v rámci rozhraní	ff01::1
všechny uzly v rámci linky	ff02::1
vyzývaný uzel	ff02::1:ff32:5ed1
přidělená skupinová	ff15::ac07

Tabulka 3.5: Příklad povinných adres pro počítač

pro směrovač

Směrovač se povinně musí hlásit ke všem adresám jako počítač a navíc k následujícím:

- výběrová adresa pro směrovače v podsíti (pro každé rozhraní, kde funguje jako směrovač)
- skupinové adresy pro všechny směrovače

Kdyby výše zmiňovaný počítač byl směrovačem, musel by na rozhraní, které popisujeme, vedle adres uvedených v obrázku 3.5 poslouchat navíc na adresách, jež shrnuje obrázek 3.6. Předpokládám v něm, že směrovač působí jako domácí agent. Proto mu byla přidělena výběrová adresa domácích agentů pro obě podsítě.

3.10 Dosahy adres

IPv4 původně počítalo výlučně s celosvětově jednoznačnými adresami. Později bylo doplněno několik lokálních adres (10.0.0.0, 192.168.0.0 a spol.),

směrovače v podsíti	2001:db8:a319:15::
směrovače v podsíti	2001:db8:a319:3::
vyzývaný uzel	ff02::1:ff00:0
přidělená výběrová	2001:db8:a319:15:fdff:ffff:ffff:fffe
přidělená výběrová	2001:db8:a319:3:fdff:ffff:ffff:fffe
vyzývaný uzel	ff02::1:ffff:fffe
všechny směrovače na rozhraní	ff01::2
všechny směrovače na lince	ff02::2
všechny směrovače v místě	ff05::2

Tabulka 3.6: Rozšíření povinných adres pro směrovač

které mají platnost omezenu na místní síť a nesmí být předávány do Internetu. Se zavedením skupinových adres se objevila otázka dosahu jednotlivých skupin. Ve světě IPv4 je řešena prostřednictvím životnosti datagramů (TTL). Pro jednotlivé linky lze definovat určitý limit a pokud má datagram TTL nižší než uvedená hodnota, nebude dotyčnou linkou odeslán.

Je celkem zřetelné, že adresy s omezeným dosahem byly do IPv4 doplňovány dodatečně a v podstatě se hledalo, jak využít existující mechanismy pro jejich implementaci. Autoři IPv6 naproti tomu popadli příležitost za pačesy a rozhodli se zapracovat koncept dosahu adres jako jeden ze standardních prvků adresace. Věnuje se mu RFC 4007: *IPv6 Scoped Address Architecture*.

Intuitivně je pojem dosahu celkem jasný. Formálně je definován jako vymezení topologické oblasti sítě, v níž je daná adresa jednoznačná.

Dostupné dosahy se liší podle druhu adresy. Nejjemnější členění mají skupinové, pro které je v současnosti definováno šest stupňů lokality (viz tabulka 3.7, níž najdete číselné hodnoty jednotlivých dosahů a jejich významy). V případě individuálních adres se rozlišují jen dva stupně: lokální pro linku a globální. Výběrové adresy spadají mezi individuální, takže mají i stejné dosahy.

V popsané hierarchii nemusí nutně platit, že větší dosah pokrývá ostře větší část sítě než dosah menší. V řadě případů mohou být dosahy na několika úrovních totožné – například linka bude v drtivé většině případů shodná se správní oblastí (podsítí). Zmíněné dva dosahy budou v reálu platit ve stejné části sítě. Pochopitelně však při postupu hierarchií dosahů směrem vzhůru nesmí dojít ke zmenšení části sítě odpovídající danému dosahu.

zóna V souvislosti s dosahy se též objevuje pojem *zóna*. Jedná se právě o tu část síťové topologie, která odpovídá danému dosahu (adresa je v zóně jednoznačná). Hranice zón procházejí počítači, nikoli linkami. Pochopitelně platí, že celá zóna je vždy zahrnuta do „nadřízené“ zóny většího dosahu. naopak zóny stejného dosahu se nemohou překrývat (jsou buď totožné nebo

dosah	*význam*
1=rozhraní	nepřekročí jediné rozhraní; používá se pro skupinové vysílání do rozhraní pro lokální smyčku (loopback, adresa ::1)
2=linka	dosah je omezen na jednu fyzickou síť (např. Ethernet či pouhou sériovou linku se dvěma účastníky)
4=správa	nejmenší dosah, který musí být konfigurován správcem (čili nelze jej automaticky odvodit z fyzické topologie či dalších informací); obvykle se jedná o podsíť
5=místo	část síťové topologie, která patří jedné organizaci a nachází se v jedné geografické lokalitě, prostě koncová zákaznická síť
8=organizace	pokrývá několik míst náležejících téže organizaci, například pobočky téže firmy v různých městech
E=globální	celosvětový dosah

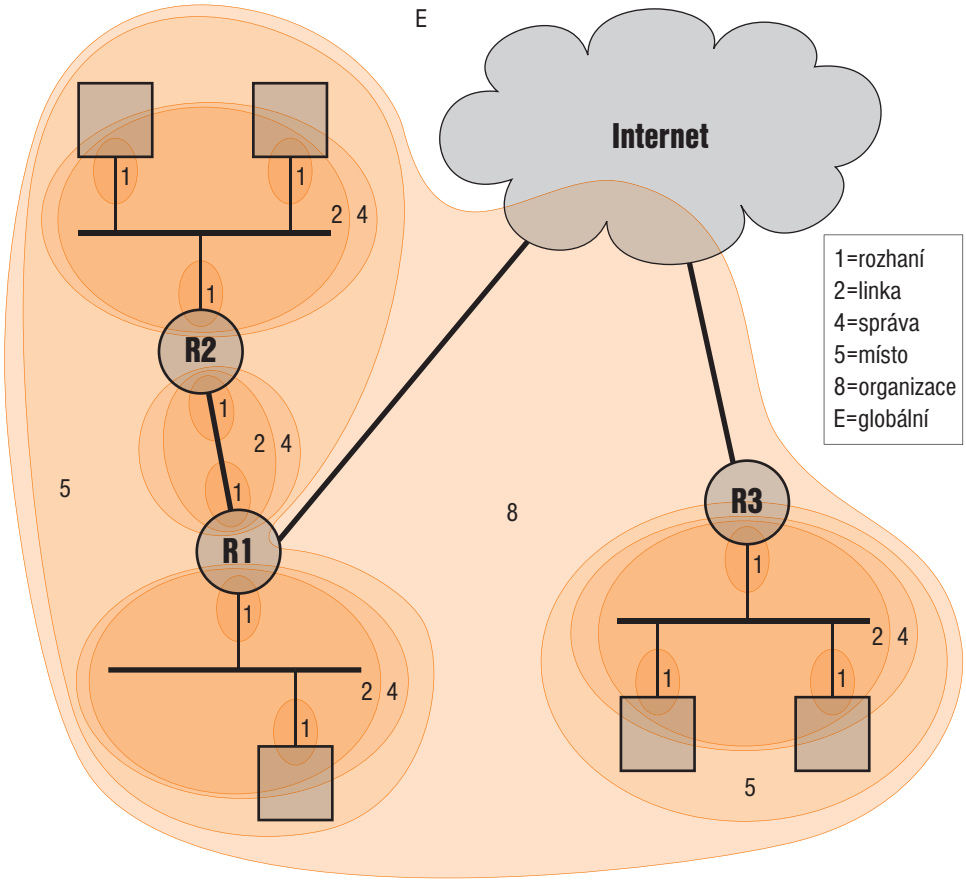
Tabulka 3.7: Dosah skupinových adres

zcela oddělené). Z hlediska směrování musí být zóna souvislá – pokud by datagram během přepravy opustil zónu, mohlo by dojít k dezinterpretaci jeho adresy.

Rozložení zón (čili dosahů) ilustruje obrázek 3.15 na příkladu jakési sítě zasahující do dvou lokalit. Síť v levé lokalitě zahrnuje tři podsítě, propojené směrovači R1 a R2, síť v pravé lokalitě má jen jednu podsíť. Všimněte si, jak jsou konstruovány zóny odpovídající oběma místům. Nejzapeklitější problém představuje zóna organizace, která má být souvislá. Nabízejí se pro ni dvě základní řešení: Buď bude ve spolupráci s poskytovatelem Internetu tažena jeho páteřní sítí (např. v podobě virtuální sítě), nebo zůstane poskytovatel zcela mimo, organizace si vytvoří tunel propojující směrovače R1 a R3 a použije jej pro přenos skupinových dat mezi oběma pobočkami. Pak by rozhraní směrovačů R1 a R3 zajišťující připojení k Internetu zůstala mimo zónu připojené organizace. Místo nich by do této zóny patřila rozhraní reprezentující propojující tunel. Obrázek 3.15 znázorňuje první variantu, i když pravděpodobnější je druhá.

Hranice některých zón jsou jasně dány z logiky věci – například pro linku. U jiných (jako je třeba místo či organizace) je třeba hranici určit konfigurací odpovídajících zařízení. Například na směrovači *R1* je třeba nastavit, že obě rozhraní vnitřní sítě patří do zóny místo, zatímco rozhraní k poskytovateli nikoli.

Jelikož je adresa jednoznačná jen v rámci zóny a počítač může být členem několika zón stejného dosahu (například *R1* patří do tří různých linkových zón), může dojít k situaci, že se stejná adresa objeví v několika zónách.



Obrázek 3.15: Zóny (dosahy) v připojené síti

identifikátor zóny

Aby se tyto adresy daly navzájem rozlišit, zavádí RFC 4007 identifikátory zón. Přestože pracovní verze dokumentu i některé implementace používají identifikátory kombinující dosah a pořadové číslo (např. *link1*), RFC dává přednost jednoduchosti a doporučuje používat přirozená čísla. Dosah zóny se odvodí z vlastní adresy.

Kompletní zápis adresy pak má tvar *adresa%zóna*. Například pokud bude spodní Ethernet připojený ke směrovači R1 reprezentován identifikátorem linkové zóny 1, bude mít skupinová adresa pro všechny uzly na této lince plný tvar ff02::1%1.

Identifikátory se přidělují interně v rámci každého počítače a nejsou navzájem synchronizovány se sousedy v téže zóně. Jejich jediným cílem je nabídnout prostředek pro identifikaci zón v rámci jednoho stroje, aby mohly figurovat ve směrovacích tabulkách a podobných strukturách.

RFC 4007 také počítá s konceptem implicitní zóny (s identifikátorem 0), která se dosadí, pokud adresa neobsahuje identifikátor zóny. Příkladem použití mohou být globální adresy, kde existuje jediná zóna a tudíž nemá smysl ji explicitně uvádět.

Drobný komentář si zaslouží změny, jimiž postupně prochází sortiment definovaných dosahů. Zatím směřují jednoznačně ke zjednodušení, během několika let zmizely dva dříve definované dosahy. Individuální adresy původně obsahovaly ještě dosah lokální pro místo, který odpovídal dosahu 5 u skupinových adres. Ovšem dlouhodobě se nedařilo najít uspokojivou definici „místa“ a projevilo se několik dalších neduhů, proto je RFC 3879 v roce 2004 zrušilo.

Pro skupinové adresy zůstal dosah pro místo zachován, zato zmizela z nabídky podsíť. Dřívější specifikace definovala trojici po sobě jdoucích dosahů (2 pro linku, 3 pro podsíť a 4 pro správu), které v praxi obvykle představovaly tutéž zónu. Každá linka obvykle bývá z hlediska IP podsítí a správcem nastavená hranice přirozeně odpovídá podsíti. Pocit určité nadbytečnosti vyústil ve vypuštění prostředního z těchto tří dosahů. Kdyby se ještě dosah „pro správu“ přejmenoval na srozumitelnější „pro podsíť“, bylo by to ideální, ovšem přednost dostala zpětná kompatibilita a konzistentnost názvů.

3.11 Výběr adresy

Přiřazení několika různých adres témuž rozhraní vyvolává nový problém: kterou z nich si vybrat. Představte si modelovou situaci: do svého WWW klienta napíšete *www.kdesi.cz*, počítač si prostřednictvím DNS zjistí jeho adresu a dostane řekněme pět odpovědí. Rozhraní, kterým se chystá odeslat data, má přiděleno šest adres. Co tedy vyplnit do hlavičky datagramu? Jakou adresu pro cíl a odesilatele zvolit?

Odpověď poskytuje RFC 3484: *Default Address Selection for Internet Protocol version 6 (IPv6)*, jež stanoví přesný postup pro výběr adres v odesílaném datagramu. Jeho cílem je, aby se všechny implementace IPv6 chovaly konzistentně a předvídatelně. Na druhé straně ovšem ponechává správci stroje možnost ovlivňovat výběr adres nastavením určitých priorit.

kandidátské adresy

Základem algoritmu je výběr z několika kandidátek, případně jejich seřazení podle vhodnosti. Aplikace, která chce komunikovat, někdy má k dispozici cílovou IP adresu. Pak je kandidátka jen jedna a výběr cílové adresy odpadá. Většinou je ale cíl zadán doménovým jménem. Aplikace v tom případě nejprve zavolá systémovou službu *getaddrinfo( )*, kterou požádá o převod DNS jména na IP adresu. Z DNS dotazu vzejde *seznam kandidátek na cílovou adresu* a ten je následně podle níže uvedených pravidel uspořádán od nejvhodnější adresy po nejméně vhodnou. Seřazený seznam tvoří výsledek volání *getaddrinfo( )*, který dostane aplikace.

Ta z něj vybere jednu adresu (slušná aplikace tu první) a požádá o odeslání dat na ni. Nyní je cílová adresa jednoznačně dána a je třeba k ní vybrat nejvhodnější zdrojovou adresu. *Kandidátkami na zdrojovou adresu* se stanou všechny individuální adresy přiřazené rozhraní, kterým budou odesílána data k danému cíli. To znamená, že různé cílové adresy mohou mít různé kandidátské sady pro odesilatele. Pokud data odesílá směrovač, může mezi kandidátky zařadit individuální adresy ze všech rozhraní, na kterých předává data. Uplatněním sady pravidel se z kandidátek vybere Miss Odesilatel a ta bude použita v datagramu.

Jestliže se komunikace nezdaří, může aplikace později zkusit další ze seznamu cílových adres a výběr odesilatele se bude opakovat.

tabulka politik

K vyjádření místních preferencí slouží tak zvaná *tabulka politik (policy table)*. Její záznamy obsahují po třech položkách: adresní prefix, prioritu (precedence) a značku (label). Vyhledává se v ní podobně jako ve směrovací tabulce – bude použita ta položka, která má nejdelší shodný prefix s posuzovanou adresou. Určí prioritu a značku posuzované adresy.

prefix	*priorita*	*značka*
::1/128	50	0
::/0	40	1
2002::/16	30	2
::/96	20	3
::ffff:0:0/96	10	4

Obrázek 3.16: Implicitní tabulka politik

Priorita obecně vyjadřuje výhodnost dané adresy jako cílové. Vyšší priorita znamená, že této adrese by se měla dát přednost. Prostřednictvím značek pak lze sdělit, že určitý pár adres spolu mimořádně dobře ladí. V jejich

případě se posuzuje jen rovnost či nerovnost. Mají-li dvě adresy shodnou značku, dobře se k sobě hodí a bude jim dána přednost.

Vhodným nastavením tabulky politik může správce systému přizpůsobit chování výběru adres svým potřebám. Jestliže tuto možnost nevyužije, použije se implicitní tabulka podle obrázku 3.16. Za pozornost v ní stojí vysoká priorita implicitní položky (druhý řádek), které vyhoví jakákoli adresa. Díky ní budou mít například globální IPv6 adresy přednost před adresami 6to4 (viz kapitola 12.3 na straně 261) s prefixem 2002::/16. Za chvilku bude jasné proč.

Algoritmus pro volbu adresy definuje dvě sady pravidel. Jedna se vztahuje na zdrojovou adresu a druhá na cílovou. Podívejme se nejprve na výběr odesilatele.

1. **Preferovat totožné adresy.**
   Pokud je některá z adres totožná s cílovou, vybere ji.
2. **Preferovat odpovídající dosah.**
   Jestliže mají zdrojové adresy rozdílný dosah, seřadí si je tak, aby $dosah(SA) < dosah(SB)$. Pak pokud je $dosah(SA) < dosah(cíl)$ vybere SB, jinak vybere SA.
3. **Vyhýbat se odmítaným adresám.**
   Je-li jedna adresa preferována a druhá odmítána (jedná se o fáze automaticky konfigurované adresy, viz strana 122), vybere preferovanou.
4. **Preferovat domácí adresy.**
   Pokud je jedna z adres zároveň domácí i dočasnou adresou (mobilní počítač je doma – viz kapitola 11 na straně 223), vybere ji. Jinak je-li SA domácí a SB dočasná, vybere SA.
5. **Preferovat odchozí rozhraní.**
   Je-li SA přidělena rozhraní, kterým budou odeslána data k danému cíli, a SB nikoli, vybere SA.
6. **Preferovat shodné značky.**
   Pokud platí $značka(SA) = značka(cíl)$ a $značka(SB) \neq značka(cíl)$, vybere SA.
7. **Preferovat veřejné adresy.**
   Když je SA veřejná a SB dočasná chránící soukromí, vybere SA.
8. **Použít nejdelší shodný prefix.**
   Nerozhodlo-li žádné z předchozích pravidel, použije tu z dvojice zdrojových adres, která má delší shodný prefix s cílem.

Obrázek 3.17: Výběr zdrojové adresy

výběr zdrojové adresy

Příslušná pravidla shrnuje obrázek 3.17. Vycházejí ze situace, že máme porovnat vhodnost dvou potenciálních zdrojových adres SA a SB pro daný cíl. Řada pravidel je symetrických a měla by obsahovat ještě jednou tutéž větu,

v níž si však SA a SB prohodí role. Pro zjednodušení toto opakování vynechávám a ponechám je na inteligenci čtenáře. Pravidla se aplikují postupně v uvedeném pořadí. Jakmile některé rozhodne, neberou se další v potaz. Jestliže nerozhodne žádné z pravidel, ponechává se volba mezi SA a SB na implementaci.

Porovnáním jednotlivých dvojic z odpovídající kandidátské množiny se určí nejvhodnější zdrojová adresa pro daný cíl. Budeme ji dále označovat *odesilatel*(*cíl*).

Podívejme se na příklad. Řekněme, že počítač má jediné síťové rozhraní, jemuž byly přiděleny následující tři adresy:

1. fe80::abcd (lokální linková)
2. 2002:93E6:3149:1::abcd (6to4)
3. 2001:db8:1:1::abcd (globální individuální)

Nyní má odeslat datagram na adresu 2002:95aa:37fe:5::4321 a potřebuje z výše uvedené trojice adres vybrat nejvhodnější. Předpokládejme, že používá standardní tabulku politik podle obrázku 3.16.

První adresa skončí na pravidle číslo 2, protože její lokální linkový dosah je menší než globální dosah cíle. Mezi druhou a třetí adresou rozhodne až pravidlo číslo 6, z hlediska pravidel 1 až 5 mezi nimi není rozdíl. Tabulka politik přiřadí cílové adrese značku 2, stejně tak jako druhé adrese. Naproti tomu třetí adresa obdrží značku 1. Adresa 2002:93e6:3149:1::abcd má stejnou značku jako cílová, dostane proto přednost a bude použita jako zdrojová adresa.

výběr cílové adresy

Seřazení kandidátů na cílové adresy je složitější. Při porovnávání vhodnosti cílových adres se mimo jiné zvažuje, jak dobře se k sobě hodí se zdrojovou adresou. Proto se pro každou z kandidátek nejprve výše uvedeným postupem určí nejvhodnější odesilatel. Seznam kandidátek se pak uspořádá podle pravidel obsažených na obrázku 3.18. Tentokrát se porovnává vhodnost dvou potenciálních cílových adres DA a DB. Opět je řada pravidel symetrických, opět rozhoduje první použitelné. Postupným porovnáním jednotlivých dvojic se seznam kandidátek na cíl uspořádá od nejvhodnějších po ty nejméně vhodné.

Jako příklad použijme opět počítač z příkladu pro výběr zdrojové adresy. Řekněme, že DNS dotaz pro určité jméno vydal následující trojici adres:

1. 2002:95aa:37fe:5::4321 (odesilatel 2002:93E6:3149:1::abcd)
2. 2001:db8:cccc:5::4321 (odesilatel 2001:db8:1:1::abcd)
3. 2001:8800:b1a:5::4321 (odesilatel 2001:db8:1:1::abcd)

1. **Vyhýbat se nepoužitelným cílům.**
   Pokud se o DB ví, že je nedosažitelná, nebo pro ni neexistuje žádná použitelná zdrojová adresa, dá přednost DA.
2. **Preferovat odpovídající dosah.**
   Je-li $dosah(DA) = dosah(odesilatel(DA))$ a $dosah(DB) \neq dosah(odesilatel(DB))$, dá přednost DA.
3. **Vyhýbat se odmítaným adresám.**
   Pokud je $odesilatel(DB)$ odmítaná adresa a $odesilatel(DA)$ preferovaná, dá přednost DA.
4. **Preferovat domácí adresy.**
   Je-li odesilatel pro některou z cílových adres zároveň domácí i dočasnou adresou, dá přednost této cílové adrese. Jinak pokud je $odesilatel(DA)$ domácí adresa a $odesilatel(DB)$ dočasná, dá přednost DA.
5. **Preferovat shodné značky.**
   Když je $značka(odesilatel(DA)) = značka(DA)$ a $značka(odesilatel(DB)) \neq značka(DB)$, dá přednost DA.
6. **Preferovat vyšší prioritu.**
   Je-li $priorita(DA) > priorita(\mathrm{DB})$, dá přednost DA.
7. **Preferovat nativní přenos.**
   Pokud je DB dosažitelná zapouzdřeným přenosem (např. tunelem), zatímco DA přímo po IPv6, dá přednost DA.
8. **Preferovat malý dosah.**
   Je-li $dosah(DA) < dosah(DB)$, dá přednost DA.
9. **Použít nejdelší shodný prefix.**
   Pokud jsou obě adresy DA a DB stejného typu (obě IPv6 nebo obě IPv4), dá přednost té z nich, která má delší společný prefix se sobě odpovídající zdrojovou adresou.
10. **Neměnit pořadí.**
    Pokud DA bylo v původním seznamu před DB, dá přednost DA.

Obrázek 3.18: Výběr cílové adresy

V jejich seznamu jsem rovnou uvedl preferované zdrojové adresy, jež jednotlivým kandidátkám vybere výše popsaný algoritmus. Tentokrát při výběru nepomohou ani značky, protože všechny tři kandidátky mají značku shodnou se svou zdrojovou adresou. První rozhodnutí přinese až pravidlo 6, protože priorita první adresy je 30, zatímco priorita zbývajících dvou je o deset vyšší. První adresa je tedy nejhorší. Mezi druhou a třetí pak rozhodne až pravidlo 9. Kandidátka číslo 2 má se svým odesilatelem shodný prefix 2001:db8::/32, zatímco kandidátka číslo 3 jen 2001::/16. Ve výsledku budou proto kandidátky na cílovou adresu seřazeny následovně:

1. 2001:db8:cccc:5::4321
2. 2001:8800:b1a:5::4321
3. 2002:95aa:37fe:5::4321

Jako ideální vychází cílová adresa 2001:db8:cccc:5::4321 a jí odpovídající zdrojová adresa 2001:db8:1:1::abcd, což odpovídá logice – použije se globální individuální adresa ze sítě adresně bližší. Prefix délky 32 bitů patří lokálnímu registru – obvykle poskytovateli Internetu. Lze proto očekávat, že přeprava paketů proběhne v rámci jeho sítě, a tedy velmi efektivně.

RFC 3484 se již dočkalo dost široké implementace, najdete je ve většině současných systémů podporujících IPv6. Ani ono však není bez chyby, v některých situacích implicitní výběr adresy nevykazuje dobré výsledky. Jeho chování sice lze konfigurovat tabulkou politik, praktické provedení tohoto kroku ale naráží na různá úskalí. Běžní uživatelé k němu zpravidla nemají dostatečné znalosti, bývá třeba provést konzistentní nastavení politik u všech strojů v určité části sítě a podobně, jak se můžete dočíst v RFC 5220: *Problem Statement for Default Address Selection in Multi-Prefix Environments: Operational Issues of RFC 3484 Default Rules*.

Proto se pokukuje po něčem novém, dynamickém, flexibilním, nízkotučném, pokud možno samokonfigurujícím se. RFC 5221: *Requirements for Address Selection Mechanisms* shrnuje požadavky na nový způsob výběru adres. Na nástupci RFC 3484 se pracuje, ale na jeho finální podobu si budeme muset ještě počkat.

3.12 Vícedomovci čili multihoming

Vícedomovci jsou opakem bezdomovců. Příklady známých vícedomovců[8] jasně naznačují, že být vícedomovcem je výhodné. V síťové praxi se tímto pojmem označují zákaznické sítě, které jsou připojeny k několika poskytovatelům připojení. Anglicky se tomuto způsobu připojení říká multihoming.

Jeho hlavním cílem je zajistit zákaznické síti spojení s Internetem i v případě, že u některého z poskytovatelů dojde k výpadku a cesta vedoucí

[8] úspěšní podnikatelé, filmové a hudební hvězdy, ...

přes něj se přeruší. Vícedomovci se proto stávají především poskytovatelé služeb, jejichž výpadek by byl kritický a znamenal ztrátu, ať už přímou finanční či poškození jména. Například Seznam by jistě nepotěšilo, kdyby byly jeho servery několik hodin nedostupné. Podobně by kterákoli banka špatně nesla ztrátu spojení svého platebního systému.

Podrobněji cíle vícedomovectví pitvá RFC 3582: *Goals for IPv6 Site-Multihoming Architectures*. Vedle redundance a odolávání výpadkům tu najdete i rozkládání zátěže a zvyšování výkonu díky paralelním přenosům různými sítěmi. Důležitým požadavkem, který zároveň eliminuje některá řešení, je transparentnost vůči vyšším vrstvám. Jestliže dojde k výpadku a provoz je převeden na jiného poskytovatele, měla by navázaná spojení normálně pokračovat a nemělo by to ani omezit možnost navazovat nová v obou směrech.

směrování

Dlužno přiznat, že vícedomovectví je problém, především z hlediska směrování. Současná běžná praxe je taková, že síť, která chce být připojena k několika poskytovatelům Internetu, si založí svůj vlastní autonomní systém (AS), opatří si adresy nezávislé na poskytovateli (Provider Independent, PI) a vstoupí do globálních směrovacích tabulek. To znamená, že má svůj záznam na nejvyšší úrovni směrování v páteřních směrovačích Internetu. Ten šíří všichni poskytovatelé, k nimž je připojena. Standardní směrovací mechanismy se postarají o nalezení optimální cesty k ní i o její aktualizace při změnách v síti. Stejné řešení se používá i ve světě IPv4.

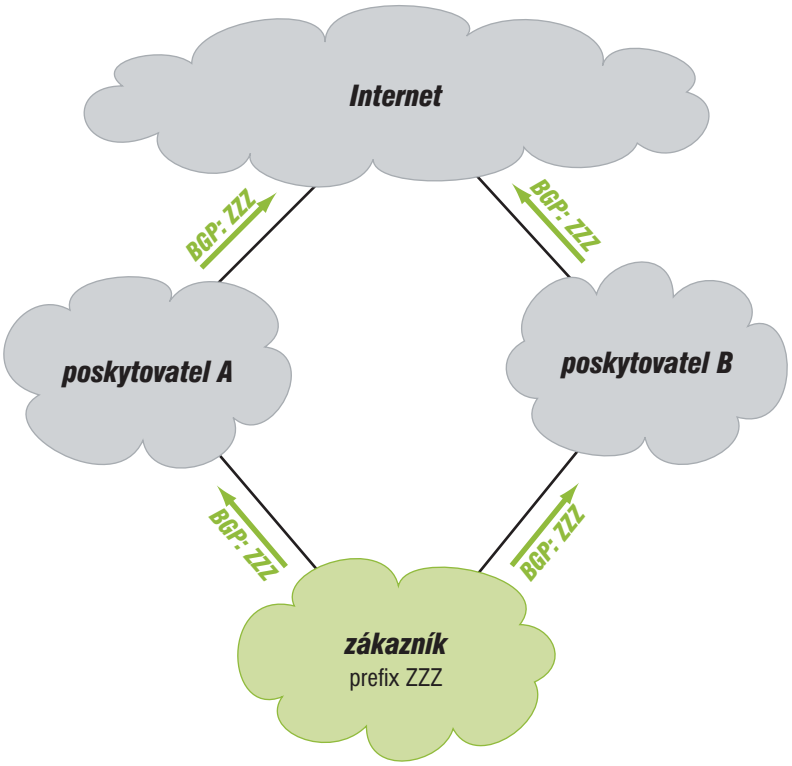
Z pohledu IPv6 je tento přístup jasné fuj. Jedním ze základních východisek při návrhu adresní struktury IPv6 bylo, aby umožňovala masivní slučování (agregaci) prefixů a snižovala tak počet záznamů ve směrovacích tabulkách páteřních směrovačů. Dosavadní praxe je ve zřejmém rozporu s tímto cílem.

Na druhé straně je třeba přiznat, že tento přístup zatím jako jediný skutečně funguje a až na škálovatelnost splňuje všechny ostatní požadavky. Navíc nevyžaduje žádné změny v používaných protokolech a systémech, jen opatření administrativního charakteru. Takže zvolejme „Hanba! Hanba! Hanba, že nám nic jiného nezbývá!“

mobilita

Některé další teoretické cesty naznačuje RFC 4177: *Architectural Approaches to Multi-homing for IPv6*. Jednou z nich je využití podpory mobilních zařízení v IPv6. Podrobněji se jí budu věnovat v kapitole 11 na straně 223. Dopředu jen naznačím, že mobilní zařízení je někde doma a pokud se zrovna toulá, poskytuje ostatním informaci „momentálně jsem k zastižení na adrese X“. Doma jej mezitím zastupuje domácí agent, který mu přeposílá datagramy přicházející na jeho domácí adresu.

Vícedomovecká síť by v tomto případě dostala dva prefixy – *PA* od prvního poskytovatele a *PB* od druhého. Jestliže stroj v ní komunikuje na adrese *PA:X* a cesta přes prvního poskytovatele padne, může využít mobilní me-



Obrázek 3.19: Multihoming prostřednictvím směrování

chanismy a sdělit svým partnerům „momentálně jsem k zastižení na adrese *PB:X*“.

To je velmi hezká myšlenka, ovšem naráží na několik ošklivých problémů. Za prvé mobilita není dosud v implementacích příliš dobře podporována. Za druhé se koncový počítač musí dozvědět, že používané spojení bylo přerušeno a že by měl partnerům oznámit změnu adresy. A třetí hřebíček do rakve představuje zabezpečení. Aby se kdokoli nemohl prohlásit za jiný uzel na cestách, následuje po přijetí zprávy „mám adresu *X* a momentálně jsem k zastižení na adrese *Y*“ test, zda její odesilatel skutečně přijímá data na obou uvedených adresách. Teď jsme ovšem v situaci, kdy jedna z adres nefunguje a test proto nemůže proběhnout úspěšně.

oddělení identifikátoru a lokátoru

Mobilita svým konceptem domácí a aktuální adresy ovšem naznačuje cestu, kterou se současné úvahy o vícedomovectví ubírají. Směřují k oddělení dvou rolí, které IP adresa hraje. Slouží jako *identifikátor*, jehož cílem je jednoznačně určit totožnost počítače, respektive jeho rozhraní. Zároveň ale

hraje úlohu *lokátoru* oznamujícího polohu zařízení v síti a používaného pro směrování datagramů k němu.

Značné úsilí je v současnosti věnováno vývoji mechanismů, které by vedly k rozdělení těchto dvou úloh. Aby každý stroj měl svůj neměnný identifikátor, který by používaly protokoly vyšších vrstev a který by nezávisel na aktuální síťové situaci. Kromě toho by ovšem měl přiřazen jeden či několik dočasných lokátorů využívaných při směrování, které by pružně měnil v závislosti na stavu sítě – při svém pohybu Internetem, při výpadcích a startech linek. Lokátory by sloužily nižším vrstvám komunikační architektury k zajištění vlastních přenosových služeb.

Tento přístup ale znamená významný zásah do významu adresy a vyžaduje úpravu síťové architektury. Některá její část musí provádět mapování mezi identifikátorem a jeho lokátory, vybírat aktuálně nejvhodnější a podobně. Zvažuje se buď přidání nového prvku nebo úprava některé ze stávajících vrstev – nabízí se síťová či transportní vrstva a jejich rozhraní. Problematice se věnuje pracovní skupina IETF nazvaná *Lisp (Locator/ID Separation Protocol)*.

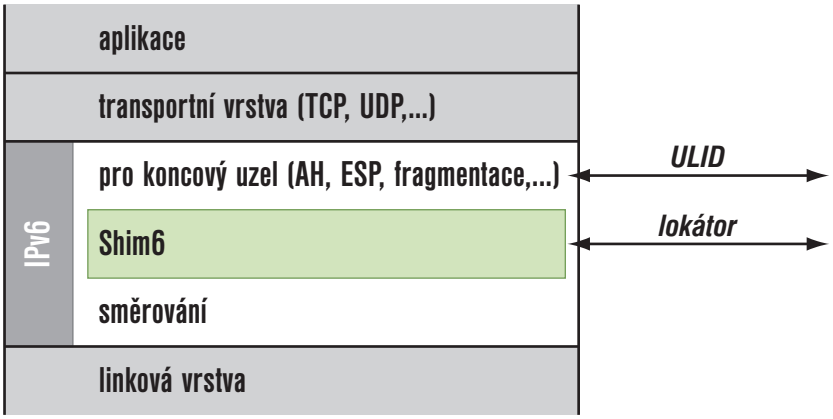
Shim6

O určitý kompromis se snaží koncept nazvaný *Shim6* definovaný v RFC 5533: *Shim6: Level 3 Multihoming Shim Protocol for IPv6*. Slovo „shim“ znamená podložku a mezi programátory se používá pro jednoduchou knihovnu, která převádí jedno aplikační rozhraní na jiné. Docela pěkně to vystihuje jeho funkci.

Shim6 nezavádí samostatný prostor pro identifikátory a neodděluje důsledně identifikátor (ve zdejší terminologii nazývaný ULID, Upper-Layer Identifier) od lokátoru. Místo toho pragmaticky používá v roli identifikátoru první lokátor, s nímž komunikace začala. Počáteční výběr adres proběhne tak jak bylo popsáno v předchozí části a teprve když se komunikace zadrhne, přijde ke slovu Shim6 a pokusí se vybrat nové funkční lokátory. Pro vyšší vrstvy (transportní počínaje) zůstávají nadále v platnosti původní adresy.

Obrázek 3.20 znázorňuje umístění Shim6 v komunikační architektuře. Je zařazen do síťové (IP) vrstvy tak, že ji rozděluje na dvě části. Nad ním se nacházejí komponenty síťové vrstvy určené koncovému uzlu – zabezpečení, fragmentace a volby pro příjemce. K případnému přepisování adres dochází až po jejich aplikaci na straně odesilatele, resp. před ní na straně příjemce. Tyto složky se o existenci Shim6 vůbec nedozvědí. Z jejich pohledu (a všech vrstev nad nimi) je Shim6 zcela transparentní. Pod ním se pak nachází ta část IP vrstvy, která zajišťuje směrování.

Shim6 se chová podle známého hesla všech tchýní „Já tady také nemusím být.“ Na začátku komunikace se vůbec neprojeví, ta je zahájena obvyklým způsobem. Spouští se teprve když výměna dat trvá určitou dobu, aby zbytečně nezatěžoval krátkodobou komunikaci (například DNS). Poté



Obrázek 3.20: Začlenění Shim6 do architektury TCP/IP

proběhne mezi oběma partnery výměna čtyř zpráv, kterými se navzájem informují o svých lokátorech a vytvoří tak zvaný Shim6 kontext a vše je připraveno pro případ nouze. Jestliže z druhé strany nedorazí odpověď na zprávu vyzývající k vytvoření kontextu, protějšek nepodporuje Shim6 a komunikace bude pokračovat bez něj.

Pokud došlo k vytvoření Shim6 kontextu, protokol se odmlčí a nijak se neprojevuje až do vzniku problému. Jestliže se komunikace zadrhne (což může zjistit Shim6 sám nebo problém ohlásí vyšší vrstva), pokusí se oba partneři najít ze známých lokátorů pár, který funguje. Když se podaří, začne Shim6 přepisovat adresy v odesílaných datagramech a zároveň k nim přidává rozšiřující hlavičku identifikující kontext. Podle ní příjemce pozná datagram upravený Shim6 a přepíše adresy v něm zpět do původní podoby, kterou měly v době zahájení komunikace a která hraje roli identifikátorů. Případné bezpečnostní prvky a další služby jsou uplatněny až na restaurovaný datagram.

Protokol je dost košatý a zahrnuje celou řadu doplňujících prvků. Jimi se například partneři informují o změnách ve složení dostupných lokátorů, umožňuje aplikacím využívat různé kontexty nebo si vyměňovat různé provozní informace.

3.13 Přidělování adres

IPv6 se do praxe prosazuje sice pomalu, ale z hlediska administrativního je dnes se svým předchůdcem srovnatelný. Procedura přidělování adres je dnes totožná pro oba protokoly: centrální autoritou je *IANA (Internet Assigned Numbers Authority)*, která přiděluje velké bloky adres *regionálním*

registrům (Regional Internet Registry, RIR). Těch je pět a na jejich počtu se nejspíš hned tak něco nezmění. Zeměkouli mají rozdělenu následovně:

AFRINIC	Afrika
APNIC	Asie a Pacifik
ARIN	Severní Amerika
LACNIC	Latinská Amerika
RIPE NCC	Evropa a Blízký východ

Jednotlivé regionální registry přidělují menší bloky *registrům lokálním (Local Internet Registry, LIR)*. Roli lokálních registrů zpravidla zastávají poskytovatelé Internetu. Od nich získávají adresy koncové instituce – zákazníci. Vzhledem k hierarchickému uspořádání přidělovaných rozsahů je zajištěna agregovatelnost.

Pravidla v jednotlivých oblastech oficiálně stanoví vždy příslušný RIR. Tyto organizace však své kroky vzájemně koordinují, navíc jim určitá omezení klade IANA. V praxi jsou proto pravidla všude dost podobná.

pravidla přidělování

Konkrétně v Evropě je aktuálně stanoví dokument ripe-523: *IPv6 Address Allocation and Assignment Policy* ze srpna roku 2011. Podle něj RIPE NCC přiděluje lokálním registrům prefixy délky 32 bitů (v odůvodněných případech i kratší). Aby jej LIR mohl získat, musí splnit podmínky uvedené v tabulce 3.8. Stručně řečeno: pokud jste LIR a plánujete do dvou let poskytovat IPv6 služby, máte nárok na 32bitový prefix.

1. Žadatel musí být lokálním registrem.
2. Žadatel musí mít plán alokovat části přiděleného IPv6 prostoru dalším organizacím a sítím do dvou let.

Tabulka 3.8: Podmínky RIPE NCC pro získání oficiálního /32 prefixu

Délku prefixu přidělovaného koncovým sítím ripe-523 konkrétně nepředepisuje, ponechává ji na uvážení poskytovatele (LIR). Zakazuje však přidělování prefixů delších než 64 bitů a v případě prefixů kratších než 48 bitů požaduje zdůvodnění. Nejobvyklejší délkou prefixů pro koncové sítě je 48 bitů, malé sítě (např. domácí) mohou mít prostor omezenější, tedy delší prefix.

Lokální registr má k dispozici alespoň 16 bitů pro rozlišení svých zákazníků. To mu dává prostor přinejmenším pro 65 536 zákaznických prefixů. Pokud by to bylo málo, politika RIR připouští v odůvodněných případech přidělit lokálnímu registru prefix kratší nebo skupinu 32bitových prefixů.

Pokud se týče délky zákaznických prefixů, základním dokumentem je RFC 6177: *IPv6 Address Assignment to End Sites*. Jedná se o poměrně obecný text, který definuje jen základní principy:

- Pro délku prefixů přidělovaných zákazníkům je rozhodující operativa. Čili pravidla, která si vytvořily regionální internetové registry (RIR), v případě Evropy tedy RIPE NCC.
- Koncová síť by měla obdržet vždy dostatečný adresní prostor na to, aby nebyla nedostatkem adres nucena k jejich mapování a překladům, tedy k nasazení IPv6 NAT. Mělo by pro ni být snadné získat dostatečný prostor k vytvoření podsítí.
- Rozhodnutí o délce přiděleného prefixu by mělo vzít v úvahu i snadnost delegace reverzního DNS (viz kapitola 9 na straně 191). V praxi toto doporučení znamená delegovat prefixy, jejichž délka je pokud možno dělitelná čtyřmi, aby zahrnovala vždy všechny bity šestnáctkových číslic tvořících poddomény v reverzním DNS.
- Dokument se staví proti přidělování adres jednotlivým zařízením, tedy 128 bitů dlouhým prefixům.

RFC 6177 nahradilo dlouhá léta platné (a ne vždy dodržované) RFC 3177. Tato starší verze požadovala, aby se koncovým sítím přidělovaly prefixy jen tří různých délek: 128 bitů pro jednotlivá zařízení, 64 bitů pro sítě, kde jistě nikdy nebude třeba vytvářet podsítě, a 48 bitů pro všechny ostatní, tedy v drtivé většině případů.

Použití unifikované délky prefixů 48 bitů pro valnou většinu koncových sítí má řadu výhod. Usnadňuje změnu poskytovatele či multihoming, protože všichni poskytovatelé přidělují stejně dlouhé prefixy. Vnitřní struktura sítě proto může zůstat stejná u všech prefixů. Odpovídá prefixům již zavedených služeb a protokolů (např. 6to4). Usnadňuje vytváření a stěhování reverzních domén pro DNS, protože lokální část adresy je stále stejně dlouhá.

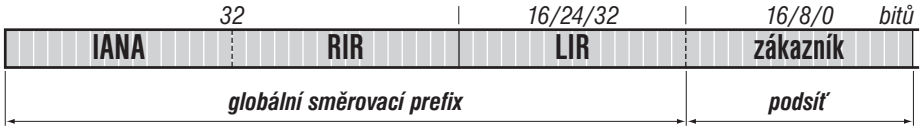
Zároveň je takových prefixů dost – podle analýzy obsažené v RFC 3177 lze na každého obyvatele země přidělit bezmála dvacet 48bitových prefixů, než se začnou objevovat problémy s nedostatkem adres.

Přesto neustále sílily hlasy, že 48bitový prefix je pro některé účely zbytečně velký, že by se mělo šetřit hned od začátku a že by bylo účelné přidělovat malým sítím prefix délky 56 bitů. Ten umožní definovat 256 podsítí (na identifikátor podsítě v něm zbývá 8 bitů), což je pořád ještě pěkná hromádka. Typickým kandidátem pro takovou alokaci je třeba domácí síť připojená přes ADSL. Ta dnes má typicky jednu až žádnou veřejnou IPv4 adresu, ADSL modem v roli NATu a uvnitř jednu síť s počítači adresovanými z neveřejného prostoru. 256 podsítí touto optikou vypadá jako pohádka.

Tomu se postupně přizpůsobila i pravidla jednotlivých regionálních a lokálních poskytovatelů Internetu. Důsledkem je, že lokální síť dnes v závislosti na svých aktuálních a potenciálních potřebách získá prefix délky 48 (běžné sítě), 56 (malé sítě) nebo 64 bitů (velmi malé sítě bez podsítí). Praktické

rozšíření IPv6 je však zatím natolik malé, že reálná pravidla pro jeho poskytování teprve vznikají.

V knize se dále budu držet především 48bitových prefixů, které jsou nejběžnější, nicméně mějte na paměti, že nejsou pravidlem.



Obrázek 3.21: Struktura adresy – kdo přiděluje jednotlivé části

struktura adresy

Když si současnou praxi přidělování promítneme do struktury adresy, dostaneme v její první polovině hezky uspořádanou kompozici podle obrázku 3.21. První část adresy přiděluje IANA regionálním registrům. Její délka kolísá, zpočátku IANA šetřila a přidělovala prefixy délky 23 bitů, v říjnu 2006 ale každý RIR dostal po jednom dvanáctibitovém prefixu. Následuje část přidělovaná regionálním registrem. Zde je hranice poměrně pevná a dohromady s počátkem od IANA tvoří 32bitový prefix přidělovaný lokálním registrům. Ty mají pod kontrolou následujících 16–32 bitů, jimiž definují 48 až 64bitový prefix zákaznické sítě. Zbývající místo do standardní délky prefixu podsítě zaplní identifikátor podsítě o maximální délce 16 bitů. S trochou zjednodušení dostáváme čtyři šestnáctibitové identifikátory, které po řadě přidělují IANA, RIR, LIR a zákazník a které představují nejčastěji se vyskytující strukturu IPv6 adres.

Podle těchto pravidel bylo do poloviny roku 2011 přiděleno přes 7000 prefixů pro lokální registry, z toho zhruba polovina v oblasti spravované RIPE NCC. Celkem v nich byl prostor na více než deset miliard zákaznických 48bitových prefixů. Aktuální čísla si můžete prohlédnout na adrese

www▸ *http://www.nro.net/statistics*

Přehled prefixů přidělených IANA najdete na stránce

www▸ *http://www.iana.org/assignments/ipv6-unicast-address-assignments*

Jak je vidět, o IPv6 adresy je zájem, i když řada přidělených prefixů zatím slouží jen pro strýčka Příhodu. Reálně bylo v polovině roku 2011 ve směrovacích tabulkách ohlašováno zhruba 5500 prefixů, tedy něco přes tři čtvrtiny přidělených. Jejich statistiky připravuje pro pravidelné RIPE Meetingy Gert Döring a najdete je na adrese

www▸ *http://www.space.net/~gert/RIPE/*

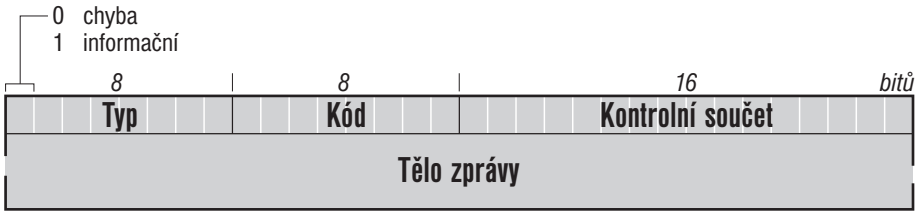
pod názvem *RXX-v6-table.pdf*, kde *XX* je číslo RIPE Meetingu. Hledejte tedy co nejvyšší.

4 ICMPv6

Internet Control Message Protocol (ICMP) je režijním protokolem Internetu. Slouží k ohlašování chybových stavů, testování dosažitelnosti a všeobecně k výměně některých provozních informací. Jeho implementace je povinná v každém zařízení podporujícím IP.

Verze pro IPv6 je definována v RFC 4443: *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification*. Tento dokument však definuje jen základy – formát paketu a základní druhy zpráv. Další typy ICMP zpráv a pravidla pro jejich generování pak doplňují různé komponenty IPv6, jako je objevování sousedů, podpora skupinových adres a podobně. Důsledkem je, že definice ICMPv6 je rozložena do několika RFC.

Skutečnost, že IP datagram nese ICMP zprávu, signalizuje hodnota 58 v položce *Další hlavička*.



Obrázek 4.1: Formát ICMP zprávy

Všechny ICMP zprávy mají jednotný základ, který vidíte na obrázku 4.1. *Typ (Type)* určuje základní druh zprávy. V jeho rámci se může vyskytovat několik podtypů, které jsou identifikovány *Kódem (Code)*. Formát *Těla zprávy (Message body)* pak pochopitelně závisí na jejím typu. Zpravidla obsahuje čtyřbajtovou položku, která buď nese užitečnou informaci, nebo je nevyužita. Za ní pak následuje co největší část datagramu, který vyvolal odeslání dané ICMP zprávy.

Zprávy jsou rozděleny do dvou tříd: na chybové (jejichž *Typ* leží v intervalu od 0 do 127) a informační (*Typ* 128 až 255). Přehled definovaných typů uvádí tabulka 4.1. Kromě hodnot v ní uvedených jsou typy 100, 101, 200 a 201 určeny pro soukromé experimenty a poslední hodnoty obou částí 127 a 225 rezervovány pro případné budoucí rozšiřování ICMP. Aktuální přehled definovaných typů najdete na adrese

www ▸ *http://www.iana.org/assignments/icmpv6-parameters*

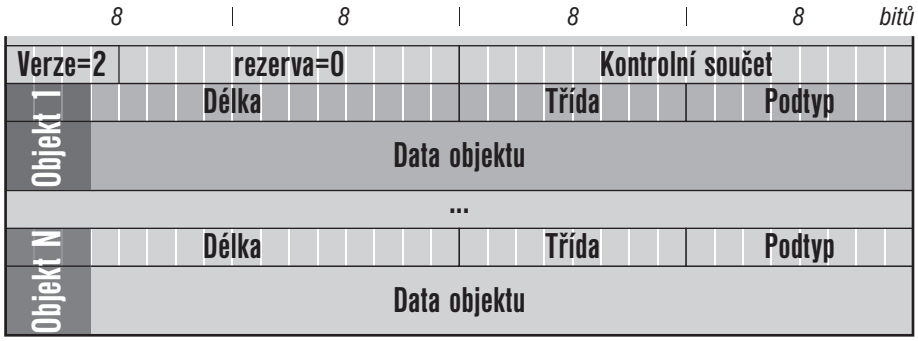
chyby	
1	cíl je nedosažitelný
2	příliš velký paket
3	vypršela životnost paketu
4	problém s parametry
echo	
128	požadavek na echo
129	odpověď na echo
MLD (skupinové adresování)	
130	dotaz na členství ve skupině
131	ohlášení členství ve skupině
132	ukončení členství ve skupině
143	ohlášení členství ve skupině (MLDv2)
objevování sousedů	
133	výzva směrovači
134	ohlášení směrovače
135	výzva sousedovi
136	ohlášení souseda
137	přesměrování
148	žádost o certifikační cestu
149	ohlášení certifikační cesty
informace o uzlu	
139	dotaz na informace
140	odpověď s informacemi
inverzní objevování sousedů	
141	IND výzva
142	IND ohlášení
mobilita	
144	žádost o adresy domácích agentů
145	odpověď s adresami domácích agentů
146	žádost o mobilní prefix
147	ohlášení mobilního prefixu
154	rychlé předávání
objevování skupinových směrovačů	
151	ohlášení skupinového směrovače
152	výzva skupinovému směrovači
153	ukončení skupinového směrovače

Tabulka 4.1: Typy ICMP zpráv

rozšířené ICMP

Formát ICMP zprávy je velice jednoduchý. Ostatně Internet představuje jednu dlouhou řadu vítězství jednoduchých, byť nedokonalých přístupů nad hypersuperdokonalými vzdušnými zámky. Přesto to lidem nedá a červíček „ještě by to mohlo umět…“ pořád hlodá. V případě ICMP vyhlodal RFC 4884: *Extended ICMP to Support Multi-Part Messages*. Definuje rozšíření, kterými lze do těla zprávy přidávat další informace. Zároveň drobně upravuje některé existující zprávy (konkrétně o nedosažitelnosti cíle a vypršení času), kde prvnímu bajtu původně nevyužité čtyřbajtové položky za základní ICMP hlavičkou přiřazuje význam délky vloženého datagramu.

Rozšíření, jehož formát vidíte na obrázku 4.2, se přidává na konec těla ICMP zprávy. Za úvodní hlavičkou poskytující jen číslo verze a kontrolní součet se nachází libovolný počet rozšiřujících objektů. Každý z nich má svou vlastní hlavičku, obsahující jeho *Délku (Length)*, *Třídu (Class-Num)* a *Podtyp (C-Type)*. Za ní pak následují vlastní data rozšiřujícího objektu.



Obrázek 4.2: Rozšíření ICMP zprávy

V době vzniku tohoto textu existoval jediný rozšiřující objekt, jehož prostřednictvím mohou směrovače přepravující paket přidávat do ICMP zprávy informace související s MPLS. Jeho definici najdete v RFC 4950: *ICMP Extensions for Multiprotocol Label Switching*. Zejména v případě nedoručitelnosti datagramu mohou být tyto údaje velmi podstatné.

4.1 Chybové zprávy

nedosažitelnost

Současná verze ICMP definuje čtyři typy chybových zpráv. Má-li položka *Typ* hodnotu 1, oznamuje nedosažitelnost cíle. Posílá ji směrovač, pokud dostal ke zpracování datagram s takovou cílovou adresou, která neumožňuje doručení. *Kód* v ICMP zprávě pak podrobněji specifikuje důvod nedoručitelnosti. Přehled definovaných kódů uvádí tabulka 4.2.

Kód 0 je evidentní. Kód 1 ohlašuje, že datagram porušil nějaká pravidla ve firewallu a jeho odeslání bylo tedy zakázáno správcem. Pokud chce filtrující

0	neznám žádnou cestu k cíli
1	správce zakázal komunikaci
2	mimo dosah zdrojové adresy
3	nedosažitelná adresa (cíl neodpovídá)
4	nedosažitelný port (cíl neodpovídá)
5	zdrojová adresa odporuje vstupně/výstupní politice
6	cesta k cíli je zakázána

Tabulka 4.2: Kódy pro nedosažitelnost cíle

zařízení poskytnout podrobnější informace, může místo jedničky použít jeden z později přidaných kódů 5 a 6. Pokud je nepřípustná zdrojová adresa, sáhne po kódu 5. Jestliže se na černé listině nachází cílová adresa, pošle kód 6.

Kód 2 se použije, pokud by směrovač měl předat datagram do rozhraní, které leží mimo dosah zdrojové adresy. To znamená, že příjemce datagramu by neměl jak poslat zpět odpověď. Kódy 3 a 4 signalizují, že z hlediska směrování je vše v pořádku, ale směrovač nebyl schopen datagram předat, protože následující prvek v cestě se nechová korektně (nepodařilo se zjistit jeho linkovou adresu, na příslušném portu nikdo nenaslouchá a podobně).

nadměrný datagram

Typ s hodnotou 2 ohlašuje příliš velký datagram. IPv6 má ve srovnání se svým předchůdcem podstatně omezený model fragmentace. Zde fragmentuje pouze odesilatel. Pokud má být paket odeslán linkou, jejíž MTU je menší než velikost paketu, směrovač jej zahodí a pošle odesilateli ICMP zprávu typu 2. Čtyřbajtová položka následující za kontrolním součtem obsahuje hodnotu MTU linky, jež problém způsobila. Tyto zprávy se používají například při objevování MTU cesty, které je popsáno v části 2.6 na straně 47.

skončila životnost

Když datagramu vyprší doba platnosti (položka *Maximum skoků* klesne na nulovou hodnotu), směrovač jej zahodí a pošle odesilateli ICMP zprávu s *Typem* 3 a *Kódem* 0. Druhou možnou příčinou pro odeslání zprávy *Typu* 3 je, pokud se příjemce nedočká v daném časovém limitu všech fragmentů skládaného datagramu. V tom případě má *Kód* hodnotu 1.

0	chybná položka v hlavičce
1	neznámý typ v poli *Další hlavička*
2	neznámá volba

Tabulka 4.3: Kódy pro chyby v parametrech

chybný datagram

Zpráva *Typu* 4 signalizuje, že příjemce obdržel datagram, s jehož parametry se nebyl schopen vypořádat. Konkrétní problém je identifikován *Kódem* – viz tabulka 4.3. Čtyřbajtová položka za kontrolním součtem identifikuje pro-

blematický údaj. Udává počet bajtů od začátku datagramu, kde začíná položka, které příjemce nerozuměl.

4.2 Informační zprávy

echo (ping)

RFC 2463 definuje pouhé dvě informační zprávy: výzva a odpověď na echo. Používá je dobře známý program *ping* (resp. *ping6*) k testování, zda je určitý stroj dostupný.

Výzva i odpověď mají stejný formát. Za kontrolním součtem následují dvě šestnáctibitové položky: *Identifikátor* a *Pořadové číslo*. Typické volání *pingu* vyvolá sekvenci žádostí se stejným identifikátorem a postupně narůstajícím pořadovým číslem. Navíc lze do těla vložit data podle potřeby.

Každý IPv6 uzel je povinen na výzvu reagovat odpovědí. V ní zopakuje identifikátor, pořadové číslo a data z výzvy, aby příjemce mohl identifikovat, ke které z jeho výzev se odpověď vztahuje.

informace o uzlu

Zatímco služba echo poskytuje spíše informace o síti (zda funguje a jak dlouho trvá obrátka k cílovému stroji a zpět), RFC 4620: *IPv6 Node Information Queries* zavádí experimentální protokol, kterým se dají získávat jednoduché informace o uzlech. Konkrétně umožňuje zeptat se uzlu na jméno nebo jeho IPv6 či IPv4 adresu. Tyto zprávy se nesnaží konkurovat DNS, ale poskytnout základní informace v případě, že DNS není k dispozici. Protokol má sloužit spíše pro správu sítě, nikoli jako běžná služba koncových počítačů.

Pro své účely zavádí dva typy ICMP zpráv. Dotaz nese *Typ* 139 a jeho *Kód* podrobněji informuje, jaký druh informací požaduje. V těle pak obsahuje vlastní data dotazu – jméno či adresu, k nimž shání protějšek. Odpověď je konstruována podobně a nese ji ICMP zpráva typu 140.

Jako možné využití zmiňuje RFC 4620 mimo jiné i objevování počítačů v síti. Samozřejmě pro ty dobré účely. Určitý problém vidím v tom, že počítače v síti bývají daleko častěji objevovány pro ty špatné účely – aby bylo na co útočit. Na masivní podporu informačního protokolu bych proto příliš nesázel.

další zprávy

Další informační zprávy jsou definovány v jiných RFC dokumentech, protože jsou součástí komplexnějších mechanismů IPv6. Konkrétně zprávy související se členstvím ve skupinách jsou prvkem skupinového adresování IPv6 (viz kapitola 8 na straně 163). Výzva a ohlášení směrovače či souseda stejně jako přesměrování patří do automatické konfigurace a objevování sousedů (kapitola 5 na straně 103). Také návrh podpory mobilních zařízení (kapitola 11 na straně 223) přišel se svými zprávami.

4.3 Bezpečnostní aspekty ICMP

ICMP ze světa IPv4 má ve své skvělé historii několik šrámů, kdy bylo zneužito k omezení funkčnosti sítě. Princip těchto útoků byl celkem jednoduchý: cílový stroj se zahltil haldou ICMP zpráv a skoro nic jiného nemělo šanci projít. Důsledkem bylo, že správci některých serverů či lokálních sítí zablokovali příjem ICMP, což je jednak proti RFC (implementace ICMP je povinná), jednak to omezuje diagnostiku chyb či parametrů sítě.

Aby se správci nemuseli uchylovat k takto drastickým metodám, má ICMPv6 implementována bezpečnostní opatření. Ta by měla zabránit zneužití ICMP pro výše uvedené účely.

První z nich spočívá v tom, že implementace by měla umožnit svému správci nastavit některé kvantitativní parametry – průměrný počet ICMP zpráv za jednotku času či jejich maximální podíl na celkové šířce pásma, které daný stroj generuje. Tato omezení zaručují, že v odesílaných datech zbude dost prostoru na reálný provoz.

Druhá skupina opatření se týká spolupráce s bezpečnostními mechanismy IPv6. Zprávy lze opatřit autentizační či šifrovací hlavičkou a uzel by to měl dělat, pokud pro cíl dané ICMP zprávy existuje bezpečnostní asociace. Pokud má přijatá zpráva bezpečnostní hlavičku, musí být prověřena a pokud neodpovídá, zahodí se. Dále by správce měl mít možnost konfigurovat uzel tak, že přijímá jen zabezpečené ICMP zprávy. Ostatní ignoruje.

5 Objevování sousedů (Neighbor Discovery)

Jedním z dobře známých problémů počítačových sítí je zjištění linkové adresy partnera. Počítač potřebuje poslat někomu data, zná jeho IP adresu, podle ní také ví, že spolu sídlí v jedné lokální síti (řekněme Ethernetu). Aby mu však mohl odeslat paket, potřebuje znát právě cílovou ethernetovou adresu.

IPv4 k tomuto účelu používá samostatný protokol nazvaný Address Resolution Protokol (ARP). V zásadě funguje tak, že odesílající rozešle na všesměrovou IP adresu 255.255.255.255 (všechny stroje v lokální síti) ARP dotaz „Kdo z vás má IP adresu XY?“ Šťastný vlastník mu pak odpoví „To jsem já a moje ethernetová adresa je HyChyKyRyDyTyNy.“

U IPv6 se rozhodli dotyčný mechanismus definovat přímo jako jednu ze základních součástí IP. A když už byli v té revoluci, rovnou vytvořili obecnější nástroj, který kromě hledání linkových adres řeší ještě celou řadu dalších problémů. Výsledek nazvali *objevování sousedů (Neighbor Discovery, ND)*. Slouží k následujícím účelům:

- zjišťování linkových adres uzlů ve stejné lokální síti
- rychlé aktualizace neplatných položek a zjišťování změn v linkových adresách
- hledání směrovačů
- přesměrování
- zjišťování prefixů, parametrů sítě a dalších údajů pro automatickou konfiguraci adresy
- ověřování dosažitelnosti sousedů
- detekce duplicitních adres

Vše je definováno v RFC 4861: *Neighbor Discovery for IP version 6*. Pro svou činnost využívá pěti typů ICMP zpráv, dvě další k nim přidává zabezpečení nazvané SEND. Jejich přehled najdete v tabulce 5.1. V této kapitole popíši jen aspekty související se zjišťováním linkových adres a testováním dosažitelnosti. Automatické konfiguraci (do níž spadá většina ostatních součástí objevování sousedů) věnuji samostatnou kapitolu.

Objevování sousedů	
výzva směrovači	router solicitation
ohlášení směrovače	router advertisement
výzva sousedovi	neighbor solicitation
ohlášení souseda	neighbor advertisement
přesměrování	redirect
SEND	
žádost o certifikační cestu	certification path solicitation
ohlášení certifikační cesty	certification path advertisement

Tabulka 5.1: Typy ICMP zpráv pro objevování sousedů

5.1 Hledání linkových adres

Zjišťování linkové adresy na základě IP se velmi podobá klasickému ARP. Změnily se vlastně jen názvy a především adresa, na kterou tazatel zasílá svůj dotaz.

adresa vyzývaného uzlu

Pro potřeby objevování sousedů byl definován hlouček skupinových adres, na něž se rozesílají dotazy. Všechny mají společný prefix

ff02:0:0:0:0:1:ff00::/104

Uzel, který hledá linkovou adresu pro určitou IPv6 adresu, vezme posledních 24 bitů z hledané IP adresy a připojí je za výše uvedený prefix. Tím získá skupinovou adresu, na kterou zašle svůj dotaz. Takže pokud například hledá linkovou adresu pro

2001:db8:1:1:022a:fff:fe32:5ed1

bude se ptát na skupinové adrese

ff02::1:ff32:5ed1

V terminologii IPv6 se takové adrese říká *adresa pro vyzývaný uzel (solicited node address)*. Skutečnost, že z hledané adresy se přebírá jen spodních 24 bitů, zmenšuje počet skupin, v nichž každý počítač musí být členem. Pokud si vzpomenete na kapitolu 3 na straně 53, IP adresy jsou konstruovány tak, že spodních 64 bitů vychází z identifikátoru rozhraní a k nim je připojen prefix pro horních 64 bitů. Totéž rozhraní může být zařazeno do několika sítí s různými prefixy, nicméně závěrečnou část adresy bude mít stále stejnou. To znamená, že všechny jeho IP adresy budou mít stejnou adresu vyzývaného uzlu.

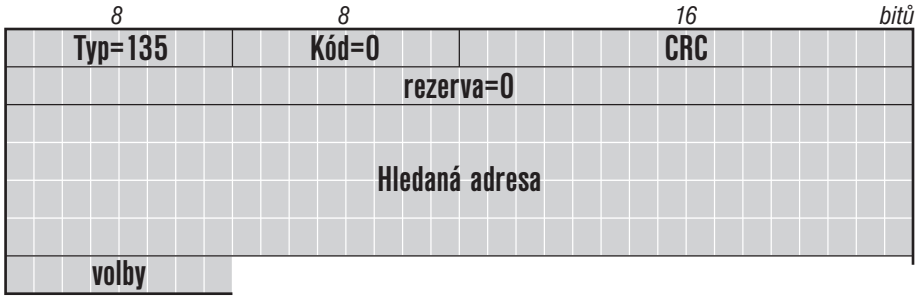
Aby objevování sousedů fungovalo, musí počítač při inicializaci IP pro síťové rozhraní vstoupit do všech skupin odpovídajících adresám vyzývaného uzlu pro všechny adresy přidělené rozhraní. Díky popsanému me-

chanismu bude zpravidla jen jedna. Na druhé straně jsou závěrečné tři bajty dostatečně dlouhé na to, aby ve skupině pro vyzývaný uzel byl zpravidla každý sám. I ve velmi velkých sítích najdete jen vzácně dvojice karet se shodnou hodnotou poslední trojice bajtů. To v praxi znamená, že při hledání linkové adresy nejsou zbytečně obtěžováni ostatní a zpravidla se osloví jen samotný její vlastník.

hledání adresy

Pokud tedy počítač (dále mu budeme říkat „vyzyvatel“) shání linkovou adresu jiného, u nějž zná pouze IP adresu, postupuje následovně: Z cílové IP adresy vytvoří výše popsaným postupem skupinovou adresu vyzývaného uzlu. Na ni pošle speciální typ ICMP zprávy nazvaný *Výzva sousedovi*. Pokud je počítač s danou IP adresou aktivní, bude zapojen do příslušné skupiny a výzvu obdrží. Reaguje na ni *Ohlášením souseda*, které pošle vyzyvateli a které obsahuje informace o jeho linkové adrese.

Každý uzel by si měl udržovat interní datovou strukturu nazvanou *cache sousedů*, ve které má uloženy jejich linkové adresy. Na základě příchodu ohlášení souseda si do této cache zanese novou položku s jeho IP adresou a odpovídající linkovou adresou.



Obrázek 5.1: Výzva sousedovi

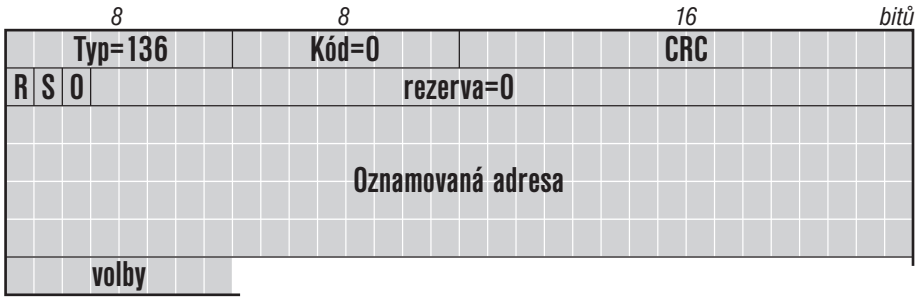
výzva sousedovi

Formát *Výzvy sousedovi* znázorňuje obrázek 5.1. V podstatě obsahuje jedinou informaci – *Hledanou adresu (Target address)*, k níž odesilatel výzvy shání linkovou. K datagramu může připojit volbu, která ohlašuje jeho vlastní linkovou adresu, aby adresát výzvy rovnou věděl, kam má odpovědět.

Obrázek 5.2: Volba *Linková adresa odesilatele*

ohlášení souseda

Na obrázku 5.3 najdete formát ohlášení souseda. Opět nese především IP adresu, které se dotyčné ohlášení týká (ať už bylo vyžádané nebo ne – viz



Obrázek 5.3: Ohlášení souseda

níže). K datagramu se připojuje volba sdělující linkovou adresu, jež k ní náleží. Kromě toho obsahuje tři příznaky: *R* (Router) signalizuje, že odesilatel ohlášení je směrovač. *S* (Solicited) nese informaci, zda ohlášení bylo vyžádáno výzvou sousedovi či nikoli. A konečně *O* (Override) určuje, zda tato informace má přepsat případné dosavadní informace spojené s danou adresou.

aktualizace Uzel může zaslat i nevyžádané ohlášení souseda. Tento přístup se používá v situacích, kdy uzel ví, že došlo ke změně jeho linkové adresy a že by bylo záhodno tuto informaci sdělit ostatním. V takovém případě zašle na skupinovou adresu pro všechny uzly (ff02::1) několik ohlášení souseda, v nichž sdělí svou novou linkovou adresu.

Pokud některý z uzlů má ve své cache sousedů položku s danou IP adresou, aktualizuje si ji. Ostatní nevyžádané ohlášení souseda ignorují (jelikož nemají tuto IP adresu v cache sousedů, s dotyčným počítačem v poslední době nekomunikovali a proto je jeho linková adresa nezajímá). Dlužno podotknout, že tento mechanismus aktualizace není zaručený a je chápán pouze jako nástroj pro zvýšení efektivity a rychlosti šíření změn. Základním nástrojem pro ověřování platnosti linkových adres je

5.2 Detekce dosažitelnosti souseda

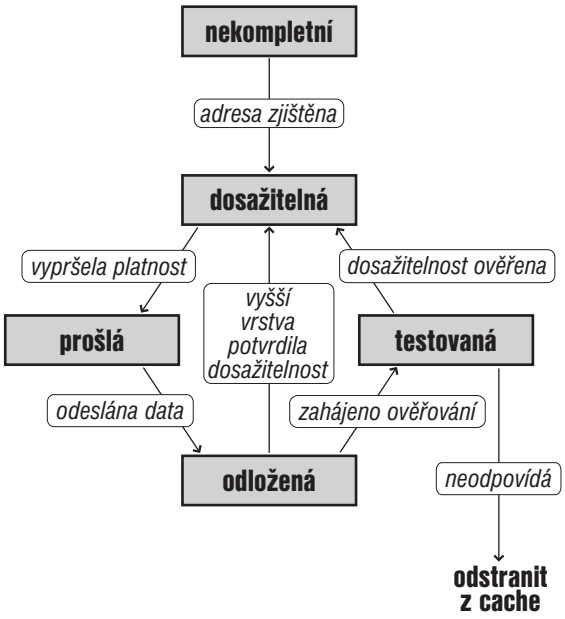
Heslo „důvěřuj, ale prověřuj“ je notoricky známé. IPv6 se jím řídí také. V případě práce se sousedy se konkrétně projevuje v tom, že uzel neustále aktivně sleduje stav dosažitelnosti sousedů, se kterými komunikuje.

K potvrzení, že soused je dosažitelný, poslouží dva základní mechanismy. Buď IP dostává zprávy od vyšší vrstvy (např. TCP), že komunikace zdárně pokračuje a tudíž soused funguje. Pokud takovéto potvrzení nemá, zbývá ještě druhá možnost – ověřit si dosažitelnost vlastními silami. Zašle výzvu sousedovi a pokud dorazí jeho ohlášení, je vše v pořádku.

Základem pro zjišťování nedosažitelnosti sousedů jsou různé stavy, které se přidělují položkám v cache sousedů. Jejich shrnutí se stručnými popisy obsahuje tabulka 5.2. Obrázek 5.4 pak znázorňuje události, které vedou ke změně stavu položky. Tento algoritmus popíši v následujících odstavcích.

nekompletní (incomplete)	linková adresa zatím není známa
dosažitelná (reachable)	cíl je považován za dosažitelný
prošlá (stale)	položce prošla platnost, ale pro cíl nemáme žádná data
odložená (delay)	položce prošla platnost, čekáme, zda vyšší vrstva potvrdí dosažitelnost
testovaná (probe)	právě se testuje

Tabulka 5.2: Stavy položek v cache sousedů



Obrázek 5.4: Změny stavu položky v cache sousedů

Stav *nekompletní* je velmi dočasný a položka jím projde pouze po krátkou dobu v samém začátku své existence. Znamená, že počítači byla odeslána výzva sousedovi s cílem zjistit jeho linkovou adresu a dosud nedorazila odpověď. Jakmile dojde, přejde položka do stavu *dosažitelná*. Pokud by snad ohlášení souseda nedorazilo, znamená to, že dotyčný soused momentálně není funkční a položka bude z cache sousedů odstraněna.

Optimálním stavem je, je-li položka považována za *dosažitelnou*. To znamená, že dosažitelnost souseda byla nedávno potvrzena a nemusíme si s ní dělat těžkou hlavu. Trvanlivost tohoto stavu je časově omezena. Doba, po kterou lze položku považovat za dosažitelnou, je jedním z parametrů sítě a připojeným uzlům ji oznamuje směrovač ve svém ohlášení (viz část 6.1 na straně 119).

Pokud od posledního potvrzení dosažitelnosti uplyne tato doba, položka bude převedena do stavu *prošlá*, ovšem jinak se nic neděje. Uzel se začne starat až v okamžiku, kdy je třeba na danou IP adresu odeslat nějaká data. Provede to stejně, jako by položka byla normálně dosažitelná, ale její stav nyní změní na *odložená*. Tento stav v podstatě říká: „Dosažitelností tohoto souseda si nejsem jist. Před chvilkou jsem mu ale odeslal data a než se pustím do vlastního ověřování, chvilku počkám, jestli mi ji nepotvrdí vyšší vrstva."

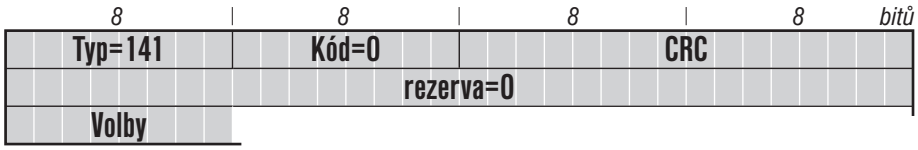
Ve stavu *odložená* položka nikdy nezůstane dlouho. Buď vyšší vrstva potvrdí, že dostala odpověď a cíl je tudíž dosažitelný (položka bude převedena do stavu *dosažitelná*). Druhou možností je, že během daného intervalu toto potvrzení nepřijde. V takovém případě musí IP vrstva dosažitelnost ověřit sama. Odešle danému cíli výzvu sousedovi a stav položky změní na *testovaná*. Odpoví-li, je vše v pořádku a položka se může vrátit do stavu *dosažitelná*.

Jestliže se odpovědi nedočká, výzvu několikrát zopakuje. Pokud soused neodpoví, je považován za nedosažitelného a jeho položka bude odstraněna z cache sousedů.

5.3 Inverzní objevování sousedů

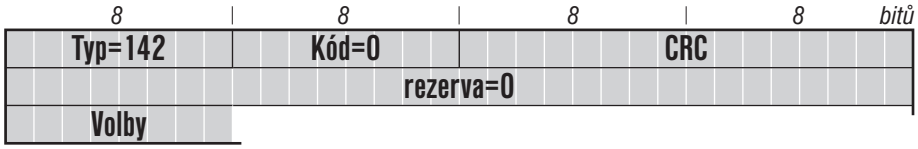
Inverzní objevování sousedů (Inverse Neighbor Discovery, IND) má přesně opačný cíl než to klasické. Řeší situaci, kdy počítač sice zná linkovou adresu svého souseda, ale nezná jeho IPv6 adresu. Původně bylo vyvinuto především pro Frame Relay sítě, kde k takovým stavům dochází, nicméně autoři nevylučují jeho použití i v jiném prostředí s podobnými vlastnostmi. Jeho definici najdete v RFC 3122: *Extensions to IPv6 Neighbor Discovery for Inverse Discovery Specification*.

Funguje velmi jednoduše. Stroj, který se chce dozvědět sousedovu adresu, mu prostřednictvím ICMP pošle *Výzvu* (viz obrázek 5.5). Z pohledu IP ji sice posílá na skupinovou adresu pro všechny uzly na lince ff02::1, ale na linkové úrovni ji adresuje pouze na linkovou adresu cílového stroje (kterou zná). Odesilatel k výzvě povinně musí přiložit volby s oběma linkovými adresami (zdrojovou i cílovou) a může přidat volby se svými IPv6 adresami pro dané rozhraní a MTU linky.



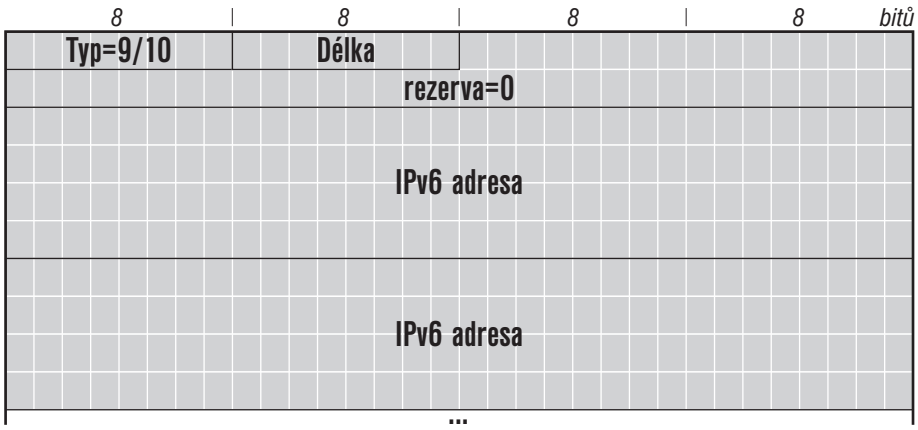
Obrázek 5.5: *Výzva* inverzního objevování sousedů

Vyzvaný počítač reaguje *Ohlášením*, jehož základní formát se (až na typ zprávy) velmi podobá výzvě. Tentokrát je povinné přibalit volbu se zdrojovou linkovou adresou ohlášení a také seznam IPv6 adres pro odpovídající rozhraní. Navíc může přidat i volbu s MTU linky. Své *Ohlášení* posílá dotázaný protokolem ICMP na adresu vyzyvatele. Ten si obdržené informace zanese do cache sousedů a může je dále používat.



Obrázek 5.6: *Ohlášení* inverzního objevování sousedů

Volba *Seznam adres* je jedním z nových prvků zavedených ve specifikaci inverzního objevování sousedů. Její formát představuje obrázek 5.7. Položka *Typ* má hodnotu 9, pokud se jedná o adresy vyzývajícího, a 10, jestliže je volba součástí *Ohlášení*. Kromě *Typu* obsahuje volba již jen obvyklou *Délku* v osmicích bajtů, zarovnání právě na násobky osmi a vlastní adresy.



Obrázek 5.7: Volba *Seznam adres* pro inverzní objevování sousedů

Zbývající volby (pro linkové adresy a MTU) jsou převzaty z klasického objevování sousedů.

5.4 Bezpečnostní prvky objevování sousedů – SEND

Pokud se zajímáte o bezpečnost počítačových systémů, pravděpodobně vás při čtení o mechanismech pro objevování sousedů napadlo, že dává těm zlým mezi námi několik nových možností, jak škodit. Bubák může odpovědět na výzvu sousedovi určenou jinému stroji a snažit se tak stáhnout na sebe jeho datový provoz. Může také předstírat, že stroj je nadále dosažitelný, přestože to již není pravda.

V následující kapitole pak uvidíte, že sortiment možných útoků je mnohem bohatší, protože do objevování sousedů patří i některé prvky automatické konfigurace. Útočník tedy může docílit toho, že si místní počítače přidělí nesmyslné adresy, může se tvářit jako implicitní směrovač pro datový provoz směřující mimo síť a může také automatickou konfiguraci adres zcela zablokovat, když bude ostatním o libovolné jimi zvolené adrese tvrdit, že už je obsazena. Podrobnou analýzu bezpečnostních problémů objevování sousedů najdete v RFC 3756: *IPv6 Neighbor Discovery (ND) Trust Models and Threats*.

Jako reakce na tyto problémy vzniklo *bezpečné objevování sousedů (SEcure Neighbor Discovery, SEND)*, jehož cílem je poskytnout dostatečnou úroveň zabezpečení vyměňovaných zpráv. Původní návrh počítal s uplatněním standardních bezpečnostních prvků IPsec. To se ale ukázalo jako nereálné, protože by stanice pro inicializaci bezpečnostních mechanismů potřebovala příliš mnoho informací. SEND se snaží minimalizovat nároky na zúčastněné.

kryptograficky generované adresy (CGA)

Než se ale pustíme do vlastního SEND, podívejme se na jeho souputníky, kterými jsou *kryptograficky generované adresy* definované v RFC 3972: *Cryptographically Generated Addresses (CGA)*. Jejich cílem je, aby se za vlastníka adresy nemohl prohlásit každý. Vycházejí z asymetrických kryptografických metod. Pokud vám tento pojem nic neříká, najdete jeho stručný popis v přiložené poznámce. Ovšem toto není kniha o kryptografii. Pokud vás zajímají podrobnosti, zkuste si přečíst knihu Williama Stallingse [17] nebo českou, praktičtěji orientovanou [4] od Libora Dostálka.

Východiskem koncepce CGA je dvojice soukromého a veřejného klíče, kterou počítač vlastní. Veřejný klíč pak použije pro generování CGA adresy – spojí jej s několika dalšími položkami, zpracuje hešovací funkcí SHA-1 a počátečních 64 bitů jejího výsledku použije po drobných úpravách jako identifikátor rozhraní, čili adresu. Průvodcem CGA adres je datová struktura nesená volbou *CGA* doplněnou do objevování sousedů. Najdete ji na ob-

Asymetrická kryptografie

Pojmem asymetrická kryptografie se obecně označují takové kryptografické metody, které k zašifrování a dešifrování zprávy používají odlišné klíče. V praxi nejčastěji používanou variantou je kryptografie s veřejným klíčem.

Jejím základem je dvojice klíčů – soukromý a veřejný. Ty jsou spolu svázány takovými matematickými vztahy, aby zpráva zašifrovaná jedním z nich šla dešifrovat pouze tím druhým. Zároveň musí platit, že z veřejného klíče se nedá odvodit soukromý.

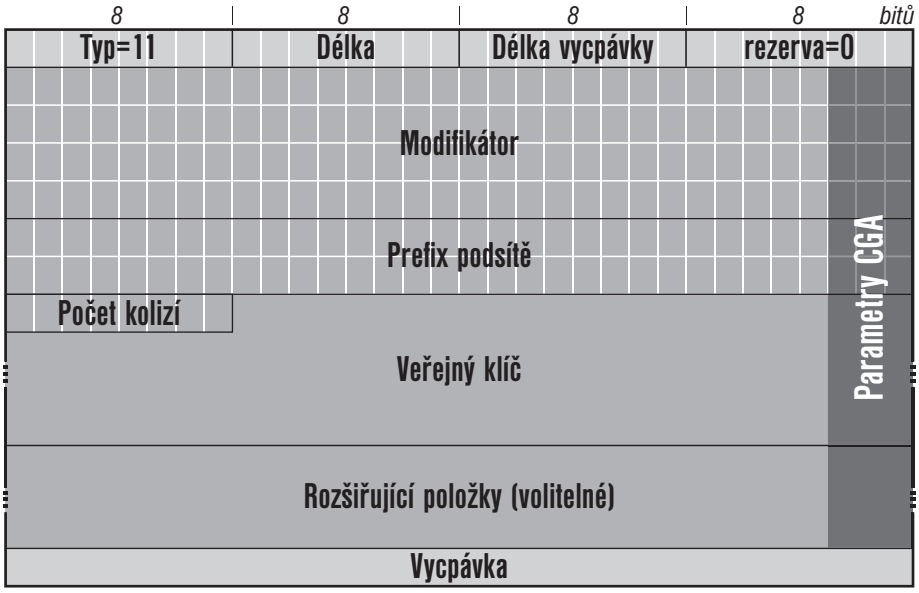
Jak název napovídá, soukromý klíč si jeho držitel uchová v tajnosti a nikdy jej nedá z ruky. Naopak veřejný klíč je volně distribuován a může jej získat každý zájemce.

Dvěma nejčastějšími aplikacemi kryptografie s veřejným klíčem jsou šifrování zpráv a digitální podpis. Jestliže odesilatel A posílá příjemci B zprávu, která má být utajena, zašifruje ji veřejným klíčem B. Tím je zaručeno, že jedině B ji může svým soukromým klíčem dešifrovat.

Pokud chce A zprávu digitálně podepsat, vypočte z ní vhodným vzorcem určitou hodnotu (která se i při drobné změně zprávy divoce mění), říkejme jí pečeť. Tu odesilatel A zašifruje svým soukromým klíčem a přiloží ke zprávě. Kdokoli ji pak může ověřit – spočítá z příchozí zprávy stejným vzorcem pečeť a dešifruje přiloženou pečeť veřejným klíčem A. Pokud se obě shodují, je zpráva autentická, poslal ji skutečně A a nebyla změněna.

Algoritmů realizujících asymetrickou kryptografii je pochopitelně celá řada. Největšího rozšíření se dočkal RSA, který v roce 1977 publikovali pánové Rivest, Shamir a Adleman.

rázku 5.8, vlastní datová struktura obsahující informace o doplňujících položkách je v něm zvýrazněna. Poslouží k ověření, zda je CGA adresa pravá.



Obrázek 5.8: Volba CGA s parametry pro ověření adresy

Aby se zkomplikoval život potenciálním útočníkům, vstupuje do výpočtu 128bitový náhodný modifikátor. Výpočet proto začíná stanovením tohoto náhodného modifikátoru. V této fázi lze navíc uplatnit omezující koeficient označovaný jako *Sec*, který požaduje, aby v určité hash hodnotě bylo 16×*Sec* bitů zleva nulových. Je-li *Sec* nenulový (jeho maximální hodnotou je 7), hledání vyhovujícího modifikátoru se může dost protáhnout. Postup pro výpočet CGA adresy je následující:

1. Uložit do modifikátoru (pseudo)náhodnou 128bitovou hodnotu.
2. Vypočíst hash algoritmem SHA-1 ze zřetězení modifikátoru, 9 nulových bajtů, veřejného klíče a případných rozšiřujících položek. Dokud nejlevějších 16×*Sec* bitů obsahuje nenulovou hodnotu, zvětšit modifikátor o jedničku a opakovat. Pokud je *Sec*=0, tento krok se vynechává.
3. Nastavit počítadlo kolizí na nulu.
4. Zřetězit modifikátor, prefix podsítě, počítadlo kolizí, veřejný klíč a případné rozšiřující položky a vypočítat z této hodnoty SHA-1 hash. Nejlevějších 64 bitů výsledku je označováno jako *Hash1* a tvoří základ adresy.

5. Vytvořit z *Hash1* identifikátor rozhraní tak, že tři jeho nejlevější bity jsou nahrazeny hodnotou *Sec* a také do 6. a 7. bitu se uloží hodnoty podle pravidel pro identifikátory rozhraní v IPv6 (příznaky globální/lokální a individuální/skupinový).
6. Zřetězením prefixu podsítě a identifikátoru rozhraní vznikne IPv6 adresa.
7. Pokud je požadováno, provést detekci duplicit (podrobněji se o ní dočtete v následující kapitole). Při neúspěchu zvýšit počítadlo kolizí a opakovat postup od kroku 4. Po třetí kolizi zastavit a ohlásit chybu.
8. Vytvořit datovou strukturu podle obrázku 5.8 a uložit do ní výsledné hodnoty.

CGA je navrženo tak, aby pravost adresy šla snadno ověřit, když máte k dispozici doprovodnou datovou strukturu. Vzhledem k jednosměrnosti hešovacích funkcí[1] je ale vyloučeno, aby si útočník k již existující adrese vytvořil vyhovující datovou strukturu s vlastními klíči. Může pouze zkopírovat informace, které poskytl její skutečný vlastník. K nim ovšem nemá soukromý klíč, takže nedokáže zprávy digitálně podepsat a jeho případné falsifikáty odesílané z této adresy budou snadno odhaleny. CGA tedy poskytuje důvěryhodné propojení mezi adresou a klíčem.

SEND

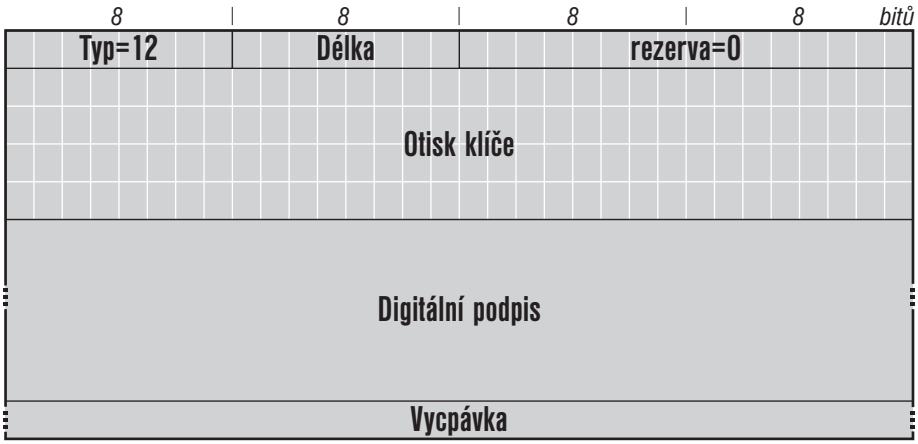
Nyní se můžeme pustit do bezpečného objevování sousedů, jak je definováno v RFC 3971: *SEcure Neighbor Discovery (SEND)*. Jeho jádrem je definice několika nových voleb pro objevování sousedů a popis chování jednotlivých účastníků.

digitální podpisy

Klíčovou volbou je *RSA podpis (RSA Signature)*, jejíž formát vidíte na obrázku 5.9. Jejím prostřednictvím lze každou zprávu související s objevováním sousedů digitálně podepsat a prokázat tak její autentičnost. Dvě nejdůležitější položky volby představují *Otisk klíče (Key hash)*, jehož prostřednictvím lze identifikovat veřejný klíč pro ověření podpisu, a vlastní *Digitální podpis (Digital signature)*. Algoritmem RSA jsou podepsány zdrojová i cílová adresa a celá zpráva ležící před podpisem (jmenovitě první řádek ICMP zprávy udávající *Typ* a *Kód*, celá základní hlavička objevování sousedů a všechny volby ležící před podpisem).

Při příchodu se podepsaná zpráva ověří. Poslouží k tomu veřejný klíč identifikovaný svým otiskem. Může dorazit ve volbě *CGA* této zprávy, nebo jej přijímající stroj mohl získat již dříve. Pokud digitální podpis odpovídá, je zpráva považována za bezpečnou. V opačném případě zpráva není bezpečná a její další osud závisí na konfiguraci příjemce. Jestliže přijímá pouze bezpečné zprávy, bude zahozena. V opačném případě ji přijme, ale zachází s ní stejně jako se zprávami, které bezpečnostní prvky vůbec nemají.

[1] Znáte-li vstupní hodnoty, snadno spočítáte výsledek. Z výsledku se ale nedají odvodit vstupní hodnoty, zbývá jen jejich hledání hrubou silou systémem pokus-omyl.



Obrázek 5.9: Volba *RSA podpis*

RFC 3971 požaduje, aby ve výchozím nastavení počítač přijímal bezpečné i nebezpečné zprávy. To je nutné zejména pro přechodné období, kdy SEND zdaleka není implementováno všude a je nutno zajistit vzájemnou spolupráci zařízení s ním a bez něj. Pochopitelně bezpečné zprávy mají přednost. Správce navíc musí mít k dispozici konfigurační nástroje, jak zakázat příjem nebezpečných zpráv. Potom jsou nepodepsané zprávy ohlašování sousedů, stejně jako zprávy s neplatným podpisem, zcela ignorovány.

Hlavní výhoda SEND proti standardním bezpečnostním mechanismům IPv6 označovaným jako IPsec (viz kapitola 10 na straně 201) spočívá v jednoduchosti a minimální režii. Zatímco IPsec vytváří bezpečnostní asociace a rafinovanými protokoly si vyměňuje použité klíče a algoritmy, zde stačí jedna zpráva obsahující rozšíření *CGA*, aby si protistrana ověřila, že odesilatel skutečně disponuje uvedenou adresou a párem klíčů, které se k ní váží.

Nevýhodou SEND je jeho těsná vazba na CGA adresy. Protokol je schopen poskytnout ochranu jen pro ně, nedokáže zabezpečit obecné IPv6 adresy. Definuje celkem čtyři volby rozšiřující sortiment voleb pro zprávy objevování sousedů:

- *CGA* – viz obrázek 5.8
- *RSA podpis* – viz obrázek 5.9
- *Časová značka (Timestamp)* – aktuální čas
- *Unikát (Nonce)* – náhodná data

Poslední dvě poskytují ochranu proti opakování, aby si vetřelec nemohl ukládat starší platné zprávy a později je znovu odesílat. Tyto prostředky

v kombinaci s CGA adresami dokáží zabránit většině neplech při objevování sousedů. Nechrání ale před lžisměrovači.

Automatická konfigurace popsaná v následující kapitole zahrnuje i nástroje pro vytvoření použitelné směrovací tabulky. Zlý stroj si může vytvořit CGA adresu a posílat podepsané korektní zprávy, v nichž se ovšem prohlásí za směrovač a protlačí do směrovacích tabulek místních počítačů záznamy, jimiž na sebe stáhne jejich datový provoz. Řešením tohoto problému je certifikace směrovačů a jimi ohlašovaných údajů prostřednictvím tak zvané *certifikační cesty (certification path)*.

certifikační cesta

Autorita, které koncový počítač důvěřuje, udělí směrovači certifikát. Může být buď obecný ve stylu „potvrzuji, že stroj s adresou X je směrovač“, nebo může udělit směrovači oprávnění ohlašovat jen určité prefixy. Tedy „potvrzuji, že stroj s adresou X je směrovač a může ohlašovat prefixy M, N a Q“. Obecný certifikát činí směrovač důvěryhodným pro všechny prefixy, konkrétní jen pro ty v něm obsažené.

Uzly, které chtějí ověřovat oprávněnost směrovačů, musí předem znát veřejný klíč autority, protože v době, kdy jej potřebují použít, ještě nemají směrovací tabulku a nemohou tedy získávat informace ze sítě. Předpokládá se, že klíč do nich uloží správce systému. Klíčů pochopitelně může být víc a navíc směrovač nemusí nutně být potvrzen přímo autoritou známou klientovi. Stejně jako v jakémkoli jiném certifikačním systému lze budovat cesty důvěry. Jestliže klient důvěřuje autoritě A (má její veřejný klíč), ta certifikuje autoritu B a autorita B certifikuje směrovač X, je z pohledu klienta směrovač X důvěryhodný, protože k němu dokáže vybudovat certifikační cestu od známého zdroje. Ten je v terminologii SEND pojmenován *kotva důvěry (trust anchor)*.

Například si lze představit, že certifikační autoritu pro směrování bude provozovat CESNET jako operátor národní akademické sítě. Tato centrální autorita bude certifikovat autority na jednotlivých univerzitách a ústavech AV ČR a ty pak budou vydávat certifikáty konkrétním směrovačům, případně budou certifikaci dále delegovat na fakulty či jiné části mateřských organizací. Libovolný připojený počítač vlastnící veřejný klíč certifikační autority CESNETu pak bude schopen ověřit jakýkoli směrovač v síti. A to i v případě, kdy se momentálně ocitne v jiné z připojených sítí, například během služební cesty.

Certifikáty nejsou vkládány přímo do zpráv ohlašujících jednotlivé prefixy. Znamenalo by to zbytečnou zátěž, protože klientovi stačí směrovač ověřit jednou a pak mu může důvěřovat až do vypršení platnosti některého z certifikátů. Proto SEND zavedl novou dvojici zpráv – *Žádost o certifikační cestu (Certification path solicitation)*, ICMP typ 148, a *Ohlášení certifikační cesty (Certification path advertisement)*, ICMP typ 149.

Když klient dostane ohlášení od směrovače, kterému zatím nemůže důvěřovat, pošle *Žádost o certifikační cestu*. Může ji poslat všem směrovačům na lince (ff02::2), na adresu pro vyzývaný uzel nebo na adresu svého implicitního směrovače. Identifikuje v ní kotvy důvěry – certifikační autority, jimž důvěřuje.

Směrovač na přijetí výzvy k certifikaci odpoví *Ohlášením certifikační cesty*. Zahrne do ní sadu certifikátů, které klientovi umožní ověřit jeho důvěryhodnost. Posloupnost certifikátů musí začínat některou z klientem uvedených autorit a pokračovat vzájemnými návaznostmi až k odesilateli ohlášení. Kdyby tedy v síti TU v Liberci některý z počítačů požádal o ověření zdejší směrovač a oznámil, že důvěřuje autoritě CESNETu, obsahovalo by ohlášení dva certifikáty. Prvním by autorita CESNETu potvrdila důvěryhodnost autority TUL, druhým by autorita TUL potvrdila důvěryhodnost směrovače.

Kombinace CGA adres, digitálních podpisů a certifikace směrovačů by měla ochránit ohlašování sousedů proti všem známým útokům. RFC 3971 obsahuje i několik implementačních opatření, jejichž cílem je obrana proti zahlcení (DoS).

5.5 Lehčí verze ochrany

SEND je sice bezpečnostně velmi silný, ale trpí některými neduhy. Je poměrně náročný, což může dělat těžkosti zařízením s omezenými zdroji. Například požární čidlo připojené k síti jistě nebude mít procesorového výkonu či paměti na rozdávání. K ověřování certifikátů je třeba klientům distribuovat veřejné klíče příslušných autorit, nemluvě o tom, že stav podpory SEND v operačních systémech má k ideálu daleko. Ve Windows chybí, pro Linux je dispozici neaktualizovaný kód[2] pro jádro verze 2.6.24.6 z konce roku 2009 dobrý nanejvýš pro experimenty.

RA-Guard

Zatím se tedy o použití SEND v původně zamýšlené podobě k přímé ochraně koncových stanic nedá ani uvažovat. Hledají se proto jiné, praktičtější cesty. Jednu popsalo RFC 6105: *IPv6 Router Advertisement Guard* – jestliže se koncové stroje nedokáží chránit samy, může to za ně udělat aktivní prvek, jehož prostřednictvím jsou připojeny. Nutno poznamenat, že RA-Guard není protokol či pevně definovaný mechanismus. Jedná se spíše o obecný popis ochranných mechanismů, které se pod různými názvy a v různých konkrétních podobách objevily v produktech jednotlivých výrobců.

Výchozí podmínkou je, že stroje nejsou v linkové vrstvě propojeny přímo a jejich komunikace prochází přes prostředníka. Typickým příkladem takové sítě je Ethernet na kroucené dvojlince, kde se kabely vedoucí od při-

[2] http://code.google.com/p/ipv6-send-cga/

pojených strojů scházejí v centrálním přepínači, jímž prochází veškerá komunikace. RA-Guard je implementován právě v tomto centrálním prvku. Zkoumá jednotlivá ohlášení a rozhoduje, zda je propustí k příjemcům či zastaví.

RFC 6105 předpokládá dva možné režimy hlídačovy práce. Může být bezstavový, což znamená, že každé ohlášení posuzuje samostatně, jen na základě informací v něm obsažených. Při rozhodování, zda paket propustit či nikoli, může bezstavový hlídač vycházet z příchozího rozhraní, linkové či IPv6 adresy odesilatele, ohlašovaných prefixů, priority směrovače nebo životnosti ohlášení.

Sofistikovanější je stavový režim práce, kdy hlídač k posuzování využívá dříve shromážděné informace. RA-Guard může příkazem správce přejít do režimu učení, kdy po omezenou dobu přijímá ohlášení a ukládá si jejich zdroje. Po ukončení této fáze bude propouštět ohlášení jen od směrovačů, které zná z fáze učení.

Zajímavější variantou stavového hlídače je využití SEND. RA-Guard v takovém případě propouští jen ohlášení ověřená pomocí SEND – odeslaná z korektních CGA adres a opatřená platnými podpisy. Jelikož je aktivních prvků řádově méně než koncových zařízení a jsou přímo řízeny správcem sítě, je takovéto nasazení „zprostředkovaného SEND“ výrazně jednodušší, než jeho plošná podpora na všech stanicích. RA-Guard představuje rozumný kompromis mezi úplným SEND a zcela nechráněnou sítí.

ochranné programy

Ani on však nemusí být dosažitelný. Vyžaduje chytré (čtěte dražší) aktivní prvky. Ty ale nemusí být k dispozici v celé síti – z ekonomických důvodů je poměrně běžné kombinovat chytré prvky v klíčových bodech infrastruktury s jednoduchými přepínači připojujícími koncové stroje uživatelů. V takové síti může zdroj falešných ohlášení stále ještě ovlivnit řadu svých sousedů.

Nic ale není ztraceno. Jestliže nedokážete zabránit šíření pirátských ohlášení, lze je alespoň sledovat a zpětně eliminovat. K tomu slouží specializované programy, jež sledují příchozí ohlášení a zkoumají, zda jsou v pořádku. Pokud zjistí nesrovnalost, pošlou do sítě vzápětí stejné ohlášení, ovšem s nulovou životností, takže odvolají účinek předchozího. Příkladem takového programu je *ramond*, jemuž se budu věnovat v části 19.2 na straně 371.

6 Automatická konfigurace

Plug and play cloumá světem. Všichni by chtěli, aby vše fungovalo pokud možno automaticky, nic se nemuselo nastavovat, konfigurovat či zapínat. IPv6 vychází tomuto trendu vstříc a jako jedno ze svých lákadel nabízí možnost automatické konfigurace počítačů.

stavová a bezstavová

Správce sítě má na výběr dokonce dva typy automatické konfigurace: stavovou a bezstavovou. *Stavová konfigurace* je stará vesta. Jejím základem je server spravující konfigurační parametry, které pak klientům na požádání sděluje. Podobné mechanismy se používají již hezkou řádku let – od RARP přes BOOTP až k dnešnímu DHCP. Pro účely stavové konfigurace IPv6 byl navržen protokol DHCPv6. V novějších textech o IPv6 se přestává termín „stavová konfigurace“ používat (prý byl matoucí) a píše se jednoduše o DHCPv6. Princip všech zmíněných mechanismů je podobný – počítač rozešle na obecnou adresu dotaz ohledně svých komunikačních parametrů a server mu je ve své odpovědi sdělí. Obvykle zahrnují vše potřebné pro rozumné zapojení do sítě – IP adresu, prefix podsítě, implicitní záznam do směrovací tabulky, adresu DNS serveru a případné další informace.

Naproti tomu *bezstavová konfigurace (Stateless Address Autoconfiguration, SLAAC)* představuje zcela nový způsob. Je založena na tom, že v síti sídlí ctnostní mudrcové (směrovače), kteří vědí vše potřebné. Proto čas od času všem sdělí, jaká je zdejší situace. Používají k tomu ohlášení směrovače. Nově přišedšímu počítači stačí jen chvíli poslouchat, případně o tyto informace aktivně požádat.

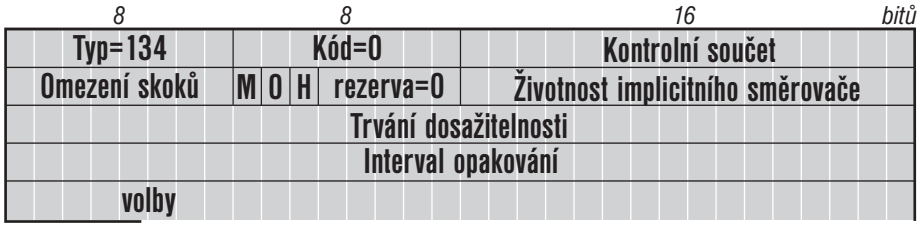
Hlavním cílem bezstavové konfigurace je automatické určení vlastní adresy uzlu. Jako taková je popsána v RFC 4862: *IPv6 Stateless Address Autoconfiguration*. Ovšem myslím, že s touto tématikou souvisí i automatická konfigurace směrování, která je oficiálně řazena do objevování sousedů. Proto se jí budu věnovat zde.

6.1 Ohlášení směrovače

Nosným pilířem bezstavové konfigurace je *Ohlášení směrovače (Router advertisement)*. Posílá je v náhodných intervalech každý směrovač, a to do všech sítí, k nimž je připojen. Náhodnost přestávky mezi ohlášeními má za cíl omezit dopady případných nešťastných časových souher (kdy dvě ohlášení doraziví v určitém nevhodném intervalu po sobě způsobí zmatení).

Ohlášení směrovače připomíná hlášení, která jsme zvyklí slýchat na nádraží. „Na ...tou kolej přijel prask škvrk z Prahy, pravidelný příjezd 14:30. Vlak dále pokr... Brno a New York.“ Po jeho absolvování vědí všichni zúčastnění – cestující ve vlaku, v ostatních vlacích i na nádraží – co se děje

a jak to bude pokračovat. Stejně tak po obdržení ohlášení směrovače vědí připojené počítače, v jaké síti se to octly, jak se zde komunikuje a kdo je implicitní směrovač.



Obrázek 6.1: Ohlášení směrovače

Na obrázku 6.1 vidíte, jak vypadá příslušný paket. Posílá se prostřednictvím ICMP. Jedna ze stěžejních informací – adresy zdejších sítí – však není na první pohled patrná, protože je umístěna mezi volbami. Z informací, které jsou obsaženy přímo v základu ohlášení, je nejdůležitější *Životnost implicitního směrovače (Router Lifetime)*. Jedná se o čas (v sekundách), jak dlouho ještě tento směrovač hodlá sloužit jako implicitní pro uzly z této sítě. Je-li hodnota nulová, směrovač nemá být používán jako implicitní. K problematice směrování se vrátíme zanedlouho.

Údaj o *Omezení skoků (Cur Hop Limit)* oznamuje zdejším uzlům, jak mají omezovat životnost odesílaných datagramů – jakou hodnotu vkládat do položky s maximálním počtem skoků.

Za ním následuje v ohlášení směrovače osmice příznaků, z nichž jsou zatím definovány jen tři. První dva se týkají DHCPv6. Příznak *M (Managed address configuration, stavová konfigurace adres)* oznamuje, že adresy i další komunikační parametry přidělí DHCPv6. Za ním následuje příznak *O (Other stateful configuration, stavová konfigurace ostatních parametrů)*, který rozhoduje o použití DHCPv6 pro ostatní parametry sítě, jako jsou například adresy lokálních DNS serverů. Významy možných kombinací příznaků M a O shrnuje tabulka 6.1.

M	*O*	*význam*
1	–	DHCPv6 poskytne vše
0	1	kombinovat bezstavovou konfiguraci (pro adresu, prefix a směrování) s DHCPv6 (pro ostatní parametry)
0	0	DHCPv6 není k dispozici

Tabulka 6.1: Význam příznaků při bezstavové konfiguraci

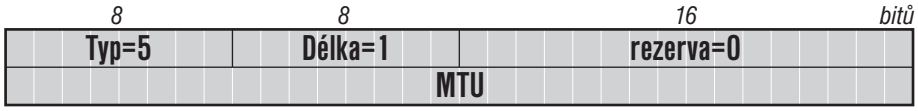
Příznak *H (Home agent, domácí agent)* slouží pro podporu mobility a byl doplněn v RFC 3775. Směrovač jeho nastavením sděluje, že je ochoten pro

místní síť pracovat jako domácí agent. Co to znamená se dočtete v kapitole 11 na straně 223.

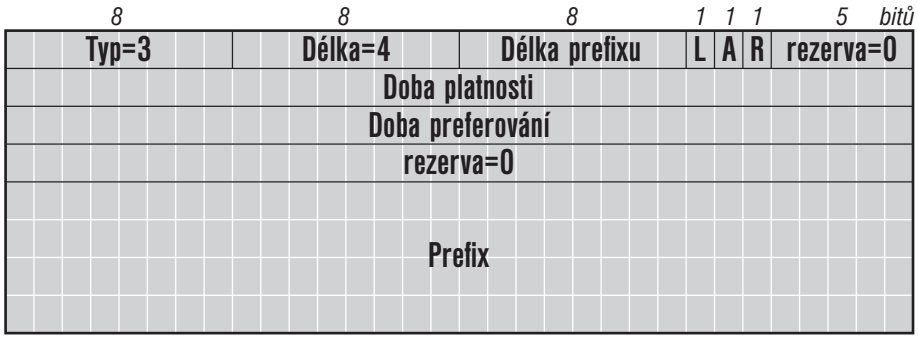
Poslední dva údaje pevné části ohlášení ovlivňují detekci dosažitelnosti sousedů (viz část 5.2 na straně 106). Oba jsou časové a jejich hodnota je uvedena v milisekundách. *Trvání dosažitelnosti (Reachable Time)* říká, jak dlouho má být uzel považován za dosažitelný poté, co byla ověřena jeho momentální dosažitelnost. *Interval opakování (Retrans Timer)* je interval mezi dvěma výzvami sousedovi.

prefixy adres

Ve *Volbách* může směrovač sdělit svou linkovou adresu (viz obrázek 5.2 na straně 105), může ohlásit MTU této sítě (obrázek 6.2) a především připojí po jedné volbě pro každý prefix IP adres, který se v dané síti používá[1]. Jinými slovy se předem počítá s tím, že jedna fyzická síť může sloužit několika různým logickým sítím a tudíž mít několik prefixů.



Obrázek 6.2: Volba *MTU*



Obrázek 6.3: Volba *Informace o prefixu*

Formát volby obsahující *Informace o prefixu (Prefix Information)* najdete na obrázku 6.3. Klíčový je pochopitelně vlastní *Prefix* a s ním spojená *Délka prefixu (Prefix Length)*, která udává, kolik bitů z něj je platných. Obvyklou hodnotou by mělo být 64. Dva časové údaje jsou uvedeny v sekundách. *Doba platnosti (Valid Lifetime)* udává jak dlouho prefix platí a *Doba preferování (Preferred Lifetime)* jak dlouho mají být preferovány adresy vzniklé automatickou konfigurací z tohoto prefixu. V obou případech hodnota 0xffffffff znamená nekonečnou trvanlivost.

[1] Mechanismy pro hierarchickou mobilitu přidávají ještě volbu pro kotevní bod mobility – viz strana 244.

Příznak *L (on-Link, na lince)* znamená, že prefix lze používat k rozhodování, který uzel je lokální – tedy přímo dosažitelný linkovou vrstvou – a který nikoli. Je-li nastaven příznak *A (Autonomous address-configuration, autonomní konfigurace adres)*, prefix lze použít k automatické konfiguraci vlastní adresy. Zde se skrývá možnost vypnout (zakázat) bezstavovou konfiguraci. Pokud všechny směrovače u všech prefixů ve svých ohlášeních vynulují příznak *A*, počítače nemají k dispozici žádné adresy, které by si mohly přidělit. Zůstanou odkázány na DHCPv6 a pochopitelně lokální linkové adresy, které si přidělují automaticky. Naopak pokud některé prefixy mají nastaven příznak A a kromě toho je v ohlášení směrovače zapnut i příznak M, může počítač použít obě cesty k získání adresy – stavovou i bezstavovou – a pokud povedou k odlišným adresám, může rozhraní přidělit všechny.

Příznak *R (Router address, adresa směrovače)* byl opět doplněn pro potřeby podpory mobilních zařízení. Je-li nastaven, obsahuje položka *Prefix* kompletní globální adresu směrovače. Pro potřeby automatické konfigurace si z ní místní stroje vezmou jen prefix sítě a identifikátor rozhraní budou ignorovat. Ovšem domácí agenti spolupracující s mobilními uzly zde najdou kompletní adresy svých kolegů. Ty pak mohou poslat uzlu na cestách, když bude dynamicky hledat domácího agenta.

fáze adresy

Dvojice časových údajů stanoví dobu trvání jednotlivých fází v životě adresy vytvořené bezstavovou autokonfigurací. Po svém vzniku je adresa *preferována (preferred)*. To znamená, že ji počítač může používat podle libosti.

Po vypršení *Doby preferování* se adresa stává *odmítanou (deprecated)*. Tato adresa je sice nadále platná, ale počítač by se jí měl pokud možno vyhýbat. Může ji použít například při pokračování již probíhající komunikace, pokud by přechod na jinou (preferovanou) působil potíže.

Konečně po uplynutí *Doby platnosti* se adresa stává *neplatnou*. Počítač ji nesmí používat a měl by odstranit její přiřazení odpovídajícímu rozhraní. Neplatná adresa jako by vůbec nebyla.

6.2 Určení vlastní adresy

Má-li prvek sítě komunikovat, musí především znát svou vlastní IP adresu. Její automatické stanovení vypadá následovně.

Uzel zahájí svou činnost tím, že si vytvoří svou lokální linkovou adresu. Udělá to tak, že ke standardnímu prefixu lokálních linkových adres fe80::/10 připojí identifikátor svého rozhraní, jehož vygenerování nepředstavuje žádný problém.

detekce duplicitních adres

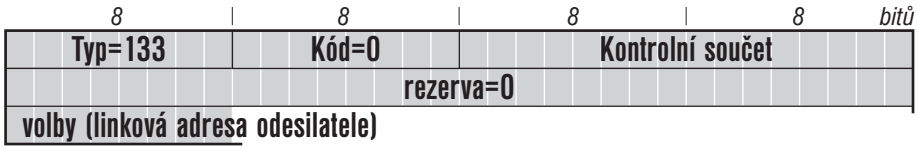
Je velmi málo pravděpodobné, že by stejnou lokální linkovou adresu mělo více uzlů, ale je vhodné se o tom přesvědčit[2]. Řešení tohoto úkolu je

[2] Už jsem zažil laciné ethernetové karty, u nichž se „fyzická“ adresa konfigurovala obslužným programem a implicitně byly nastaveny všechny stejně. Z nich odvozené EUI-64 identifi-

snadné: použije se standardní objevování sousedů. Uzel rozešle výzvu sousedovi, v níž hledá vlastníka adresy, kterou sám sobě vygeneroval. Pokud dorazí ohlášení souseda, vznikne potíž. Znamená to, že někdo má stejný identifikátor rozhraní a automatická konfigurace tudíž nemůže pokračovat dál. Popsaný postup se nazývá *detekce duplicitních adres*.

přidělení adres

V normálním případě však bude odezva negativní a uzel si vytvořenou lokální linkovou adresu přidělí. Pro pokračování v automatické konfiguraci bude potřebovat znalosti o svém okolí. Proto počká na ohlášení směrovače, případně o ně sám aktivně požádá prostřednictvím *Výzvy směrovači*.



Obrázek 6.4: Výzva směrovači

Z příznaků v *Ohlášení směrovače* se dozví, zda má použít stavovou konfiguraci pro svou adresu a další parametry sítě. Dále pak je u každého ze zdejších prefixů uveden příznak, zda se pro tento prefix má použít bezstavová konfigurace adres. Pokud ano, připojí si k prefixu svůj identifikátor rozhraní a tuto adresu si přidělí. Už ji netestuje, protože jednoznačnost lokálního prefixu byla prověřena hned v počáteční fázi, když si uzel přiděloval lokální linkovou adresu.

Jednoznačnost se testuje také u adres konfigurovaných manuálně nebo získaných prostřednictvím stavové konfigurace. Použije se výše popsaný postup – uzel pošle výzvu sousedovi se svou vlastní adresou.

6.3 Konfigurace směrování

V IP verze 6 se jednotlivé uzly dovedou naučit i směrování ve své síti. Návrh předpokládá, že si uzel bude udržovat následující datové struktury:

Cache cílů (Destination Cache) obsahuje směrovací informace pro konkrétní cílové adresy. Ke každému cíli je v této tabulce umístěna první adresa po cestě k němu (next hop). Datagramy směřující k uvedenému cíli se mají předat na tuto adresu.

Seznam prefixů (Prefix List) slouží k posuzování, kdo je a kdo není umístěn ve stejné síti.

kátory by byly totožné. Také u náhodně generovaných identifikátorů rozhraní nelze zcela vyloučit shodu.

Seznam implicitních směrovačů (Default Router List) obsahuje informace o všech směrovačích, které ve svém ohlášení nastavily příznak implicitního směrovače.

Dá se říci, že seznam prefixů a seznam implicitních směrovačů představují obecný mechanismus, zatímco cache cílů ztělesňuje výjimky z něj. Datové struktury jsou ideové, lze je realizovat také všechny v jednom – například směrovací tabulkou.

Podpora mobility pak přidává ještě cache vazeb, která říká, že dotyčný počítač momentálně pobývá na úplně jiné adrese. Tato skutečnost znamená, že datagram bude zcela přepracován (změní se cílová adresa a přibyde hlavička *Směrování*), jak je popsáno v části 11.5 na straně 238.

Když je rozhodnuto o příjemci datagramu, přijde ke slovu cache sousedů, v níž se bude hledat jeho fyzická (linková) adresa. Té jsem se podrobně věnoval v kapitole 5 na straně 103. Celý postup při odesílání datagramu je znázorněn na obrázku 6.5.

hledání zprostředkovatele

Když má uzel odeslat datagram, musí nejprve určit IP adresu, na kterou jej má předat. Své úsilí zahájí pohledem do cache cílů, zda tato adresa není explicitně definována. Pokud ano, použije ji. Jestliže se daný cíl nevyskytuje v cache cílů, srovná jeho adresu s jednotlivými položkami v seznamu prefixů. Podle nich určí, zda se jedná o adresu lokální či vzdálenou. U lokální adresy se datagram posílá rovnou adresátovi. Pro vzdálenou použije jeden z implicitních směrovačů.

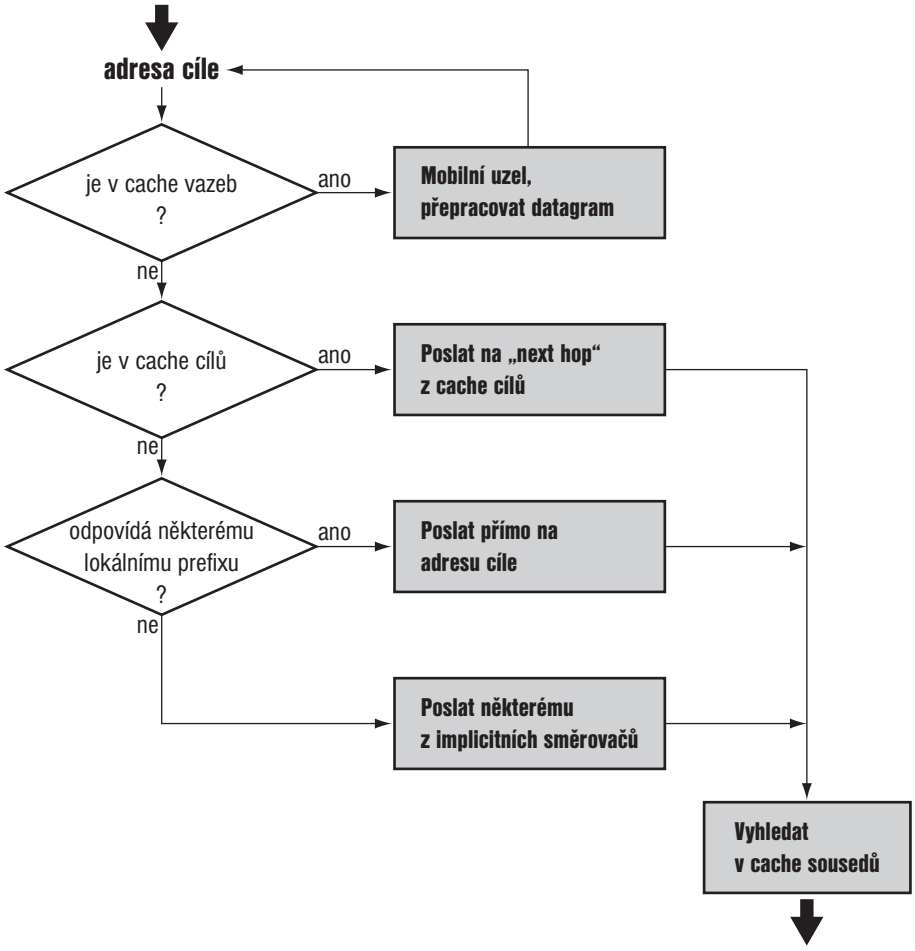
přesměrování

Zvolí-li nevhodný směrovač nebo je daný cíl ve skutečnosti lokální, pošle směrovač odesilateli paketu ICMP zprávu *Přesměrování*. Údaje v ní obsažené by si odesilatel měl poznamenat do cache cílů, aby příště posílal datagramy určené tomuto cíli vhodnější cestou.

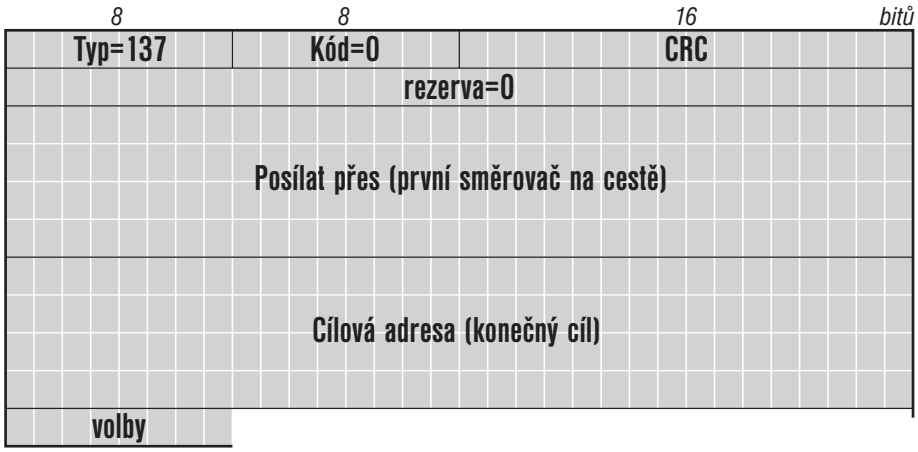
Formát přesměrování najdete na obrázku 6.6. Obsahuje jen nejzákladnější informace: *Cílovou adresu (Destination Address)* a *Posílat přes (Target Address)*, což je adresa směrovače (nebo cíle samotného), na kterou se mají posílat datagramy určené pro tento cíl. Do voleb lze zařadit fyzickou adresu směrovače, pokud ji odesilatel přesměrování zná. Měl by také přidat hlavičku datagramu, který přesměrování vyvolal. Její velikost je omezena tak, aby celkově datagram s přesměrováním nepřekročil délku 1280 B.

6.4 Konfigurace DNS

Bezstavová automatická konfigurace dokáže nastavit adresu rozhraní i jednoduchou směrovací tabulku. K rozumnému zapojení počítače do sítě zbývá jediná věc: adresa místního DNS serveru, na nějž se má obracet se svými dotazy. Dlouhá léta se ovšem DNS nacházelo mimo dosah bezstavové konfigurace, jež pro ně nenabízela žádné možnosti. Při rozkošné po-



Obrázek 6.5: Postup při odesílání datagramu



Obrázek 6.6: Přesměrování

době IPv6 adres je ale počítač bez funkčního DNS téměř nepoužitelný, proto bylo třeba problém nějak řešit.

Jedinou standardní možností bylo nechat bezstavovou konfiguraci být a informace o DNS (i případné další) do ní doplnit stavovou cestou. K tomu slouží příznak O v ohlášení směrovače, jak se podrobněji dočtete v části 6.6 na straně 134. Nevýhoda tohoto přístupu je zjevná: musíte provozovat další server a v konfiguraci spojovat informace získané různými cestami. Ne každá implementace IPv6 se s tím dokázala vyrovnat.

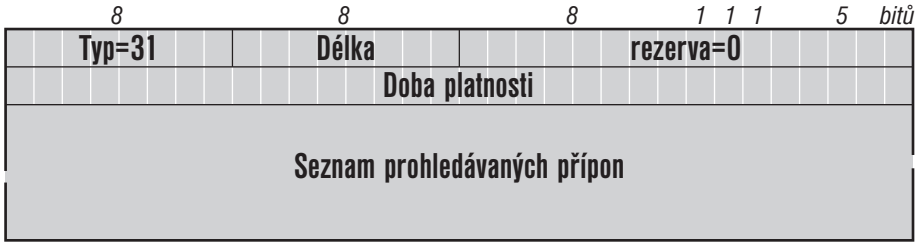
V roce 2010 proto vyšlo RFC 6106: *IPv6 Router Advertisement Options for DNS Configuration*, jež do automatické bezstavové konfigurace doplnilo volby pro DNS. Jsou dvě, RDNSS poskytuje seznam místních rekurzivních DNS serverů, na něž se klient má obracet se svými dotazy, zatímco DNSSL obsahuje seznam přípon, které může při hledání přidávat na konec relativních doménových jmen.

Obrázek 6.7: Volba *Rekurzivní DNS server*

Formát volby *Rekurzivní DNS server (Recursive DNS Server, RDNSS)* vidíte na obrázku 6.7. Poskytuje IPv6 adresy místních serverů v libovolném počtu (je odvozen z *Délky*) a počet sekund jejich *Životnosti*. Ta je společná pro všechny uvedené adresy. Ovšem jedno ohlášení může obsahovat více exemplářů této volby s odlišnými životnostmi.

Specifikace doporučuje, aby se životnost pohybovala v rozmezí od maximálního intervalu mezi ohlášeními směrovače do jeho dvojnásobku. Dá se očekávat, že během doby životnosti dorazí další ohlášení směrovače, které ji prodlouží. Speciální význam má životnost 0, která zakazuje příslušný server nadále používat. Naopak životnost 0xffffffff je nekonečná.

Klient si udržuje seznam používaných serverů. Kdykoli dorazí nové ohlášení s volbou *Rekurzivní DNS server*, seznam si aktualizuje – servery s nulovou životností vyřadí, u existujících si aktualizuje její hodnotu a nové přidá. RFC 6106 požaduje, aby existoval systémový parametr určující dostatečný počet serverů (a doporučuje, aby měl hodnotu 3). Jestliže přidáním nových serverů překročí délka seznamu tuto hodnotu, budou z něj vyřazeny adresy s nejkratší životností tak, aby byl doporučený počet serverů zachován.



Obrázek 6.8: Volba *Prohledávací seznam DNS*

Volba *Prohledávací seznam DNS (DNS Search List, DNSSL)* vypadá velmi podobě a prakticky stejně s ní klient také pracuje. Liší se jen formát a význam její datové části. Tentokrát neobsahuje seznam IPv6 adres, ale doménových jmen ve standardním přenosovém formátu. Když klient neuspěje při hledání údajů k určitému jménu a dané jméno nebylo zadáno absolutně (nekončí tečkou), může za ně postupně připojovat jednotlivá jména z prohledávacího seznamu.

Pokud například prohledávací seznam mého stroje obsahuje *tul.cz*, mohu se na web naší univerzity dostat prostým zadáním *www*. Klient se pokusí nejprve najít v DNS adresu pro jméno *www* a když neuspěje, bude opakovat pokus s příponou podle seznamu. Bude tedy hledat adresu pro jméno *www.tul.cz*.

Aktualizace prohledávacího seznamu podle přicházejících ohlášení probíhají stejným způsobem jako v případě seznamu místních DNS serverů.

RFC 6106 počítá i se situací, kdy klient dostává relevantní DNS informace jak pomocí ohlášení směrovače, tak protokolem DHCPv6. V tom případě by je měl kombinovat tak, aby si v seznamu zanechal alespoň jednu hodnotu z každého aktivního mechanismu. Pokud by se některý pomátl (nebo byl napaden), zůstanou mu díky tomu k dispozici alespoň nějaké použitelné hodnoty.

Při vzájemném kombinování hodnot do společného seznamu by klient měl dodržovat následující pořadí priorit:

DHCPv6 > ohlášení chráněné SEND > ohlášení bez SEND

6.5 DHCPv6

Automatická konfigurace s dopomocí je ve světě IPv4 celkem běžnou záležitostí. Zajišťuje ji *Dynamic Host Configuration Protocol (DHCP)*, jehož prostřednictvím může startující počítač získat všechny údaje potřebné pro plnohodnotný síťový život (IP adresu, masku podsítě, adresu DNS serveru a implicitní směrovač pro odchozí provoz). DHCP je dnes všudypřítomné – typický operační systém bývá čerstvě po instalaci nastaven na jeho použití, pomocí DHCP se konfigurují síťové tiskárny, najdete je v podnikových sítích i v domácnostech (protože ADSL modem obvykle funguje jako DHCP server pro počítače připojené k němu Ethernetem či Wi-Fi). Získání síťových parametrů prostřednictvím DHCP má čtyři fáze:

1. **Objevování (discover):** Klient pošle všesměrově (čili na IP adresu 255.255.255.255) dotaz obsahující jeho ethernetovou adresu.
2. **Nabídka (offer):** Servery, k nimž se dotaz dostane (často bývá jeden, ale obecně jich může být libovolné množství), nahlédnou do svých tabulek, zda pro tohoto klienta mají nějaké použitelné parametry. Pokud ano, pošlou mu nabídku „Ode mne bys mohl mít tohle ...“
3. **Požadavek (request):** Klient posbírá nabídky, vybere si tu, která je jeho srdci nejbližší a příslušnému serveru pošle požadavek, v němž žádá o přidělení nabídnutých parametrů.
4. **Potvrzení (acknowledge):** Server mu potvrdí, že jeho žádosti vyhověl. Tím okamžikem může klient začít příslušné parametry používat. Jejich přidělení je však pouze dočasné (v terminologii DHCP se jedná o pronájem, lease), po vypršení platnosti musí klient požádat o prodloužení nebo získat zcela nové parametry.

Ve světě IPv6 se tento přístup nazývá stavovou konfigurací a zajišťuje jej nová verze DHCP. Protokol pochopitelně doznal jistých změn – IPv6 nezná všesměrové (broadcast) adresy, na druhé straně si však každá stanice umí sama nastavit lokální linkovou adresu, takže odpadá vazba na adresy nižších komunikačních vrstev (Ethernet apod.).

Vzhledem k všeobecnému rozšíření DHCP a spoustě zkušeností s jeho provozem je překvapující, že definice nové verze protokolu vznikala neskutečně dlouho. Od prvního návrhu *draft-ietf-dhc-dhcpv6-00* do výsledného RFC uběhlo osm a půl roku! V roce 2003 jsme se konečně dočkali RFC 3315: *Dynamic Host Configuration Protocol for IPv6 (DHCPv6)*.

účastníci DHCP

Na DHCPv6 se podle něj podílejí tři kategorie zařízení: *klient* je stroj, který chce získat informace; *server* je ten, kdo mu je poskytne; *zprostředkovatel (relay)* pak zprostředkovává styk mezi nimi, pokud se klient a server nacházejí na různých linkách. Pod společný pojem *agent* pak bývají zahrnovány servery a zprostředkovatelé. Agent je zkrátka někdo, kdo poskytne DHCP odpověď (ať už svou vlastní či zprostředkovanou) a sídlí na lokální lince.

DUID

Významnou roli v DHCP hraje otázka identifikace – jak serverů, tak především klientů. Dříve se k tomuto účelu používala ethernetová adresa, DHCPv6 však zavádí pojem *DHCP Unique Identifier (DUID)*. Jedná se o jednoznačný identifikátor účastníka DHCP života, právě jeden DUID má každý klient i server. Měl by být pokud možno stálý a neměnit se ani při výměně síťové karty počítače.

Autoři protokolu vzdali snahu o vytvoření univerzálního identifikátoru, který by vyhověl ve všech případech. Místo toho definovali několik způsobů, jak jej lze vytvořit. Navíc připustili do budoucna rozšiřování sortimentu typů DUID.

Výše uvedenou podmínku stálosti snadno splňuje identifikátor přidělený výrobcem. Předpokládá, že výrobce přidělí zařízení jednoznačnou identifikační hodnotu (výrobní číslo). DUID je pak tvořen touto hodnotou a doménou výrobce a zařízení si jej ponese po celý svůj život. Další dva definované typy využívají linkovou adresu. První ji kombinuje s časem vytvoření a předpokládá, že zařízení má k dispozici trvanlivou zapisovatelnou paměť. Čili že si DUID jednou vygeneruje, uloží do této paměti a pak bude trvale používat tutéž uloženou hodnotu. Konečně poslední ze základní trojice typů používá jen samotnou linkovou adresu a tedy odpovídá praxi ze světa IPv4. Na rozdíl od něj by ale měl stejný DUID používat pro všechna síťová rozhraní, jež chce pomocí DHCPv6 konfigurovat. Výměnou síťové karty se tento typ DUID pochopitelně změní. RFC 6355 poději doplnilo DUID postavený na univerzálním identifikátoru UUID.

IA

Druhou identifikační konstrukcí je tak zvaná *identifikační asociace (identity association, IA)*. Jedná se typicky o shluk konfiguračních informací přidělených jednomu rozhraní, opatřený jednoznačným identifikátorem (IAID). Tyto identifikátory přiděluje klientský počítač každému rozhraní, pro něž chce použít DHCPv6. Opět by měly být konzistentní a neměnit se v čase. Čili by si je buď měl ukládat do trvanlivé paměti nebo používat takový algoritmus pro jejich vytváření, který zajistí pokaždé stejné hodnoty.

Otázky identifikace jsou tedy v DHCPv6 narýsovány takto: klient je jednoznačně identifikován svým DUID, rozhraní v rámci klienta jsou pak rozlišována prostřednictvím IA. S IA jsou také spojeny přidělované parametry (některé však mohou stát i mimo IA, pokud jsou obecné pro celého klienta).

fáze

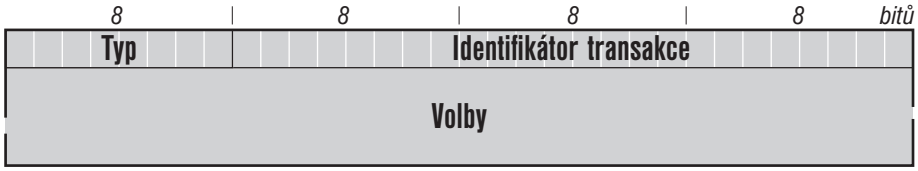
Základní fáze DHCPv6 dialogu se proti předchůdci nijak významně nezměnily. Klient se poptá po dostupných parametrech, dostane nabídky, jednu si vybere a se serverem si dohodne její přidělení. Konfigurace začíná hledáním vstřícných serverů. Jelikož na začátku klient neví nic o síti, ve které se nachází, posílá své počáteční zprávy na standardní skupinové adresy. Pro DHCPv6 byly definovány následující:

všichni DHCP agenti a servery	ff02::1:2
všechny DHCP servery	ff05::1:3

hledání serverů

Klient zahájí svou činnost tím, že vytvoří IA pro svá rozhraní a opatří je jednoznačnými identifikátory. Na adresu všech DHCP agentů (ff02::1:2, všimněte si, že má dosah jen v rámci linky) pak pošle tak zvanou *výzvu (solicit)*, ke které přibalí svůj DUID i všechny IA. Významem výzvy je „hledám všechny DHCP servery, které jsou ochotny mi poskytnout adresu“. Aby systém fungoval, musí být v každé lokální síti umístěn alespoň jeden agent. Součástí výzvy je i *lokální linková adresa* (s prefixem fe80::), kterou si klient přidělil.

Formát DHCP zprávy najdete na obrázku 6.9. Ve srovnání se staršími verzemi návrhu byl naprosto minimalizován. Skoro všechny informace byly přesunuty do voleb, zůstaly jediné dvě společné položky: *Typ (Msg-type)* identifikující o jakou zprávu se vlastně jedná a *Identifikátor transakce (Transaction-id)*, který umožňuje párovat dotazy a odpovědi.



Obrázek 6.9: Formát zprávy DHCPv6

Pokud výzvu obdrží server, rovnou klientovi odpoví. Znamená to, že sídlí společně s klientem na téže lince a pro doručení odpovědi proto použije jeho lokální linkovou adresu. Odpovědí na výzvu je *ohlášení serveru (advertise)*. Jeho součástí bývá *preference*, která udává ochotu serveru poskytnout své služby danému klientovi. Zároveň server přibalí konfigurační parametry, které by přidělil jednotlivým IA. V podstatě říká „Kdybych já dostal tento požadavek, nabídl bych toto...“

1	výzva (solicit)
2	ohlášení serveru (advertise)
3	žádost (request)
4	potvrzení (confirm)
5	obnovení (renew)
6	převázání (rebind)
7	odpověď (reply)
8	uvolnění (release)
9	odmítnutí (decline)
10	rekonfigurace (reconfigure)
11	žádost o informace (information request)
12	předání (relay forward)
13	odpověď k předání (relay reply)

Tabulka 6.2: Typy zpráv DHCPv6

Zprostředkovatel naproti tomu má konfigurován seznam serverů, kterým má předávat dotazy (součástí tohoto seznamu může být i obecná skupinová adresa všech DHCP serverů daného místa ff05::1:3). Dorazí-li k němu výzva, předá ji na všechny adresy ze seznamu. Dotaz přitom zabalí do nové zprávy typu *předání (relay forward)*, v níž uvede svou vlastní adresu. Server svou odpověď zabalí do podobné zprávy (*odpověď k předání, relay reply*) a pošle zpět zprostředkovateli. Ten vybalí data a předá je opět na lokální linkovou adresu klienta.

Klient posbírá doraziví ohlášení a vytvoří si tak seznam DHCP serverů, které má k dispozici. Měl by dát přednost tomu, kdo má nejvyšší preferenci. Pokud se hodnoty shodují, může se rozhodnout podle dalších poskytnutých informací (např. nabídnutých adres a podobně).

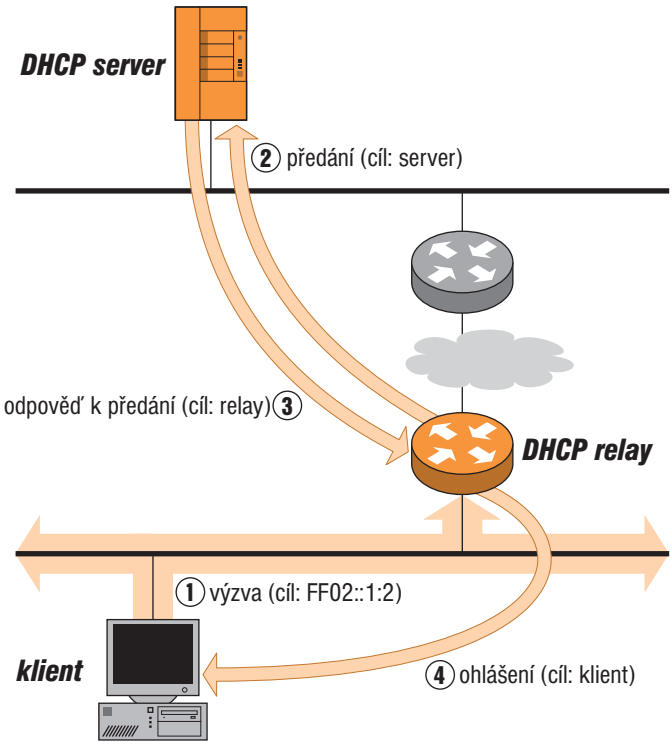
získání adresy

Když si klient vybral server, nastává druhá fáze: získání komunikačních parametrů. Odešle tedy novou zprávu, tentokrát se jedná o DHCP *žádost (request)*. V ní uvede DUID serveru, kterému je určena (identifikátor získal z jeho ohlášení). Zprávu opět pošle na obecnou adresu všech DHCP agentů. Stále ještě nezná zdejší síť, takže neví, jak doručit konkrétně adresovaný datagram. Servery, kterých se netýká, žádost ignorují.

Cílový server žádost vyhodnotí a pošle zpět *odpověď (reply)*. Při přidělování adresy server bere v úvahu především linku (fyzickou síť), ke které je klient připojen, a DUID klienta. Především podle těchto dvou informací vybere adresu (či adresy), které oznámí klientovi.

Součástí odpovědi je i stav, kterým sděluje, zda žádosti vyhověl nebo ne (a z jakého důvodu). Vzhledem k tomu, že klient jej vybíral z pozitivních ohlášení na svou výzvu, mělo by být odmítnutí velmi nepravděpodobné.

Komunikace opět může proběhnout přímo nebo přes zprostředkovatele. Druhý případ je znázorněn na obrázku 6.10.



Obrázek 6.10: Zprostředkovaná žádost a odpověď v DHCPv6

Klient si přidělené adresy ověří (standardním postupem pro detekci duplicitních adres) a pokud zjistí, že je někdo již používá, může je odmítnout. Slouží k tomu DHCP zpráva *odmítnutí (decline)*.

životnost adres

Stejně jako v IPv4 jsou adresy přidělovány na omezenou dobu. Po jejím uplynutí musí klient požádat o prodloužení. Nejprve žádá server, který adresu přidělil (zpráva *obnovení, renew*). Pokud neodpovídá, obrátí se později na všechny dostupné servery, zda některý z nich není ochoten adresu prodloužit (*převázání, rebind*). Když naopak klient končí svou síťovou existenci, měl by o tom server informovat zprávou *uvolnění (release)*, aby mohla být jeho adresa přidělena případnému novému zájemci.

Další situací, jejímž řešením se DHCPv6 zabývá, je návrat klienta do sítě. Může k němu dojít například po restartu, usnutí a probuzení počítače či jeho dočasném fyzickém odpojení. V takovéto situaci si klient musí ověřit, jestli jeho stávající síťové parametry jsou správné. Proto pošle na adresu všech DHCP agentů zprávu *potvrzení* (*confirm*), v níž sdělí aktuální parame-

try svých IA. Příslušný server reaguje *odpovědí*, ve které platnost přiřazení potvrdí nebo naopak odmítne.

konfigurace vyžádaná serverem

Aktivita v DHCP typicky vychází od klienta. Návrh však pamatuje i na případy, kdy vyvolání konfiguračního dialogu požaduje server. Zpravidla se jedná o situace, kdy došlo ke změně síťových parametrů a server chce, aby se klienti přizpůsobili nové situaci.

V tom případě rozešle zprávu *rekonfigurace* (*reconfigure*). Posílá se individuálně každému z klientů, kterých se týká. Jelikož si server vede přehled o přidělených parametrech, bez problémů si v něm najde potřebné adresy. Klient pak reaguje odesláním požadavku na *obnovení* svých parametrů a server ve své *odpovědi* sdělí vše potřebné.

bezpečnost

Na závěr jako tradičně vystupují bezpečnostní otázky DHCPv6. Aneb jak se bránit proti partyzánským serverům poskytujícím nesmyslné údaje či proti změnám DHCP zpráv při přenosu. Protokol za tímto účelem zavádí volbu *Autentizace (Authentication)*. Pokud klient chce ověřovat pravost DHCP komunikace, přibalí ji hned k úvodní výzvě. V ní určí, jaké metody pro ověřování chce používat, základní je HMAC v kombinaci s MD5. Server následně připojí autentizační informace ke svému ohlášení a i dále bude tato volba součástí vyměňovaných zpráv. Obsahuje digitální podpis, díky němuž může příjemce ověřit jak totožnost odesilatele (zná klíč), tak obsah zprávy.

Slabinou tohoto mechanismu je, že staví na symetrické kryptografii, kde obě strany používají stejný klíč. Vyžaduje konfiguraci na obou stranách – klient musí znát klíč pro komunikaci se serverem a server musí znát stejný klíč pro komunikaci s daným klientem. V praxi bude mít DHCPv6 server uloženy klíče všech známých klientů, což vyžaduje neustálou aktualizaci a navíc představuje bezpečnostní riziko při kompromitaci serveru. Obávám se, že autentizační prvky DHCPv6 jsou spíše teoretickou než reálně použitelnou konstrukcí.

Výrazně jednodušší je situace při zabezpečení komunikace mezi agenty a servery. K ní dochází v době, kdy oba účastníci již mají nastaveny síťové parametry a normálně komunikují. Díky tomu není třeba vyrábět kompromisy a lze použít IPsec (viz kapitola 10 na straně 201). Navíc agentů i serverů bývá v síti málo, takže i související konfigurace bude poměrně jednoduchá.

praxe

Dlouhou dobu vzniku RFC 3315, na niž jsem si stěžoval, lze vnímat i jako předzvěst problémů, jimiž je DHCPv6 zatíženo a které výrazně omezují jeho používání.

První z problematických míst představují identifikátory. Generuje a ukládá si je operační systém, takže i když daný systém používá konzistentní DUID, jiné systémy na tomtéž počítači budou mít jiný. Přechod na novou verzi Windows, více systémů na jednom stroji či prostá přeinstalace systému ty-

picky znamenají, že jeden a tentýž počítač vystupuje pod různými DUID. Chcete-li pomocí DHCPv6 přidělovat pevné síťové parametry, znamená každá z výše uvedených změn nutnost upravit konfiguraci DHCP serveru. A naopak klonování systému způsobí stejný DUID na různých počítačích.

Obtížné může být už samotné jeho zjištění. Pro laického uživatele může nahlédnutí do registru *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip6\Parameters\Shcpv6DUID* představovat dobrodružnou cestu do končin, o jejichž existenci dosud neměl tušení.

Druhým významným odpuzovačem zájemců je rozhodnutí autorů nezařadit do DHCPv6 volbu pro implicitní cestu. Oficiálním důvodem je, že směrovací tabulky se v IPv6 plní lokálními linkovými adresami, které platí jen v rámci linky a je proto nesystémové poskytovat je zvenčí. Dopadem tohoto rozhodnutí je, že směrovače musí posílat svá ohlášení. A – světe, div se – řada správců si řekla, že když už v síti mají základ bezstavové konfigurace, budou její pomocí konfigurovat vše.

Důsledkem je, že plnohodnotného DHCPv6 je v sítích jako šafránu. A pravděpodobně se na tom ani do budoucna mnoho nezmění.

6.6 Bezstavové DHCPv6

Hlavní nevýhodou bezstavové automatické konfigurace je velmi omezený sortiment informací, které lze jejím prostřednictvím získat. Nejpalčivější byla absence adres místních DNS serverů, jež sice byla doplněna, ale na implementace si musíme ještě nějakou dobu počkat. Občas by se klientům hodily i jiné věci. Proto obsahuje bezstavová konfigurace možnost, jak doplnit další informace jiným (stavovým) způsobem.

Připomenu, že k tomuto účelu slouží kombinace voleb *M*=0 a *O*=1 (viz tabulka 6.1 na straně 120) v *Ohlášení směrovače*. Znamená, že počítač si má adresu a směrování nastavit bezstavově a doplnit k nim další informace získané stavovým protokolem. A právě k tomuto účelu slouží bezstavové DHCPv6 definované v RFC 3736: *Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6*.

Jedná se o *velmi* zjednodušenou verzi DHCPv6. Z něj přebírá formáty zpráv i pravidla chování všech účastníků, ovšem snaží se o maximální jednoduchost a díky tomu snadnou implementovatelnost. Používá jen dva typy DHCP zpráv: žádost o informace a odpověď[3]. Také sortiment dostupných voleb je značně omezen.

Celá transakce začíná odesláním zprávy *Žádost o informace (Information request)* ze strany klienta. Její součástí je volba identifikující parametry,

[3] Dobře, lhát se nemá. Ve skutečnosti jsou čtyři. Aby se daly zprávy bezstavového DHCP předávat dál, potřebuje ještě předání a odpověď k předání pro komunikaci mezi serverem a zprostředkovatelem, pokud server není na lokální lince.

které požaduje. Server podle nich sestaví *Odpověď (Reply)* a pošle ji klientovi, který informace v ní obsažené využije. Toť vše. Vzájemná komunikace je jednodušší, protože na rozdíl od klasického DHCPv6 klient již má (bezstavově) nastavenu adresu a směrování.

Bezstavové DHCPv6 je jednodušší nejen pro implementátory, ale také pro správce. Zpravidla bude třeba poskytovat klientům v místní síti jen adresy lokálních DNS serverů, které bývají pro všechny stejné a příliš se nemění. Lze očekávat, že konfigurace bezstavového DHCPv6 serveru bude obsahovat pár řádků a vydrží roky.

6.7 Jak tedy konfigurovat?

Když projdete možnosti automatické konfigurace, jako obvykle zjistíte, že každá z dostupných variant má své klady a zápory a těžko se najde univerzální odpověď na otázku v názvu této části.

Bezstavová konfigurace láká svou jednoduchostí. Nikdo nemusí nic nastavovat (všude je podporována, v implicitní konfiguraci operačních systémů a zařízení bývá zapnutá), stroj se prostě zapojí do sítě a funguje. Tedy funguje za předpokladu, že nějak získá informace o DNS. Až bude široce implementováno RFC 6106, budeme mít vyhráno a všeobecnému bezstavovému konfigurování nebude už nic stát v cestě.

Zatím lze údaje pro DNS doplnit bezstavovým DHCPv6, ovšem v tomto případě bych byl velmi opatrný v prohlášeních o jeho všeobecné podpoře a rozšíření na straně klientů. Druhou možností je vyvěsit návod pro ruční konfiguraci, který bude sice stejný pro všechny, takže relativně snadno vytvořitelný, ovšem zvolna se vytrácí ona lákavá bezpracnost.

Druhým problémem bezstavové konfigurace je, že některým správcům sítí při slovech „nikdo nemusí nic nastavovat, zařízení se prostě zapojí do sítě a funguje“ vyráží studený pot na čele. Chaos v síti. Divocí uživatelé zapojující bez jakékoli kontroly a evidence nejrůznější aparáty. Jak v takové situaci řešit problémy, které způsobí? Jak odpovídat na dopisy „Tehdy a tehdy byl na počítači s adresou X z Vaší sítě nabízen P2P službou Y film „Zachraňte vojína Ryana“. Zbylo nám tu z natáčení pár tanků, tak s tím koukejte rychle něco udělat, nebo s nimi přijedeme!“

Protipólem je regulérní DHCPv6, doplněné minimalistickým bezstavovým základem nastavujícím směrování. Umožňuje udržovat v síti jakýs takýs pořádek[4], ale pokud se chcete vyhnout náhodně přidělovaným adresám, musíte si udržovat databáze počítačů v síti a trápit se s DUID. Správa takového systému je pracná, navíc implementace DHCPv6 jsou pořád ještě méně časté a méně kvalitní, než by si člověk přál. Kromě toho beztak musíte

[4] Zejména pokud na směrovačích zakážete komunikaci počítačů, které svou adresu nedostaly pomocí DHCP.

provozovat i – alespoň omezenou – bezstavovou automatickou konfiguraci, ze které si připojené stroje nastaví implicitní směrovače.

Kdybych si měl zavěštit do budoucna, očekával bych spíše příklon k plug-and-play přístupu. Tedy buď bezstavovou konfiguraci doplněnou bezstavovým DHCPv6 pro zbývající parametry, nebo plnohodnotné DHCPv6 v liberálním nastavení typu „přidělím adresu každému, kdo si o ni řekne“. O pořádek v síti se pak postará autentizace uživatelů protokolem IEEE 802.1X nebo podobným – počítač sice dostane síťové parametry volně, ale jeho komunikace bude zablokována, dokud *uživatel* neprokáže svou totožnost.

6.8 Jednoduchá detekce připojení

Výše popsané konfigurační mechanismy potřebují ke své práci určitý čas. Většina z nás si pravděpodobně nebude dělat těžkou hlavu ze „zpoždění až 3,5 s“, jímž je zatížena bezstavová konfigurace, nicméně existují snahy o rychlejší nastavení síťových parametrů. Jejich výsledkem je algoritmus *jednoduché detekce síťového připojení* (v originále též zkracovaný jako *simple DNA*) definovaný v RFC 6059: *Simple Procedures for Detecting Network Attachment in IPv6*.

Nejedná se o nic fundamentálního, dá se žít docela dobře bez něj (a žije se, protože zatím to s implementacemi nevypadá nijak slavně). Na druhé straně ale nepotřebuje žádné úpravy okolního prostředí, celý postup staví na standardních konfiguračních zprávách a odehrává se výlučně v koncovém zařízení. Pokud detekci implementuje, nakonfiguruje se v řadě případů rychleji. V opačném případě se nic neděje a vše probíhá konvenčními postupy.

Jednoduchá detekce připojení prospívá zejména uzlům, které střídají malý počet sítí. Typickým příkladem je notebook, který náhodně zapojujete v práci a doma. V důsledku toho střídá dvě sady síťových parametrů. Díky jednoduché detekci může rychle zjistit, že se nachází v jedné z těchto známých sítí a nastavit si parametry, které v ní používal minule.

SDAT

Základní datovou strukturou jednoduché detekce připojení je tabulka adres zvaná *Simple DNA Address Table (SDAT)*. Je indexována identifikátory lokálních směrovačů, s nimiž se stroj během své nedávné síťové historie setkal. Jako identifikátor se používá kombinace lokální linkové IPv6 adresy a linkové (MAC) adresy daného směrovače. Obsahuje adresu (či adresy), již stroj ve spojitosti s tímto směrovačem používal, doprovodné informace k ní (jakým mechanismem byla získána a informace specifické pro použitý mechanismus) a zda je použitelná.

Když stroj dostane informaci od linkové vrstvy, že se právě podařilo připojit k lince, provede paralelně několik kroků:

- Všechny právě nastavené IPv6 adresy označí jako nepoužitelné. V cache sousedů označí položky všech směrovačů jako prošlé.
- Na skupinovou adresu všech směrovačů na lince zašle výzvu směrovači (čili zahájí bezstavovou automatickou konfiguraci).
- Pokud adresy získal protokolem DHCPv6, zahájí DHCP výměnu k jejich obnovení.
- Projde SDAT a každému směrovači, pro který existuje v tabulce alespoň jedna platná adresa, pošle výzvu sousedovi.

Dostane-li jako odpověď ohlášení směrovače nebo DHCP zprávu, postupuje dál obvyklým způsobem, odpovídajícím bezstavové či stavové konfiguraci. Jedinou specialitou je, že nastavené adresy zároveň ukládá do SDAT ve spojení s příslušným směrovačem. To se samozřejmě týká i jakýchkoli pozdějších změn v automaticky nastavených adresách, ať už k nim dojde z jakýchkoli důvodů.

Nad rámec obvyklých mechanismů jde zmíněné oslovování směrovačů s platnými adresami v SDAT. Pokud odpoví některý z nich, zařízení aktivuje s ním spojené adresy z SDAT a nastaví je příslušnému rozhraní. Pokud by snad došlo ke konfliktu mezi informacemi získanými touto cestou a některým ze standardních mechanismů automatické konfigurace, mají vždy přednost standardní mechanismy.

RFC 6059 předpokládá, že si zařízení bude v SDAT uchovávat vždy jen údaje z několika naposledy navštívených sítí a starší bude průběžně mazat.

Významným omezením jednoduché detekce připojení je, že se zabývá výlučně adresami. Nestará se o směrování, DNS servery ani jiné konfigurační parametry. To do značné míry omezuje její užitečnost – stroj s platnou adresou, který nesměruje, si svou adresu příliš neužije. Jednoduchá detekce může být drobným přínosem, ale základ bude vždy spočívat na bezstavové či stavové automatické konfiguraci.

7 Směrování a směrovací protokoly

Ve všech možných nadstavbách a podpůrných mechanismech by člověk skoro mohl zapomenout, že základním úkolem IP je směrování. Tedy hledání cest, kudy odeslat datagram k danému cíli. Teď se podíváme, jak to IPv6 dělá.

7.1 Elementární směrování

Přestože profesionály v oboru směrování jsou – jak jinak – směrovače, směrovat musí každé zařízení podporující IPv6. Ten nejjednodušší přístup ke směrování lze realizovat na bázi automatické konfigurace (viz část 6.3 na straně 123). Pokud si to však zařízení může kapacitně dovolit, je vhodnější použít jemnější přístup. Stejný dnes používá valná většina zařízení pro IPv4.

směrovací tabulka

Jeho výchozí datovou strukturou je *směrovací tabulka*. Ta obsahuje informace „k cíli A se chodí tudy, k cíli B tamtudy, k cíli C támhletudy a všechno ostatní předáváme semhle“. Jednotlivé cíle jsou identifikovány prostřednictvím IPv6 prefixů (ukládá se jak vlastní prefix, tak jeho délka). Daný cíl může být buď přímo připojen, nebo je ve směrovací tabulce uloženo, že data směřující k němu se mají předávat přímo připojenému směrovači na určité adrese.

Poněkud speciální roli hraje *implicitní cesta (default route)*, která slouží jako „krabička poslední záchrany“ pro ty adresy, ke kterým v tabulce neexistuje specifický záznam. Implicitní cesta je dána prefixem s nulovou délkou. Prefix by v principu mohl být libovolný – vzhledem k nulové délce na něm nezáleží. Doporučuje se však používat samé nuly, což je běžně zažitá konvence. Implicitní cesta je tudíž dána prefixem ::/0.

Tabulka může obsahovat i individuální záznamy pro jednotlivé adresy. V takovém případě je prefixem celá adresa a má délku 128. Tím se de facto implementuje cache cílů. Přidáním lokálních prefixů pak směrovací tabulka plně nahradí rozhodování popsané na obrázku 6.5 na straně 125 a nabídne mnohem více.

Jednoduchý příklad směrovací tabulky vidíte na obrázku 7.1. Pochází z běžného uživatelského počítače s jedinou ethernetovou kartou (rozhraní *eth0*), který je zapojen do lokální sítě. Z ní vede jediná cesta ven, a sice přes směrovač s (lokální linkovou) adresou fe80::21b:8fff:feff:ff. V podobné situaci se nachází drtivá většina strojů v Internetu. Jejich tabulka proto bude, až na konkrétní hodnoty adres, velice podobná[1].

[1] Samozřejmě za předpokladu, že na dotyčném počítači neběží MS Windows Vista. Tento štědrý systém si vytváří neuvěřitelné množství rozhraní, většinou pro tunely, čemuž odpovídá i košatost jeho směrovací tabulky.

	cíl	*předat na adresu*	*rozhraní*
1	::1/128	::	lo
2	fe80::215:cff:fe5d:a/128	::	lo
3	fe80::/64	::	eth0
4	2001:db8:1:3:215:cff:fe5d:a/128	::	lo
5	2001:db8:1:3::/64	::	eth0
6	ff00::/8	::	eth0
7	::/0	fe80::21b:8fff:feff:ff	eth0

Obrázek 7.1: Příklad směrovací tabulky koncového počítače

První záznam najdete v každém zařízení podporujícím IPv6 – jedná se o lokální smyčku (loopback), tedy možnost hovořit sám se sebou. U počítačů je to normální, na rozdíl od lidí. Záznamy 2 a 3 řeší směrování datagramů posílaných na lokální linkové adresy. Adresu fe80::215:cff:fe5d:a počítač přidělil svému rozhraní *eth0* (je odvozena z jeho ethernetové adresy 00:15:0c:5d:00:0a) a směrování pro ni řeší záznam 2 předáním do lokální smyčky. Záznam 3 pak sděluje, že všechny ostatní lokální linkové adresy (prefix fe80::/64) se doručují přímo rozhraním *eth0*. Záznamy 4 a 5 řeší totéž pro globální individuální adresu počítače. Ta se nachází v podsíti 2001:db8:1:3::/64 (položka 5) a její identifikátor rozhraní je samozřejmě shodný s identifikátorem v lokální linkové adrese. Skupinově adresované datagramy mají prefix ff00::/8 a podle záznamu 6 jsou opět doručovány přímo do rozhraní *eth0*, podrobněji se jim budu věnovat v kapitole 8 na straně 163. A konečně sedmý záznam oznamuje, že všechny ostatní adresy mají být předávány implicitnímu směrovači s adresou fe80::21b:8fff:feff:ff (všimněte si, že je použita jeho lokální linková adresa).

směrovací rozhodnutí

Když má daný stroj předat datagram, vyhledá ve své směrovací tabulce všechny záznamy, jejichž cíl odpovídá cílové adrese datagramu. Může jich být několik s různou délkou prefixů (například prefix ::/0 odpovídá libovolné adrese, implicitní směrovací záznam bude proto pokaždé zařazen mezi kandidáty). Z nich vybere ten, jehož prefix je nejdelší, a datagram odešle podle údajů v něm obsažených.

Toto je základní směrovací algoritmus, podle kterého se řídí každé zařízení používající IPv6. Ve srovnání s IPv4 se elementární směrování nezměnilo ani o chlup.

7.2 Směrovací protokoly

Otázkou je, jak je směrovací tabulka rozsáhlá a jakým způsobem vzniká. Ve většině případů (koncové počítače) je velmi jednoduchá, jak jste viděli na obrázku 7.1. Nic složitého není potřeba – zdejším počítačům se data odesílají přímo, o vše ostatní se postará implicitní směrovač. Takováto tabulka bývá

statická a může vzniknout na základě bezstavové automatické konfigurace, DHCPv6 nebo být pevně nakonfigurována.

Směrovače často bývají v komplikovanější pozici. Propojují větší či menší množství sítí a tomu odpovídá i rozsah jejich směrovacích tabulek. Měly by reagovat na změny v topologii sítě a směrování jí dynamicky přizpůsobovat.

směrovací protokol

Proto používají *směrovací protokoly*, kterými se navzájem informují o situaci a na základě těchto údajů pak upravují své tabulky. Směrovací protokol je tedy nástroj, který slouží k výměně informací o topologii sítě a k adaptaci směrovacích tabulek podle ní. V tomto případě hovoříme o dynamickém směrování.

hierarchické směrování

Nejhorší je pochopitelně role těch směrovačů, které mají zvládat celý Internet. Jejich tabulky jsou skutečně monumentální. Aby se daný objem vůbec dal zvládnout, používá se hierarchické přidělování adres a odpovídající směrování. Síť (ať už koncová, regionální či rozlehlá síť poskytovatele připojení) dostane přidělen určitý prefix a veškeré její adresy z něj vycházejí. Mimo danou síť pak směrovačům stačí znát dotyčný prefix a podle něj dopraví data do cílové sítě. Teprve v ní se začne posuzovat podle podrobnějších (delších) prefixů, do které konkrétní části sítě má datagram směřovat[2].

Praktický příklad: Můj počítač s adresou 2001:718:1c01:20:21c:32ff:fe5c:b5e1 se nachází v budově A sítě Technické univerzity v Liberci, která je připojena do akademické sítě CESNET2. Když mi odesílá datagram server řekněme z Japonska, je zdejším směrovačům srdečně jedno, v jaké je budově a k jaké univerzitě patří. Mají jediný prefix pro celou síť CESNET2 (konkrétně 2001:718::/32) a podle něj datagram dopraví až do ní. Jakmile dorazí do CESNET2, zdejší směrovače už mají v tabulkách delší prefixy pro sítě jednotlivých univerzit. Budova (tedy podsíť) je z jejich pohledu stále ještě nezajímavá, bude se směrovat nejprve podle prefixu libereckého uzlu (2001:718:1c00::/40) a po příchodu do něj podle prefixu TU v Liberci 2001:718:1c01::/48. Až když dorazí do univerzitní sítě, začne se při směrování posuzovat prefix sahající až po adresu podsítě (2001:718:1c01:20::/64), datagram bude dopraven do patřičné podsítě a v ní pak mému oblíbenému počítači. Stejný přístup se dnes používá i v IPv4 pod názvem CIDR.

Toto shlukování prefixů provádějí směrovače. Například směrovač, jehož prostřednictvím je připojena síť naší univerzity, sice zná topologii zdejších podsítí, ale směrem ven (tedy ostatním směrovačům sítě CESNET2) ohlašuje jen jedinou položku: prefix celé naší sítě 2001:718:1c01::/48.

autonomní systém

Aby se vše dalo organizačně zvládnout, je Internet rozdělen na tak zvané *autonomní systémy (AS)*. Autonomní systém je tvořen skupinou sítí, které mají jednotnou správu a směrovací politiku. Například běžný poskytovatel Internetu má přidělen svůj autonomní systém, do nějž patří jeho vlastní páteřní síť a sítě zákazníků k ní připojené.

[2] Trochu zjednodušuji, ale v zásadě je to tak.

IGP Existence autonomních systémů rozděluje směrovací protokoly do dvou skupin. Jako *Internal Gateway Protocol (IGP)* jsou označovány ty protokoly, které slouží ke správě směrovacích tabulek uvnitř jednoho autonomního systému. Zde bývá situace relativně jednoduchá a dotyčné protokoly se snaží především o to, aby rychle reagovaly na změny v síti. V současné době existují tři IGP protokoly použitelné pro IPv6: RIPng, IS-IS a OSPFv3.

EGP *External Gateway Protocol (EGP)* naproti tomu slouží k výměně směrovacích informací mezi autonomními systémy. Protokoly této kategorie drží Internet pohromadě – jejich prostřednictvím se směrovače dozvídají, kudy do kterého autonomního systému a jaké prefixy jsou v něm dostupné. Vstupní směrovač autonomního systému tudíž má ve svých tabulkách prefixy do celého Internetu a musí mít adekvátní kapacitu.

Protokoly ze skupiny EGP usilují především o to, aby vůbec zvládly obrovské objemy informací, které musí přenášet. Proto jsou ve srovnání s IGP konzervativnější a reagují pomaleji. V současné době se používá jediný protokol této třídy – BGP4. Pro IPv6 slouží jeho upravená verze BGP4+.

7.3 RIPng

Routing Information Protocol (RIP) patří mezi internetové veterány. Byl navržen a implementován již ve velmi raných dobách a dodnes nachází využití především v koncových sítích.

Protokol má několik závažných omezení (především velmi malou maximální délku cesty a pomalejší reakci na změny), je však velmi jednoduchý. Díky tomu jeho implementaci najdete prakticky v každém systému. Máte-li nepříliš rozsáhlou síť, kterou chcete dynamicky směrovat, je RIP nejjednodušší cestou.

RIPng představuje reinkarnaci starého dobrého protokolu. Vychází z druhé verze RIPu, která byla definována v první polovině devadesátých let ve snaze odstranit některé nedostatky původního protokolu. Ve srovnání s RIPv2 se RIPng liší prakticky jen v tom, že používá adresy ve formátu IPv6. Je definován v RFC 2080: *RIPng for IPv6*.

základní princip Jedná se o protokol založený na vektoru vzdáleností. Linky a sítě, propojující jednotlivé směrovače, mají přiřazenu určitou cenu. Má-li datagram projít jistou cestou, určí se její celková cena (vzdálenost, metrika) součtem cen linek, kterými prošel. RIPng se snaží, aby datagramy k danému cíli vždy dorazily cestou s nejmenší celkovou cenou.

šíření informací Daný ideál je realizován tak, že stroj používající RIP si k jednotlivým cílům ve směrovací tabulce poznamenává i údaj o tom, jak dlouhá je cesta k nim. V půlminutových intervalech údaje ze své tabulky pošle všem sousedům. Když sám obdrží tabulku od některého ze sousedů, přičte si k ní cenu linky, kterou ohlášení dorazilo, a porovná s údaji ze své tabulky. Pokud se dozví

o novém cíli, lepší cestě nebo se cesta vedoucí přes tento sousední směrovač zhoršila, upraví svou tabulku.

RIPng je určen pro menší sítě, čemuž odpovídá zvolená metrika pro oceňování cest. Povolenými hodnotami jsou celá čísla v rozsahu 1–15, hodnota 16 už představuje nekonečno, tedy nedosažitelný cíl. Důsledkem je, že správce sítě nemá prakticky žádný prostor k tomu, aby vhodně zvolenou cenou vyjádřil průchodnost. Ve valné většině případů se všechny linky oceňují hodnotou 1 a cena cesty pak vyjadřuje, kolika linkami datagram po své cestě projde (anglicky se tomu říká hop count).

Směrovací tabulka musí pro potřeby RIPng ke každému cíli obsahovat následující údaje:

- prefix cíle (jeho hodnotu a délku)
- metriku odpovídající celkové ceně cesty
- adresu dalšího směrovače na cestě (komu má předávat datagramy směřující k tomuto cíli)
- příznak změny (zda se v poslední době změnila)
- časovače: dobu platnosti a likvidační interval

Pokud se týče sítí, ke kterým je dotyčný stroj přímo připojen, tam adresa dalšího směrovače chybí a metrika je rovna ceně této linky (zpravidla 1).

Údaje ze své tabulky posílá v následujících případech:

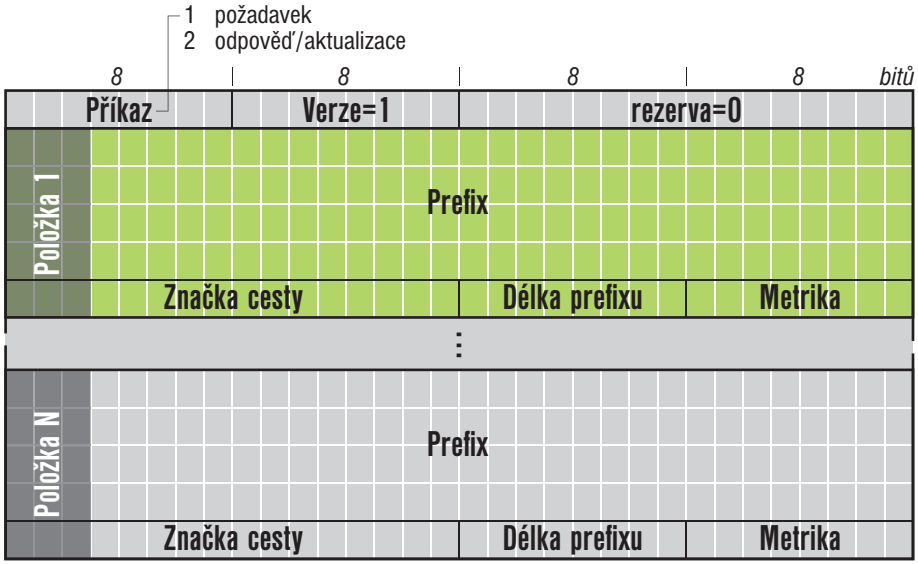
- *pravidelná aktualizace* zasílaná každých 30 s; aby nedocházelo k nežádoucí synchronizaci mezi zprávami jednotlivých směrovačů, je tento interval vždy posunut o náhodný čas z intervalu od –15 s do 15 s
- *aktualizace vyvolaná změnou (triggered update)*, kterou zasílá, když došlo ke změně jeho směrovacích tabulek
- *odpověď na požadavek*, když některý ze sousedů požádá o informace

V prvních dvou případech se aktualizace posílá všem sousedům. Konkrétně ji stroj zašle do všech sítí, k nimž je přímo připojen, na skupinovou adresu pro všechny RIPng směrovače ff02::9. Odpověď se pak posílá jen tomu, kdo zaslal požadavek.

formát zprávy

Formát zprávy protokolu RIPng vidíte na obrázku 7.2. Je poměrně jednoduchý – v prvních dvou bajtech je identifikována *Verze (Version)* protokolu (v současnosti 1) a *Příkaz (Command)*. Ty jsou k dispozici jen dva. Hodnota 1 signalizuje požadavek, hodnota 2 odpověď či aktualizaci. Zbytek zprávy tvoří záznamy odpovídající položkám ze směrovací tabulky.

Položky popisující jednotlivé cíle obsahují v první řadě *Prefix* a jeho *Délku (Prefix len)*, které určují cíl této položky. Dále je v něm uvedena *Metrika*



Obrázek 7.2: Formát zprávy pro RIPng

(Metric), tedy vzdálenost k cíli ze směrovače, jenž položku odeslal. *Značka cesty (Route tag)* pak může obsahovat atribut, který je cestě přiřazen, musí s ní být uchován a dále redistribuován. Slouží k tomu, aby se jeho prostřednictvím mohly předávat informace získané z jiných směrovacích protokolů, které v RIPng nemají žádný význam, ale při konverzi do jiného směrovacího protokolu na dalším směrovači by jej opět mohly nabýt. Tyto informace se tedy mechanicky předávají, protože možná někomu dalšímu k něčemu budou.

obsah zprávy

Výše uvedené tři případy rozesílání tabulky se liší především tím, jaké informace bude obsahovat. Při *pravidelné aktualizaci* se odešle kompletní směrovací tabulka[3].

Aktualizace vyvolaná změnou bude obsahovat jen ty položky, které mají nastaven příznak změny. Následně se příznak vynuluje, protože změny byly právě ohlášeny. Tuto aktualizaci stroj odešle, když dojde ke změně jeho směrovacích tabulek (například po obdržení informací od souseda nebo změní-li se stav některé z přímo připojených sítí). Cílem je, aby se sousedé o této změně dozvěděli pokud možno co nejdříve. Lze očekávat, že u nich také vyvolá změnu směrovací tabulky a následně i oni zašlou vyvolanou aktualizaci svým sousedům a tak dále. Informace se díky tomu celkem rychle rozšíří po celé síti.

[3] Některé cesty mohou být vynechány – viz algoritmus rozděleného horizontu, který je popsán dále.

Potenciální problém spočívá v tom, aby ke změnám nedocházelo až příliš často. Proto se vždy po odeslání zprávy s vyvolanou aktualizací nastaví čítač na náhodnou dobu z rozsahu 1–5 s. Po tuto dobu je zablokováno zasílání vyvolaných aktualizací. Všechny změněné údaje budou odeslány až po jeho vypršení v jedné společné aktualizaci.

Odpověď se váže na požadavek a obsahuje ta data, o která si vyzyvatel řekl. Požadavek je konstruován stejně jako aktualizace, až na to, že obsahuje kód příkazu 1. Jeho položky stanoví, o které cesty má tazatel zájem. Existuje speciální případ, kdy dotazovaný má zaslat svou kompletní směrovací tabulku. Takový požadavek musí obsahovat právě jednu položku, v níž má *Prefix* i *Délka prefixu* hodnotu 0 a *Metrika* je rovna 16.

Jako odpověď na tento speciální požadavek pošle dotázaný směrovač celou svou tabulku, jako při pravidelné aktualizaci. V opačném případě projde položky v požadavku a sestaví z nich odpověď. Pokud má daný cíl ve své směrovací tabulce, vloží odpovídající informace. Jestliže cíl ve směrovací tabulce nemá, zařadí pro něj do odpovědi metriku 16. Sestavenou odpověď pak zašle žadateli.

Sekvenci požadavků a odpovědí lze využít například ke sledování stavu a činnosti směrovače. Nejčastější využití však najde při startu směrovače. Aby se co nejrychleji naučil směrovat, rozešle všem sousedům žádost o jejich kompletní tabulky a na jejich základě sestaví své.

Je-li směrovací tabulka rozsáhlejší, může vzniknout problém s velikostí zprávy. Ta nesmí překročit MTU linky, do které je zasílána. Pokud by měla být větší, musí se rozdělit do několika datagramů.

zpracování zprávy

Při příchodu aktualizace či odpovědi od některého ze sousedů ji příjemce porovná se svými záznamy. Prochází jeden záznam z aktualizace za druhým a zpracovává jej. Nejprve k metrice přičte cenu linky, kterou zpráva dorazila. Pokud je výsledná hodnota větší než 16, upraví ji na 16 (nekonečno).

Následně pak záznam porovná se svou tabulkou. Jestliže v ní daný cíl ještě nemá a metrika je menší než 16, přidá si cíl do směrovací tabulky, nastaví mu příznak změny cesty a signalizuje odesílací části, že je třeba vyslat aktualizaci vyvolanou změnou.

Pokud daný cíl už má v tabulce, zajímá se o adresu dalšího směrovače pro něj. Pokud se liší od odesilatele aktualizace a metrika v aktualizaci je kratší, přepíše si záznam ve směrovací tabulce. Nastaví v ní metriku a další směrovač podle aktualizace, nastaví příznak změny a požádá o zaslání vyvolané aktualizace.

Jestliže má ve směrovací tabulce jako další směrovač pro daný cíl uvedeného odesilatele aktualizace, postupuje poněkud odlišně. Pokud se metrika

v aktualizaci liší od směrovací tabulky, uloží si ji[4], nastaví příznak změny a také tentokrát požádá o odeslání vyvolané aktualizace. Je-li metrika stejná, jen si poznamená, že dotyčná cesta je stále platná.

Každá položka má totiž přidělen časovač, udávající její platnost. Při jejím založení, změně či potvrzení platnosti se nastaví na 180 s. Pokud potvrzení nepřichází, doba platnosti klesá až k nule. Dojde-li do této hodnoty, znamená to, že položka je neplatná a bude odstraněna.

odstranění položky

Likvidace položky může být vyvolána dvěma událostmi: když vyprší její doba platnosti nebo když směrovač, přes nějž vedla, ohlásí metriku 16. V obou případech je inicializován likvidační interval na hodnotu 120 s. Kromě toho se položce nastaví metrika 16 a příznak změny. Výstupní části se předá požadavek na odeslání vyvolané aktualizace.

Během likvidačního intervalu je položka zařazována do všech odpovědí a aktualizací. Pokud v této době dorazí nová položka pro daný cíl, zapíší se odpovídající údaje a likvidační interval bude zrušen. V opačném případě interval vyprší a položka bude odstraněna ze směrovací tabulky.

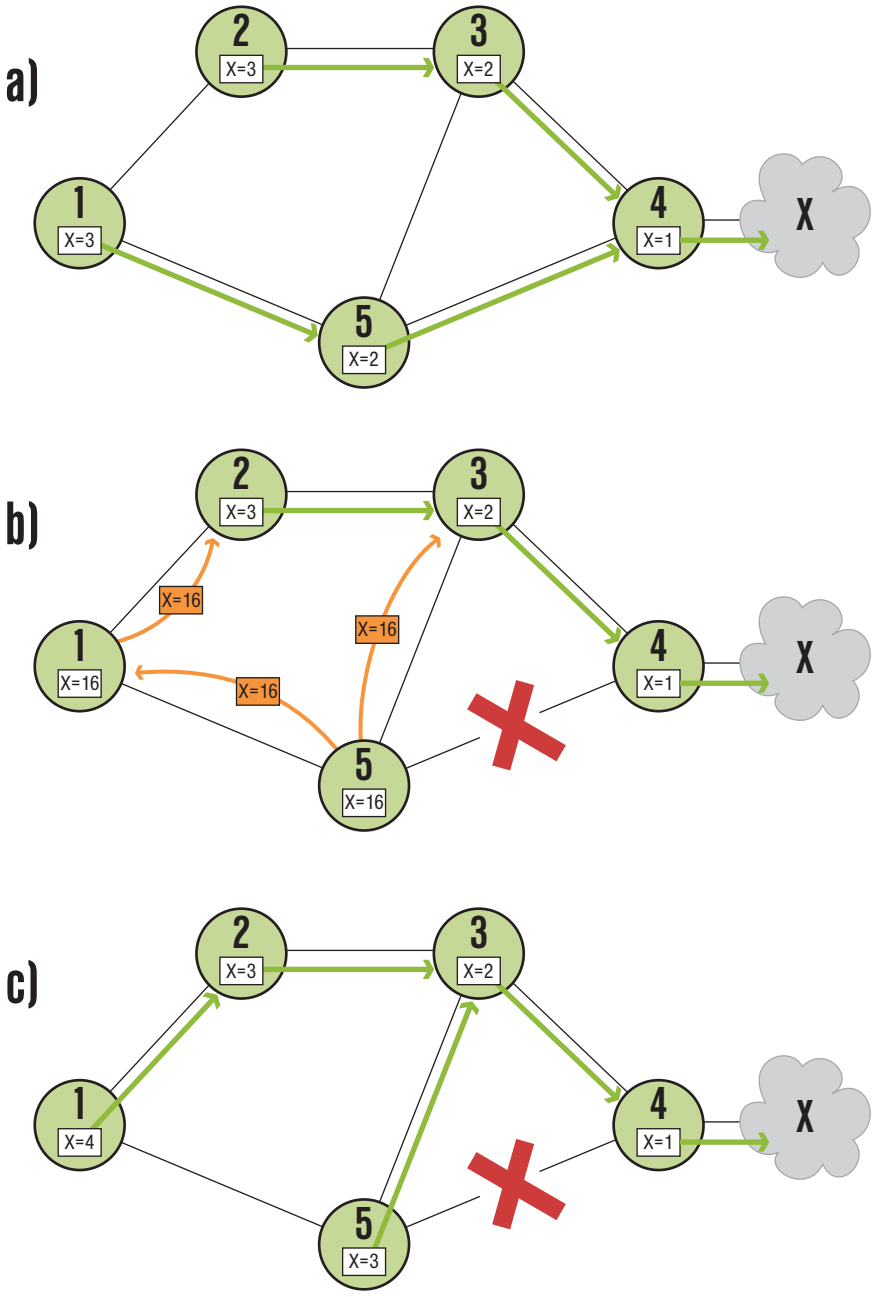
Chování RIPng v konkrétní situaci ilustruje obrázek 7.3. Jeho část a) znázorňuje výchozí situaci pro cílovou síť X. U každého směrovače je uvedena metrika, kterou danému cíli přiřadil, a šipka znázorňuje směrovač, přes nějž vede cesta s touto metrikou.

Na obrázku 7.3b došlo k přerušení spoje mezi směrovači 4 a 5. Pro směrovač 5 to znamená, že cíl X se pro něj stal nedosažitelným (směrovač, přes nějž vedla cesta, je nyní nedostupný). Musí si změnit metriku na 16 a informuje o tom své sousedy vyvolanou aktualizací. Směrovač 3 tuto zprávu ignoruje, protože zná lepší cestu k X. Naproti tomu nejlepší cesta ze směrovače 1 vedla přes směrovač 5. Proto si musí poznamenat novou metriku do své tabulky a informovat o této změně svého souseda 2. Ten však opět zná lepší cestu, takže svou tabulku nezmění.

Pro směrovače 1 a 5 je nyní síť X nedostupná. Teprve až jim dorazí aktualizace od směrovačů 2 a 3, dozvědí se o existenci alternativní cesty. Výsledné cesty v daném případě závisí na tom, v jakém pořadí budou aktualizace odeslány, protože se směrovači 1 nabízejí dvě stejně dlouhé cesty.

Obrázek 7.3c vznikl takto: Nejprve odeslal svou aktualizaci směrovač 2. Směrovač 1 se tak dozvěděl o cestě k cíli X, která vede přes 2 a má délku 4. Zanesl si ji do směrovací tabulky a ihned odeslal vyvolanou aktualizaci. Díky ní si směrovač 5 poznamenal cestu k X vedoucí přes směrovač 1 s metrikou 5 a také on odeslal vyvolanou aktualizaci, která však žádnou senzaci nezpůsobila. Když k němu za chvilku dorazila aktualizace od směrovače 3,

[4] I v případě, že je větší než stávající hodnota ve směrovací tabulce. Cesta vede přes tento směrovač a pokud se zhoršila, je třeba to akceptovat. Třeba to dá šanci cestě vedoucí jinudy, která se teď prosadí.



Obrázek 7.3: Reakce RIPng na změnu v síti

dozvěděl se o lepší cestě s délkou 3. Tu si hbitě poznamenal a vyvolanou aktualizací o tom informoval směrovač 1.

Ten se nyní ocitl v situaci „osel a dvě kupky sena“. Byla mu ohlášena jiná cesta k cíli X, jejíž délka se shoduje s cestou v jeho směrovacích tabulkách. RIPng v takovém případě doporučuje být konzervativní a zůstat věrný cestě, která je obsažena ve směrovací tabulce, aby se nezasílalo zbytečně mnoho vyvolaných aktualizací. Jedinou výjimkou je, pokud se dotyčné položce ve směrovací tabulce krátí životnost. Hrozí-li její brzké vypršení, je doporučeno přejít na novou stejně dlouhou cestu a předejít tak hrozící krátkodobé nedostupnosti daného cíle.

Námět k zamyšlení: Jak by se vyvíjely metriky a cesty, kdyby směrovač 3 poslal svou pravidelnou aktualizaci dříve než směrovač 2?

rozdělený horizont

Jeden z problémů původního RIPu byl způsobován ohlašováním cest těm směrovačům, přes něž vedly. Například směrovače 3 a 5 z obrázku 7.3a by ohlašovaly směrovači 4, že znají cestu k síti X s metrikou 2, která vede právě přes směrovač 4. Kdyby ten ztratil spojení se sítí X a vzápětí dostal toto ohlášení, zaznamenal by si do směrovací tabulky, že k cíli X zná cestu například přes směrovač 5 s metrikou 3. Tím se však směrování zacyklí – datagramy směřující do sítě X si směrovače 4 a 5 budou předávat mezi sebou, dokud nevyprší jejich životnost. S dalšími aktualizacemi se metrika pro síť X bude zvětšovat (směrovač 4 nyní ohlásí metriku 3, takže 3 a 5 si svou metriku zvětší na 4 atd.), ale potrvá určitý čas, než dospěje k hodnotě 16, která podle pravdy vyjadřuje, že síť X je nedostupná.

Vyvolané aktualizace (které původní RIP neměl) podstatným způsobem zkracují dobu, kterou toto poznání pravdy trvá. Nicméně je lepší, aby podobné situace byly z principu vyloučeny. O to se stará mechanismus nazvaný *rozdělený horizont (split horizon)*. Jeho myšlenka je prostá: cíle se neohlašují tomu, od koho je máme. Exaktně řečeno se z ohlášení zasílaného do určité sítě vynechají všechny cíle, pro které leží následující směrovač v dané síti.

Existuje ještě jiná varianta tohoto algoritmu nazvaná *otrávený návrat (poisoned reverse)*. V ní se dotyčné cíle do aktualizace sice zařazují, avšak přiřadí se jim metrika 16. Je doporučeno, aby směrovač používal některou z těchto dvou variant.

7.4 OSPF

Open Shortest Path First (OSPF) je podstatně mladší, složitější a rafinovanější protokol než starý dobrý RIP. Není založen na vektoru vzdáleností, ale na stavu linek. To znamená, že každý směrovač v síti používající OSPF si udržuje aktuální mapu této sítě. Obsahuje informace o tom, kdo je s kým jak propojen, jaké jsou prefixy jednotlivých podsítí, ceny linek a podobně.

OSPF dbá na to, aby všechny směrovače v síti měly stejnou mapu. Kdykoli u některého z nich dojde ke změně, okamžitě o tom informuje všechny ostatní směrovače, aby si aktualizovaly svou mapu. Z ní si každý spočítá strom nejkratších vzdáleností ke všem známým cílům, jehož kořenem je on sám. Tak zjistí, kudy od něj vede nejkratší cesta ke každému cíli a zanese si ji do směrovací tabulky.

Základními výhodami OSPF je velmi rychlá reakce na změny a schopnost zajistit směrování i v poměrně rozsáhlých sítích. V současnosti je pro IPv4 široce používána jeho druhá verze, jejíž definici najdete v RFC 2328: *OSPF Version 2*. Úpravy OSPF pro směrování IPv6 určuje RFC 5340: *OSPF for IPv6*. Nová verze protokolu je označována jako OSPFv3. Vedle podpory IPv6 by měla zahrnovat i různá rozšíření, navržená pro IPv4, jako je využití OSPF pro směrování skupinově adresovaných datagramů (RFC 1584) a další.

mapa sítě

Mapa sítě (též databáze linek) je vlastně orientovaný graf. Jeho vrcholy představují směrovače a skupinové sítě. Pro zjednodušení budu brát v úvahu jen dva druhy linek: dvoubodové, které navzájem spojují dvojici směrovačů, a skupinové, k nimž může být připojeno směrovačů několik a podporují skupinové adresování. Typickou dvoubodovou linkou je ADSL připojení domácí sítě k Internetu, typickou skupinovou Ethernet.

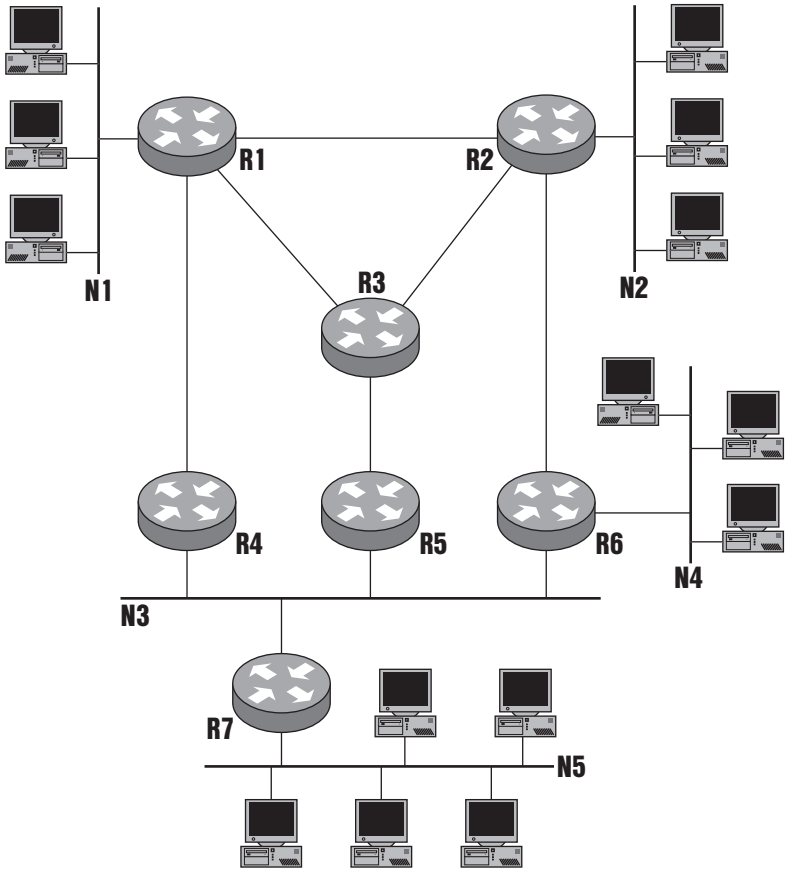
Pokud jsou dva směrovače propojeny dvoubodovou linkou, vede mezi nimi v síťovém grafu obousměrná cesta. Jestliže cesta propojuje směrovač a skupinovou síť, znamená to, že dotyčný směrovač má rozhraní vedoucí do této sítě. Takže například síť z obrázku 7.4 bude z pohledu OSPF reprezentována grafem 7.5.

Všimněte si, že do sítí N1, N2, N4 a N5 vede pouze jednosměrná šipka. Je to proto, že se jedná o tak zvané *koncové sítě* (anglicky stub networks, čili pahýly). Jejich charakteristickou vlastností je, že nevedou nikam dál. Veškeré datagramy, které se v nich objeví, byly buď odeslány některým ze zdejších strojů nebo jsou mu určeny. Naproti tomu N3 je typickým příkladem *tranzitní sítě*, kterou může procházet i provoz, který tu ani nevznikl, ani nekončí. Hrany, které ji spojují s připojenými směrovači, jsou obousměrné.

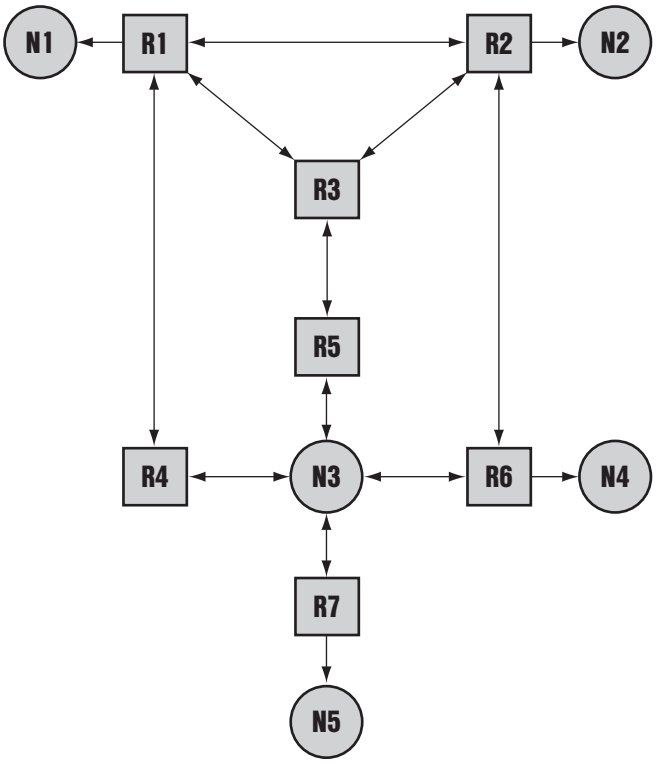
Hrany v grafu sítě jsou ohodnoceny celými čísly v rozmezí 0–65 535. Číslo představuje „délku“ příslušné cesty (oficiálně se v OSPF mluví o ceně cesty). Vzhledem ke značnému rozpětí dostupných hodnot se v ocenění hrany projevuje i rychlost linky. OSPF při hledání optimálních cest sčítá ceny linek a hledá ty, které mají nejnižší součet. Snadno tak třeba odhalí, že do daného místa bude výhodnější přepravit datagram po třech gigabitových Ethernetech než jednou ADSL linkou s rychlostí 2 Mb/s.

sousedé a okolní

Životně důležitým prvkem OSPF je výměna informací o změnách v topologii. Vychází ze sítě sousedů, kterou si protokol vytvoří. Vznikne tak, že každý směrovač si automaticky zjišťuje, které další směrovače se nacházejí v jeho okolí. Zařadí si je do jedné ze dvou kategorií:



Obrázek 7.4: Příklad sítě



Obrázek 7.5: OSPF reprezentace sítě z obrázku 7.4

- *Okolní směrovače (neighbors)* jsou takové, se kterými má přímé spojení. To znamená, že je s nimi spojen dvoubodovou linkou nebo mají připojení k téže skupinové lince.
- *Sousedé (adjacent routers)* se vybírají z okolních směrovačů (ne každý okolní směrovač se stane sousedem, viz níže). Se sousedy si vyměňuje informace o mapě sítě.

Jak si směrovač vybírá sousedy? Pro dvoubodové spoje zcela jednoduše. Tam se navzájem propojená dvojice směrovačů vždy stane sousedy. Složitější situace vzniká u skupinových linek. Směrovače připojené k téže skupinové lince si ze svého středu zvolí *pověřený směrovač (designated router)*. Ten se stane sousedem pro všechny zdejší stroje. Ostatní mezi sebou sousedské vztahy nenavazují[5].

[5] Mám-li být upřímný, ještě si zvolí záložní pověřený směrovač, který je připraven kdykoli nahradit pověřence, pokud by zanikl. Také on se stane sousedem pro všechny ostatní na téže lince, ale už nikdo další, vážně.

Informace o svém okolí směrovač získá tak, že opakovaně posílá na všechna svá rozhraní zprávu oznamující jeho přítomnost – tak zvaný *Hello* paket. Do něj uvede identifikaci pověřeného směrovače pro danou síť a také identifikátory všech směrovačů, o nichž ví (nedávno od nich obdržel Hello paket). Nováček se z těchto paketů dozví, kdo všechno je na dané lince přítomen a kdo je pověřeným směrovačem (pokud je).

synchronizace map

Mapa sítě, alias databáze linek, je vlastně kolekce jednotlivých oznámení o stavu v určitém místě. Toto oznámení se v OSPF jmenuje *Link State Advertisement (LSA)* a posílá je vždy ten, u nějž informace vzniká. Existuje několik typů LSA, které shrnuje tabulka 7.1.

kód	*název*	*význam*
1	směrovač	stav rozhraní daného směrovače, posílá každý směrovač
2	síť	seznam směrovačů připojených k síti, posílá pověřený směrovač sítě
3, 4	souhrn	cesta k cíli, jenž leží mimo danou oblast (ale uvnitř AS), posílají hraniční směrovače oblasti
5	externí	cesta k cíli z jiného AS, posílají hraniční směrovače AS

Tabulka 7.1: Typy LSA zpráv

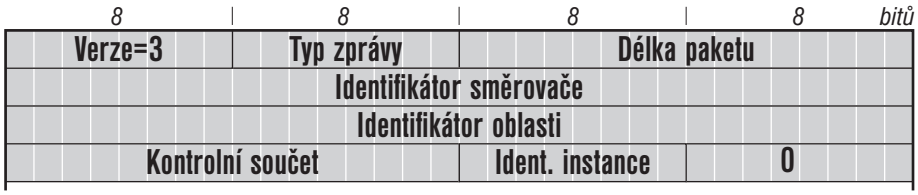
Když dva směrovače nově naváží sousedský vztah, musí si nejprve synchronizovat své mapy sítě. Dělají to tak, že pošlou svému protějšku sadu OSPF zpráv *Popis databáze (Database description)*, ve kterých vyjmenují identifikátory a verze LSA, tvořících jejich databázi. Protějšek si poznamená ta LSA, která dosud neznal nebo má jejich starší verzi. Následně o ně požádá pomocí *Žádosti o stav linky (Link state request)* a očekává, že mu soused pošle *Aktualizaci stavu linky (Link state update)* pro všechny požadované. Jakmile se tak stane, jsou databáze synchronní a oba sousedi vědí, že jejich pohled na síť je totožný.

Kdykoli později dojde ke změně (například se k některému směrovači připojí nová síť), odpovídající směrovač o tom neprodleně informuje všechny své sousedy prostřednictvím *Aktualizace stavu linky*. Ta obsahuje příslušné LSA, které si sousedé poznamenají do mapy a okamžitě předají dále všem svým sousedům a ti zase svým... Informace se tak velmi rychle dostane do všech směrovačů v síti a mapy se opět stanou synchronními.

Tento postup, kdy se doraziví novinka okamžitě předá všem ostatním sousedům, se nazývá záplavový algoritmus (flooding). Má tu příjemnou vlastnost, že nevyžaduje žádné velké přemýšlení a přitom se dostane všude, a to nejkratší cestou (protože použije všechny). Aby nedocházelo k cyklům a opakování aktualizací, předávají se dále jen ta LSA, která dotyčný dosud

neznal nebo měl jejich starší verzi. Doručení aktualizace se potvrzuje (zprávou *Potvrzení stavu linky, Link state acknowledgment*), aby soused věděl, že jeho oznámení dorazilo v pořádku.

Synchronizace map je tedy velmi rychlá, je garantována a navíc se posílají jen novinky, takže OSPF má jen velmi malou režii. Kromě toho se pro jistotu stavová informace pošle ještě čas od času, i když se nezměnila. Jistota je jistota.



Obrázek 7.6: Hlavička OSPF zprávy

typ	*název*	*význam*
1	Hello	zjištění okolních směrovačů
2	Popis databáze	shrnuje obsah databáze
3	Žádost o stav linky	požaduje LSA
4	Aktualizace stavu linky	aktualizace databáze (posílá LSA)
5	Potvrzení stavu linky	potvrzuje aktualizaci

Tabulka 7.2: Typy OSPF zpráv

oblasti

Tím jsme ovšem se schopnostmi OSPF neskončili. Protokol byl navržen s cílem zvládnout i opravdu velké sítě, v nichž by se synchronizace map mohla stát velmi náročnou. Proto byl do OSPF zařazen koncept oblastí, které rozdělují autonomní systém na části a omezují objem přenášených směrovacích informací.

Za *oblast (area)* je v OSPF označována skupina souvislých sítí a strojů v nich a také všechny směrovače, které mají rozhraní do některé z těchto sítí. Každá oblast provozuje svůj nezávislý exemplář směrovacího algoritmu a udržuje si mapy, které pokrývají pouze její vlastní síť. Všechny operace, které jsem popsal výše, tedy probíhají jen v rámci jedné oblasti. Díky tomu významně klesá režie spojená s aktualizací map.

Směrovač, který má rozhraní do více než jedné oblasti, se nazývá *hraniční směrovač (border router)*. Musí provozovat nezávislou kopii směrovacího algoritmu a samostatnou mapu sítě pro každou z oblastí, do nichž je zapojen. V podstatě se tváří jako několik různých směrovačů – pro každou oblast jeden.

Celý autonomní systém drží pohromadě *páteřní oblast* s identifikačním číslem 0.0.0.0 (identifikátory oblastí jsou 32bitové a pro jejich zápis se vžila stejná konvence jako pro IPv4 adresy). Aby bylo směrování jednoduché, požaduje OSPF, aby všechny hraniční směrovače patřily do páteřní oblasti.

To znamená, že cestu mezi libovolnou dvojicí počítačů v různých oblastech lze rozdělit na tři části: První prochází oblastí s odesilatelem a končí na některém jejím hraničním směrovači. Druhá vede páteřní oblastí k hraničnímu směrovači, který spojuje páteřní a cílovou oblast. A na ni navazuje závěrečná část cesty, která vede cílovou oblastí od jejího hraničního směrovače k cíli.

Příklad rozdělení naší známé sítě na oblasti uvádí obrázek 7.7. Lehce jsem ji rozšířil přidáním směrovače R8 a sítě N5 vpravo nahoru, aby byla zajímavější. Všimněte si, že například směrovač R1 má jedno rozhraní v oblasti 0.0.0.1 a tři rozhraní v páteřní oblasti 0.0.0.0. Většina směrovačů na obrázku je hraničních. Vnitřní jsou jen R3, R7 a R8, jejichž všechna rozhraní leží vždy v jediné oblasti.

směrování mezi oblastmi

Směrovače v oblasti nemusí znát detailně situaci za jejími hranicemi, ale musí mít alespoň rámcový přehled o tom, jaké sítě se tam nacházejí. Vlastně jim stačí vědět, že k cíli *X* vede nejvýhodnější cesta přes zdejší hraniční směrovač *Y*.

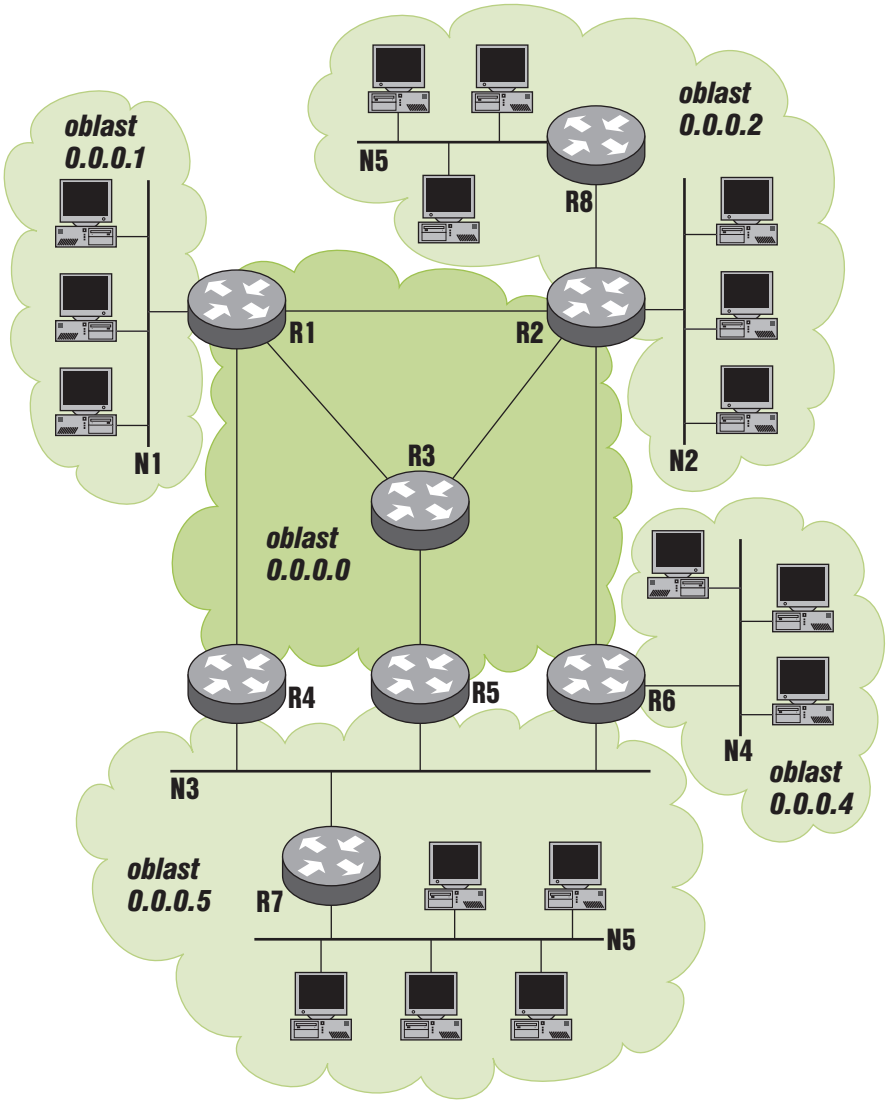
Proto hraniční směrovače předávají do okolí informace o situaci v oblasti. Neposílají však její kompletní mapu, ale pouze jakýsi souhrn. V ideálním případě shrnou celou připojenou oblast do jediného LSA záznamu, který říká třeba „za mnou leží síť s prefixem 2001:db8:abcd::/48“. Míra agregace je nastavitelná, takže si správce sítě může řídit, jak detailní informace se budou přenášet.

Tyto souhrnné LSA záznamy jsou v ostatních oblastech šířeny stejně jako všechny ostatní. Díky nim se zdejší stroje dozvědí, jak mají směrovat datagramy určené počítačům z jiných oblastí. Například v oblasti 0.0.0.5 na obrázku 7.7 se díky nim všichni dozvědí, že nejlepší cesta do sítě N1 vede přes hraniční směrovač R4, zatímco do N2 a N4 to bude nejkratší přes R6.

Podobně se řeší i distribuce cest k cílům z jiných autonomních systémů. Hraniční směrovač AS, který je spojen s okolním světem, se je dozvídá prostřednictvím externího směrovacího protokolu (typicky BGP) a předává je v podobě externích LSA do páteřní oblasti. Hraniční směrovače je pak předávají dále do ostatních oblastí.

Tedy, pokud je to třeba. Oblast totiž lze definovat jako *koncovou (stub area)*[6] a v takovém případě se do ní externí LSA nepředávají. Směrování za hranice AS je zde prováděno pomocí implicitní cesty. Hraniční směrovač (nebo směrovače) propaguje do koncové oblasti implicitní cestu a říká „všechno

[6] zase ty pahýly



Obrázek 7.7: Rozdělení sítě na oblasti

ostatní posílejte přese mne“. Žhavými kandidáty na koncové oblasti v naší ukázkové síti budou 0.0.0.1, 0.0.0.2 a 0.0.0.4. Všechny mají jen jediný hraniční směrovač, takže nemají o čem přemýšlet.

IPv6 Všechny výše popsané mechanismy jsou společné pro obě verze OSPF. Podpora IPv6 znamenala jen malé úpravy protokolu a paradoxně přinesla jeho jisté zjednodušení. OSPFv2 totiž definuje bezpečnostní prvky, kterými se chrání před záškodnickými směrovači. V OSPFv3 tento prvek mizí a je nahrazen standardním IPsec přímo na úrovni IP.

Došlo samozřejmě k úpravě LSA pro dlouhé adresy. Ze všech ostatních míst (jako např. identifikace směrovače či sítě) byly odstraněny IP adresy a lišácky nahrazeny 32bitovými identifikátory. Čili je tam totéž, ale jmenuje se to jinak. Terminologie se změnila také u podsítí, které byly nahrazeny linkami. Sečteno a podtrženo: došlo k mírnému pokroku v mezích zákona a OSPFv3 se hodně podobá svému předchůdci.

7.5 IS-IS

Protokol nazvaný *Intermediate system to intermediate system (IS-IS)* má v internetovém světě zcela zvláštní pozici, jedná se totiž o vetřelce zvenčí. Původně byl vyvinut firmou Digital Equipment Corporation pro její síťovou architekturu DECnet Phase V, budiž jim země lehká. Následně jej převzala mezinárodní standardizační organizace ISO jako směrovací protokol pro svůj referenční model OSI, budiž mu země lehká.

Protokoly kolem OSI bývaly něco jako švýcarský kapesní nožík s 287 nástroji. Nabízí sice úplně všechno, ale váží 5 kg a po hodinové námaze s ním možná ukrojíte krajíc chleba. Všeobecný neúspěch rodiny OSI s sebou strhl i IS-IS, který dlouhá léta zůstával zcela na okraji zájmu internetové komunity, přestože se jedná o docela povedený protokol. Přesněji řečeno, internetová komunita si jej všimla, uznala jej za zdařilý a vyvinula si vlastní variaci na stejné téma – OSPF. Ta potom v míře nasazení nechala svůj vzor daleko za sebou.

Přesto ale IS-IS nelze odbýt jako historickou kuriozitu, protože zejména v poslední době začíná zvolna získávat na popularitě. A to zejména v souvislosti s IPv6.

Základní koncept IS-IS je totožný s OSPF. Jedná se o protokol založený na stavu linek, v němž si všechny směrovače udržují aktuální mapu sítě a z ní vypočítávají algoritmem pro hledání nejkratších cest svou směrovací tabulku. Pokud dojde ke změně, informace o ní se okamžitě šíří záplavovým algoritmem, aby byly mapy sítí všech směrovačů co nejrychleji aktualizovány. Technické detaily se liší (terminologie, formáty zpráv, postupy pro jejich zpracování), ale principiálně jsou si OSPF a IS-IS velmi blízko.

hierarchické směrování

Významnější rozdíly mezi oběma protokoly panují v oblasti hierarchického směrování. IS-IS také dělí síť na oblasti a udržuje kompletní mapu topologie jen v rámci jedné oblasti. Zde ovšem celý směrovač patří vždy do jedné oblasti a hranice mezi oblastmi procházejí linkami. Naproti tomu v OSPF jsou do oblastí zařazována jednotlivá rozhraní a hranice procházejí uvnitř směrovačů. V IS-IS proto nelze sestavit přímou analogii obrázku 7.7. Původní hraniční směrovače R1, R2, R4, R5 a R6 je třeba zařadit jen do jedné oblasti a tomu přizpůsobit hranice. Aby v IS-IS vznikla oblast, musí obsahovat alespoň jeden směrovač. Naproti tomu v OSPF lze vytvořit oblast jediným rozhraním na směrovači, jako třeba oblast 0.0.0.1 na obrázku 7.7. Nechme stranou úvahy, zda je taková oblast smysluplná, když z hlediska OSPF nemá vlastně žádný obsah.

Směrovače jsou v IS-IS rozděleny do dvou úrovní. Jako směrovače úrovně 1 jsou označeny ty, které se nacházejí uvnitř oblasti a starají se především o její vnitřní topologii. Směrovače úrovně 2 pak zajišťují komunikaci a výměnu informací mezi oblastmi. Na obrázku 7.8 znázorňujícím možné oblasti v naší síti jsou tyto směrovače zvýrazněny zeleně. Spolu se baví vždy jen směrovače stejné úrovně. Aby byla možná vzájemná komunikace mezi vnitřním životem oblasti a jejím okolím, může mít směrovač obě úrovně. Je pak označován jako L1/L2 směrovač. Pokud je oblast triviální, jako například oblast 1 na obrázku 7.8, může obsahovat jen jediný směrovač úrovně L2.

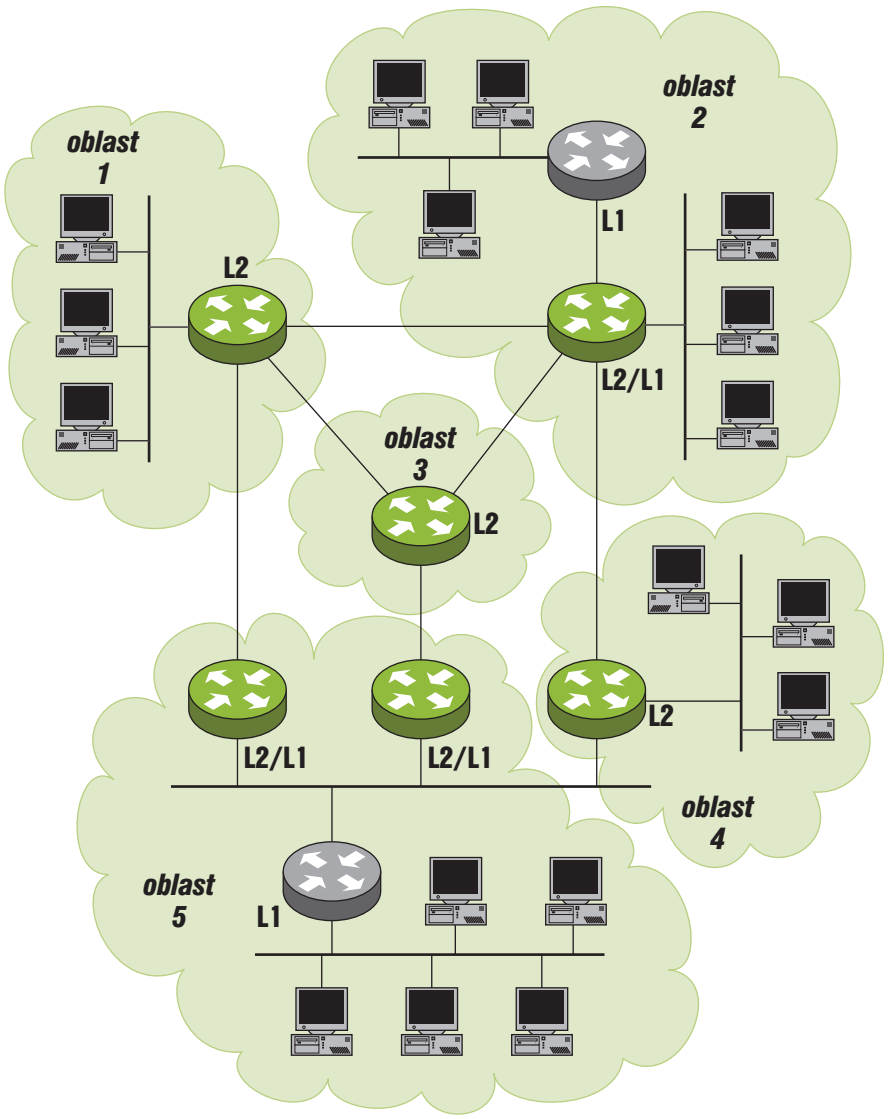
Na rozdíl od OSPF nemá IS-IS žádnou páteřní oblast. Zde je páteř tvořena souvislou topologií L2 směrovačů, která může oblastmi procházet celkem libovolně. Směrování mezi oblastmi zde tedy vypadá tak, že po odeslání je datagram doručen L1 směrovači odesilatelovy oblasti do vhodného L2 směrovače, dále putuje L2 infrastrukturou do cílové oblasti, kde je místními L1 směrovači doručen adresátovi. Zatímco hierarchie v OSPF je stavěna hvězdicově, kdy se koncové oblasti nabalují na oblast páteřní, IS-IS umožňuje téměř libovolné kompozice.

Základní definici IS-IS najdete v RFC 1142: *OSI IS-IS Intra-domain Routing Protocol*, které je převzatým standardem ISO 10589. Jeho adaptaci pro použití v Internetu obsahuje RFC 1195: *Use of OSI IS-IS for routing in TCP/IP and dual environments*. Oba dokumenty pocházejí z roku 1990, v žádném případě se tedy nejedná o horkou novinku.

IPv6

S nástupem IPv6 začalo IS-IS těžit ze své švýcarské nožíkovitosti. Zatímco OSPF vzniklo adaptací jeho myšlenek do světa IPv4 a pro nový protokol se muselo upravovat, IS-IS bylo od počátku koncipováno jako obecné pro libovolný síťový protokol a nezávislé na jeho adresách. Díky tomu bylo jeho použití pro IPv6 jednodušší a rychleji implementovatelné.

Příjemnou vlastností je i to, že v sítích podporujících oba protokoly spravuje informace o nich pod jednou střechou. Naproti v OSPF jsou IPv4 a IPv6



Obrázek 7.8: Oblasti podle protokolu IS-IS

dva zcela oddělené světy. V posledních letech proto některé sítě přecházejí z jiných protokolů na IS-IS. Jedním z významných příkladů je evropská akademická páteř GÉANT2.

7.6 BGP4+

Bez velkého přehánění lze prohlásit, že *Border Gateway Protocol (BGP)* drží pohromadě celý současný Internet. Patří mezi externí směrovací protokoly, jejichž prostřednictvím se vyměňují směrovací informace mezi různými autonomními systémy (nicméně jej lze použít i uvnitř jednoho AS). Čtvrtá verze je v současné době standardem a kdokoli chce mít svůj AS a komunikovat s okolím, musí tak činit prostřednictvím BGP4.

Jeho původní definice z RFC 1771 byla v roce 2006 revidována v RFC 4271: *A Border Gateway Protocol 4 (BGP-4)*. RFC 1772: *Application of the Border Gateway Protocol in the Internet* pak popisuje jeho nasazení v Internetu. Klasická verze BGP4 se zabývala výlučně směrováním IPv4. Později byla rozšířena prostřednictvím RFC 4760: *Multiprotocol Extensions for BGP-4* tak, že umožňuje směrování prakticky libovolného protokolu síťové vrstvy – včetně IPv6. Tato verze bývá označována jako *BGP4+*.

Výchozím bodem celého procesu je, že směrovač shrne do jedné „cesty" všechny prefixy ze svého autonomního systému a ohlásí je svým sousedům v jiných AS. Ti je předají dál a tak se postupně šíří informace, co je kudy dosažitelné. Z nich si pak jednotlivé směrovače musí spočítat, která cesta je pro ně nejlepší.

Ve srovnání s předchozí dvojicí je BGP velmi konzervativní. Nesnaží se aktivně vyhledávat sousední směrovače, ale staví na statické konfiguraci. Správce mu zadá, kdo jsou jeho sousedé.

S každým ze sousedů si směrovač udržuje trvale navázané TCP spojení. Na začátku si jeho prostřednictvím vymění kompletní směrovací informace a později zasílají jejich aktualizace. Dojde-li k ukončení spojení, směrovače na obou jeho koncích považují příslušného souseda za nedosažitelného a odstraní si z tabulek všechny cesty, které od něj pocházely.

Z pohledu BGP se směrovací informace ukládají do tak zvaných bází směrovacích informací (*Routing Information Base, RIB*). Jsou tři:

Vstupní báze obsahuje informace, které směrovači zaslal některý z jeho sousedů. Tyto informace se posuzují a na jejich základě je modifikována lokální báze.

Lokální báze představuje zdejší směrovací tabulku. Toto jsou pravidla, podle kterých směrovač posílá jednotlivé datagramy.

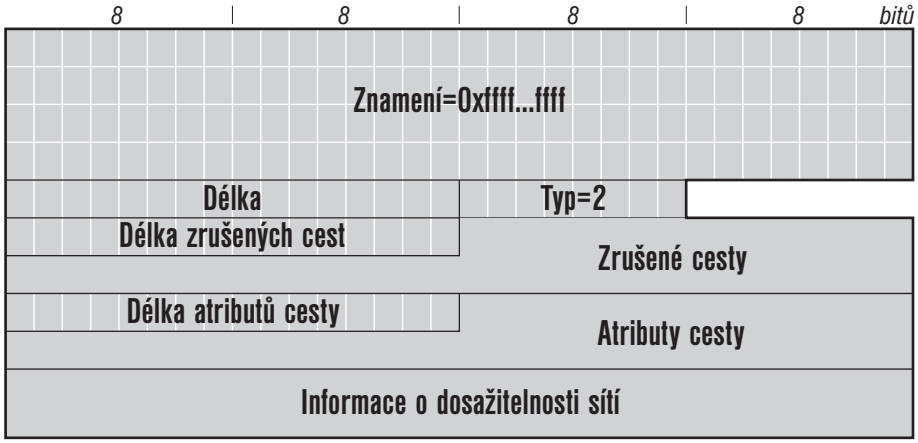
Výstupní báze pak zahrnuje informace, které se směrovač rozhodl ohlásit svým sousedům.

aktualizace BGP definuje několik typů zpráv. Mají shodný začátek, který obsahuje *Znamení*, *Délku* a *Typ* zprávy. Počáteční *Znamení (Marker)* podle nové definice obsahuje samé jedničky. Dále pak následuje celková *Délka (Length)* BGP zprávy a její *Typ (Type)*. Sortiment a účel jednotlivých typů zpráv uvádí tabulka 7.3. Formát zbytku zprávy se liší v závislosti na typu.

typ	*určení*
1=OPEN	zahájení vzájemné komunikace
2=UPDATE	zprávy o změnách ve směrování
3=NOTIFICATION	chybové hlášení
4=KEEPALIVE	udržování komunikace, když není o čem mluvit

Tabulka 7.3: Typy zpráv BGP

Nejdůležitějším typem je UPDATE, jehož prostřednictvím se ohlašují změny ve směrovacích tabulkách. Řada jeho položek má proměnlivou délku, jak vidíte z obrázku 7.9.



Obrázek 7.9: BGP – formát zprávy UPDATE

Za společnými položkami následují informace o *Zrušených cestách (Withdrawn routes)*. Například když směrovač ztratí kontakt s některým ze svých BGP sousedů, považuje všechny cesty od něj za nefunkční a tudíž je ostatním ohlásí jako zrušené. Lze zrušit několik cest v jediné zprávě. Jsou identifikovány svými prefixy.

Zbytek zprávy je věnován ohlašované cestě (jedna zpráva může ohlašovat nanejvýš jednu cestu). Nejprve jsou uvedeny *Atributy cesty (Path attributes)*, které poskytují doplňkové informace. Jejich přehled najdete v tabulce 7.4. Dále pak následuje seznam cílů (prefixů), které do dané cesty

spadají[7]. Tato informace se skrývá pod lehce tajuplným názvem *Informace o dosažitelnosti sítí (Network layer reachability information)*.

atribut	*význam*
ORIGIN	odkud pochází informace (IGP, EGP, odjinud)
AS_PATH	seznam AS, kterými prošla směrovací informace o této cestě
NEXT_HOP	adresa směrovače, který je první na cestě k danému cíli
MULTI_EXIT_DISC	používá se pro rozhodování mezi několika cestami do téhož sousedního AS; pokud jsou ostatní kritéria shodná, použije se cesta s nejmenší hodnotou tohoto atributu
LOCAL_PREF	posílá jen směrovačům ve svém vlastním AS; informuje o preferenci, kterou dával odesilatel této cestě
ATOMIC_AGGREGATE	informuje, že spojil několik cest do jedné obecnější (s kratším prefixem)
AGGREGATOR	adresa a číslo AS směrovače, který spojil cesty

Tabulka 7.4: Atributy cesty v BGP

zpracování zprávy

Když směrovači dorazí od některého ze sousedů zpráva s aktualizací, upraví podle jejího obsahu příslušnou vstupní bázi. Následně pak spustí rozhodovací proces, jehož cílem je

- vybrat cesty pro lokální bázi (a tedy lokální směrovací tabulku)
- vybrat cesty, které ohlásí sousedům ve stejném AS
- vybrat cesty, které ohlásí sousedům v jiných AS
- agregovat (spojit) cesty a redukovat objem směrovacích informací

Rozhodování probíhá ve třech fázích. V první spočítá preferenci ke všem cestám z každé vstupní báze. Jedná se o matematickou funkci, která ke každé cestě přiřadí míru její výhodnosti. Na rozdíl od interních směrovacích protokolů není jednoduché takovou funkci stanovit, protože různé AS mohou používat různé metriky. Je proto záležitostí lokální konfigurace a může brát v potaz celou řadu faktorů (počet AS po cestě, zvýhodňovat či znevýhodňovat určité konkrétní AS, upřednostňovat stabilní cesty a podobně).

Ve druhé fázi podle vypočtených preferencí určí nejvýhodnější cestu pro každý ze známých cílů a zavede ji do lokální báze. A konečně třetí fáze vyjde

[7] Typicky seznam prefixů z cílového AS.

ze změněné lokální báze a naplní výstupní báze pro jednotlivé sousedy. Důležité je, že posílá jen ty cesty, které sám používá (jsou z jeho pohledu nejvýhodnější). Navíc zde může uplatnit směrovací politiku a například ohlášení určitých cest potlačit. Dojde-li v nich ke změnám, je třeba informovat ostatní zasláním odpovídajících aktualizací.

8 Skupinové radovánky čili multicast

Práce s datagramy směřujícími na skupinovou adresu se do značné míry liší od zpracování běžných individuálních paketů. Proto jsem se rozhodl vyčlenit pro ně samostatnou kapitolu.

Hlavní problém pochopitelně představuje směrování. V případě skupinového vysílání se jedná o vybudování distribučního stromu, kterým budou data šířena co nejefektivněji ke všem příjemcům.

8.1 Doprava po Ethernetu a Wi-Fi

Nejprve se ale podívejme, jak se skupinové datagramy dopravují po linkové vrstvě. Nejzajímavější je kombinace s Ethernetem, který má své vlastní mechanismy pro skupinové vysílání a IPv6 je využívá. Stejně je na tom i Wi-Fi, které sice používá odlišný způsob fyzického přenosu dat, ale jeho struktura adres a podpora skupinového vysílání se shodují s Ethernetem. Pro zjednodušení budu nadále psát pouze o Ethernetu, pro Wi-Fi jednoduše platí totéž. Ostatní technologie jsou buď celkem jasné, protože používají jen dvoubodové spoje, nebo natolik málo rozšířené, že nemá valný smysl se o nich zmiňovat.

Ethernet podporuje skupinovou komunikaci. Jeho skupinové adresy jsou charakteristické tím, že mají v prvním bitu jedničku, zatímco u individuálních tu najdete nulu.

Skupinové adresy z IPv6 se do Ethernetu mapují celkem přímočaře: vezmou se poslední čtyři bajty z cílové (skupinové) IPv6 adresy a před ně se přidá předpona 3333 (hexadecimálně). Ze skupinové IPv6 adresy pro vyzývaný uzel ff02::1:ff32:5ed1 tak vznikne ethernetová adresa

33:33:ff:32:5e:d1

Samozřejmě se může stát, že několik skupinových IPv6 adres splyne do jedné ethernetové – například adresy, které se liší jen dosahem. Teoreticky jich je velmi mnoho, v praxi bude k těmto případům docházet jen zřídka, protože jednotlivé adresy se zpravidla liší především v závěrečném identifikátoru skupiny. Tomuto splývání se nedá zabránit. Na úrovni Ethernetu stroj prostě přijme všechny rámce přicházející na danou adresu a je úkolem IPv6 vrstvy rozlišit, které z nich skutečně přijme a které sem dorazily jen shodou okolností.

Kdykoli začne počítač přijímat některou skupinu, musí nastavit své ethernetové rozhraní tak, aby přijímalo data přicházející na odpovídající ethernetovou adresu. Došlé datagramy dostává IPv6 vrstva a ta standardním způsobem vyřadí všechny, které dorazily na nesprávnou IPv6 adresu.

Definice přenosu skupinových IPv6 datagramů po Ethernetu je součástí RFC 2464: *Transmission of IPv6 Packets over Ethernet Networks*. Pro IEEE 802.11 alias Wi-Fi žádné RFC neexistuje, vzhledem k podobnosti logických prvků s Ethernetem však ani není potřeba. Specifikace přenosu IPv6 po několika dalších, méně častých linkových technologiích shrnuje příloha B na straně 391.

8.2 Multicast Listener Discovery (MLD)

Jednou ze základních informací, které potřebuje znát každý směrovač zapojený do skupinového života, je seznam skupin, jež má vysílat do daného rozhraní. V podstatě se jedná o ty skupiny, které zde mají alespoň jednoho příjemce.

Ke zjišťování příjemců sloužil ve světě IPv4 Internet Group Management Protocol (IGMP). IPv6 pro něj změnilo název na MLD, základní principy však zůstaly podobné. Ostatně samotní jeho autoři prohlašují MLD za „překlad IGMP pro IPv6“.

MLD se momentálně vyskytuje ve dvou verzích. MLDv1 (RFC 2710) je protipólem IGMPv2. Novější MLDv2 pak odpovídá IGMPv3. Jeho definici obsahuje v RFC 3810: *Multicast Listener Discovery Version 2 (MLDv2) for IPv6*. Hlavním rozdílem je, že novější verze protokolu umožňuje filtrovat zdroje. Zatímco v MLDv1 lze pouze ohlásit zájem o příjem skupiny G, v MLDv2 může počítač požadovat data pro skupinu G vysílaná konkrétním strojem nebo naopak příjem skupiny z určitých adres odmítnout. Obě verze spolu mohou spolupracovat, ovšem formáty jejich zpráv i používané postupy se poněkud liší.

formát zprávy

MLD je podprotokolem ICMPv6 a jeho zprávy jsou tedy jen jedním typem zpráv ICMPv6. Jejich tvar vychází ze základního formátu ICMP (viz obrázek 4.1 na straně 97), který však konkretizuje. Posílají se vždy z lokální linkové adresy příslušného rozhraní, jejich maximální počet skoků je nastaven na jedničku a musí nést rozšiřující hlavičku *Upozornění směrovače*, aby si jich všímaly i směrovače, které samy neposlouchají cílovou skupinu příslušného datagramu.

zpráva	*MLDv1*	*MLDv2*
dotaz (query)	130	130
hlášení (report)	131	143
ukončení (done)	132	–

Tabulka 8.1: Typy MLD zpráv pro obě verze protokolu

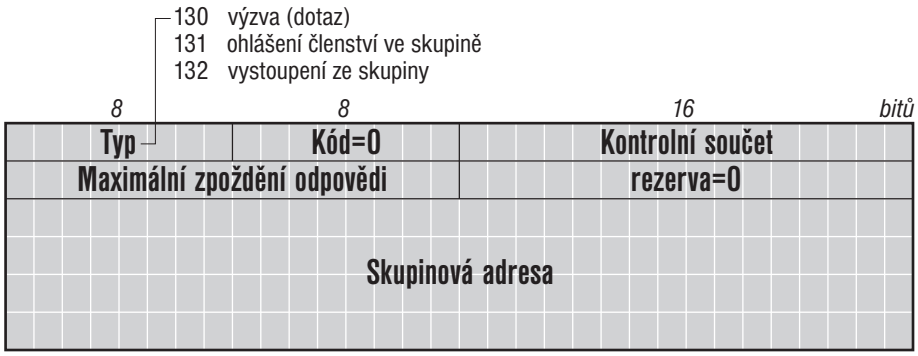
V prvním poli zprávy je v souladu s pravidly ICMPv6 vždy uveden typ. Typy i formáty zpráv se poněkud liší v závislosti na verzi protokolu. V tabulce 8.1

najdete jejich přehled. Vzhledem k nezanedbatelným odlišnostem popíši obě verze protokolu samostatně.

8.2.1 MLD verze 1

Jak již bylo řečeno, MLDv1 se nestará o odesilatele. Zajímá jej pouze informace, zda pro danou skupinu existuje někdo, kdo ji přijímá. Směrovač si pro každé rozhraní udržuje seznam skupinových adres, pro které se zde nachází alespoň jeden posluchač. Z těchto informací se pak vychází při stanovení distribučních stromů pro jednotlivé skupiny a směrování skupinově adresovaných datagramů. Na rozhraní, které podporuje skupinový provoz, je směrovač povinen přijímat data adresovaná kterékoli skupině – včetně těch, které sám nezná.

Absence informací o odesilatelích se odráží i v jednodušším formátu zpráv protokolu. Ten je pro všechny tři jejich typy společný a vidíte jej na obrázku 8.1.

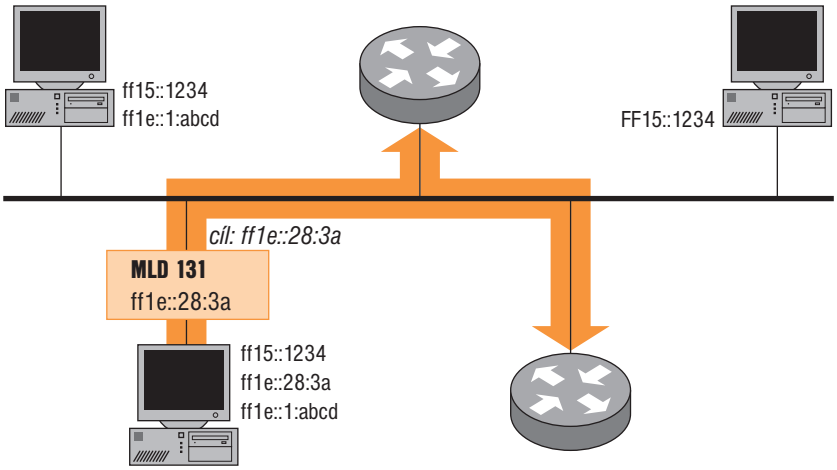


Obrázek 8.1: Formát zprávy protokolu MLD verze 1

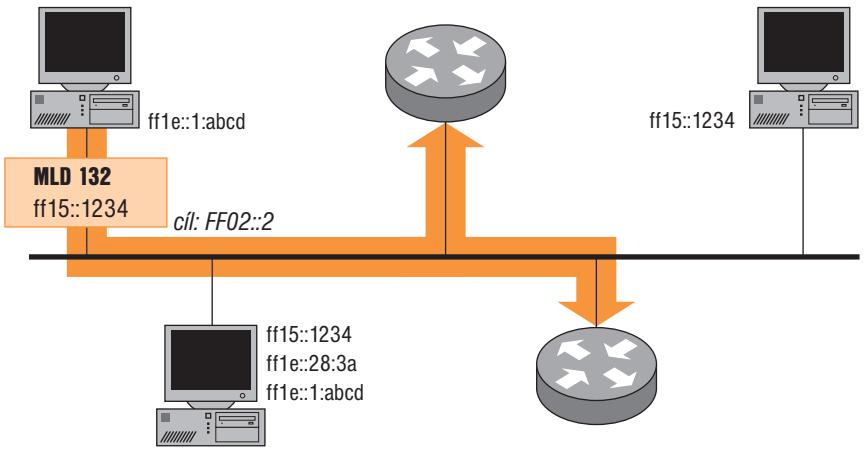
Pro *Typ (Type)* přicházejí v úvahu tři hodnoty: 130 slouží k výzvě (dotazu) směrovače, prostřednictvím 131 ohlašují stanice své členství ve skupinách a číslem 132 sdělují ukončení příjmu dané skupiny.

vstup do skupiny Když počítač vstoupí do nové skupiny, pošle na její adresu MLD zprávu typu 131 ohlašující členství v této skupině. Směrovače si u příslušného rozhraní přidají skupinovou adresu do seznamu vysílaných (pokud ji tam ještě nemají). Doporučuje se poslat toto ohlášení opakovaně, aby se eliminovala případná ztráta datagramu.

vystoupení ze skupiny Pokud počítač naopak ukončuje svou účast ve skupině, pošle MLD zprávu typu 132, tedy ukončení členství. Tato zpráva se neposílá na adresu skupiny, ale na ff02::2, což je skupinová adresa pro všechny směrovače na dané lince.



Obrázek 8.2: Počítač vstupuje do skupiny



Obrázek 8.3: Počítač opouští skupinu

Směrovače, které zprávu obdržely, musí posoudit, zda skupina má ještě nějakého posluchače. Pokud své členství v ní naposledy ohlašoval jiný počítač než ten, který se právě odhlásil, je rozhodování snadné – zjevně existuje i jiný posluchač a proto si skupinu ponechá v seznamu.

V opačném případě (příjem skupiny naposledy ohlašoval právě ten počítač, který se odhlašuje) si směrovač musí udělat jasno. Pošle proto na adresu skupiny dotaz – MLD zprávu typu 130, v níž je jako *Skupinová adresa (Multicast address)* uvedena adresa zjišťované skupiny. Aby se vzápětí nesesypala hromada ohlášení od všech přijímajících, je reakce na dotaz definována následovně:

Každý počítač, který dostane dotaz[1], si nastaví časovač na náhodný interval. Jeho horní hranici udává položka *Maximální zpoždění odpovědi (Maximum response delay)* v dotazu. Po vypršení intervalu pošle na adresu skupiny ohlášení svého členství. Jakmile dostane ohlášení od jiného člena, časovač zruší a sám své ohlášení už posílat nebude. To znamená, že ze skupiny odpoví vždy jen jeden náhodně vybraný stroj.

typ	*zpráva*	*adresa*
130	obecný dotaz	ff02::1
130	konkrétní dotaz	adresa skupiny
131	ohlášení členství	adresa skupiny
132	ukončení členství	ff02::2

Tabulka 8.2: MLD zprávy a jejich cílové adresy

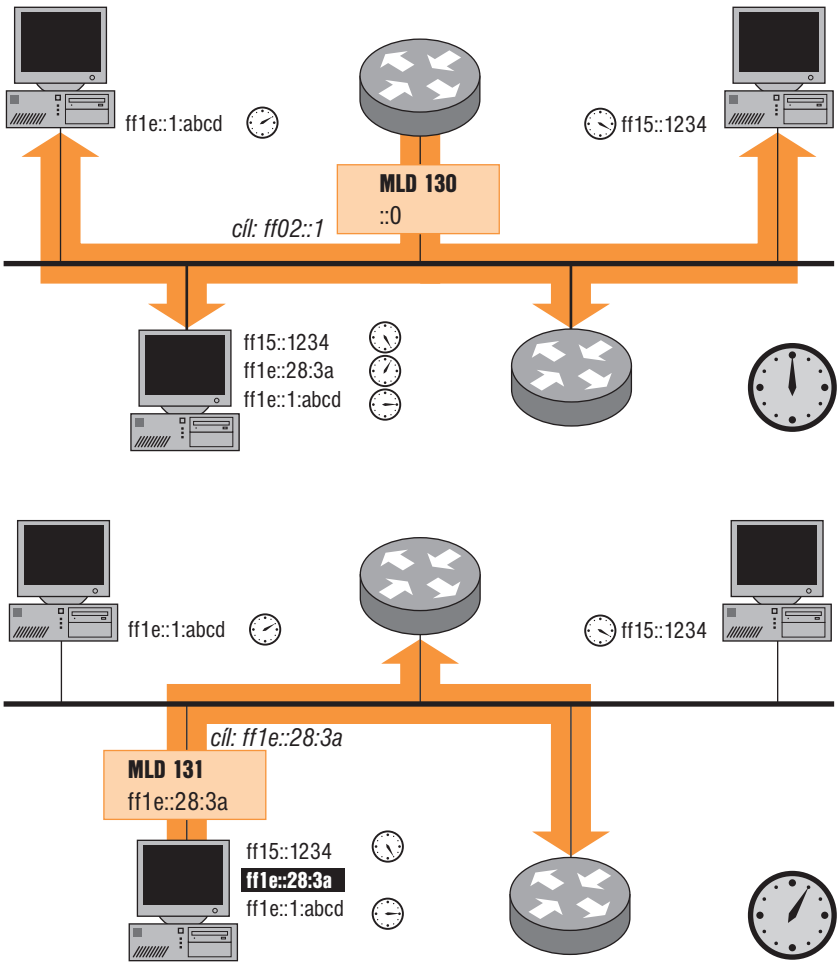
obecný dotaz (objevování skupin)

Ovšem život není ideální a je třeba počítat s tím, že občas některý počítač přestane poslouchat skupinu, aniž by to korektně ohlásil. Proto směrovače opakovaně posílají do připojených sítí obecné dotazy. Jedná se o MLD zprávu typu 130, kde *Skupinová adresa* je nulová. Směrovač tím říká „chtěl bych vědět, které všechny skupiny zde mají posluchače“. Dotaz zašle na adresu ff02::1 (všechny uzly na lince).

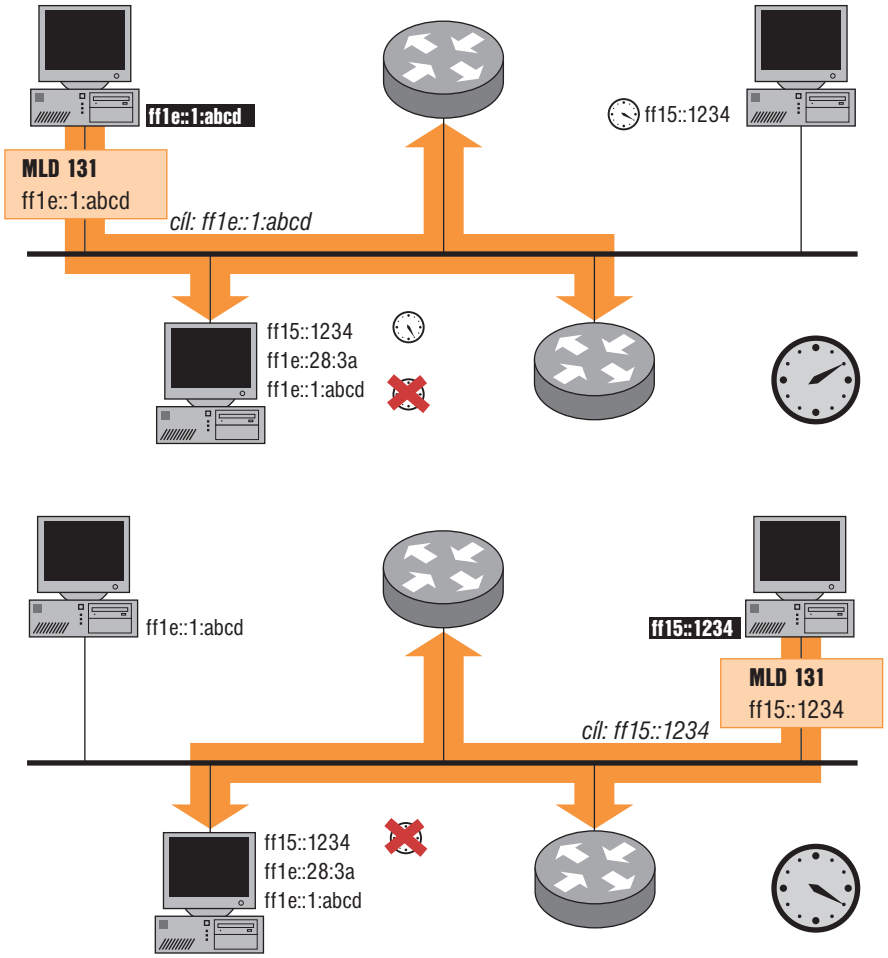
Počítač se chová stejně, jako kdyby naráz dostal konkrétní výzvu pro všechny své skupiny. Pro každou skupinu, jejímž je členem, (kromě ff02::1 a skupin s dosahem 0 nebo 1), si nastaví samostatný časovač. Každý dostane jinou náhodnou hodnotu. Když vyprší a nedorazilo ještě ohlášení od jiného člena, pošle své ohlášení.

Aby se počet dotazů udržoval v rozumných mezích, posílá je vždy jen jeden ze směrovačů připojených k dané lince – ten, který má nejmenší IP adresu. Implementace jeho výběru je velmi jednoduchá. Každý směrovač poslouchá a pokud příliš dlouho nedorazí obecný dotaz od někoho s menší adresou než je jeho vlastní, pošle jej sám.

[1] To znamená, že je členem dotyčné skupiny.



Obrázek 8.4: Směrovač zjišťuje aktivní skupiny



Obrázek 8.5: Směrovač zjišťuje aktivní skupiny (dokončení)

Dotazy tedy posílá jen jeden směrovač, ale odpovědi dostávají a zpracovávají všechny. Díky tomu si všechny směrovače připojené k dané lince udržují konzistentní informace o zdejších skupinách.

8.2.2 MLD verze 2

Druhá verze protokolu přináší možnost omezit příjem skupinových dat v závislosti na jejich zdroji. Přesněji řečeno dává na výběr dvě varianty, jak skupinově adresované datagramy filtrovat. Buď je možné požadovat doručování skupinových dat jen od vybraných stanic. Takový režim je označován jako INCLUDE(L), kde L představuje seznam přijímaných adres. Druhou možností je příjem dat od všech kromě několika uvedených zdrojů, který se označuje jako EXCLUDE(L).

Toto filtrování je důsledně promítnuto do celého systému. Začíná už v aplikačním rozhraní, kde si program může poručit, od koho chce či nechce dostávat skupinová data. Síťová vrstva počítače si musí vést evidenci o požadavcích aplikací pro jednotlivé sokety a podle nich pak vytvářet stav příjmu skupinových dat na každém ze svých síťových rozhraní. Podobně směrovač musí kombinovat zprávy od různých klientů a dát z nich dohromady přehled o tom, jaké skupinové datagramy předávat do sítí, k nimž je připojen.

INCLUDE(X) + INCLUDE(Y)	→	INCLUDE(X ∪ Y)
EXCLUDE(X) + INCLUDE(Y)	→	EXCLUDE(X – Y)
EXCLUDE(X) + EXCLUDE(Y)	→	EXCLUDE(X ∩ Y)
příklady	→	
INCLUDE(a, b, c) + INCLUDE(c, d, e)	→	INCLUDE(a, b, c, d, e)
EXCLUDE(a, b, c) + INCLUDE(c, d, e)	→	EXCLUDE(a, b)
EXCLUDE(a, b, c) + EXCLUDE(c, d, e)	→	EXCLUDE(c)

Tabulka 8.3: Pravidla pro kombinování filtrů

Je proto potřeba stanovit pravidla, jak kombinovat jednotlivé požadavky. Ty přitom mohou obsahovat nejen různé adresy zdrojů, ale mohou používat i různé režimy filtrování. Základní pravidla shrnuje tabulka 8.3. Jestliže se spojují dva filtry typu INCLUDE, vznikne filtr stejného typu zahrnující sjednocení adres původních dvou – je třeba přijímat data ze zdrojů, o něž projevil zájem alespoň jeden z posluchačů. Při kombinaci EXCLUDE s INCLUDE dostane přednost první z nich, protože požaduje data od všech kromě několika uvedených. Z nich budou vyloučeny ty adresy, o které žádá filtr typu INCLUDE. A konečně spojením dvou filtrů typu EXCLUDE vznikne filtr stejného typu vylučující jen ty adresy, které nechce ani jeden z původních filtrů.

Toto spojování probíhá v několika místech. Provádí je každý příjemce, když dává dohromady požadavky jednotlivých aplikací, které pak souhrnně ohlašuje jako požadavky pro své síťové rozhraní. Podobně pak postupuje i směrovač, který slučuje požadavky jednotlivých klientů.

Za zmínku stojí dva speciální případy, které odpovídají situacím podle MLDv1 a jsou v praxi asi nejčastější. Prvním je příjem skupinové adresy ze všech zdrojů, tedy bez jakéhokoli filtrování. Tato situace je v MLDv2 vyjádřena jako EXCLUDE( ), tedy EXCLUDE s prázdným seznamem odmítaných adres. Jakmile se objeví při kombinování požadavků, je jasné, že i výsledkem bude EXCLUDE( ). Druhým případem je vystoupení ze skupiny, tedy ukončení jejího příjmu. Z pohledu MLDv2 příslušný filtr přejde do režimu INCLUDE( ).

změna příjmu

MLDv1 rozlišuje na straně příjemce skupinových dat dvě základní situace: vstup do skupiny a její opuštění. Naproti tomu MLDv2 má jen jednu událost tohoto typu – změnu v příjmu skupin. Zahájení či ukončení příjmu skupiny představují její speciální případy, kromě nich může změna zahrnovat i rozšíření nebo zúžení počtu přijímaných zdrojů nebo změnu režimu filtrování příslušné skupiny. Pokud dojde k jakékoli z těchto událostí, příjemce pošle MLD zprávu typu *Hlášení (Report)* na adresu ff02::16 pro všechny MDLv2 směrovače na lince[2].

Formát hlášení MLDv2 vidíte na obrázku 8.6. Srovnejte si jej s formátem zprávy první verze z obrázku 8.1 na straně 165. Verze 1 používala velmi jednoduchou logiku. Zpráva se týkala vždy jedné skupiny, jejíž adresa v ní byla uvedena. Typ zprávy rozhodoval o tom, zda odesilatel zahájil či ukončil příjem skupiny, nebo se na ni ptá.

Do jednoho hlášení lze v MLDv2 vložit celou řadu informací. Ty jsou obsaženy v tak zvaných záznamech a úvodní hlavička hlášení obsahuje především údaj o tom, kolik záznamů se nachází uvnitř. Záznamy mají různý význam, který je určen položkou *Typ záznamu (Record type)*. Jejich souhrn najdete v tabulce 8.4 a hned se k nim dostaneme podrobněji. Záznam nese také skupinovou adresu, jíž se týká, a případně seznam odesilatelů (zdrojů) skupinových dat.

Pokud se u stanice něco změní na příjmu skupiny (či několika skupin), okamžitě odešle MLD hlášení s popisem změn. Došlo-li ke změně režimu filtrování skupiny, pošle pro ni záznam typu 3 (při změně z EXCLUDE na INCLUDE) nebo 4 (naopak). Vloží do něj i adresy zdrojů, které nový filtr obsahuje.

Pokud režim zůstal zachován, ale došlo ke změně filtrovaných adres, posílá stanice záznamy typu 5 (chce přijímat nové adresy) a 6 (končí příjem dříve

[2] V cílové adrese je změna, MLDv1 posílá zprávy tohoto typu na adresu skupiny, jíž se zpráva týká. Ovšem hlášení v MLDv2 se může týkat více skupin zároveň.

Obrázek 8.6: Hlášení (report) protokolu MLD verze 2

typ	*význam*
1	MODE_IS_INCLUDE
2	MODE_IS_EXCLUDE
3	CHANGE_TO_INCLUDE
4	CHANGE_TO_EXCLUDE
5	ALLOW_NEW_SOURCES
6	BLOCK_OLD_SOURCES

Tabulka 8.4: Typy záznamů v MLDv2 hlášení

přijímaných). Může poslat i oba záznamy současně, pokud některé zdroje přidal a jiné odebral. Záznam typu ALLOW_NEW_SOURCES znamená, že odesilatel chce přijímat data od daného zdroje. Pokud například má filtr typu EXCLUDE a jeho obsah se změní z EXCLUDE(X,Y) na EXCLUDE(X), pošle záznam ALLOW_NEW_SOURCES(Y), protože zdroj Y byl dříve blokován a nyní již není.

Tabulka 8.5 shrnuje, jaké záznamy se posílají pro ohlášení změny filtru ze stavu „před“ do stavu „po“. A a B v ní reprezentují seznamy adres. Zbytečné záznamy se samozřejmě neposílají – dojde-li k rozšíření filtru z INCLUDE(X) na INCLUDE(X,Y), pošle se pouze ALLOW_NEW_SOURCES(Y). Pokud pro danou skupinu žádný stav neexistuje, chápe se, jako by měla stav INCLUDE( ).

před	*po*	*odeslané záznamy*
INCLUDE(A)	INCLUDE(B)	ALLOW(B–A), BLOCK(A–B)
INCLUDE(A)	EXCLUDE(B)	TO_EXCLUDE(B)
EXCLUDE(A)	EXCLUDE(B)	ALLOW(A–B), BLOCK(B–A)
EXCLUDE(A)	INCLUDE(B)	TO_INCLUDE(B)

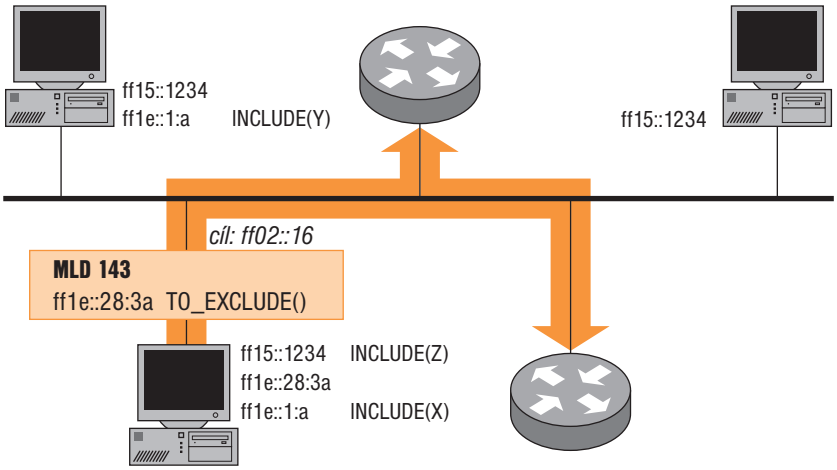
Tabulka 8.5: Záznamy posílané při změnách filtru pro skupinu

Podívejme se na dvě nejčastější situace. Jestliže stanice vstoupí do skupiny X, ve které předtím nebyla, a přijímá pro ni data ze všech zdrojů, oznámí tuto skutečnost podle druhého řádku tabulky 8.5 MLD hlášením obsahujícím pro skupinu X záznam CHANGE_TO_EXCLUDE( ). Příklad vidíte na obrázku 8.7, kdy dolní stanice vstupuje do skupiny ff1e::28:3a.

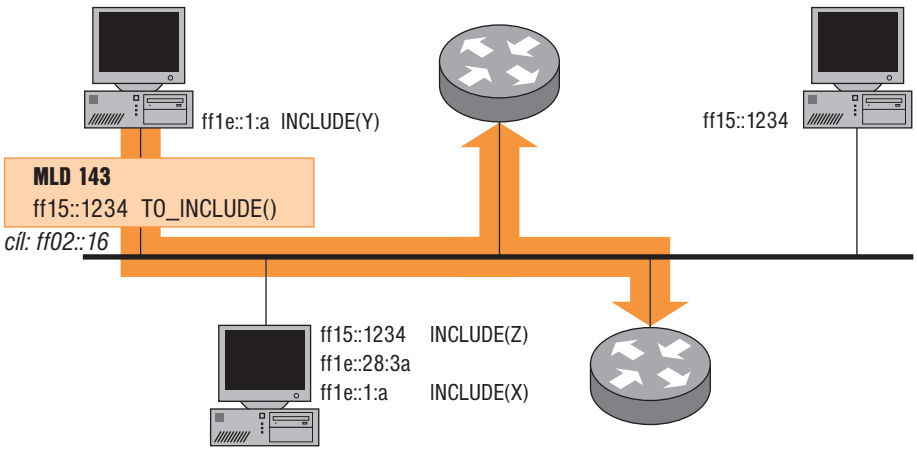
Pokud stanice opouští skupinu, přechází z pohledu MLD do režimu INCLUDE( ). Jestliže ji předtím přijímala bez omezení, tedy byla ve stavu EXCLUDE( ), pošle podle posledního řádku tabulky 8.5 hlášení se záznamem CHANGE_TO_INCLUDE( ). Na obrázku 8.8 levá stanice vystupuje ze skupiny ff15::1234.

zjišťování skupin

Hlášení o změnách či jiné události mohou vést k situaci, kdy si směrovač není jist, jaký je vlastně aktuální stav příjmu skupinových dat na lince.



Obrázek 8.7: MLDv2 – počítač vstupuje do skupiny



Obrázek 8.8: MLDv2 – počítač opouští skupinu

K jeho zjištění slouží MLD dotaz (query). Velmi se podobá dotazu z MLDv1, pouze přibyla možnost přidat do něj adresy zdrojů.

V MLDv2 může směrovač podle potřeby položit jeden ze tří typů dotazů: Obecný dotaz s nulovou adresou skupiny znamená „Sháním všechny posluchače skupinových dat“ Pokud uvede adresu skupiny, zajímají jej jen příjemci této skupiny. Jestliže navíc přidá i adresy zdrojů, ptá se „Poslouchá někdo tuto skupinu z uvedených zdrojů?“ Obecný dotaz se posílá všem (na adresu ff02::1), konkrétní na adresu skupiny.

Obecný dotaz a reakce na něj představuje obrázek 8.9 a jeho pokračování. Stejně jako v MLDv1 počítače své odpovědi náhodně zpozdí, existují tu však dvě významné změny: Počítač do jednoho hlášení zařadí všechny své skupiny[3] a posílá je MLDv2 směrovačům, nikoli na adresu skupiny. Koncové stanice svá hlášení neslyší a nereagují na ně. Jestliže v MLDv1 je přeneseno jedno hlášení pro každou skupinu, kterou někdo na lince přijímá, v MLDv2 se posílá jedno hlášení za každý počítač zapojený do skupinového příjmu[4].

Pro odpověď na dotaz slouží speciální typy záznamů 1 a 2 (viz tabulka 8.4 na straně 173), jimiž počítač ohlásí „pro skupinu X mám filtr daného typu a obsahuje následující adresy zdrojů“. Směrovače přicházející odpovědi shromažďují a podle nich sestaví filtr pro příslušné rozhraní. Výsledek vidíte ve spodní části obrázku 8.10.

Lightweight MLDv2

Praxe ukázala, že příjem skupiny od kohokoli s výjimkou vyjmenovaných adres je koncept ryze teoretický a v reálném provozu se neobjevuje. Režim EXCLUDE se v praktickém provozu vyskytuje jen s prázdným seznamem odesilatelů, pokud je skupina přijímána odkudkoli.

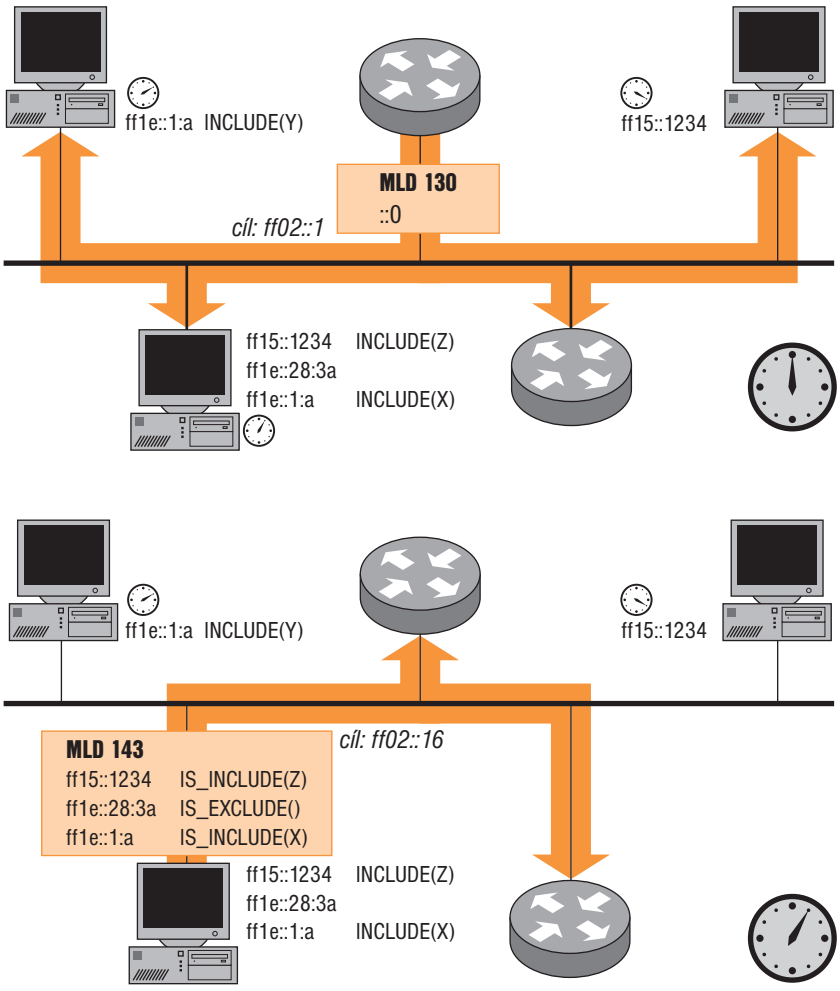
Tuto skutečnost kodifikuje RFC 5970: *Lightweight Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Version 2 (MLDv2) Protocols* zavedením odlehčené verze *Lightweight MLDv2 (LW-MLDv2)*. Jedná se o zpětně kompatibilní podmnožinu plného MLDv2, která výrazně omezuje režim EXCLUDE a připouští jej pouze s prázdným seznamem zdrojů. Chování protokolu se nemění, ale do lehké verze jsou zařazeny jen jeho části. Vzhledem k tomu, že odstraněny byly části reálně nepoužívané, neměla by koexistence prvků podporujících MLDv2 a LW-MLDv2 působit problémy.

8.3 Směrování skupinových datagramů

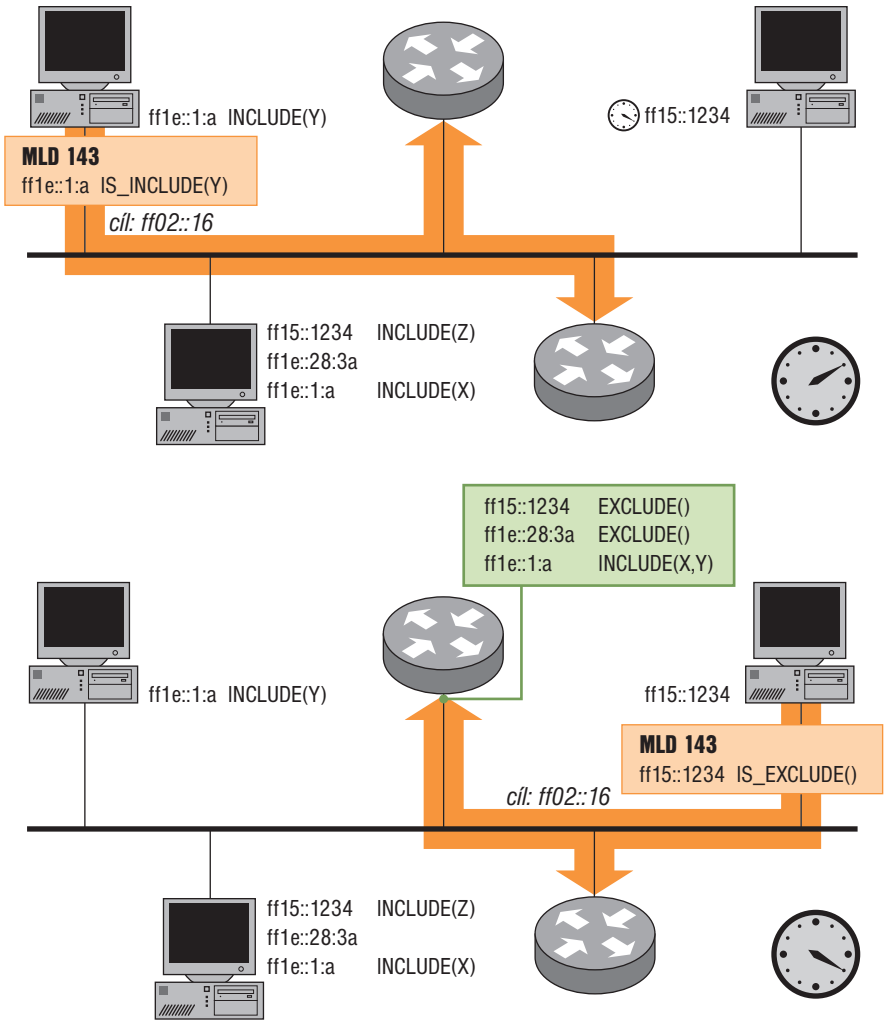
Protokol MLD umožňuje klientům vstoupit do skupiny a přihlásit se k odběru datagramů, směřujících na její adresu. Zbývá už jen to nejtěžší: koordinovat tyto informace, zjistit rozložení příjemců jednotlivých skupin v síti

[3] V MLDv1 obsahuje hlášení jen jednu skupinu, každá skupina má svůj vlastní zpožďovací časovač.

[4] Za předpokladu, že neposílá příliš mnoho záznamů a vejdou se vždy do jednoho hlášení.



Obrázek 8.9: MLDv2 – směrovač zjišťuje aktivní skupiny



Obrázek 8.10: MLDv2 – směrovač zjišťuje aktivní skupiny (dokončení)

a vytvořit co nejefektivnější cesty, jak jim doručovat data. Tuto činnost mají na starosti skupinové směrovací protokoly.

Prodělaly určitý vývoj, v němž poměrně významnou roli hrál *Distance Vector Multicast Routing Protocol (DVMRP)*, který ale není příliš efektivní a nehodí se pro rutinní nasazení. V současné době před ním dostává přednost skupina protokolů s názvem *Protocol Independent Multicast (PIM)*. Toto společné označení skrývá čtyři dost odlišné protokoly:

- **PIM – Dense Mode (PIM-DM)** je vhodný pro situace s vysokou hustotou příjemců, kdy je třeba datagramy distribuovat skoro do všech částí sítě. Jeho použitelnost je prakticky omezena na lokální sítě a v současné době je považován za překonaný.
- **PIM – Sparse Mode (PIM-SM)** naopak předpokládá, že příjemci jsou v síti roztroušeni jen zřídka. Vytváří pro ně distribuční stromy na základě žádostí o příjem skupinových dat. To jej činí vhodným pro rozlehlé sítě a široko daleko nejpoužívanějším protokolem současnosti.
- **Bidirectional PIM (BIDIR-PIM)** představuje variantu PIM-SM. Jeho distribuční stromy jsou ovšem obousměrné, zatímco PIM-SM používá skupinu jednosměrných stromů.
- **PIM Source Specific Multicast (PIM-SSM)** rozlišuje při doručování nejen skupinovou adresu, ale i adresu zdroje. Je určen především pro komunikaci, kdy vysílá jediný zdroj, jehož data přijímá řada klientů (něco jako internetová televize či rádio).

K některým svým činnostem skupinové směrování využívá i směrovací informace pro individuální (unicastové) pakety. Nestará se však o to, jak individuální směrovací tabulka vznikla a není vázáno na konkrétní směrovací protokol. Odtud pochází ono „Protocol Independent“ v názvu PIM.

Podíváme-li se na individuální směrování, najdeme Internet rozdělený na části (autonomní systémy), které provozují určitý interní směrovací protokol (OSPF, IS-IS, RIP a podobně) a vzájemně si vyměňují informace externím protokolem BGP. Při skupinovém směrování je situace obdobná. V Internetu najdeme oblasti, uvnitř nichž běží vnitřní směrovací protokol (zpravidla PIM-SM). Říká se jim PIM domény, mají svá shromaždiště (RP) a organizují si doručování skupinových dat po svém. Představují určitou analogii autonomních systémů.

Většina skupinového provozu probíhá uvnitř PIM domény, ovšem je třeba umožnit jí i komunikaci s okolním světem. V IPv4 si proto jednotlivé PIM domény vzájemně vyměňují informace o svých skupinových zdrojích. To umožňuje, aby vznikaly skupiny s příjemci a zdroji roztroušenými v několika PIM doménách. Pro vzájemnou výměnu informací o existujících skupinách a zdrojích slouží *Multicast Source Discovery Protocol (MSDP)*, který má úlohu podobnou BGP ve světě individuálního směrování.

IPv6 díky svým dlouhým adresám může k problému přistoupit jinak a zařadit adresu shromaždiště přímo do skupinové adresy. Žádná výměna informací o dění uvnitř PIM domén pak není potřeba, kdokoli v celém Internetu se přímo z adresy skupiny dozví, kde je její shromaždiště a kam má tedy poslat žádost o příjem dat. Práce na adaptaci protokolu MSDP pro IPv6 byly proto zastaveny a nahrazeny konceptem vložených adres. Jak taková skupinová adresa vypadá jsem popsal na straně 71.

8.3.1 PIM Sparse Mode (PIM-SM)

Začněme nejpoužívanějším a bohužel i nejsložitějším z celé skupiny. V různých situacích jsou nároky na co nejefektivnější distribuci skupinových dat odlišné, proto v sobě PIM-SM kombinuje několik odlišných mechanismů. Díky nim může nabídnout adekvátní prostředky pro jednotlivé případy, ale jeho celková složitost tím nepříjemně narůstá. Jeho definici najdete v RFC 4601: *Protocol Independent Multicast – Sparse Mode (PIM-SM): Protocol Specification*.

Základní myšlenkou PIM-SM je, že skupinově adresovaná data se doručují jen tam, kde si o ně někdo řekl. Používá k tomu speciální typ zpráv *Připojení (PIM Join)*, jejichž prostřednictvím ohlašuje zájem o odběr skupiny směrovač, kterému se ohlásil alespoň jeden její příjemce.

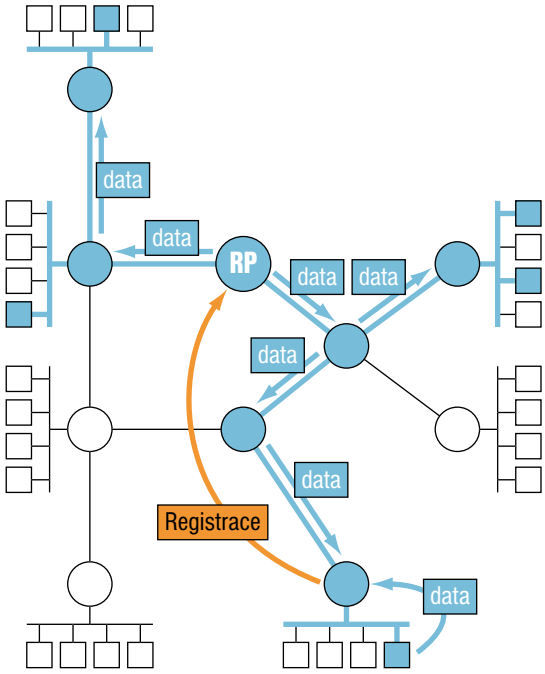
shromaždiště aneb rendezvous point (RP)

Jinou otázkou je, kam žádosti o příjem skupiny adresovat. PIM-SM pro tento účel zavádí speciální směrovač, označovaný jako *shromaždiště (rendezvous point, RP)*. Jak název napovídá, představuje místo v síti, kde si dávají dostaveníčko odesilatelé dat určité skupiny s jejich příjemci. Teoreticky může mít každá skupina své vlastní shromaždiště, v praxi ale bývá v každé části jeden či několik málo strojů vyhrazených jako RP pro všechny zdejší skupiny.

sdílený strom

PIM-SM vytváří pro skupinu tak zvaný *sdílený strom (shared tree)*, jehož kořenem je shromaždiště a větve dosahují do všech směrovačů, které se přihlásily k odběru skupiny. Základní distribuce dat vypadá tak, že odesilatel skupinových dat pošle datagram, jeho přilehlý směrovač jej zabalí do PIM zprávy nazývané z nevyzpytatelných důvodů *Registrace (Register)* a pošle na individuální adresu shromaždiště. Zde se příchozí datagram vybalí a rozešle sdíleným stromem příjemcům. Sdílený strom je podle této představy jednosměrný a slouží k distribuci dat od RP k příjemcům, příklad vidíte na obrázku 8.11.

Když se směrovači protokolem MLD ozve zájemce o novou, zde dosud nepřijímanou skupinu, směrovač pošle shromaždišti dané skupiny zprávu *Připojení*, kterou žádá o zapojení do jejího sdíleného stromu. Skupina, do níž se hlásí, bývá označována jako (*,G). Tento zápis znamená, že odesilatel má zájem o pakety ze všech zdrojů (hvězdička) zaslané na skupinovou adresu G.

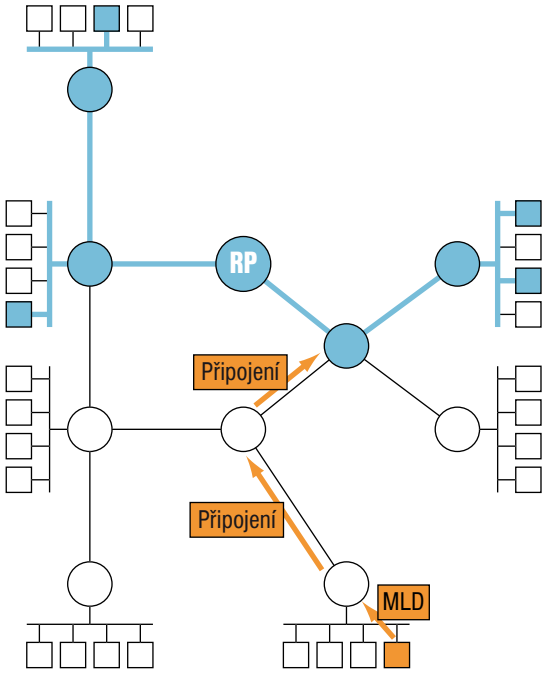


Obrázek 8.11: Distribuce dat sdíleným stromem PIM-SM

Připojení se posílá na individuální adresu RP. Směrovače po cestě si poznamenají, že do rozhraní, ze kterého přišla, mají do budoucna posílat danou skupinu. Jakmile zpráva dorazí ke směrovači, který již je součástí jejího sdíleného stromu, žádost o vstup se zahodí, protože právě byla naplněna. Celá posloupnost směrovačů, jimiž na své cestě prošla, se přidá do sdíleného stromu skupiny a bude se do budoucna účastnit předávání jejích dat.

Proces zapojení do sdíleného stromu ilustrují obrázky 8.12 (zaslání požadavku o příjem skupiny) a 8.13 (rozšíření stromu na základě této žádosti). Dokud směrovač má přímé či nepřímé posluchače skupiny, posílá v určitých intervalech *Připojení*, aby u nadřízených občerstvil své členství ve sdíleném stromu. Jestliže z určité větve dlouho nepřicházejí připojovací zprávy, odřízne se.

Při vstupu do sdíleného stromu i rozesílání dat hrají důležitou roli směrovače poblíž příjemců a zdrojů dat. Ty lze určit snadno, pokud je dotyčný stroj připojen dvoubodovou linkou. Většina počítačů je dnes ale připojena sítěmi s mnoha účastníky, jako jsou Ethernet či Wi-Fi. Do nich může být zapojeno i několik směrovačů a je třeba vyjasnit, který z nich má mít tuto úlohu. PIM-SM pro tento účel definuje jednoduchý mechanismus, kterým si skupina směrovačů připojených ke stejné lince vybere mezi sebou tak



Obrázek 8.12: PIM-SM – žádost o zapojení do sdíleného stromu

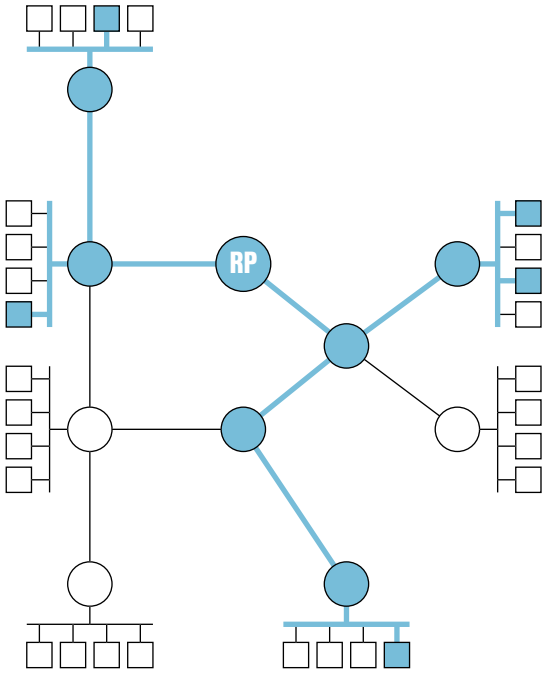
zvaný *zodpovědný směrovač (designated router, DR)*, který zastupuje tuto síť a počítače k ní připojené. Zodpovědný směrovač příjemce posílá *Připojení*, zodpovědný směrovač odesilatele posílá *Registraci*.

Už z jednoduchého příkladu na obrázku 8.11 je patrné, že rozesílání dat sdíleným stromem je neefektivní. Do míst ležících poblíž zdroje se dostávají přes RP, takže z Olomouce do Ostravy se jezdí přes Prahu. PIM proto obsahuje několik optimalizačních mechanismů.

Konec registrace

Prvním z nich je *Konec registrace (Register-Stop)*. Vychází z toho, že odesilatel skupinových dat bývá zároveň příjemcem dané skupiny. Jedna větev sdíleného stromu proto vede směrem k němu a data po ní proudí dvakrát – nejprve registrovaná směrem k RP, později rozbalená směrem k příjemcům. To by se dalo odstranit.

Shromaždiště proto po obdržení registrovaného datagramu ze zdroje S pro skupinu G pošle zodpovědnému směrovači, který odeslal *Registraci*, žádost o vstup do stromu (S,G). Jedná se o nový distribuční strom pro datagramy směřující na skupinovou adresu G odeslané ze zdroje S. Použije k tomu zprávu *Připojení* a stejně jako ve výše popsaném případě sdíleného stromu pro (*,G) se nyní vytvoří větev stromu pro (S,G). Směrovače na ní budou zpravidla také součástí sdíleného stromu (*,G) a budou datagramy



Obrázek 8.13: PIM-SM – sdílený strom po zpracování žádosti z obrázku 8.12

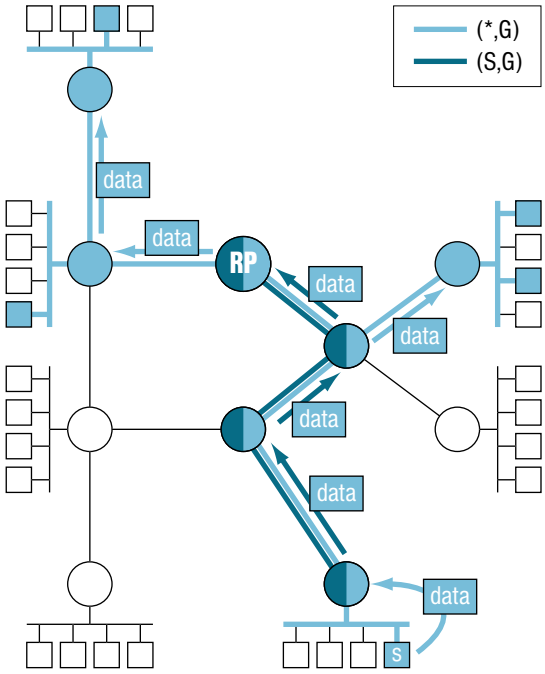
přicházející stromem (S,G) předávat do stromu (*,G). Jakmile shromaždišti dorazí první datagram ze stromu (S,G), pošle zodpovědnému směrovači pro S zprávu *Konec registrace*, kterou říká „už mi neposílej registrované datagramy, dostávám je přímo“. Distribuci dat po provedení Register-Stop najdete na obrázku 8.14.

Efektivita šíření dat se tak výrazně zlepší, ale k ideálu mívá stále daleko. Pro některé příjemce může být cesta přes RP slušnou oklikou. Směrovač zapojený do sdíleného stromu skupiny G se může rozhodnout[5] zapojit se přímo do stromu (S,G), jehož kořenem je zodpovědný směrovač zdroje S. Pošle tedy PIM zprávu *Připojení* pro vstup do stromu (S,G) směrem ke zdroji S. Stejně jako dříve to znamená, že se někde (možná až v kořeni) napojí na tento strom.

strom nejkratších cest

V terminologii PIM je strom (S,G) označován jako *strom nejkratších cest (Shortest-Path Tree, SPT)*. Po rozšíření začne některý směrovač v něm dostávat datagramy dvojmo – jednou přímo od zdroje stromem nejkratších cest, podruhé se zpožděním od RP sdíleným stromem. Na to reaguje odříznutím ze sdíleného stromu pro tento konkrétní zdroj. Pošle svém nadraze-

[5] Pravidla, kdy tak má udělat, nejsou oficiálně definována. Záleží na místní politice, například může takový krok udělat, když objem provozu překročí určitou mez.



Obrázek 8.14: PIM-SM – distribuce dat po Register-Stop

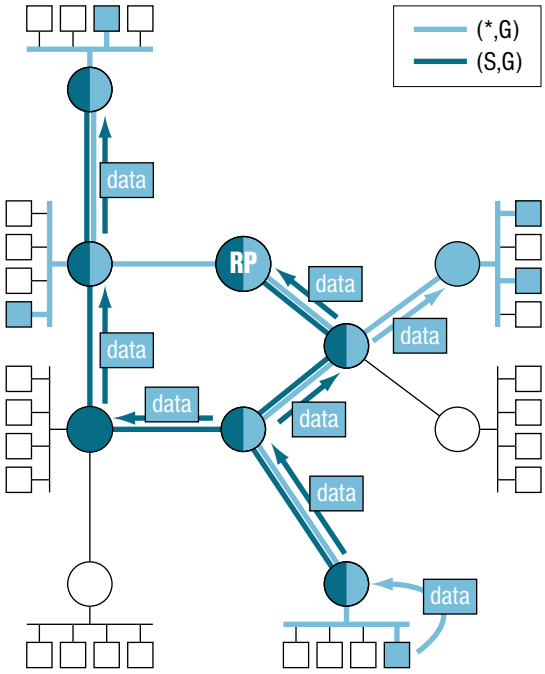
nému uzlu ve sdíleném stromě zprávu *Odříznutí (Prune)* pro (S,G), kterou říká „data do skupiny G pocházející od S už mi neposílej“. Pokud nadřízený nemá jiného odběratele, předá zprávu dál nahoru – ze sdíleného stromu se tak pro zdroj S odřízne celá nepotřebná větev[6]. Situaci naší ukázkové skupiny poté co směrovač vlevo nahoře vstoupil do stromu nejkratších cest znázorňuje obrázek 8.15.

K podobnému kroku, tedy k přihlášení pouze do stromu (S,G), může sáhnout směrovač, kterému se protokolem MLD přihlásil odběratel pouze pro data z tohoto konkrétního zdroje.

opuštění stromu

Pokud směrovač zjistí, že pro určitou skupinu už nemá žádné zájemce, pošle ke kořeni příslušného stromu[7] žádost o likvidaci své větve, zprávu *Odříznutí*. Ta cestuje stromem vzhůru tak dlouho, dokud nedorazí na směrovač s jiným odběratelem, který nadále má zájem o příjem skupiny. Celá větev distribučního stromu, jíž zpráva *Odříznutí* prošla, bude odstraněna.

[6] Pro konkrétní zdroj. Dat pocházejících z jiných zdrojů se toto odříznutí netýká.
[7] Tedy na adresu RP pro strom sdílený, na adresu zdroje pro strom nejkratších cest.



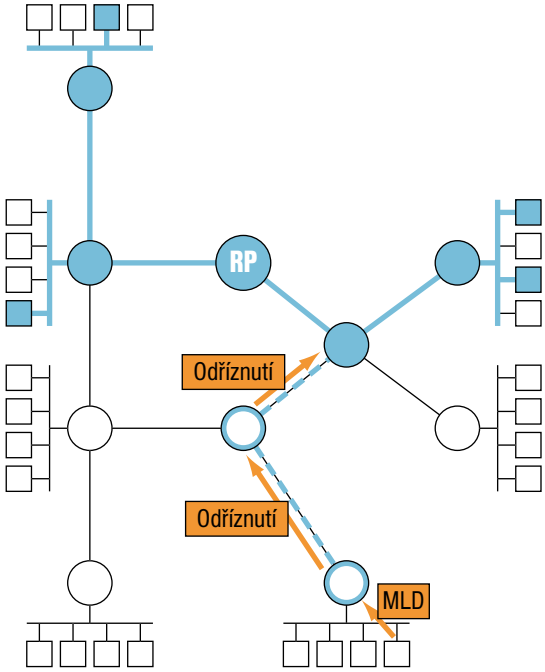
Obrázek 8.15: PIM-SM – strom nejkratších cest

Situaci, kdy počítač vpravo odhlásí příjem skupiny, ilustruje obrázek 8.16. Čárkovaně jsou v něm zobrazeny větve sdíleného stromu, které budou v důsledku této akce zrušeny.

PIM-SM je poměrně minimalistický ohledně počtu různých typů zpráv. Vystačí s pouhými šesti základními typy (jejich přehled uvádí tabulka 8.6). Dost složitá jsou však pravidla pro jejich vysílání a zpracování – kdo komu a za jakých podmínek posílá kterou z nich a jak se má chovat její příjemce.

určení shromaždiště

Aby PIM-SM dobře fungoval, musí umět najít adresu shromaždiště pro každou skupinu. Navíc musí být tato adresa v rámci dosahu skupiny jednoznačná, jinak by vznikaly oddělené sdílené stromy. Tento problém vůbec není jednoduchý a PIM-SM pro něj nabízí několik alternativních řešení. V první řadě je možná statická konfigurace RP pro jednotlivé skupiny, kterou podle RFC 4601 musí podporovat každý směrovač implementující PIM-SM. Druhou alternativou je vložit adresu RP přímo do skupinové adresy (tzv. embedded RP, viz strana 71). Na rozdíl od ostatních je tato možnost dostupná jen pro IPv6, protože v IPv4 adrese na něco takového prostě není dost místa. Má příjemné vlastnosti, protože je na jedné straně dost pružná – umožňuje vytvářet RP podle potřeby – a zároveň jednoduchá. Z adresy skupiny se snadno sestaví adresa jejího RP a může se komunikovat.



Obrázek 8.16: PIM-SM – opuštění sdíleného stromu

Haló	*Hello*	oznámení existence směrovače
Registrace	*Register*	zabalený skupinový datagram odeslaný RP k distribuci
Konec registrace	*Register-Stop*	ukončit odesílání registrovaných datagramů
Připojení	*Join*	vstup do stromu
Odříznutí	*Prune*	odpojení od stromu
Prohlášení	*Assert*	řešení problémů s několika směrovači v jedné LAN

Tabulka 8.6: Typy zpráv protokolu PIM-SM

Třetí varianta je nejsložitější. Definuje postup pro dynamické určování shromaždišť jednotlivých skupin. Jeho definici najdete v RFC 5059: *Bootstrap Router (BSR) Mechanism for Protocol Independent Multicast (PIM)*. Spočívá v tom, že směrovače v jedné PIM doméně si mezi sebou vyberou jednoho „rozhodčího“, označovaného jako *bootstrap router (BSR)*. U něj se pak ostatní směrovače ucházejí o roli RP. BSR vybere shromaždiště pro jednotlivé skupiny a tuto informaci pak rozšíří všem směrovačům v PIM doméně.

RP je pro skupinu unikátní a představuje tak zároveň její slabé místo. Pokud přestane pracovat, zastaví se činnost sdíleného stromu (data do něj posílá jen RP) a skupina bude mít vážné potíže. Jednou z možností, jak tento problém obejít, je realizovat shromaždiště skupinou směrovačů se společnou výběrovou adresou (anycast RP). Ze stromu se pak vlastně stane les, příjemci i vysílající jsou napojeni vždy na nejbližšího zástupce RP, který pak datové pakety předává i dalším dílčím RP k odeslání do jejich stromů. Podrobněji toto uspořádání popisuje RFC 4610: *Anycast-RP Using Protocol Independent Multicast (PIM)*.

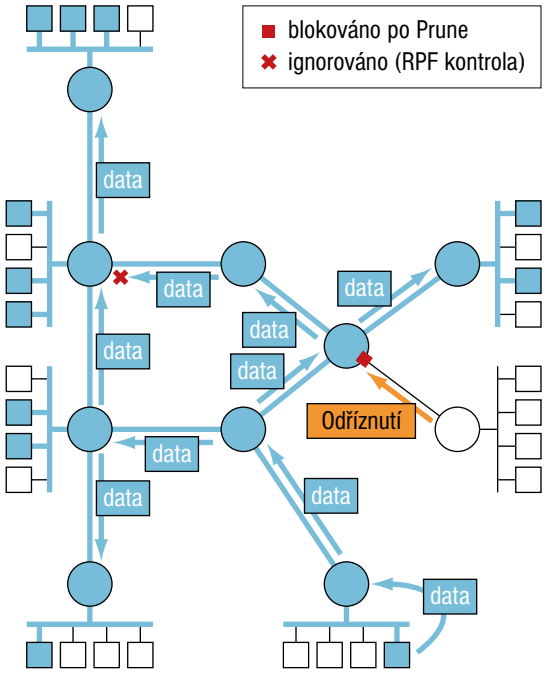
8.3.2 PIM Dense Mode (PIM-DM)

Tato odrůda PIMu předpokládá, že příjemci skupinového vysílání jsou téměř všude. Není-li řečeno jinak, rozesílá skupinová data do všech rozhraní kromě toho, ze kterého dorazila[8]. Tento způsob distribuce ovšem vede k tomu, že data mouhou přicházet z nečekaných směrů a pokud síť obsahuje cyklus, mohla by po ní kroužit i věčně. Aby předávání probíhalo alespoň trochu rozumně, používá PIM-DM tak zvanou *RPF kontrolu (Reverse Path Forwarding check)*. Pokud dorazí skupinově adresovaný datagram, podívá se do individuální směrovací tabulky, kudy vede cesta k jeho odesilateli. Jestliže vede rozhraním, jímž datagram dorazil, přijme jej a rozešle do všech ostatních rozhraní. V opačném případě jej ignoruje, protože přišel jakousi oklikou.

Pokud některý ze směrovačů nechce určitou skupinu přijímat, pošle ve směru jejího zdroje PIM zprávu typu *Odříznutí*. Stejně jako v případě PIM-SM znamená požadavek o odpojení z distribuce. Odříznutí se provádí vždy pro konkrétní dvojici zdroje a skupiny (S,G) a posílá se směrem k S. Směrovač se musí postupně odpojit ze skupiny G pro všechny zdroje, jež do ní vysílají. Ve stavových informacích pro distribuci si PIM-DM ukládá informace o odříznutých směrovačích. Datagram ze zdroje S pro skupinu G pak pošle do všech rozhraní kromě těch, kde sousední směrovač odhlásil příjem (S,G).

Odhlášení je dočasné. Po vypršení doby platnosti se zruší (pokud je soused neobnoví) a data opět začnou být zasílána. Ovšem zájem o ně může vzniknout dříve – pokud se odříznutému směrovači ohlásí příjemce dané

[8] Opět záplavový algoritmus, roztékání aneb flooding.



Obrázek 8.17: Distruce datových paketů podle PIM-DM

skupiny. Pro tento případ je definován nový typ PIM zprávy nazvaný *Roubování (Graft)*, která zhruba znamená „rozmyslel jsem si to, o skupinu (S,G) mám zase zájem“. Směrovač jej pošle všem sousedům, u nichž se dříve odřízl.

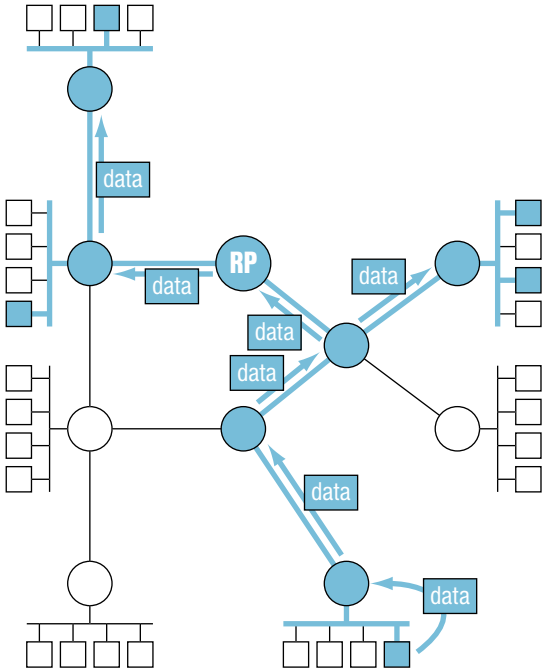
Podrobný popis protokolu najdete v RFC 3973: *Protocol Independent Multicast – Dense Mode (PIM-DM): Protocol Specification*.

8.3.3 Bidirectional PIM (BIDIR-PIM)

Protokol PIM-SM staví dva druhy stromů: jednosměrný sdílený strom, do nějž data posílá pouze RP a paket se mu musí poslat zprávou *Registrace*, a opět jednosměrné stromy nejkratších cest z jednotlivých zdrojů. Znamená to, že pro jednu skupinově adresovanou videokonferenci s deseti účastníky bude v PIM-SM existovat 11 stromů (po jednom pro každý zdroj plus sdílený). Pro zúčastněné směrovače je to dost zatěžující.

BIDIR-PIM naproti tomu staví pro skupinu jediný strom, který je sdílený a zároveň obousměrný. Datagram do něj může zaslat libovolný ze zapojených směrovačů a ten se pak rozšíří do všech ostatních větví, jak vidíte na obrázku 8.18. Protokol i jeho implementace je díky tomu podstatně jedno-

dušší, než v případě PIM-SM. Cenou za jednoduchost je nižší efektivita směrování dat – distribuce sdíleným stromem může vést cestami, které zdaleka nejsou optimální a které nesnesou srovnání se stromem nejkratších cest.



Obrázek 8.18: Obousměrný sdílený strom v BIDIR-PIM

Definice BIDIR-PIM je ze všech zúčastněných nejmladší. Je obsažena v RFC 5015: *Bidirectional Protocol Independent Multicast (BIDIR-PIM)*.

8.3.4 Source-Specific Multicast (PIM-SSM)

Zjednodušeně řečeno: SSM vznikne, když z PIM-SM použijeme jen stromy nejkratších cest. SSM řeší doručování datagramů odeslaných vždy konkrétním zdrojem S na skupinovou adresu G. Bohužel používá poněkud odlišnou terminologii, dvojici (S,G) se zde říká *kanál (channel)*.

SSM nepotřebuje shromaždiště, žádosti o vstup do distribučního stromu (přihlášení se k odběru kanálu) se posílají na individuální adresu zdroje S. Stejně jako v případě stromu nejkratších cest u PIM-SM se zastaví, jakmile dorazí ke směrovači, který již je součástí kanálu, případně až ke směrovači, k němuž je připojen zdroj S.

Příjemnou vlastností SSM je, že nevyžaduje koordinaci přidělování skupinových adres. Jestliže si stroje X a Y zvolí pro odesílání skupinových dat

stejnou cílovou adresu G, jedná se z pohledu SSM o dva různé kanály (X,G) a (Y,G), které nemají nic společného a jsou zpracovávány zcela odděleně. Vysílajícímu tedy stačí, aby si sám udržel pořádek ve skupinových adresách, na něž posílá data, a nemusí brát ohled na adresy používané jinými zdroji.

Adresy pro SSM skupiny mají svůj vlastní prostor, jejich formát jste viděli na obrázku 3.9 na straně 70. RFC 3307 navíc klade omezení na skupinové identifikátory (viz tabulka 3.3 na straně 69). V praxi pro jejich dynamické přidělování podle potřeb aplikací slouží rozmezí ff3*x*::8000:0 až ff3*x*::ffff:ffff.

PIM-SSM je popsán v RFC 4607: *Source-Specific Multicast for IP*.

9 Domain Name System

Domain Name System (DNS) slouží pro pohodlí uživatelů. Umožňuje používat místo IP adres symbolická jména počítačů, uspořádaná do hierarchické struktury. Jeho dvě nejzákladnější funkce jsou: převod jména na IP adresu a naopak převod IP adresy na odpovídající jméno.

V případě IPv6 je role DNS velmi významná, protože zdejší adresy jsou nechutně dlouhé a obtížně se pamatují i zapisují. Příběh IPv6 a DNS má sice za sebou dost klopotnou historii zahrnující převraty, návraty i slepé uličky, nicméně v poslední době se situace stabilizovala a dnes už běží podpora IPv6 v DNS hladce.

Symbióza má dvě stránky: za prvé je třeba ukládat IPv6 adresy do DNS, aby bylo možné získávat ke jménům adresy obou protokolů a naopak, za druhé se pak musí DNS servery zpřístupnit po IPv6, aby se jich klienti hovořící novým protokolem vůbec měli jak zeptat. Tyto dva problémy – IPv6 obsažené v DNS záznamech a IPv6 jako transport pro DNS – jsou do značné míry nezávislé. Klidně je možné hovořit se serverem protokolem IPv4 a vyměňovat si přitom informace o IPv6 a naopak. Ten první vyžaduje standardizaci ze strany IETF, ten druhý je spíše otázkou implementací DNS serverů a úsilí jejich správců. Bohužel v obou se objevily problémy, jejichž odstranění trvalo dlouhé roky.

historie

V roce 1995 vyšlo RFC 1886: *DNS Extensions to support IP version 6*, které definovalo jednoduché, ale nepříliš silné konstrukce pro ukládání IPv6 informací do DNS. Po pěti letech následovalo RFC 2874: *DNS Extensions to Support IPv6 Address Aggregation and Renumbering*, které zavedlo jiné (podstatně vypečenější) mechanismy. Zároveň prohlásilo předchozí specifikaci za zastaralou.

A začala zuřivá debata. Objevily se pochybovačné hlasy, které kritizovaly některé problémy RFC 2874. Zároveň vázla jeho implementace a správci řady sítí odmítali nově definované DNS záznamy používat.

Schizma se prohloubilo v létě 2001, kdy setkání IETF vyhlásilo novější specifikaci za experimentální a pro produkční sítě doporučilo vrátit se k RFC 1886. Potěšilo zejména ty, kteří se od původních mechanismů nikdy neodklonili. Tento názor o rok později potvrdilo RFC 3363: *Representing Internet Protocol version 6 (IPv6) Addresses in the Domain Name System (DNS)*. Snad poslední slovo do několikaletého sporu vneslo v roce 2003 RFC 3596: *DNS Extensions to Support IP Version 6*. Představuje návrat na původní pozice (s velmi drobnými změnami), o RFC 2874 se vůbec nezmiňuje. To má statut experimentálního protokolu, stejně jako jeho souputník RFC 2673 pro zápis binárních prefixů v reverzních adresách.

Kromě ukládání IPv6 informací do DNS je také třeba zajistit jeho přenos novým protokolem. To vypadá jako banalita, nicméně nový protokol proniká mezi autoritativní servery jen zvolna. V minulém vydání jsem popisoval situaci z poloviny roku 2008, kdy záznam typu AAAA existoval pro osm ze třinácti kořenových serverů, dva ze třinácti serverů domény *com* a například doména *no* neměla ani jeden autoritativní server hovořící IPv6.

O tři roky později je vidět určitý pokrok. Ze třinácti kořenových serverů nabízí nový protokol již deset, v doméně *com* zůstala situace 2/13 a doména *no* se zlepšila na 2/6. Jak je vidět, k ideálu má podpora IPv6 v horních patrech DNS hierarchie stále daleko. Podíváme-li se na domácí scénu, uvidíme nadprůměrný obraz. Všechny autoritativní servery domény *cz* mají registrované IPv6 adresy a více než 10 % všech domén druhé úrovně v *cz* má alespoň jeden autoritativní server podporující IPv6. Věci se postupně mění k lepšímu, ale jde to pomalu.

9.1 IPv6 adresy v DNS

Jak již bylo řečeno, ukládání IPv6 informací do DNS řeší RFC 3596. Je jednoduché a drží se celkem přímočaře řešení, které se používá pro IPv4.

dopředné dotazy (AAAA)

Pro dopředné dotazy (tedy zjišťování adresy k danému jménu) zavedlo nový typ záznamů nazvaný *AAAA*. V IPv4 k tomuto účelu slouží záznamy *A* a jelikož je délka IPv6 adresy čtyřnásobná, odrazila se tato skutečnost v názvu nového záznamu.

Má-li počítač *pc.kdesi.cz* adresu 2001:db8:89ab:1:123:45ff:fe67:89ab, bude v zónovém souboru[1] pro doménu *kdesi.cz* obsažen záznam

```
pc        AAAA  2001:db8:89ab:1:123:45ff:fe67:89ab
```

Celá definice domény *kdesi.cz*, v níž se nacházejí dva autoritativní DNS servery ve dvou různých podsítích a jeden počítač, by mohla vypadat následovně:

```
$ORIGIN kdesi.cz.
@         SOA ns1.kdesi.cz. root.ns1.kdesi.cz. (
          2008011200 ; serial
          28800      ; refresh
          14400      ; retry
          3600000    ; expire
          86400      ; default_ttl
          )

;DNS servery
```

[1] Zónový soubor obsahuje definici dané domény. Tento pojem má původ v programu BIND, nejpoužívanějším DNS serveru.

```
          NS  ns1
          NS  ns2

;adresy počítačů
ns1       AAAA  2001:db8:89ab:1:2a0:ecff:fe12:3456
ns2       AAAA  2001:db8:89ab:2:2a0:ecff:fe12:7890
pc        AAAA  2001:db8:89ab:1:123:45ff:fe67:89ab
```

Nepříjemnou vlastností tohoto přístupu je, že pokud síť používá několik prefixů a počítače tedy mají několik adres, musí mít i odpovídající počet AAAA záznamů. Kdyby například síť z předchozího příkladu kromě prefixu 2001:db8:89ab::/48 používala navíc třeba prefix 2002:a00:1::/48 pro automatické tunelování 6to4 (dočtete se o něm na straně 261), zónový soubor by rázem nabobtnal:

```
$ORIGIN kdesi.cz.
@         SOA ns1.kdesi.cz. root.ns1.kdesi.cz. (
          2008011200 ; serial
          28800      ; refresh
          14400      ; retry
          3600000    ; expire
          86400      ; default_ttl
          )

;DNS servery
          NS  ns1
          NS  ns2

;adresy počítačů
ns1       AAAA  2001:db8:89ab:1:2a0:ecff:fe12:3456
          AAAA  2002:a00:1:1:2a0:ecff:fe12:3456
ns2       AAAA  2001:db8:89ab:2:2a0:ecff:fe12:7890
          AAAA  2002:a00:1:2:2a0:ecff:fe12:7890
pc        AAAA  2001:db8:89ab:1:123:45ff:fe67:89ab
          AAAA  2002:a00:1:1:123:45ff:fe67:89ab
```

zpětné dotazy (PTR)

Zpětný dotaz vychází ze známé IPv6 adresy a snaží se k ní získat jméno. Stejně jako ve světě IPv4 se používají záznamy typu PTR. Dotaz je položen prostřednictvím doménového jména sestaveného tak, že se obrátí pořadí šestnáctkových číslic v adrese a na konec se připojí doména *ip6.arpa*[2]. Nuly se pochopitelně nesmí vynechávat, adresa musí být kompletní. Takže pro výše zmiňovanou adresu by reverzní dotaz měl tvar

[2] Původně byla v RFC 1886 pro tento účel použita doména *ip6.int*. Pozdější RFC 2874 použilo pro tento účel *ip6.arpa*, což navíc odpovídá praxi z IPv4. Aby se domény uvedly do souladu, prohlásilo RFC 3152: *Delegation of IP6.ARPA* doménu *ip6.int* za zavrženou a předepsalo použití *ip6.arpa* pro veškeré reverzní domény IPv6. Na tom trvá i RFC 3596.

b.a.9.8.7.6.e.f.f.f.5.4.3.2.1.0.1.0.0.0.b.a.9.8.8.b.d.0.1.0.0.2.ip6.arpa

Díky obrácenému pořadí číslic se obecná část adresy (prefix) dostává na konec a lze tedy realizovat distribuovanou správu reverzních domén. Například síť instituce vlastnící doménu *kdesi.cz* má prefix 2001:db8:89ab::/48 a tudíž dostane do správy reverzní doménu *b.a.9.8.8.b.d.0.1.0.0.2.ip6.arpa*. Pro počítač *pc.kdesi.cz* by její zónový soubor obsahoval záznam

```
b.a.9.8.7.6.e.f.f.f.5.4.3.2.1.0.1.0.0.0  PTR  pc.kdesi.cz.
```

Celá reverzní zóna odpovídající výše uvedené doméně *kdesi.cz* by musela obsahovat:

```
$ORIGIN b.a.9.8.8.b.d.0.1.0.0.2.ip6.arpa.
@          SOA ns1.kdesi.cz. root.ns1.kdesi.cz. (
           2008011200 ; serial
           28800      ; refresh
           14400      ; retry
           3600000    ; expire
           86400      ; default_ttl
           )

;DNS servery
           NS  ns1.kdesi.cz.
           NS  ns2.kdesi.cz.

;reverzní záznamy
6.5.4.3.2.1.e.f.f.f.c.e.0.a.2.0.1.0.0.0  PTR  ns1.kdesi.cz.
0.9.8.7.2.1.e.f.f.f.c.e.0.a.2.0.2.0.0.0  PTR  ns2.kdesi.cz.
b.a.9.8.7.6.e.f.f.f.5.4.3.2.1.0.1.0.0.0  PTR  pc.kdesi.cz.
```

Reverzní domény jsou samozřejmě obludně dlouhé, což je důsledkem dlouhých IPv6 adres. Není třeba psát je úplně celé (díky $ORIGIN), ale i tak zbyde v zónovém souboru 16 až 20 číslic.

Pracné, ale robustní. Tak by se stručně dalo shrnout krédo, v jehož duchu je postaveno RFC 3596 (a jeho předchůdce RFC 1886). Varianta, se kterou přišlo RFC 2874, umožňovala, aby se při dopředném i zpětném hledání část informace doplnila z jiného záznamu. To umožňovalo velmi elegantně přidávat, odebírat či měnit prefixy v adresách. Bohužel ale zároveň zvyšovalo zátěž DNS systému dotazy na další části a zároveň i riziko, že se řetězec navazujících záznamů někde přetrhne a výsledek se stane nedobytným. Věnujme mu tichou vzpomínku.

Rozsah IPv6 záznamů si de facto vynucuje strojové generování zónových souborů. Pokud není cílová síť opravdu malá a adresy a jména v ní se mění obvyklou rychlostí, je ruční editace zónových souborů nepříjemná a riziko

zanesení chyb vysoké. Vřele doporučuji poohlédnout se po vhodném programu, který povede databázi počítačů a zónové soubory podle ní na přání vygeneruje.

9.2 Obsah domén

Technické řešení pro poskytování IPv6 informací v DNS je celkem jednoduché, přesto však zbývá pár témat k přemýšlení. Patří mezi ně především rozhodnutí, jaké IPv6 adresy vlastně do DNS ukládat. Kromě toho je třeba si ujasnit, jak koncipovat vztah jmen a adres obou protokolů[3].

adresy v DNS

Z kapitoly 3.9 na straně 77 víte, že rozhraní počítače nese několik IPv6 adres s různým dosahem i životností. Které z nich patří do DNS a které nikoli? Začněme pozitivně. Do DNS rozhodně zařaďte:

- Všechny globální individuální adresy stroje s dlouhodobější platností.
- Dlouhodobě platné adresy přechodových mechanismů, jako je třeba 6to4.

Do DNS naopak nepatří:

- Lokální linkové adresy – mají platnost jen pro místní linku, DNS klient pochází odkudkoli a nemá žádnou šanci zjistit, která linka je ta pravá a zda se jej adresa týká či nikoli. Obecně se dá říci, že adresy omezeného dosahu nemají v DNS co dělat.
- Náhodně generované krátkodobé adresy pro zachování soukromí (psal jsem o nich na straně 62) – jednak by bylo třeba obsah zóny neustále aktualizovat, ale především by se vazbou na stejné jméno zcela zabil jejich účel zamezit sledování pohybu stroje v Internetu.

Otevřenou otázkou zůstávají adresy, které správce sítě nemá pod přímou kontrolou. Mám na mysli adresy získané bezstavovou automatickou konfigurací či přidělované liberálním DHCP serverem nastaveným ve stylu „dám nějakou adresu každému, kdo si o ni řekne“. Vznikají víceméně náhodně a navíc mívají krátkodobý charakter. Pro ně neexistuje univerzální doporučení. Svět se asi nezboří, když v DNS nebudou. Pokud je tam chcete mít, bude zřejmě třeba nasadit dynamické aktualizace DNS, jejichž pomocí si klient se serverem dohodne doménové jméno a server je následně zařadí do DNS. Toto rozhodnutí je čistě na vás.

strategie pojmenování

Pokud má počítač jen IPv4 nebo IPv6 adresu, je jeho zařazení do DNS jednoznačně dáno. Jak se ale zachovat, když komunikuje oběma protokoly a má samozřejmě jejich adresy? Nabízejí se dva základní přístupy.

[3] Dovolím si předpokládat, že nejste natolik progresivní, že byste provozovali čistě IPv6 síť. Většina současných sítí funguje „pod obojí“ a odtud pak plynou zde naznačená dilemata.

stejné jméno Podle prvního přidělíte IPv4 i IPv6 adresu stejnému jménu. Pokud naše experimentální *pc.kdesi.cz* bude mít IPv4 adresu 10.1.2.3, budou jeho dopředné záznamy následující (v IPv6 jsem zachoval dvojici prefixů):

```
pc          A      10.1.2.3
            AAAA   2001:db8:89ab:1:123:45ff:fe67:89ab
            AAAA   2002:a00:1:1:123:45ff:fe67:89ab
```

Ano, tak je to správné, tak je to čisté, tak to má být. Pokud někdo projeví touhu komunikovat s *pc.kdesi.cz*, dodá mu DNS všechny tři adresy a jeho stroj si podle svého připojení vybere. Komunikace novým protokolem proběhne zcela transparentně, aniž by se kdo co dozvěděl.

Až na případy, kdy upadne na ústa. Současné operační systémy totiž při dostupnosti obou protokolů dávají většinou přednost IPv6. Pokud tedy stroj navazující spojení bude také mít IPv6 adresu, dá pravděpodobně přednost novému protokolu a použije některý z AAAA záznamů (z nich vybere podle pravidel popsaných v části 3.11 na straně 82). Jenže když nebude dobře fungovat IPv6 spojení mezi oběma stroji, dopadne to špatně. Taková situace bohužel v současnosti není až tak vzácná. IPv6 se vyskytuje pořád poněkud ostrůvkovitě, často má experimentální charakter. Různé dohledové programy a systémy upozorňující na nefunkčnosti v síti si často hledí jen IPv4.

Pokud nastane takový problém, spojení se nepodaří navázat. To ovšem musí zjistit až transportní protokol TCP, protože IP si doručování dat nijak nepotvrzuje. Čeká se tedy na vypršení trpělivosti TCP, což trvá desítky vteřin až minuty, teprve pak vzdálený počítač zkusí jinou alternativu a nakonec se snad dostane k něčemu fungujícímu. Jenže tou dobou už nejspíš uživateli došla trpělivost a komunikaci přerušil.

Kromě toho se mohou objevit větší či menší zádrhele na aplikační úrovni. Mohu posloužit ilustračním příkladem: Pro vzdálený přístup k serveru používám SSH. Nezadávám heslo, mám vygenerován soukromý a veřejný klíč, jejichž prostřednictvím se můj klient autentizuje. Jednoho rána se ovšem znenadání začal dožadovat hesla. Důvodem bylo, že jsme serveru výše popsaným způsobem přiřadili IPv6 adresu. Můj stroj také disponuje IPv6 adresou, SSH tedy automaticky přešlo na nový protokol, ale já jsem u svého veřejného klíče na serveru měl nastaveno, že platí jen z IPv4 adresy mého počítače. Musel jsem přidat také jeho IPv6 adresu, abych mohl klíč nadále používat. Občas se zkrátka IPv4 skrývá i pod kameny, kde byste je nečekali.

odlišná jména Druhou možnou strategií je používat pro IPv6 adresy odlišná jména. Bývá obvyklé zavést speciální poddoménu (často nazvanou *ip6* nebo *ipv6*) a do ní je zařadit. V našem experimentálním případě by tedy staré adresy zůstaly v doméně *kdesi.cz*, jména přiřazená IPv6 by končila *ip6.kdesi.cz*. Konkrétně pro stroj *pc* by definice vypadala následovně:

```
pc          A      10.1.2.3

pc.ip6      AAAA   2001:db8:89ab:1:123:45ff:fe67:89ab
            AAAA   2002:a00:1:1:123:45ff:fe67:89ab
```

V tomto uspořádání z použitého jména přímo vyplývá komunikační protokol. Zadá-li někdo DNS k řešení jméno *pc.kdesi.cz*, dostane pouze IPv4 adresu. Jestliže použije *pc.ip6.kdesi.cz*, získá dvojici IPv6 adres.

Tahle cesta představuje sázku na jistotu. Použil ji například *Google*, když začal poskytovat svůj slavný vyhledávač po IPv6. Případné prodlevy při problémech se síťovým protokolem, které jsem popsal výše, jsou zde zcela neakceptovatelné. Proto *www.google.com* vede pouze na IPv4 adresy[4], zatímco za jménem *ipv6.google.com* se skrývají IPv6 adresy téže služby.

Oddělení adres samozřejmě znamená útlum IPv6 provozu – většina uživatelů bude používat stará známá jména, která mohou být i vestavěna v aplikacích, a IPv6 zůstane na ocet. Může vám také vadit (nebo vyhovovat), že z údajů poskytnutých DNS se nedá nijak zjistit, že *pc.kdesi.cz* a *pc.ip6.kdesi.cz* mají něco společného. Jsou to dvě zcela oddělená jména se svými reverzními záznamy. Je *www.google.com* a *ipv6.google.com* stejný stroj? To vědí jen u Google.

kompromis

Samozřejmě nikde není řečeno, že všechna jména v doméně musí používat stejnou koncepci. V síti TU v Liberci kombinujeme oba postupy: Pro běžné uživatelské počítače používáme oddělená jména, protože na jedné straně chceme umožnit experimenty s IPv6, ale na straně druhé se u nich na kvalitní podporu IPv6 nelze spolehnout. Pouze u vybraných serverů, u kterých za dostupnost IPv6 ručíme a chceme transparentně používat nový protokol kdykoli to je možné, používáme společné jméno pro A i AAAA záznamy.

Variantu s oddělenými jmény lze také vnímat jako dočasné řešení. Až si budete jisti, že IPv6 služba je dostatečně robustní a dostupná, opustíte její samostatné jméno a přejdete na první variantu. Nicméně s jednotkou dočasnosti 1 furt máme v naší zemi nemalé zkušenosti.

9.3 Provozní záležitosti

podporovat také IPv4

Několik informativních RFC se zabývá pragmatickými otázkami soužití IPv6 s DNS. Příjemně minimalistické je RFC 3901: *DNS IPv6 Transport Operational Guidelines*, jehož hlavní myšlenka by se dala shrnout do věty „Nebuďte nadměrně progresivní, nedělejte DNS závislé na IPv6.“

Obsahuje dvě doporučení. První se týká rekurzivních serverů v koncových sítích, které řeší dotazy místních klientů. Podle RFC 3901 by měly mít možnost komunikovat oběma protokoly, aby nezůstaly bezmocné, pokud pro

[4] Pokud nejste zapojeni do IPv6 programu firmy Google.

některou doménu dostanou jen IPv4 adresy autoritativních serverů. Třeba pro výše zmiňované Norsko. V polovině roku 2008 nemá žádný z autoritativních serverů domény *no* IPv6 adresu. To samozřejmě neznamená, že v celém Norsku nemají IPv6, ani že někde v norských doménách nejsou AAAA záznamy. Ale aby se k nim server dostal, musí po cestě pohovořit s některým ze serverů pro doménu první úrovně *no*, a to protokolem IPv4 – jiný není k mání. Pokud by váš server hovořil jen IPv6, nedokázal by to a celé norské DNS by pro něj zůstalo nedostupné.

Dostupnost IPv4 pro místní rekurzivní server lze zařídit i zprostředkovaně. Pokud se jedná o čistou IPv6 síť, mohlo by být přivedení IPv4 k jejímu serveru zbytečně komplikované. Naštěstí implementace DNS serverů obvykle počítají s alternativou, že dotyčný počítač nemá přímý přístup k Internetu. V takovém případě k němu přistupují přes prostředníka (forwarder). Koncový DNS server pak spravuje pouze svou vyrovnávací paměť a jakékoli dotazy, které její pomocí není schopen zodpovědět, předává prostředníkovi. V našem případě by komunikace mezi koncovým serverem a jeho prostředníkem probíhala protokolem IPv6, zatímco prostředník by hovořil oběma protokoly a dokázal se domluvit s kterýmkoli DNS serverem potřebným k vyřešení dotazu.

Druhým konkrétním doporučením obsaženým v RFC 3901 je nedělat opačnou věc. Tedy nevytvářet domény, jejichž všechny autoritativní servery hovoří pouze IPv6. V Internetu existuje řada klientů a serverů podporujících pouze IPv4. Není rozumné házet je přes palubu. Alespoň jeden ze serverů každé domény by měl podporovat starší verzi IP. Ideální samozřejmě je poskytnout dostatečný počet autoritativních serverů jak pro IPv4 tak pro IPv6, aby nikdo neměl problém domluvit se s nimi svým nativním protokolem.

chyby serverů

Pokud v novinách nejprve obracíte na černou kroniku, určitě si nenechte ujít RFC 4074: *Common Misbehavior Against DNS Queries for IPv6 Addresses*. Dočtete se v něm, co všechno jsou schopny udělat DNS servery, když jim pošlete dotaz na záznam typu AAAA. Některé omdlí (vůbec neodpoví). Jiné lžou – zatloukají existenci jména, i když pro ně existují záznamy typu A, tváří se jako neautoritativní či pošlou zkomolenou odpověď. Další ten dotaz natolik zmate, že odpoví chybovým kódem. A pak se zkuste na něco spolehnout ...

Nejrozsáhlejší a nejčerstvější soubor doporučení a úvah ohledně provozování IPv6 DNS najdete v RFC 4472: *Operational Considerations and Issues with IPv6 DNS*. Probírá komplexně celou problematiku, mimo jiné i otázky adres vhodných pro zařazení do DNS či strategii pojmenování, kterým jsem se věnoval v předchozí části.

AAAA záznamy fungujícím strojům

Zdůrazňuje se zde, že AAAA záznam je vhodné ke jménu přiřadit, až když IPv6 na daném stroji skutečně funguje. Tedy má nakonfigurovánu příslušnou adresu, je připojen do dosažitelné sítě s fungujícím IPv6 a nový proto-

kol podporují i provozované služby. Toto doporučení platí dvojnásob, pokud používáte strategii společného jména pro oba protokoly.

Dozvíte se také, co se stane, když v DNS nastavíte různou dobu životnosti (TTL) záznamům typu A a AAAA. Prozradím rovnou ponaučení celé bajky: nedělejte to. Dost zevrubně jsou diskutovány otázky dynamické aktualizace DNS při stavové či bezstavové konfiguraci klientů. A pokud byste měli ambice napsat DNS klienta, najdete zde kapitolu s doporučeními, jak by se měl vůči IPv6 chovat.

10 IPsec čili bezpečné IP

Klasické IP neobsahovalo vůbec žádné bezpečnostní mechanismy. Je to překvapující, protože se jednalo o technologii vyvíjenou původně pro armádu, nicméně je to tak. Postupem času se ukázalo, že svět umí být i ošklivý a začaly se hledat cesty, jak komunikaci v Internetu zabezpečit.

Přístupy jsou různé – od hardwarových (utajovače na linkách) až po aplikační (jako je S/MIME v elektronické poště). Z pohledu této knihy jsou nejzajímavější snahy o poskytnutí bezpečnostních prvků přímo na úrovni IP, pro které se vžilo označení IPsec. Existuje jak pro IPv4, tak pro IPv6. Jako jedna z výhod novějšího protokolu se často uvádí, že zde je implementace *IPsec* povinná[1].

Pro uživatele či aplikace nabízí IPsec dvě základní služby: autentizaci a šifrování. Cílem autentizace je ověřit, že data odeslal skutečně ten, kdo to o sobě tvrdí. Zamezí se tak například útokům, kdy odesilatel falšuje svou IP adresu a vydává se za počítač z lokální sítě cílového stroje. Autentizace také zaručí, že obsah datagramu je původní a nebyl během průchodu sítí změněn. Šifrování navíc umožňuje utajit obsah korespondence. Data nesená v zašifrovaných datagramech dokáže rozluštit jen jejich příjemce.

Současná definice bezpečnostní architektury IP je již třetí generací. První byla soustředěna kolem RFC 1825 vydaného v roce 1995. O tři roky později ji nahradila generace kolem RFC 2401. Zatím poslední specifikaci představuje RFC 4301: *Security Architecture for the Internet Protocol* vydané koncem roku 2005. Během vývoje nedocházelo k žádným radikálním změnám, kroky jsou spíše evoluční a reagují na zkušenosti získané s implementacemi IPsec a reálnými požadavky na ně kladenými.

10.1 Základní principy

Vlastní realizace bezpečnostních služeb spočívá na dvou rozšiřujících hlavičkách: *AH (Authentication Header)* a *ESP (Encapsulating Security Payload)*. První má na starosti autentizaci datagramu, tedy především ověření pravosti jeho adres a obsahu. Druhá umí zajistit podobné služby, navíc k nim přidává možnost zašifrovat obsah. Datagram může být opatřen jednou či oběma bezpečnostními hlavičkami v závislosti na požadovaném zabezpečení.

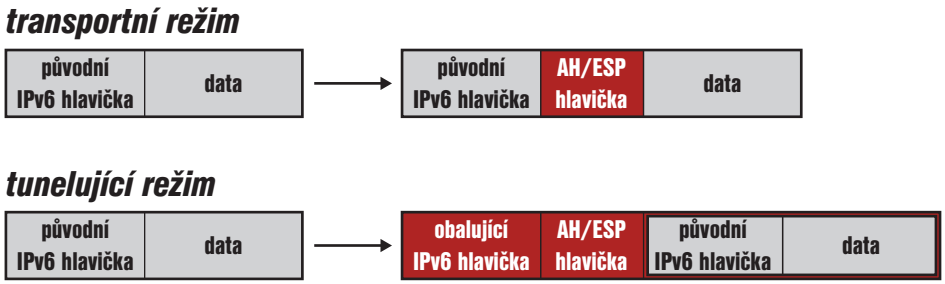
V současné době jsme svědky zřetelného odklonu od hlavičky AH. Důvodem je, že ESP dokáže nabídnout stejné služby a ještě něco navíc. Proto

[1] Což ovšem nijak nebrání tomu, aby v řadě implementací chyběla.

je podle RFC 4301 implementace ESP povinná, zatímco AH pouze dobrovolná. Dá se očekávat, že postupem času se AH ocitne zcela na vedlejší koleji a zmizí.

režimy ochrany

Bezpečnostní hlavičky lze doplňovat ve dvou režimech. V *transportním režimu* se vkládají přímo jako součást datagramu mezi jeho rozšiřující hlavičky. Naproti tomu *tunelující režim* pracuje tak, že celý stávající datagram zabalí jako data do nového datagramu, který opatří novými hlavičkami, včetně bezpečnostních. Rozdíl mezi oběma režimy ilustruje obrázek 10.1.

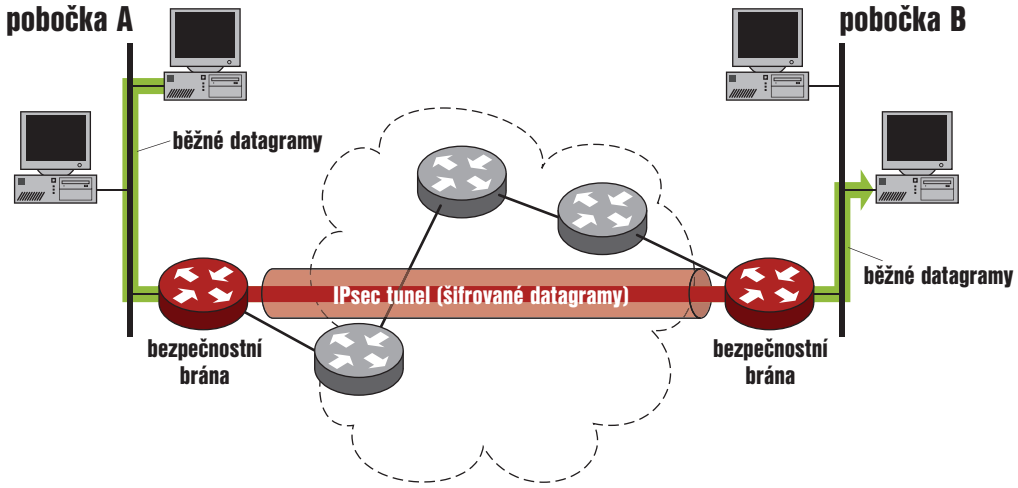


Obrázek 10.1: Režimy IPsec

Aby to bylo ještě komplikovanější, nemusí být datagramy chráněny po celé své cestě. Vezměme si jako příklad firmu, která bude mít dvě filiálky v různých městech, propojené běžným Internetem. Aby chránila svá interní data, vytvoří mezi těmito pobočkami šifrovaný tunel.

Pokud počítač v pobočce A posílá data svému kolegovi v pobočce B, odesílá je v otevřené formě. Když datagramy dorazí na hraniční směrovač lokální sítě, který zajišťuje přechod do Internetu, ten z cílové adresy pozná, že jsou určena pro pobočku B. Odešle je tedy šifrovaným tunelem – zabalí je do nových datagramů, určených hraničnímu směrovači lokální sítě B a opatřených hlavičkou ESP. Hraniční směrovač pak dešifruje původní datagramy a předává je opět v otevřené formě cílovému počítači ve zdejší síti. Takto fungující směrovač bývá označován jako *bezpečnostní brána (security gateway)*. Koncové počítače se tedy vůbec nemusí dozvědět, že při jejich komunikaci bylo použito šifrování.

Tunelující režim je z hlediska bezpečnosti o něco silnější, zejména při šifrování. V transportním režimu je šifrována celá část datagramu za hlavičkou ESP. To ovšem znamená, že jeho základní hlavička a případně část rozšiřujících hlaviček zůstávají v otevřeném tvaru. Špion sice nerozumí obsahu, ale má k dispozici IP adresy obou partnerů a může například analyzovat charakter jejich komunikace, zpracovávat si statistiky a podobně. Ostatně už samotná informace o tom, že proběhla libovolná komunikace mezi určitými adresami může být zneužitelná. Lze z ní odvozovat, že uživatel určitého počítače u něj nejspíš zrovna sedí a vyrazit mu vykrást byt. Jestliže z počítače



Obrázek 10.2: Zabezpečení po cestě (bezpečnostní brány)

pana vedoucího proběhla čilá výměna paketů se serverem zaměřeným na perverzní sex, není až tak podstatné, že jejich obsah zůstal utajen...

Naproti tomu v tunelovacím režimu je zašifrován celý původní datagram včetně všech hlaviček a opatřen hlavičkami novými. Jen ty jsou otevřené. Když tedy například bude někdo odposlouchávat šifrovaný tunel z obrázku 10.2, nedozví se, které konkrétní počítače z koncových sítí spolu hovoří. Má k dispozici jen obalující hlavičku a tedy informaci, že datagram posílá jedna bezpečnostní brána druhé.

bezpečnostní asociace

Důležitou roli v IPsec hraje tak zvaná *bezpečnostní asociace (Security Association, SA)*. Jedná se o jakési virtuální spojení dvou počítačů, které zajišťuje zabezpečený přenos dat. Součástí bezpečnostní asociace jsou všechny potřebné informace – použitý bezpečnostní protokol (AH nebo ESP, nikoli oba) a jeho režim, šifrovací algoritmus a klíče platné pro toto spojení, čítače, doba životnosti, ochranné prvky proti opakování a podobně.

Bezpečnostní asociace jsou jednosměrné. Při komunikaci je nutno navazovat je vždy po dvojicích – jednu pro vysílání, druhou pro příjem. Praktickým dopadem tohoto opatření je, že se v každém směru používají jiné klíče. Navíc jsou asociace jednoúčelové. Chcete-li opatřit datagramy jak AH, tak ESP hlavičkou, budete potřebovat samostatnou asociaci pro každou z těchto služeb. RFC 2401 zavedlo pro tuto situaci pojem *svazek bezpečnostních asociací (security association bundle)* a RFC 4301 jej zase opustilo. Bezpečnostní architektura v jeho podání sice použití několika bezpečnostních asociací pro jeden datagram připouští, ale nijak konkrétně nedefinuje a ponechává řešení na implementaci.

databáze bezpečnostní politiky

K úplnému štěstí už schází jediné – řízení. Tuto činnost má na starosti tak zvaná *databáze bezpečnostní politiky (Security Policy Database, SPD)*. Jedná se vlastně o sadu pravidel, podle kterých jsou posuzovány *všechny* přicházející či odesílané datagramy. Uplatnění pravidel vede k jednomu z následujících tří rozhodnutí:

- zahodit datagram
- zpracovat datagram – přijmout či odeslat, žádné IPsec služby se na něj nevztahují
- podrobit datagram IPsec – v tom případě databáze vydá i odkaz na bezpečnostní asociaci, která na něj má být uplatněna

Každé pravidlo obsahuje selektor určující pakety, pro něž je použitelné. Selektorem může být rozsah adres odesilatele a/nebo příjemce, protokol vyšší vrstvy a jeho porty a podobně. Systém postupně prochází pravidla v databázi a použije první, jehož selektoru posuzovaný datagram vyhovuje. Poněkud to připomíná posuzování datagramu ve firewallu. RFC 4301 požaduje, aby implicitní politikou každé SPD bylo „zahodit“. Pokud se tedy nenajde žádné specifické pravidlo použitelné pro daný datagram, bude bez milosti zlikvidován.

Například hraniční směrovač výše zmiňované pobočky A by mohl mít ve své databázi bezpečnostní politiky následující pravidla pro datagramy odcházející do Internetu:

1. Datagramy směřující na adresu X zahodit (známý phishingový server).
2. Datagramy směřující do sítě pobočky B podrobit IPsec; pravidlo odkazuje na odpovídající bezpečnostní asociaci, která předepisuje použití ESP v tunelovém režimu, kde cílem tunelu je hraniční směrovač sítě B, používá se algoritmus AES-CBC s klíčem K1.
3. Všechny ostatní datagramy jsou propouštěny bez použití IPsec.

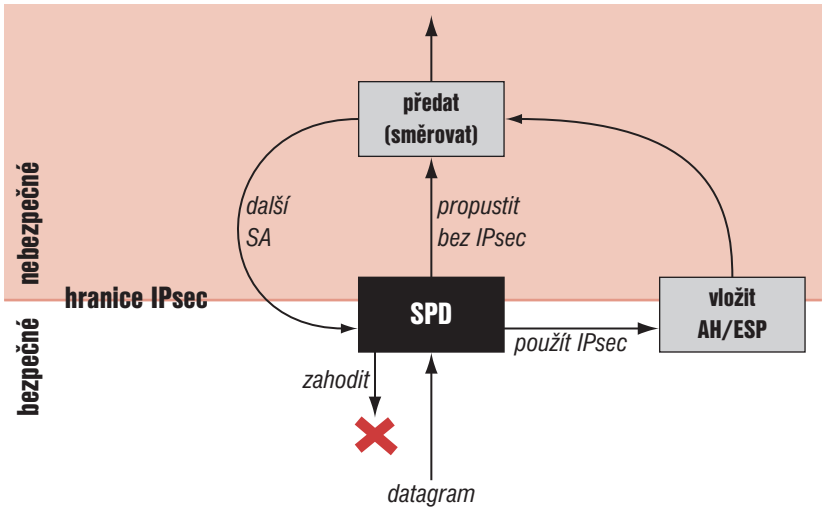
Jak je vidět, je databáze bezpečnostní politiky velmi těsně svázána s bezpečnostními asociacemi. Vznikne-li v ní nové pravidlo, často to znamená, že současně musí vzniknout i odpovídající asociace, na kterou/které se pravidlo bude odkazovat. Asociace jsou ukládány do *databáze bezpečnostních asociací (Security Association Database, SAD)*. Jsou v ní identifikovány pomocí *indexu bezpečnostních parametrů (Security Parameters Index, SPI)*, případně v kombinaci s adresami. SAD obsahuje pro každou asociaci kompletní informace – režim, kryptografické algoritmy, klíče, čítače, životnost a další.

Bezpečnostní architektura popsaná v RFC 4301 koncepčně rozděluje stroj na část bezpečnou (například interní porty zdejších aplikací) a nebezpeč-

nou (rozhraní vedoucí do veřejného Internetu). Mezi nimi vede hranice IPsec a každý datagram, který ji překračuje, musí být posouzen podle databáze bezpečnostní politiky a podle výsledku s ním naloženo.

odchozí datagramy

Jako odchozí jsou označeny ty datagramy, které opouštějí bezpečnou část a vstupují do nebezpečné. Typickým příkladem je datagram odeslaný zdejší aplikací kamsi do Internetu. Jejich zpracování je celkem přímočaré, jak naznačuje obrázek 10.3. Databáze bezpečnostní politiky rozhodne, zda bude zahozen, volně propuštěn nebo podroben IPsec operacím. V posledním případě vydá odkaz do databáze bezpečnostních asociací, která poskytne parametry pro úpravu datagramu – doplnění bezpečnostní hlavičky v transportním či tunelujícím režimu. Následně je datagram předán k odeslání. Pokud má mít několik bezpečnostních hlaviček, může to znamenat opakování celého postupu.

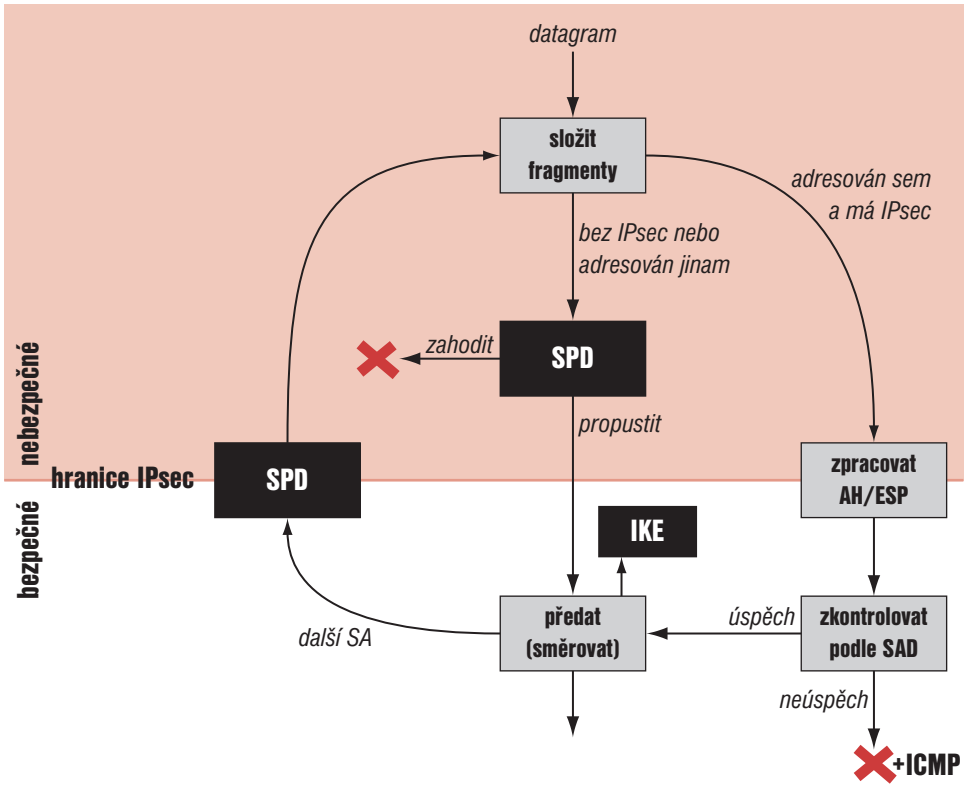


Obrázek 10.3: Zpracování odchozího datagramu podle IPsec

příchozí datagramy

Datagramy směřující opačným směrem přicházejí z nebezpečné části sítě a mají projít IPsec hranicí do bezpečné části. Může se jednat o data určená místním aplikacím, ale také o data předávaná dál. Například když bezpečnostní brána přijme tunelovaný datagram určený pro některý z místních počítačů – po uplatnění IPsec bude původní datagram vybalen a předán do vnitřní sítě svému koncovému adresátovi.

Z obrázku 10.4 vidíte, že situace je tentokrát složitější. Zpracování začíná složením datagramu, pokud dorazil ve fragmentech. O použití IPsec hlaviček rozhodl jeho odesilatel, proto se databáze bezpečnostní politiky uplatní jen pokud datagram nemá bezpečnostní hlavičky nebo je adresován jinam



Obrázek 10.4: Zpracování příchozího datagramu podle IPsec

a daný stroj pro něj hraje roli běžného směrovače. V takovém případě bude v souladu s bezpečnostní politikou zahozen nebo propuštěn k doručení.

Pokud obsahuje hlavičku AH a/nebo ESP a jako příjemce je uveden zdejší stroj, bude zpracován IPsec mechanismem. Na základě údajů z datagramu se vyhledá odpovídající bezpečnostní asociace a podle ní se provedou bezpečnostní operace. Jestliže nedopadnou dobře nebo se nepodaří najít použitelnou asociaci, datagram bude zahozen[2]. Dopadne-li vše dobře, datagram je předán k doručení.

správa bezpečnostních asociací

Klíčovým problémem pro použití IPsec je samozřejmě správa bezpečnostních asociací mezi komunikujícími partnery. Jejím nejjednodušším, ale nejméně praktickým modelem je manuální konfigurace. Správce systému ručně nastavuje, co a jak se bude dít v oblasti bezpečnosti. To je ovšem po-

[2] Pokud vám vrtalo hlavou, že příchozí datagram opatřený IPsec hlavičkou obchází databázi bezpečnostní politiky, zde je řešení. Když se s jeho odesilatelem nechceme bavit, odmítneme jeho snahu o vytvoření bezpečnostní asociace a příchozí pakety pak zahazujeme, protože pro ně není v SAD žádná použitelná asociace.

užitelné jen v malém a stabilním prostředí. Pokročilejší nasazení vyžaduje protokoly, jimiž se komunikující partneři domlouvají na dynamickém vytváření bezpečnostních asociací. Tomuto účelu slouží protokol IKEv2, k němuž se dostanu v části 10.4.1.

Pokud se vám to celé zdá poněkud krkolomné, máte naprostou pravdu. IPsec není zrovna procházka růžovou zahradou. Že v tom neplavete sami, dokazuje jen pomalu se zlepšující stav implementací IPsec v současných systémech (a to jak v IPv4, tak v IPv6).

10.2 Authentication Header, AH

Základním cílem této hlavičky je autentizovat odesilatele a obsah datagramu. Čili ověřit, že datagram odeslal skutečně ten počítač, jehož adresa je uvedena v hlavičkách, a že jej odeslal v té podobě, ve které dorazil k cíli. Kromě toho zajišťuje nepovinně (ale doporučeně) ochranu proti opakování.

odesilatel

Stroj, který opatřuje datagram hlavičkou AH, se chová zhruba následovně:

1. Vloží do datagramu hlavičku AH. Její formát a obsah najdete na obrázku 10.5.
2. Vyplní její položky – typ *Další hlavičky*, svou vlastní *Délku (Payload len)*, odpovídající *Index bezpečnostních parametrů (Security parameters index, SPI)* a *Pořadové číslo (Sequence number)*, to zvětšuje o jedničku pro každý následující datagram. *Autentizační data (Integrity check value, ICV)* prozatím vynuluje.
3. Následuje výpočet autentizačních dat. Pro jeho potřeby je však třeba datagram poněkud upravit, aby jej příjemce mohl ověřit. Některé hlavičky upraví na hodnoty, které budou mít při příchodu datagramu (např. cílovou adresu, pokud se používá hlavička *Směrování*). Ty, které předvídat nelze (třída, značka toku, počet skoků), vynuluje. Pro takto změněný datagram vypočítá *Autentizační data* a uloží je do odpovídající položky AH.

algoritmy

Pro výpočet autentizačních dat se nejčastěji používají jednosměrné hešovací funkce, jako je MD5 či SHA-1. Jejich funkce se poněkud podobá elektronickému podpisu, ale na rozdíl od něj se pracuje se sdíleným klíčem. Obě strany mají stejný tajný klíč (je součástí dat bezpečnostní asociace) a obě provádějí stejný výpočet. Algoritmy se sdíleným klíčem jsou totiž o poznání rychlejší než algoritmy s klíčem veřejným, používané u klasického elektronického podpisu.

Implementace IPsec povinně musí pro výpočet AH podporovat kombinaci algoritmů HMAC a SHA-1 (HMAC-SHA1-96 podle RFC 2404). Kromě toho by

Obrázek 10.5: Hlavička *Authentication Header*

měla podporovat AES v CBC režimu kombinovaný s MAC (AES-XCBC-MAC-96 podle RFC 3566). Naproti tomu původně povinná implementace MD5 je dnes už jen dobrovolná (HMAC-MD5-96 podle RFC 2403).

příjemce Příjemce vynuluje výše zmíněné položky s nepředvídatelnou hodnotou a *Autentizační data* z hlavičky AH. Na základě identifikátoru SPI získá klíč a algoritmus a s upraveným datagramem provede stejný výpočet, jako odesilatel. Výslednou hodnotu pak porovná s tou, která byla uvedena v hlavičce AH. Pokud se liší, znamená to, že datagram byl změněn a autentizace selhala. Takový datagram IPsec zahodí, aniž by o tom informoval jeho odesilatele (aby případný útočník neměl zpětnou vazbu).

Je-li zapnuta ochrana proti opakování, ještě předtím zkontroluje *Pořadové číslo* datagramu. Opakuje-li se číslo již použité, datagram zahodí a vůbec se jím nezabývá.

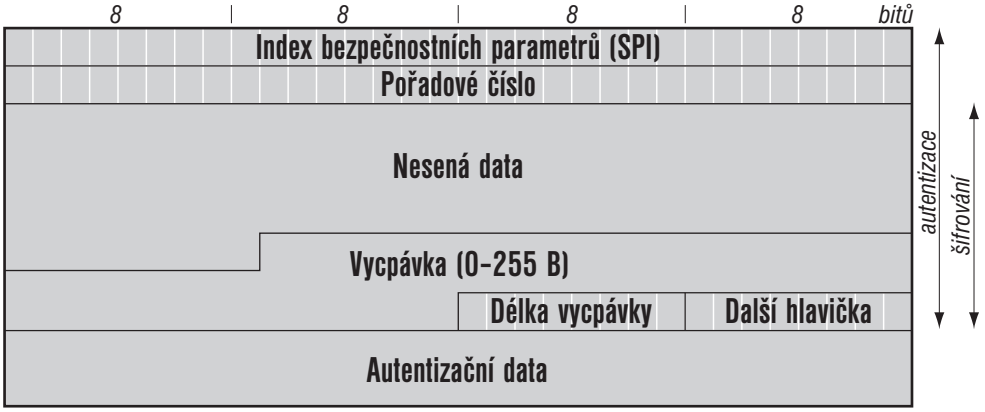
Podrobnou definici AH najdete v RFC 4302: *IP Authentication Header*. Popisuje formát hlavičky a způsob jejího zpracování, zatímco použité kryptografické algoritmy jsou definovány samostatným dokumentem. S vývojem kryptografických poznatků totiž postupně podléhají změnám, které se ale netýkají vlastních bezpečnostních hlaviček. Nebylo by praktické vydávat novou definici AH či ESP jen kvůli změně použitých algoritmů. Jejich kryptografické mechanismy proto definuje RFC 4835: *Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)*.

10.3 Encapsulating Security Payload (ESP)

Základní službou hlavičky ESP je šifrování. Kromě něj však nabízí i služby odpovídající AH – autentizaci odesilatele, kontrolu původnosti dat a ochranu proti opakování. Její dvě základní služby – šifrování a autentizace – si oba komunikující partneři dohodnou během vytváření bezpečnostní asociace. ESP může poskytnou jen jednu z nich nebo obě současně. Nicméně nedoporučuje se používat jen samotné šifrování bez autentizace.

Důvodem je, že sice chrání proti odposlechu, ale nikoli proti padělání obsahu datagramů. Proto je po implementacích IPsec požadováno, aby povinně pro ESP podporovaly jak samotnou autentizaci, tak kombinaci šifrování s autentizací. Naproti tomu podpora samotného šifrování je volitelná a jejímu využívání je rozumné se vyhnout.

ESP je hlavička dosti nezvyklá, protože „spolkne“ skoro celý datagram do sebe. Bere se jako hlavička typu end-to-end a proto se umisťuje za hlavičky určené pro každého po cestě, směrování a fragmentaci[3]. Celý zbytek původního datagramu (včetně případných dalších rozšiřujících hlaviček) je zašifrován a vložen jako *Nesená data (Payload data)* do ESP hlavičky, jejíž podobu vidíte na obrázku 10.6. Za hlavičkou již tedy nenásleduje nic dalšího, vše je uvnitř. Obrázek zároveň znázorňuje, které části datagramu jsou chráněny šifrováním a které autentizací.



Obrázek 10.6: Hlavička *Encapsulating Security Payload*

Hlavička pochopitelně obsahuje *Index bezpečnostních parametrů (Security parameters index, SPI)*, podle kterého si příjemce vyhledá bezpečnostní asociaci pro zpracování datagramu. *Pořadové číslo (Sequence number)* slouží k ochraně proti opakování a *Autentizační data (Integrity check value, ICV)* k ověření odesilatelovy totožnosti a autentičnosti datagramu (jsou-li tyto služby zapnuty).

Zašifrovaný datagram a případné další pomocné údaje vyžadované šifrovacím algoritmem tvoří vlastní *Nesená data (Payload data)*. Některé algoritmy vyžadují, aby celková délka zpracovávaných dat byla násobkem určité hodnoty. V takovém případě se k datům připojí *Vycpávka (Padding)* vhodné délky, aby se dosáhlo požadované hodnoty. Za šifrovaná data se připojí ještě údaj o *Délce vycpávky (Pad length)*.

[3] Toto platí v případě, kdy je ESP používáno v transportním režimu. V tunelovém režimu se prostě datagram obalí novým, jehož je ESP poslední hlavičkou.

Další hlavička pak identifikuje první hlavičku v zašifrovaných datech. Může se jednat o identifikaci protokolu vyšší vrstvy, jehož data datagram nese, nebo o některou z rozšiřujících hlaviček určených koncovému příjemci. To závisí na původním datagramu.

odesilatel

Odesílání datagramu opatřeného ESP hlavičkou vypadá následovně:

1. Odesilatel najde vhodnou pozici pro vložení ESP hlavičky. Zbytek datagramu případně doplní vycpávkou a zašifruje podle parametrů daných bezpečnostní asociací.
2. Vygeneruje *Pořadové číslo* (zvětšuje vždy o jedničku).
3. Pokud je požadována autentizace a kontrola integrity, vypočítá kontrolní hodnotu pro nesená data a uloží ji do ESP jako *Autentizační data*.

IPsec operace se provádějí vždy pro celý datagram. Případná fragmentace probíhá až po šifrování. Naopak příjemce si nejprve datagram poskládá a teprve pak jej bude dešifrovat.

algoritmy

Opět lze použít několik různých kryptografických algoritmů. Jejich výběr je složitější, protože autentizace a šifrování vyžadují odlišné algoritmy a navíc jsou k dispozici postupy, které zajistí obě služby pod jednou střechou. Podle RFC 4835 je pro šifrování povinná podpora AES v CBC režimu se 128bitovými klíči (AES-CBC podle RFC 3602) a Triple DES v CBC režimu (TripleDES-CBC podle RFC 2451). Dále by měl být podporován AES Counter Mode (AES-CTR podle RFC 3686). Naopak by neměl být podporován DES v CBC režimu, který předchozí generace bezpečnostních protokolů povinně vyžadovala (RFC 2405).

Pro autentizaci se nabídka algoritmů shoduje s hlavičkou AH: povinně HMAC+SHA-1, měl by být AES CBC+MAC a může být HMAC+MD5. Implementace také musí podporovat prázdný šifrovací a autentizační algoritmus označovaný jako NULL, kterým lze vyjádřit, že jedna ze dvou služeb hlavičky není poskytována. Tuto hodnotu ale nesmí nést šifrovací a autentizační algoritmus současně.

Kombinované algoritmy pro současné šifrování a autentizaci zatím nejsou striktně požadovány žádné. Za zajímavé jsou považovány CCM a Galois/Counter režimy AES (RFC 4309 a 4106), ale výzkum v této oblasti zatím pokračuje.

příjemce

Příjem datagramu vypadá takto:

1. Nejprve si příjemce vyhledá odpovídající bezpečnostní asociaci. Pokud neexistuje, datagram zahodí.
2. Následuje kontrola *Pořadového čísla* (je-li zapnuta ochrana proti opakování). Jestliže datagram nese číslo již použité, opět bude zahozen.

3. Dalším krokem je autentizace – příjemce vypočte *Autentizační data* a porovná je s obdrženou hodnotou. Pokud se liší, pryč s datagramem.
4. Teprve když všechny předchozí kroky byly úspěšné, dešifruje paket, odstraní z něj ESP hlavičku a výsledek předá k dalšímu zpracování.

Chcete-li znát podrobnosti o ESP, nahlédněte do RFC 4303: *IP Encapsulating Security Payload (ESP)*. Kryptografické algoritmy jsou definovány společně s AH v RFC 4835: *Cryptographic Algorithm Implementation Requirements for Encapsulating Security Payload (ESP) and Authentication Header (AH)*.

10.4 Správa bezpečnostních asociací

Nosným pilířem celého IPsec jsou bezpečnostní asociace. Ty určují, jaké algoritmy mají být nasazeny, s jakými parametry a s jakými klíči. Pokud neexistuje rozumný mechanismus pro jejich správu, je praktická použitelnost IPsec velmi problematická.

manuální konfigurace

Pochopitelně vždy je k dispozici primitivní možnost ruční konfigurace. Prostě si jednotlivé asociace nakonfigurujete odpovídajícími příkazy operačního systému. Takové schéma je zaručeně funkční, ale velmi omezeně použitelné. Má dvě významné nevýhody:

- Špatně škáluje. Ručně lze bez problémů nakonfigurovat šifrované tunely mezi několika filiálkami firmy, jak byly popsány výše. Na druhé straně je nemyslitelné postavit třeba šifrovaný přístup klientů k bankovnímu serveru na IPsec s manuální konfigurací. Klientských počítačů jsou haldy a neustále se mění. Udržovat ručně tohle klubko bezpečnostních asociací by pro správce serveru představovalo volňáska do Bohnic.
- Neumožňuje často měnit parametry (především klíče). Jedna z cest ke zvýšení bezpečnosti komunikace je častá změna klíčů. Pokud se vetřelci podaří některý z nich uhodnout či ukrást, poslouží mu jen pro malou část komunikace. Obě strany za chvíli přejdou na jiný klíč a komunikace se stane opět bezpečnou[4].

Čili je silně žádoucí, aby existovaly automatické mechanismy pro správu bezpečnostních asociací, které by byly schopny asociace vytvářet a rušit podle potřeby. Předchozí generace IPsec používala pro tento účel dvojici protokolů: *Internet Security Association and Key Management Protocol (ISAKMP)* definovaný v RFC 2408 vytvořil obecný rámec vzájemné dohody o parametrech bezpečnostních asociací, zatímco *Internet Key Exchange*

[4] Extrémní příklad krátkodobého používání klíčů představuje protokol TKIP pro šifrování ve Wi-Fi sítích, jenž mění klíč po každém paketu.

(IKE) verze 1 (RFC 2409) zajišťoval výměnu klíčů pro kryptografické algoritmy.

Současná generace vše nahradila jedním společným protokolem nazvaným IKEv2, který obsahuje kompletní funkce potřebné pro správu bezpečnostních asociací a nastavení jejich parametrů i používaných klíčů. Jeho definici najdete v RFC 4306: *Internet Key Exchange (IKEv2) Protocol*. Později byla doplněna vysvětlujícími komentáři v podobě RFC 4718: *IKEv2 Clarifications and Implementation Guidelines* a ještě později byly oba dokumenty shrnuty a dále vyjasněny v RFC 5996: *Internet Key Exchange Protocol Version 2 (IKEv2)*, který představuje aktuálně platnou specifikaci protokolu IKEv2.

10.4.1 IKEv2

Bezpečnostní asociace vyžaduje vzájemnou koordinaci obou stran – musí se dohodnout na používaných kryptografických algoritmech, jejich parametrech a na klíčích, které budou využívat. IKEv2 jim k tomu poskytuje prostředky.

Jeho komunikace probíhá v tak zvaných *výměnách (exchange)* zahrnujících vždy dvě zprávy – požadavek a odpověď. Pro přenos dat IKEv2 používá nezaručený protokol UDP, konkrétně porty 500 a 4500, a případné ztráty paketů si řeší sám. Jednoduchost výměn mu umožňuje dělat to zcela triviálně: pokud na dotaz nepřijde včas odpověď, tazatel jej prostě zopakuje. RFC 4306 definuje čtyři základní výměny. Dvě se používají při zahájení, jedna ke správě asociací a jedna pro vzájemnou výměnu informací.

kryptografické algoritmy

Dohoda o kryptografických algoritmech pro nově vytvářenou bezpečnostní asociaci využívá osvědčené schéma, kdy jedna strana navrhne možné alternativy a protějšek si z nich vybere. Jeden z partnerů pošle svému protějšku požadavek na vytvoření nové bezpečnostní asociace[5] obsahující sadu návrhů pro její zabezpečení, které jsou podle něj pro tento případ myslitelné. Teď to bude trochu složité, takže si to rozeberme:

- *Návrh (proposal)* je kompletní popis bezpečnostních mechanismů pro asociaci. Obsahuje jeden či několik protokolů (zpravidla ale jen jeden).
- *Protokol (protocol)* je IPsec protokol použitý pro zabezpečení. Tedy AH nebo ESP. Obsahuje jednu či několik transformací.
- *Transformace (transform)* definuje použitý kryptografický algoritmus. Skupina transformací dává protější straně na výběr, kterému dává přednost. Parametry transformace lze případně doplnit prostřednictvím atributů.

[5] Nebo změnu stávající, IKEv2 umožňuje například změnit používané klíče.

- *Atribut (attribute)* je nepovinný. Přidává se k transformaci jen v případě, kdy je třeba upřesnit některé její vlastnosti.

Řekněme, že iniciátor chce založit bezpečnostní asociaci chráněnou jak autentizací, tak šifrováním. Navrhne tedy dvě varianty: buď použití ESP s oběma funkcemi, nebo nasazení obou hlaviček AH a ESP, kde první se postará o autentizaci a druhá o šifrování. Jeho požadavek na vytvoření bezpečnostní asociace by obsahoval následující návrhy:

Návrh 1:

- protokol ESP
  - transformace AES-256, AES-128 a 3DES pro šifrování, AES-XCBC-MAC a HMAC-SHA1 pro autentizaci

Návrh 2:

- protokol AH
  - transformace AES-XCBC-MAC a HMAC-SHA1
- protokol ESP
  - transformace AES-256, AES-128 a 3DES

Protější strana si z návrhů vybere podle svých schopností a priorit ten, který považuje za optimální. Zvolí právě jeden návrh, který zašle ve své odpovědi. Do něj zařadí všechny jeho protokoly (protokoly v návrhu se mají použít současně, nejsou to alternativy) a pro každý vždy vybranou transformaci. Kdyby stroj oslovený předchozí žádostí považoval za ideální zajistit obě služby společnou hlavičkou ESP s použitím protokolů AES-256 pro šifrování a AES-XCBC-MAC pro autentizaci, zařadil by do své odpovědi návrh:

Návrh 1:

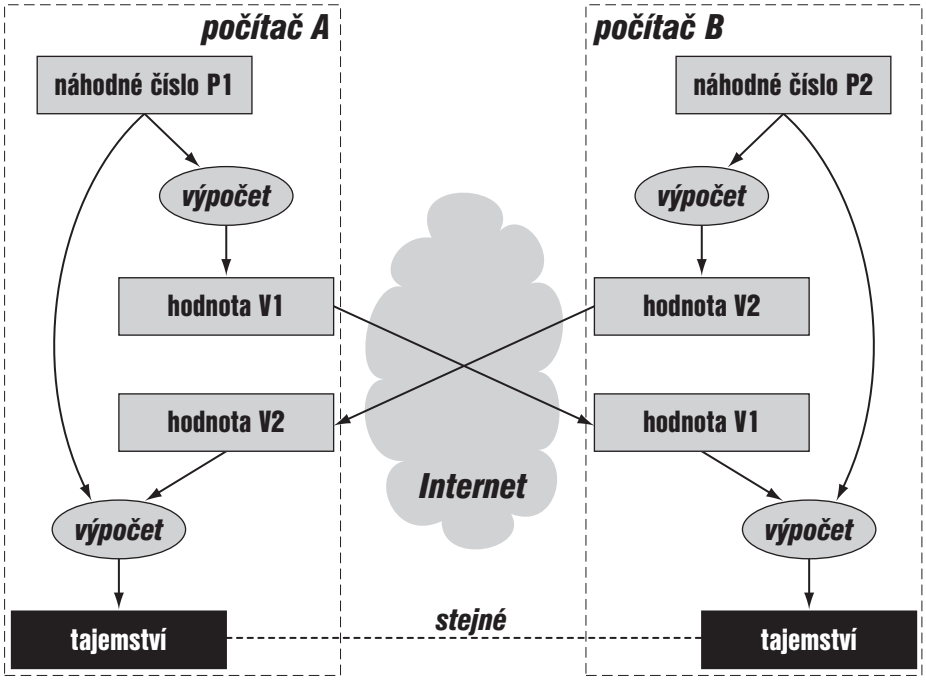
- protokol ESP
  - transformace AES-256 a AES-XCBC-MAC

klíče

Tím je vybráno, jakými mechanismy bude asociace chráněna. Složitější problém ale představují klíče. Pro ochranu datagramů využívá IPsec symetrické kryptografické algoritmy. Jsou mnohem rychlejší, ale vyžadují, aby obě strany používaly stejný klíč. Právě synchronizace klíčů v prostředí, kdy komunikační kanál může být odposlouchávána, představuje tvrdý oříšek.

Louskáček použitý v IKEv2 nese název Diffie-Hellmanův algoritmus. Je postaven na ošklivém triku: žádné klíče se Internetem nepřenášejí, generuje si je každý sám. Procedura začíná tím, že si každý z účastníků vygeneruje

náhodné číslo, které se stane jeho tajnou hodnotou. Z něj pak vypočítá druhou hodnotu, která je veřejná a kterou pošle svému protějšku. Na oplátku od něj dostane jeho veřejné číslo.



Obrázek 10.7: Diffie-Hellmanův algoritmus

Vybrané matematické vztahy zajišťují, že když oba dva účastníci zkombinují svou tajnou hodnotu s veřejnou hodnotou od svého protějšku, dostanou stejný výsledek. Ten se však nedá odvodit z dvojice veřejných hodnot (které někdo mohl odposlouchat). Oba tedy získali shodné tajné číslo, ze kterého následně odvozují klíče pro použité šifrovací algoritmy.

zahajovací fáze

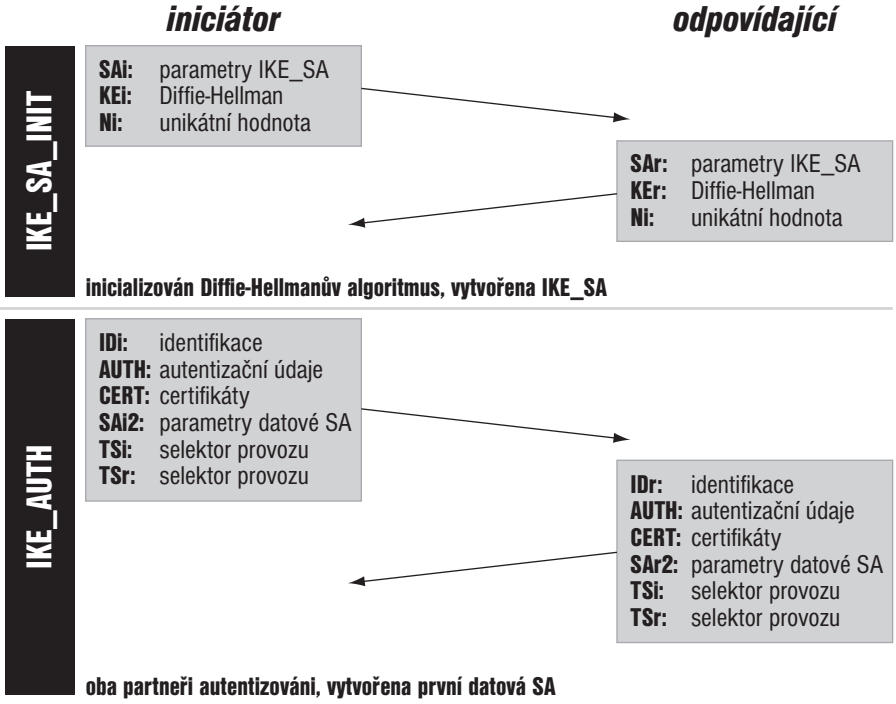
Do správy bezpečnostních asociací mezi dvojicí IPsec strojů nikomu nic není, proto IKEv2 své zprávy šifruje. Svou činnost zahajuje vytvořením jedné bezpečnostní asociace pro vlastní potřebu. Ta je v dokumentaci označována jako IKE_SA a slouží k šifrování a autentizaci zpráv samotného protokolu IKEv2. Pro asociace sloužící vlastnímu přenosu dat mezi IPsec partnery se používá termín potomci (child SA).

Když IKEv2 zahajuje svou činnost, musí nejprve vytvořit režijní IKE_SA. Úvodní fáze protokolu zahrnuje dvě výměny (tedy celkem čtyři zprávy), jejichž úkolem je:

1. vyměnit si veřejné hodnoty Diffie-Hellmanova algoritmu

2. vytvořit IKE_SA
3. ověřit vzájemně svou totožnost
4. vytvořit prvního SA potomka

Proces, který tyto činnosti obstará, je znázorněn na obrázku 10.8 (pro součásti zpráv používá zkratky podle tabulky 10.1). Zahajující výměna nese název IKE_SA_INIT a má na starosti první dva body. Iniciátor IKEv2 komunikace pošle protějšku svůj návrh parametrů pro režijní asociaci IKE_SA, svou veřejnou hodnotu pro Diffie-Hellmanův algoritmus a unikátní hodnotu, jež zabraňuje opakovanému vysílání dříve zachycených paketů.



Obrázek 10.8: Zahajovací fáze protokolu IKEv2

Protějšek[6] vybere ideální z navržených parametrů a pošle zpět odpověď obsahující vybraný návrh parametrů IKE_SA, jeho veřejnou hodnotu pro Diffie-Hellmanův algoritmus a opět unikátní hodnotu proti opakování.

V tomto okamžiku mají oba partneři dohodnuté parametry IKE_SA a znají vzájemně veřejné hodnoty Diffie-Hellmanova algoritmu. Mohou tedy vypočíst sdílené tajemství a z něj pak definovaným způsobem odvozují jednot-

[6] Pokud je ochoten s iniciátorem mluvit.

livé klíče pro kryptografické algoritmy. Stojí za zmínku, že pro každý směr komunikace používají odlišné klíče. Ostatně, IKE_SA jsou ve skutečnosti dvě bezpečnostní asociace – pro každý směr jedna, se stejnými parametry, ale odlišnými klíči. Od tohoto okamžiku jsou všechny další zprávy IKEv2 opatřeny IPsec ochranou podle IKE_SA.

Zahajovací úkoly jsou po první výměně splněny z poloviny. Zbývá ověřit totožnost zúčastněných (zatím se za protějšek mohl vydávat kdekdo) a vytvořit první asociaci pro data. To má za úkol výměna IKE_AUTH, jež následuje bezprostředně po IKE_SA_INIT. Tentokrát iniciátor ve svém požadavku pošle informaci o vlastní identitě a autentizační data, jimiž prokáže znalost tajného klíče pro danou identitu. Může přibalit i sadu certifikátů, umožňujících jejich ověření. Zároveň zařadí i návrh parametrů pro prvního SA potomka a může pomocí tak zvaných selektorů provozu (traffic selector) popsat pakety, na něž se má vztahovat. Selektory (například rozsahy adres odesilatele či příjemce) pak budou převzaty do databáze bezpečnostní politiky pro tuto asociaci.

Příjemce ověří autentizační údaje. Tento okruh problémů stojí poněkud stranou vlastního jádra IKEv2, proto se k němu vrátím až později. Dopadne-li autentizace úspěšně, pošle zpět odpověď, do níž zahrne analogické údaje o sobě (identitu, autentizační data, případně certifikáty) a vybere nejvhodnější návrh parametrů datové SA. Iniciátor opět ověří identitu svého partnera a v případě úspěchu považuje IKEv2 komunikaci za navázanou. Oba si vygenerují klíče pro datovou SA, kterou si přidají do databáze bezpečnostních asociací a začnou používat.

IKEv2 proti svému předchůdci znatelně zlepšil svou efektivitu. Po výměně pouhých čtyř zpráv je komunikace navázána, inicializován generátor klíčů, ověřena identita obou partnerů a založeny dvě bezpečnostní asociace – jedna režijní pro IKEv2, druhá pro běžná data.

další asociace

Kdykoli později je třeba navázat další bezpečnostní asociaci nebo změnit klíče u stávající, poslouží k tomu výměna typu CREATE_CHILD_SA. Může ji zahájit každý z partnerů, jejich práva jsou zcela rovnocenná. Do svého požadavku zařadí návrhy jejího zabezpečení, unikátní hodnotu proti opakování a případné selektory provozu. Protějšek odpoví vybraným návrhem, doplněným o unikátní hodnotu. Výsledkem je založení dvojice SA (po jedné v každém směru) s dohodnutými parametry a klíči vygenerovanými na základě Diffie-Hellmanova algoritmu.

CREATE_CHILD_SA poslouží i pro změnu klíčů již existující asociace. Formát je podobný, jen se do požadavku přidá identifikace asociace, jíž se změna týká.

zprávy

IKEv2 zprávy nemají pevný tvar. Protokol pouze definuje základní stavební bloky, tak zvané *obsahy (payloads)*, které nesou jednotlivé části informace. Z nich si pak odesilatel poskládá zprávu tak, jak potřebuje. Přehled defi-

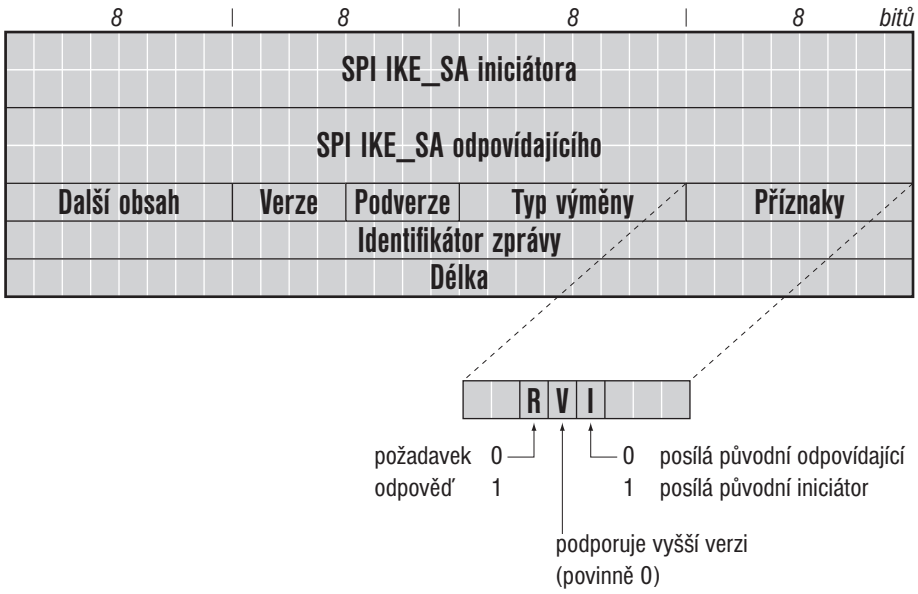
novaných obsahů uvádí tabulka 10.1. O jejich složení rozhoduje typ výměny a okolnosti, za nichž probíhá. Příklady si můžete prohlédnout na obrázku 10.8.

Kód	*Význam*	
0		nic
1–32		rezervováno (používaly starší verze)
33	SA	bezpečnostní asociace (popisuje situaci, ve které se bude používat)
34	KE	výměna klíčů (Diffie-Hellman)
35	IDi	identifikace iniciátora
36	IDr	identifikace odpovídajícího
37	CERT	certifikát
38	CERTREQ	žádost o certifikát
39	AUTH	autentizační data
40	Ni, Nr	unikátní data (nonce)
41	N	oznámení (chyby, informace)
42	D	vymazání (oznamuje vymazání SA z databáze)
43	V	identifikace výrobce/implementace (umožňuje experimentovat s rozšířeními)
44	TSi	selektor provozu – iniciátor (identifikuje datový tok, pro který má být SA použita)
45	TSr	selektor provozu – odpovídající
46	E	zašifrováno (zašifrované ostatní obsahy)
47	CP	konfigurace
48	EAP	autentizační data EAP
49–127		rezervováno
128–255		pro privátní využití

Tabulka 10.1: Typy obsahů definované v IKEv2

Jediným společným jmenovatelem všech zpráv IKEv2 je jednotná základní hlavička, kterou je zahájena každá z nich. Její formát vidíte na obrázku 10.9. Začíná dvojicí identifikátorů, které oba partneři přiřadili IKE_SA a podle nichž rozpoznají, ke kterému probíhajícímu IKEv2 dialogu zpráva patří. Důležitou roli hraje *Typ výměny (Exchange type)*, kterým se řídí jak možné složení obsahů, tak reakce příjemce při obdržení zprávy. Přehled dostupných typů uvádí tabulka 10.2.

Identifikátor zprávy (Message ID) má několik úloh. Především umožňuje párovat dotazy a odpovědi – odpověď vždy nese stejný identifikátor zprávy jako dotaz, který ji vyvolal. Dále lze podle něj rozpoznat opakované pakety při výpadcích a navíc slouží jako ochrana proti přehrávání dříve zachycené komunikace. Pokud dorazí zpráva se zcela nesmyslným identifikátorem, bude zahozena.



Obrázek 10.9: Hlavička IKEv2 zprávy

0–33	rezervováno
34	IKE_SA_INIT
35	IKE_AUTH
36	CREATE_CHILD_SA
37	INFORMATIONAL
38–239	rezervováno
240–255	pro privátní využití

Tabulka 10.2: Typy výměn IKEv2

Nesené obsahy jsou seskupeny za základní hlavičkou podle potřeby. Pro jejich identifikaci se používá podobné řetězení jako u rozšiřujících hlaviček: každý obsahuje typ obsahu, který následuje za ním.

10.4.2 Autentizace

Autentizace, čili ověření totožnosti je posledním otevřeným problémem. Řekněme, že se mi ohlásil počítač s adresou 1.2.3.4 a chce se mnou prostřednictvím IKEv2 navázat zabezpečenou komunikaci. Jak se dozvím, že se jedná skutečně o „ten pravý počítač 1.2.3.4“? Někdo si mohl nečistým trikem opatřit jeho IP adresu (např. využil jeho vypnutí, odpojil mu síť a připojil ji k sobě, sedí v síti mezi námi a procházející datagramy si mění podle libosti a podobně) a teď se snaží vystupovat jeho jménem.

Obecným řešením je opatřit data digitálním podpisem, který příjemce považuje za důvěryhodný. To znamená, že data byla podepsána soukromým klíčem, k němuž má příjemce klíč veřejný (a může si tudíž podpis ověřit) a věří, že skutečně patří počítači 1.2.3.4. Existují dvě základní cesty, jak získat důvěryhodný veřejný klíč: manuální konfigurace a certifikát.

Při manuální konfiguraci byl klíč přenesen nějakou jinou cestou (např. doručen osobně) a správce přijímajícího počítače jej uložil do konfigurace. Obvyklým problémem je škálovatelnost takového řešení. Můžete je použít ve firmě, kdy zaměstnaneckým počítačům instalujete veřejný klíč svého účetního serveru. Pokud byste však chtěli tímto způsobem řešit zabezpečení veřejného serveru, jehož klienti pocházejí odkudkoli, narazíte.

certifikát

Certifikáty jsou daleko zajímavější. Certifikát je v podstatě zpráva sdělující „Přísahám na holý pupek, že počítač s IP adresou 1.2.3.4 má veřejný klíč HyChyKyRyDyTyNy.“ a případné další údaje. Tato zpráva je opatřena digitálním podpisem instituce či osoby, která ji vydala. Instituce vydávající certifikáty jsou označovány jako *certifikační autority (CA)*.

Odesilatel tedy přibalí ke svým datům certifikát (oznamující jeho veřejný klíč) a opatří je digitálním podpisem vycházejícím z jeho soukromého klíče. Příjemce se podívá na certifikát a na autoritu, která jej vydala. Pokud dotyčné autoritě důvěřuje[7], ověří certifikát jejím veřejným klíčem. Vyhovuje-li, má k dispozici důvěryhodnou informaci o odesilatelově veřejném klíči. Jeho pomocí pak ověří digitální podpis ve zprávě. Jestliže je správný, pak zprávu skutečně poslal ten jedině pravý počítač 1.2.3.4.

Tento postup je docela dobře použitelný v omezeném dosahu. Například poskytovatel Internetu si vybuduje certifikační autoritu, které budou důvěřovat všichni jeho zákazníci a mohou tedy mezi sebou používat její cer-

[7] To znamená: důvěřuje způsobu vydávání certifikátů, který dotyčná autorita používá, a má k dispozici její veřejný klíč, aby je mohl ověřovat.

tifikáty. Problém vznikne, když potřebujete komunikovat s někým, jehož certifikát pochází od jiné autority.

veřejné klíče, PKI

Jeho řešení by měla nabídnout tak zvaná *infrastruktura veřejných klíčů (Public Key Infrastructure, PKI)*. Předpokládá, že z certifikačních autorit se vybuduje stromová hierarchie. Autority na vyšší úrovni pak certifikují ty, které jsou pod nimi. Například mi pošle data počítač a doprovodí je certifikátem vydaným CA podniku, kterému náleží. Tato certifikační autorita se prokáže certifikátem od CA jejího poskytovatele Internetu, která se prokáže certifikátem centrální autority pro Českou republiku. Této CA důvěřuji a mám její veřejný klíč, takže následně mohu postupně ověřit veřejné klíče v celém řetězci certifikátů a dojít k závěru, že zpráva je autentická. Sekvence certifikátů, která umožňuje postupně ověřit autentičnost požadovaného klíče, bývá označována jako *řetězec důvěry*. Stejný princip využívá i bezpečné objevování sousedů (SEND), jež navazující řetězec certifikátů pojmenovalo certifikační cesta.

Z uživatelského hlediska by bylo ideální, kdyby existovala jedna velká PKI s centrální autoritou. Její veřejný klíč by měl předinstalován každý operační systém a umožňoval by postupně ověřit cokoli. Bohužel „centrální certifikační autorita“ se v americké výslovnosti čte jako „velké peníze“ a v evropských jazycích jako „politický vliv“. Výsledkem je, že existuje celá řada certifikačních autorit[8] s omezenou působností.

Zajímavá alternativa se rodí v rámci pracovní skupiny *DNS-based Authentication of Named Entities (Dane)*, která směřuje k ukládání kryptografického materiálu do DNS. Zatím je však tato aktivita příliš mladá na konkrétní výsledky – pracovní skupina byla založena koncem roku 2010.

Do IKEv2 se tyto skutečnosti pochopitelně také promítají, konkrétně do obsahů výměny IKE_AUTH. Jejím cílem je ověřit vzájemně identitu komunikujících partnerů. Odesilatel požadavku či odpovědi vloží do zprávy jednak informaci o své identitě, jednak autentizační údaje (digitální podpis), které prokazují, že zná tajný klíč této identity. K ověření podpisu druhá strana potřebuje získat z důvěryhodného zdroje veřejný klíč příslušející identitě. Odesilatel proto může do zprávy přibalit i sekvenci certifikátů, jejichž prostřednictvím lze ověřit veřejný klíč podle některé z dobře známých autorit. Odesilatel požadavku navíc může informovat partnera, které CA považuje za důvěryhodné, a tím ovlivnit sekvenci certifikátů v odpovědi.

EAP

Špatná škálovatelnost certifikátů motivuje k využití alternativního autentizačního mechanismu. Přímo v definici IKEv2 je proto otevřena možnost použít k autentizaci *Extensible Authentication Protocol (EAP)*. Jedná se o obecnou skořápku definující základní formát zpráv, jejichž konkrétní obsah pak naplňují různé autentizační protokoly použité v kombinaci s EAP. V sou-

[8] Podívejte se do svého webového prohlížeče, kolik jich má přednastavených. Já jsem napočítal něco přes stovku.

časnosti se EAP běžně používá v bezdrátových sítích k ověření totožnosti uživatelů.

Výměna IKE_AUTH podle RFC 4306 může místo certifikátů použít EAP, ovšem pouze ve směru od iniciátora spojení k odpovídajícímu. Počet vyměněných zpráv pak narůstá, protože zájem o použití EAP iniciátor projeví tím, že do své zprávy zahajující IKE_AUTH nevloží autentizační data. Druhá strana ve své odpovědi pošle svá obvyklá autentizační data (a případné certifikáty) a pokud je schopna a ochotna použít k autentizaci EAP, indikuje to ve své odpovědi. Naopak zatím neposílá vybrané parametry SA a selektory provozu. Následuje výměna přinejmenším dvojice zpráv, kdy iniciátor pošle EAP data zabalená do IKEv2 a protějšek mu odpoví výsledkem EAP operace. Byla-li úspěšná, iniciátor se následně autentizuje, dostane kýžené parametry pro první datovou bezpečnostní asociaci a zahajovací fáze IKEv2 je tím dokončena.

Tento přístup ovšem zdaleka nevyužívá všech možností, které EAP dokáže nabídnout – například lze jeho prostřednictvím autentizovat obě komunikující strany. RFC 5998: *An Extension for EAP-Only Authentication in IKEv2* proto popsalo výraznější zapojení EAP, kdy mu IKEv2 svěřuje vzájemnou autentizaci a výměnu klíčů.

11 Mobilita

Mobilní telefony, mobilní počítače a přenosná či pojízdná zařízení všeho druhu cloumají naší přítomností. Rychle se šíří a nabývají na kvalitě i schopnostech, možnost komunikovat je pro ně životně důležitá. Je třeba uživatelům umožnit, aby si během cesty do Prahy vyřídili elektronickou poštu, přečetli svůj oblíbený WWW magazín či se podívali na aktuální kurzy akcií.

Podpora mobilních zařízení je v IPv6 velmi promyšlená a měla by hrát roli jednoho z významných trumfů při prosazování tohoto protokolu do praxe. Plány mobilních operátorů jdou do miliard připojených zařízení, což se s adresním prostorem IPv4 rozumně zvládnout nedá. Mechanismy pro podporu mobility byly sice vymyšleny i pro starší protokol, ale z hlediska efektivity a nabízených možností za IPv6 zřetelně zaostávají. Používá se pro ně označení *Mobile IPv6 (MIPv6)*.

Bohužel však mobilita zároveň představuje jedno z bolavých míst IPv6. Je krásná, úžasná, blýskavá a najdete ji v každém propagačním letáku nového protokolu. V praxi už daleko méně. Jednou z příčin je i mimořádně dlouhý vznik její specifikace. Když se po několika letech práce v roce 2001 konečně zdálo být RFC na spadnutí, shodila IETF se stolu navržený způsob zabezpečení a začínalo se znovu.

Dočkali jsme se v polovině roku 2004, kdy vyšlo RFC 3775: *Mobility Support in IPv6* a společně s ním i doprovodné RFC 3776: *Using IPsec to Protect Mobile IPv6 Signaling Between Mobile Nodes and Home Agents*. V roce 2011 byla základní specifikace aktualizována v RFC 6275, došlo ale jen ke kosmetickým změnám, které vyjasňovaly některé pasáže a reagovaly na změny, k nimž od roku 2004 v IPv6 došlo. Dodnes je bohužel stav podpory mobilního IPv6 dost neutěšený a zlepšuje se jen ppoommaalluu.

Příslušné dokumenty má na starosti pracovní skupina IETF *Mobility EXTensions for IPv6 (mext)* založená v roce 2007. Vedle nových standardů pracuje i na textech provozního charakteru.

11.1 Základní princip

Podpora mobility se opírá o základní myšlenku, že i pohyblivé zařízení je někde doma. Existuje pro ně tak zvaná domácí síť a v ní má registrovánu svou domácí adresu. Například notebook, na kterém píši tento text, je „doma“ v síti naší univerzity a z jejího adresního prostoru pochází i jeho domácí adresa.

adresy

Domácí adresa (home address) je neměnná a pod ní je stroj zaveden v DNS. Jejím prostřednictvím je také trvale dostupný – i když se zrovna nenachází

v domácí síti. Když počítač vyrazí na cesty, bude dostávat další, *dočasné adresy (care-of address, CoA)*.

Například když se posadíte do vlaku a připojíte se k Internetu prostřednictvím mobilního telefonu, notebooku bude přidělena adresa v rámci sítě daného mobilního operátora. Jak pojedete, bude váš mobil postupně přecházet od jedné BTS ke druhé. To může znamenat i přechod mezi jednotlivými podsítěmi mobilního operátora (a pokud vyjedete do zahraničí a změníte tak operátora, změní se adresa vašeho počítače od základu).

domácí agent

Aby byl mobilní stroj dosažitelný, ustaví si doma tak zvaného *domácího agenta (home agent)*. Jedná se o jeden ze směrovačů v domácí síti, který na sebe stahuje datagramy směřující k mobilnímu uzlu a předává mu je tunelem. Jakmile mobilnímu uzlu dorazí tunelovaný datagram od domácího agenta, dozví se z něj, že se jej někdo pokoušel kontaktovat na domácí adrese.

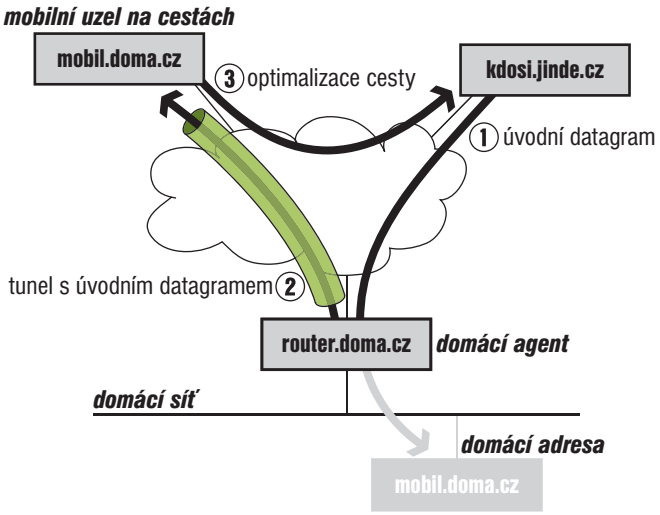
Mobilní IPv6 připouští i režim, kdy tímto způsobem probíhá veškerá komunikace s mobilním uzlem. Jeho partner, v terminologii mobilního IPv6 nazývaný korespondent, posílá data na domácí adresu, kde je sbírá domácí agent a předává tunelem mobilnímu uzlu. Ten ve svých odpovědích používá jako odesilatele svou domácí adresu, předává je tunelem domácímu agentovi, jenž je z domácí sítě odesílá korespondentovi. Cesta datagramů v tomto případě má samozřejmě k ideálu daleko, navíc hrozí přetížení připojení domácí sítě a domácího agenta. Tento režim se proto používá jen v případech, kdy korespondent nepodporuje mobilitu a nic lepšího nedovede.

optimalizace cesty

V normálních případech přichází ke slovu optimalizace cesty. Mobilní uzel ji zahájí okamžitě při příchodu prvního datagramu od nového korespondenta. Cílem optimalizace je seznámit korespondenta s aktuální dočasnou adresou mobilního uzlu, aby bylo možné další data posílat přímo. K ohlášení dočasné adresy slouží *Aktualizace vazby*. Než ji však může odeslat, musí si nejprve se svým novým partnerem vyřídit oficiality a prokázat, že je skutečně ten, za koho se vydává.

Když je vše hotovo a protější stroj přijme aktualizaci vazby, poznamená si informaci v ní obsaženou. Své další datagramy směřující k mobilnímu uzlu opatřuje rozšiřující hlavičkou *Směrování*. Jsou tedy finálně určeny pro domácí adresu mobilního uzlu, ale posílají se na adresu dočasnou. Díky tomu další komunikace mezi oběma stroji probíhá přímo, bez účasti domácího agenta. Navázání spojení s mobilním uzlem znázorňuje obrázek 11.1.

Domácí adresa je pevným bodem, za který můžete mobilní uzel kdykoli uchopit. Ji také používají protokoly vyšších vrstev. Z jejich pohledu je mobilita zcela transparentní a o nějakých dočasných adresách se nikdy nedozvědí.



Obrázek 11.1: Navázání spojení s mobilním uzlem

Vztah mezi domácí a aktuální adresou mobilního uzlu se nazývá *vazba (binding)*. Jsou definovány metody, které se snaží, aby všichni zúčastnění znali aktuální vazbu pro mobilní uzel, s nímž komunikují.

11.2 Hlavičky a volby

Než se pustím do podrobnějšího popisu jednotlivých mechanismů, představím vám formát zpráv, kterými se domlouvají jednotliví účastníci. Je to poněkud komplikované, protože zasahují do různých částí IPv6. RFC 3775 zavedlo novou hlavičku *Mobilita*, jejímž prostřednictvím především spravuje vazby a ověřuje důvěryhodnost mobilního uzlu při optimalizaci cesty. Dále definuje nový typ hlavičky *Směrování* a novou volbu pro příjemce *Domácí adresa*. K některým dalším účelům využívá protokol ICMP, jemuž doplňuje čtyři nové typy zpráv. Jako třešničku na dortu pak lehce rozšiřuje ohlášení směrovače. Pokusil jsem se vám trochu usnadnit orientaci tabulkou11.1, kde najdete přehled základních zpráv využívaných při komunikaci s mobilními zařízeními.

hlavička Mobilita

Klíčovým prvkem je správa vazeb, kde všechnu práci zastane nová rozšiřující IPv6 hlavička *Mobilita (Mobility)*. Je identifikována hodnotou 135 v položce *Další hlavička* své předchůdkyně. Její formát představuje obrázek 11.2.

Protokol obsahu (Payload protocol) identifikuje obsah následující za touto hlavičkou – typ následující hlavičky nebo protokol, kterému patří data. Není

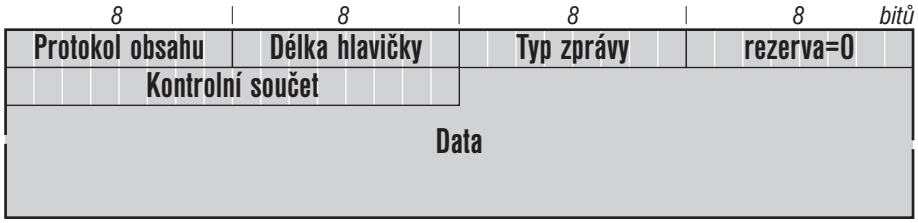
Hlavička Mobilita
- test domácí adresy
- test dočasné adresy
- správa vazeb

Volby pro příjemce
- domácí adresa

Hlavička Směrování typu 2
- doručování dat mobilnímu uzlu

ICMPv6
- objevování adresy domácí agenta
- objevování mobilních prefixů

Tabulka 11.1: Přehled zpráv pro podporu mobility



Obrázek 11.2: Rozšiřující hlavička *Mobilita*

mi jasné, proč se autoři odchýlili od standardního názvu „další hlavička“, ale znamená de facto totéž.

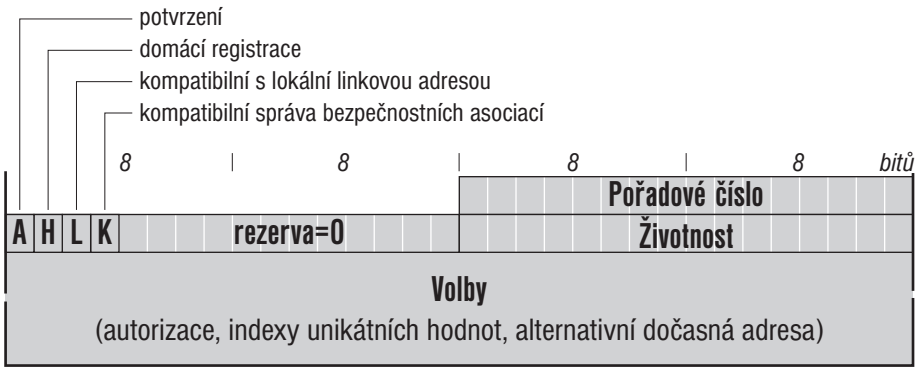
Nejvýznamnější položkou je *Typ zprávy (MH type)*, který určuje, jaká zpráva je touto hlavičkou přenášena. Na její hodnotě závisí struktura nesených *Dat (Message data)*. Definované typy zpráv shrnuje tabulka 11.2, do níž jsem zařadil i typy definované pozdějšími specifikacemi. Jejich formáty začínají vždy určitou pevnou částí, za níž pak následují volby. Každá volba má svůj vlastní formát, platný pro všechny zprávy, v nichž se může vyskytnout. *Typ zprávy* společně se situací odesilatele rozhoduje, které volby budou do zprávy zařazeny. V obrázcích znázorňujících formát jednotlivých typů zpráv vždy uvedu přípustné volby. Nenechte se zaskočit „zubatým“ horním okrajem zpráv. Tvoří obsah hlavičky *Mobilita* a pokračují tedy za jejím *Kontrolním součtem*.

aktualizace vazby

Klíčovou roli hraje *Aktualizace vazby* (obrázek 11.3). Její základní funkcí je oznámit aktuální adresu mobilního uzlu – buď partnerskému počítači nebo domácímu agentovi.

Typ	*Význam*
0	žádost o obnovení vazby (binding refresh request)
1	zahájení testu domácí adresy (home test init)
2	zahájení testu dočasné adresy (care-of test init)
3	test domácí adresy (home test)
4	test dočasné adresy (care-of test)
5	aktualizace vazby (binding update)
6	potvrzení vazby (binding acknowledgement)
7	chyba vazby (binding error)
8	rychlá aktualizace vazby (fast binding update), RFC 5568
9	rychlé potvrzení vazby (fast binding acknowledgement)
10	rychlé ohlášení souseda, zavrženo
11	experimentální hlavička, RFC 5096
12	změna domácího agenta (home agent switch), RFC 5142
13	kontrola spojení (heartbeat), RFC 5847
14	zahájení předávky (handover initiate), RFC 5568
15	potvrzení předávky (handover acknowledge), RFC 5568
16	zrušení vazby (binding revocation), RFC 5846

Tabulka 11.2: Typy zpráv pro hlavičku *Mobilita*



Obrázek 11.3: Zpráva *Aktualizace vazby*

V pevné části svého formátu obsahuje dvojici položek *Pořadové číslo (Sequence #)*, *Životnost (Lifetime)* a sadu příznaků. Obě položky představují reakci na skutečnost, že dočasné adresy se postupem času mění. Proto musí být každá aktualizace vazby opatřena pořadovým číslem, které musí být větší než číslo poslední aktualizace směřující ke stejnému cíli[1]. Díky tomu se nestane, že by se cílový počítač díky zpoždění datagramu vrátil k předchozí (teď již neplatné) dočasné adrese. *Životnost* pak stanoví dobu platnosti dočasné adresy v sekundách. Vyprší-li, musí být vazba považována za neplatnou.

Možná vás překvapuje, že aktualizace v sobě neobsahuje žádné adresy[2]. Ty se doplní z ostatních částí datagramu. *Aktualizaci vazby* mobilní uzel posílá ze své dočasné adresy. Příjemce si ji proto vyzvedne ze *Zdrojové adresy* v základní IPv6 hlavičce. Domácí adresu, ke které se vazba vztahuje, pak oznámí volbou pro příjemce *Domácí adresa*.

Čtveřice příznaků na začátku druhého řádku ovlivňuje příjemcovo chování. Příznak *A* (acknowledge) znamená, že mobilní uzel požaduje od svého protějšku potvrzení vazby. Je-li nastaven příznak *H* (home registration), představuje hlavička žádost, aby se její příjemce stal domácím agentem mobilního uzlu. V tom případě se jedná o datagram sloužící k ustanovení domácího agenta. Adresátem musí být směrovač sídlící v domácí síti mobilního uzlu.

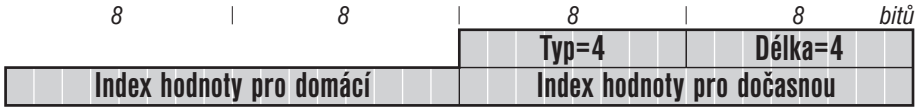
Pokud má domácí adresa mobilního uzlu stejný identifikátor rozhraní jako jeho lokální linková adresa, nastaví v aktualizaci vazby příznak *L* (link-local address compatibility). Je to signál pro domácího agenta, zda se má starat o obě adresy, nebo jen o jednu.

Příznak *K* (key management mobility capability) má smysl jen u aktualizací posílaných domácímu agentovi. Sděluje mu, že mobilní uzel podporuje protokol pro dynamickou správu bezpečnostních asociací pro IPsec, kterému nevadí přesuny uzlu v síti (čili IKEv2). Při manuální konfiguraci bezpečnostních prvků musí být vynulován. Problematiku použití protokolu IKEv2 pro správu bezpečnostních asociací v mobilitě řeší RFC 4877: *Mobile IPv6 Operation with IKEv2 and the Revised IPsec Architecture*.

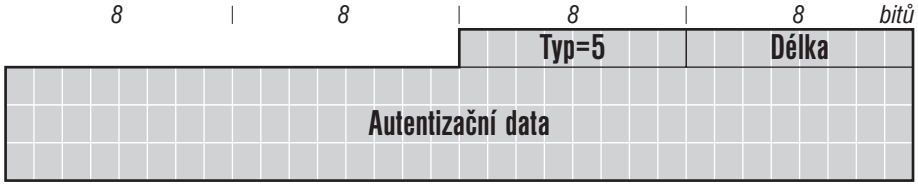
volby

Volby umožňují připojit k aktualizaci doplňující informace, které nejsou potřeba vždy. Například když mobilní uzel posílá aktualizaci běžnému stroji, musí prokázat svou identitu. Proto připojí *Autorizační data (Authorization Data)*. V takovém případě je třeba připojit i *Indexy unikátních hodnot (Nonce Indices)*. Pomocí *Alternativní dočasné adresy (Alternate Care-of Address)* lze předepsat, že ve vazbě se nemá použít zdrojová adresa ze základní IPv6 hlavičky, ale adresa zde uvedená.

[1] Modulo 65 536 – položka má omezenou kapacitu.
[2] Alternativní dočasná adresa ve volbách je nepovinná.



Obrázek 11.4: Volba *Indexy unikátních hodnot*



Obrázek 11.5: Volba *Autorizační data*

potvrzení vazby

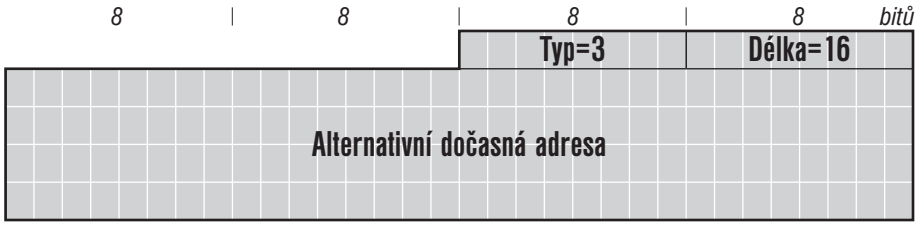
Jako reakce na aktualizaci vazby slouží *Potvrzení vazby* (viz obrázek 11.7). Jeho odeslání je povinné, pokud aktualizace obsahovala příznak *A* (žádost o potvrzení) nebo *H* (žádost o domácí registraci) a zpravidla také při odmítnutí, kdy poskytuje odesilateli aktualizace informaci o důvodech jeho neúspěchu.

Nejvýznamnější položkou je *Stav (Status)*. Obsahuje informaci o tom, zda aktualizace byla akceptována či nikoli. Konkrétní kódy najdete v tabulce 11.3. Obecně platí, že hodnota menší než 128 znamená, že aktualizace byla přijata, zatímco hodnoty od 128 výše představují odmítnutí. Díky *Pořadovému číslu (Sequence #)* mobilní uzel pozná, ke které aktualizaci se toto potvrzení vztahuje.

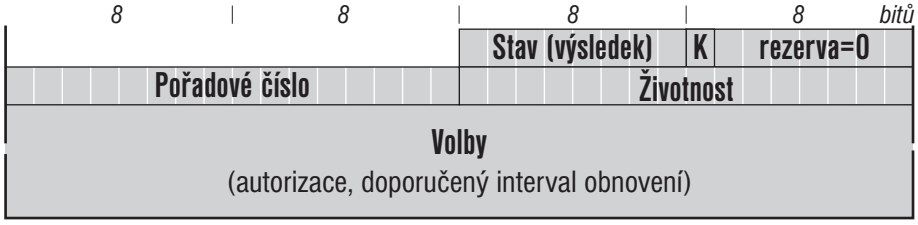
Položka *Životnost (Lifetime)* udává dobu v sekundách, po kterou si příjemce zaručeně podrží aktualizaci vazby ve svých datových strukturách. Pokud potvrzuje žádost o domácího agenta, znamená to současně, že minimálně po tuto dobu bude mobilnímu uzlu dělat domácího agenta. Zároveň může volbou *Doporučený interval obnovení (Binding refresh advice)* doporučit kratší interval, po jehož uplynutí má mobilní uzel znovu aktualizovat vazbu a prodloužit tak služby svého domácího agenta. Příznakem *K* sdělí, zda se bude používat dynamický protokol pro správu bezpečnostních asociací. Potvrzení vazby od komunikačního partnera musí být provázeno volbou *Autorizační data*, který umožní ověřit jeho platnost a autentičnost.

žádost o obnovení vazby

Aktualizaci vazby si lze vyžádat prostřednictvím zprávy *Žádost o obnovení vazby*. Po této možnosti sáhne partner mobilního uzlu, když se platnost vazby blíží svému konci, ale komunikace dosud probíhá. Může samozřejmě nechat vazbu projít, začít posílat data na domácí adresu bez optimalizace a sjet si s mobilním uzlem znovu stejnou proceduru jako na začátku. Je ale výhodnější raději požádat o obnovení vazby předem a vyhnout se tak zbytečnému předávání několika datagramů přes domácího agenta.



Obrázek 11.6: Volba *Alternativní dočasná adresa*



Obrázek 11.7: Zpráva *Potvrzení vazby*

0	akceptováno
1	akceptováno, ale je třeba objevit prefix
128	odmítnuto bez uvedení příčiny
129	zakázáno správcem
130	nedostačující zdroje
131	domácí registrace není podporována
132	není domácí podsíť
133	není domácí agent pro tento uzel
134	selhala detekce duplicitní adresy
135	chybné pořadové číslo
136	prošlý index hodnoty pro domácí adresu
137	prošlý index hodnoty pro dočasnou adresu
138	prošlé unikátní hodnoty
139	nedovolená změna typu registrace
174	neplatná dočasná adresa

Tabulka 11.3: Hodnoty položky *Stav*

Žádost o obnovení vazby sama o sobě neobsahuje žádná data. Mobilní uzel má po jejím příchodu na výběr tři alternativy: Chce-li vazbu zachovat, zahájí proces optimalizace cesty (a svého ověření), který vyústí její aktualizací. Pokud vazbu nehodlá dále udržovat, může buď poslat zpět ohlášení vazby s nulovou životností a tak ji explicitně zrušit, nebo si žádosti jednoduše nevšímat a nechat vazbu propadnout. Mimochodem, zrušit vazbu odesláním aktualizace s nulovou životností může mobilní uzel kdykoli během komunikace, pokud to z nějakého důvodu uzná za žádoucí.

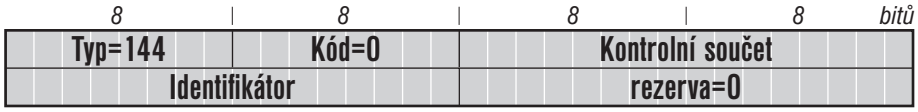
11.3 Získání domácího agenta

Když mobilní uzel vyrazí na cesty, musí ve své domácí síti získat domácího agenta, aby jej zastupoval. Tuto roli může hrát libovolný ze zdejších směrovačů, pokud je k tomu konfigurován a má dostatek volných prostředků. Jelikož se situace může průběhem času měnit, byl navržen postup, kterým si mobilní uzel svého agenta dynamicky vyhledá. Odpadá tak nutnost statické konfigurace domácích agentů a její údržby ve všech mobilních strojích.

Získání domácího agenta tudíž probíhá ve dvou fázích. V první se mobilní uzel dozví adresy všech potenciálních domácích agentů a ve druhé se domluví s jedním z nich, že pro něj bude tuto funkci skutečně vykonávat. Plné znění algoritmu, jehož cílem je nalezení vhodného domácího agenta, zní *dynamické objevování adresy domácího agenta (dynamic home agent address discovery)*. Mobilní uzel jej může, ale nemusí používat.

hledání potenciálních agentů

Zahájí jej odesláním ICMP zprávy *Žádost o adresy domácích agentů*, jejíž tvar vidíte na obrázku 11.8. Tuto zprávu zašle na výběrovou (anycast) adresu domácích agentů v síti. Definuje ji RFC 2526: *Reserved IPv6 Subnet Anycast Addresses*. Vypadá tak, že v horní polovině obsahuje prefix domácí sítě mobilního uzlu, zatímco spodních 64 bitů obsahuje hodnotu fdff:ffff:ffff:fffe.

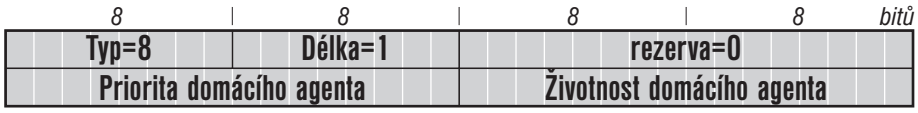


Obrázek 11.8: ICMP zpráva *Žádost o adresy domácích agentů*

Žádost dodržuje standardní tvar ICMP zpráv. Jako data obsahuje jen *Identifikátor (Identifier)*, jehož prostřednictvím lze rozpoznat příslušnou odpověď. Jelikož se jedná o výběrovou adresu, bude doručena jednomu z domácích agentů dotyčné sítě.

Každý ze směrovačů, které pracují jako domácí agenti, zná své kolegy. Mobilita rozšiřuje *Ohlášení směrovače* o několik prvků, sloužících tomuto účelu. V první řadě se jedná o příznak *H*, kterým směrovač sděluje „Jsem ochoten být domácím agentem pro stroje na této lince.“ Kromě toho pak přidává

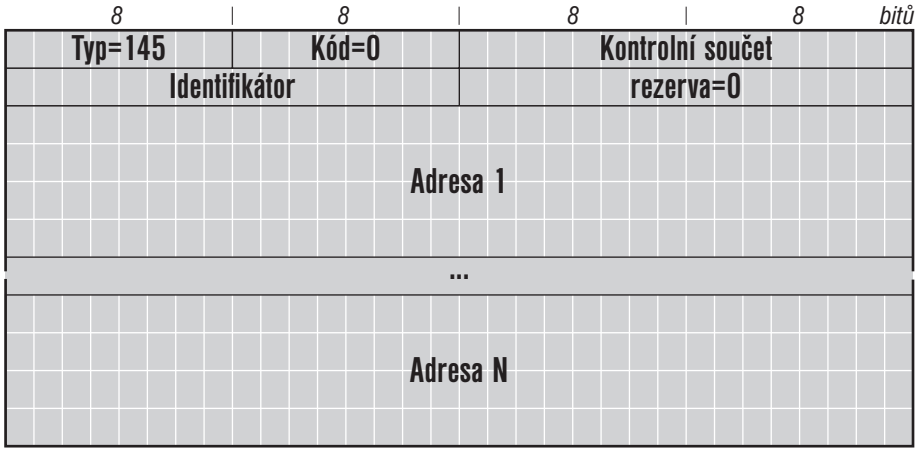
ke svému ohlášení volbu *Informace o domácím agentovi (Home Agent Information)*, jejímž prostřednictvím ohlásí *Prioritu (Home Agent Preference)* nastavenou správcem a *Životnost (Home Agent Lifetime)*, tedy jak dlouho je ochoten tuto činnost vykonávat. Formát volby s informacemi o domácím agentovi vidíte na obrázku 11.9.



Obrázek 11.9: Volba *Informace o domácím agentovi* v *Ohlášení směrovače*

Každý domácí agent je povinen si udržovat datovou strukturu nazvanou *seznam domácích agentů*. V ní si na základě ohlášení ostatních udržuje přehled o všech domácích agentech na dané lince.

Z těchto dat sestaví ICMP zprávu *Odpověď na objevování adresy domácího agenta (Home Agent Address Discovery Reply)*, kterou pošle odesilateli žádosti. Seznam by měl uspořádat sestupně podle priority jednotlivých agentů. Pokud se u některých priorita shoduje, uvádí je v náhodném pořadí, aby se zátěž rozkládala rovnoměrně. Zároveň jejich seznam zkrátí tak, aby se vešel do jedné zprávy. V reálném životě ale lze očekávat, že na lince bude jeden, nanejvýš několik málo domácích agentů.



Obrázek 11.10: ICMP zpráva *Odpověď s adresami domácích agentů*

dohoda s agentem

Mobilní uzel tedy získal seznam potenciálních domácích agentů a zbývá se s jedním dohodnout. Vybere uchazeče s nejvyšší prioritou a zašle mu *Aktualizaci vazby* s nastaveným příznakem *H* – buď mým domácím agentem!

Tento krok se nazývá registrace primární (momentálně používané) dočasné adresy.

Pokud dotyčný domácí agent nemá nic proti, zkontroluje pomocí detekce duplicit (je popsána na straně 122), zda domácí adresu někdo místní nepoužívá. Má-li aktualizace vazby nastaven příznak *L*, provede totéž s lokální linkovou adresou odvozenou z identifikátoru rozhraní domácí adresy. Když testy dopadnou dobře, zanese si mobilní uzel do svých datových struktur a pošle mu kladné *Potvrzení vazby*. Od tohoto okamžiku pracuje jako domácí agent mobilního uzlu, a to minimálně po dobu uvedenou v potvrzení.

Jakmile se stane domácím agentem, rozešle do domácí sítě několik nevyžádaných *Ohlášení souseda*, v nichž uvádí domácí IP adresu mobilního uzlu a svou vlastní linkovou adresu[3]. Také bude od tohoto okamžiku odpovídat na výzvy sousedovi s IP adresou mobilního uzlu. Díky tomu budou datagramy adresované na domácí adresu mobilního uzlu předávány jemu.

Blíží-li se vypršení doby, po kterou domácí agent potvrdil své fungování, je na mobilním uzlu, aby zaslal další žádost o registraci primární adresy a pokusil se tak prodloužit svůj vztah s domácím agentem. Pokud by byl neúspěšný, může spolupráci rozvázat (viz níže), zopakovat dynamické hledání domácího agenta a dohodnout se s jiným strojem.

zabezpečení

Jedním z nebezpečí mobility je, že se ošklivý počítač zaregistruje u domácího agenta a bude předstírat, že je některý ze zdejších strojů na cestách. Tím by na sebe přesměroval provoz určený pro dotyčný stroj a získal by neoprávněný přístup k cizím datům.

Proto je třeba domácí registraci chránit. Jelikož lze předpokládat, že domácí agent a počítače z jeho sítě se znají, není problém použít zde klasické IPsec, konkrétně ESP se zapnutou autentizací. Distribuce klíčů (přinejhorším statických) je v lokální síti jistě vyřešena a proto by neměl být s nasazením těchto prostředků problém.

změny adres v domácí síti

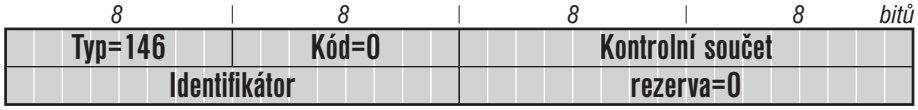
„Během mé přítomnosti se nic zvláštního nestalo“ zní standardní vojenské hlášení a něco podobného ohledně adres ohlašuje domácí agent. Pokud se snad náhodou stalo a v domácí síti došlo k mimořádce, tedy ke změně adres, mobilní mechanismy pamatují i na to. Když došlo ke změně ještě dříve, než se ozval mobilní agent z cest, zařadí do své domácí registrace neplatnou adresu[4]. Domácí agent registraci sice potvrdí, ale použije v *Potvrzení vazby* stavový kód 1. Jím žádá mobilní uzel, aby si vyhledal prefixy.

Mobilní uzel reaguje ICMP zprávou *Žádost o mobilní prefix (Mobile Prefix Solicitation)*, kterou domácího agenta žádá „Řekni mi, jak to tedy s prefixy mé

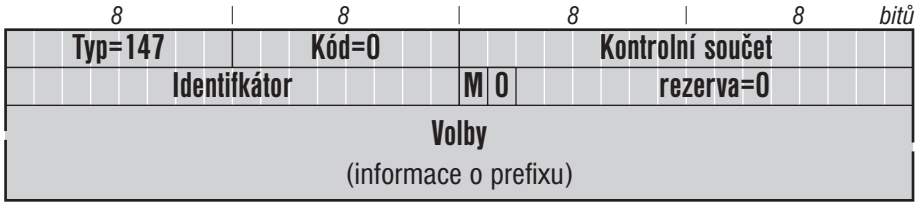
[3] Pokud byl v aktualizaci vazby nastaven příznak *L*, provede totéž i s lokální linkovou adresou mobilního uzlu.

[4] Předpokládám, že směrování pro původní prefix sítě zůstalo zachováno. V opačném případě nemá mobilní uzel jak se se svým domácím agentem spojit a stává se z něj bezdomovec.

milené domácí sítě vypadá.“ Odměnou mu bude *Ohlášení mobilního prefixu (Mobile Prefix Advertisement)*. Formát obou zpráv vidíte na obrázcích 11.11 a 11.12. Mají společný *Identifikátor (Identifier)*, podle nějž si mobilní uzel přiřadí agentovu odpověď ke své výzvě.



Obrázek 11.11: ICMP zpráva *Žádost o mobilní prefix*



Obrázek 11.12: ICMP zpráva *Ohlášení mobilního prefixu*

Klíčová informace v odpovědi – platné globální prefixy domácí sítě – je vysunuta do voleb. Musí obsahovat alespoň jednu volbu *Informace o prefixu (Prefix information)*. Jedná se o stejnou volbu, která se používá v ohlášení směrovače. Její formát jste mohli vidět na obrázku 6.3 na straně 121. K prefixům přidá domácí agent ještě dva příznaky informující o metodách automatické konfigurace v domácí síti. Příznak *M (Managed)* znamená, že zdejší stroje používají kromě bezstavové také stavovou konfiguraci (DHCPv6) a příznak *O (Other)*, že si stavově konfigurují ostatní informace, nikoli adresy.

Domácí agent posílá *Ohlášení mobilního prefixu* i bez výzvy. Opakuje je v určitých intervalech a navíc je zašle pokaždé, když v domácí síti dojde k významnější změně. Mobilní agent je tedy průběžně informován o tom, jaké adresy se doma používají.

Připomínám, že veškerá komunikace mezi mobilním uzlem a jeho domácím agentem je chráněna IPsec, konkrétně hlavičkou ESP. Na informace z tohoto zdroje tedy je spolehnutí.

11.4 Optimalizace cesty

Bezpečnostní problém (možnost prohlásit se za jiný uzel na cestách a zcizit jeho provoz) provází i vytvoření vazby u komunikačního partnera (korespondenta). Tentokrát je však situace složitější, protože partnerem mobilního uzlu se může stát libovolný počítač. To činí IPsec v podstatě nepou-

žitelným, protože celosvětová důvěryhodná distribuční síť klíčů se zatím vyskytuje jen v divokých snech bezpečnostních expertů.

Autoři mobilního IPv6 proto museli připravit vlastní návrh ověření totožnosti mobilního uzlu. Zahrnuje i ověření dočasné adresy, aby někdo nemohl delegovat svá data jinému počítači, když by prohlásil, že teď vlastně cestuje a má cizí adresu. Tato odrůda falšování by se dala využít k distribuovaným zahlcujícím útokům (DDoS).

IPv6 přichází s metodou, jak ověřit, že mobilní uzel skutečně poslouchá na domácí i dočasné adrese, které uvedl ve své *Aktualizaci vazby*. Nazývá se *zpětná směrovatelnost (return routability, RR)*. Nejprve se však podívejme, co jí předchází.

zahájení komunikace

Když někdo odesílá data počítači, o jehož mobilitě zatím nemá tušení, přijde ke slovu výše naznačený proces. Datagram je zaslán na domácí adresu a tudíž dorazí do domácí sítě mobilního uzlu. Směrovač, který jej přijme, se začne prostřednictvím objevování sousedů shánět po fyzické adrese cílového stroje. Místo něj však odpoví jeho domácí agent a datagram se tak dostane k němu.

Následuje doprava tunelem chráněným ESP od domácího agenta k mobilnímu uzlu. Domácí agent tedy vytvoří nový IPv6 datagram, do nějž zabalí ten původní jako nesená data. Odesilatelem obalujícího datagramu bude domácí agent, cílem bude dočasná adresa mobilního uzlu.

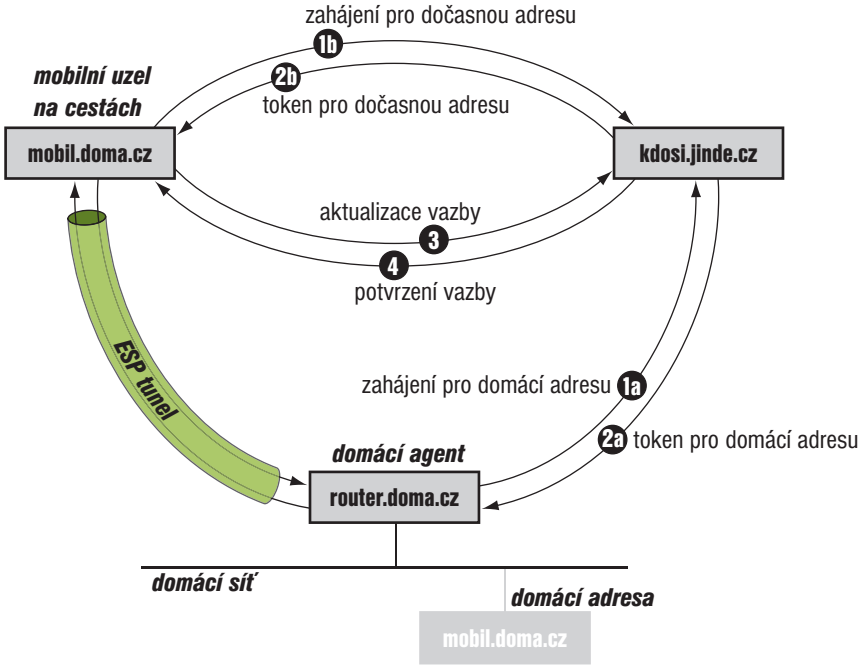
zahájení testu

Když mobilní uzel obdrží takto tunelovaný datagram, ví, že odesilatel netuší o jeho aktuální adrese. Proto se jej pokusí informovat. Nejprve však musí prokázat, že je skutečně ten pravý. Pošle dva navzájem nezávislé datagramy: *Zahájení testu domácí adresy (Home Test Init)* a *Zahájení testu dočasné adresy (Care-of Test Init)*. Jedná se o zprávy přenášené v hlavičce *Mobilita*, jejichž formát je prakticky totožný (vidíte jej na obrázku 11.14. Každá z nich obsahuje jinou náhodnou hodnotu, pojmenovanou *Cookie*. Korespondent ji zkopíruje do své odpovědi, aby prokázal, že skutečně reaguje na zahájení testu vyvolané mobilním uzlem.

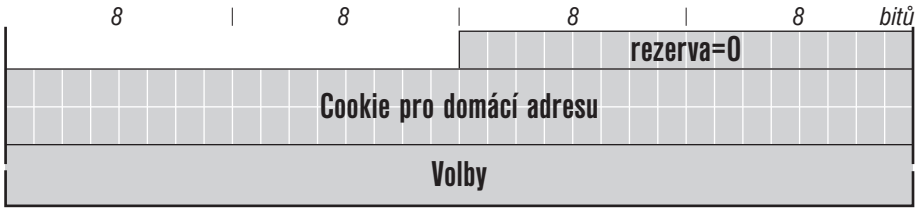
Zahájení testu domácí adresy mobilní uzel odešle v datagramu, kde jako odesilatele uvede svou domácí adresu. Pošle je tunelem domácímu agentovi, který je rozbalí a přepošle korespondentovi. Tunelování je nutné, protože dnes řada sítí používá filtrování a nedovolí ze sítě odeslat datagramy, jejichž zdrojová adresa neleží v dané síti. *Zahájení testu dočasné adresy* naproti tomu obsahuje jako odesilatele dočasnou adresu mobilního uzlu a posílá se rovnou komunikačnímu partnerovi.

partnerova odpověď

Ten na příchod obou zpráv reaguje odesláním speciální hodnoty, označované pro změnu *Token*. K jejímu výpočtu využívá několik hodnot. Především si každý IPv6 uzel podporující mobilitu musí udržovat soukromý



Obrázek 11.13: Autentizace mobilního uzlu při aktualizaci vazby

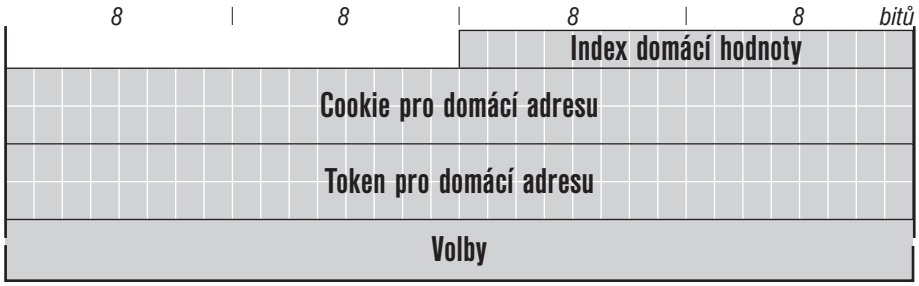


Obrázek 11.14: Zpráva *Zahájení testu domácí adresy*

klíč *K*. Ten se nikam nezasílá, takže není třeba zajišťovat distribuci. Může se také v čase měnit.

Vedle něj si řádově v několikaminutových intervalech generuje náhodné unikátní hodnoty, označované v angličtině jako *nonce*. Jelikož se tyto hodnoty poměrně rychle mění, jsou opatřeny pořadovými čísly (indexy). Unikátní hodnota s pořadovým číslem *i* se označuje jako *N(i)*. Uzel je povinen uchovávat si několik posledních unikátních hodnot, jejichž platnost dosud nevypršela[5].

Když dorazí *Zahájení testu domácí adresy*, vybere z něj domácí adresu, připojí k ní aktuální hodnotu *N(i)* a konstantu 0 a to celé předloží hešovací funkci HMAC SHA1 se soukromým klíčem *K*. Prvních 64 bitů jejího výsledku tvoří *Token pro domácí adresu (Home Keygen Token)*. Počítač je zabalí do zprávy *Test domácí adresy*, přidá index použité unikátní hodnoty *i* v položce *Index domácí hodnoty (Home nonce index)* a odešle na domácí adresu mobilního uzlu. Formát testu domácí adresy vidíte na obrázku 11.15. Domácí agent předá zprávu mobilnímu uzlu (je povinen ji šifrovat pomocí ESP), který tak získá první token.



Obrázek 11.15: Zpráva *Test domácí adresy*

Druhý token počítá korespondent úplně stejně, jen k výpočtu použije dočasnou adresu, kterou dostal v *Zahájení testu dočasné adresy*. Jelikož tato zpráva cestuje jinudy, dorazí o chvilku dříve či později než výzva k testu domácí adresy. Proto může teoreticky použít jinou unikátní hodnotu *N(j)*, pokud došlo k její změně. Token vzniklý z dočasné adresy, *N(j)* a konstanty 1 pošle společně s *j* ve zprávě *Test dočasné adresy* na dočasnou adresu mobilního uzlu.

Do tohoto okamžiku si korespondent pro daný mobilní uzel nic specifického neukládá. Má jen několik málo svých uložených unikátních hodnot, které jsou pro všechny stejné a spotřebují jen minimum prostředků.

aktualizace vazby

Jestliže mobilní uzel nelhal a uvedl své skutečné adresy, dostal oba tokeny.

[5] Maximální doba platnosti nonce je dána konstantou MAX_NONCE_LIFETIME, jejíž hodnotu RFC 6275 stanoví na 240 sekund.

Z nich definovaným způsobem vypočítá klíč pro *Aktualizaci vazby* a konečně ji může poslat. Pomocí HMAC SHA1 s tímto klíčem vypočítá autentizační hodnotu (určitou formu digitálního podpisu) pro aktualizaci vazby a přidá ji do volby *Autorizační data*[6] (obrázek 11.5 na straně 229). Zároveň přibalí volbu *Indexy unikátních hodnot*, aby protější uzel věděl, jaké náhodné hodnoty použít.

Aktualizaci vazby již posílá přímo. Protější stroj si podle indexů vyzvedne ze své paměti odpovídající hodnoty *N(i)*, *N(j)* a zopakuje celý výpočet – spočítá si oba tokeny, z nich klíč pro aktualizaci a s jeho pomocí pak autentizační data. Výsledek porovná s hodnotou obsaženou v aktualizaci a pokud se shodnou, zajásá, protože mobilní uzel prokázal znalost obou tokenů. To dokazuje, že skutečně sídlí na obou adresách, které uvedl. Protější uzel teď může otřít pot s čela a zaznamenat si, že daný mobilní uzel má takovou a makovou dočasnou adresu.

O co by byl ten svět jednodušší, kdybychom si mohli důvěřovat...

Popsaný test není stoprocentní. Dokáže jej padělat stroj, jímž procházejí datagramy mezi korespondentem a domácí i dočasnou adresou mobilního uzlu. Ovšem vytvořením vazby dokáže jen získat provoz směřující k mobilnímu uzlu, ke kterému se beztak dostane. Žádné nové bezpečnostní riziko se tedy neotvírá.

seznam aktualizací vazby

Mobilní uzel si musí udržovat přehled o tom, komu všemu zaslal *Aktualizace vazby*. Pokud totiž změní svou dočasnou adresu, měl by je o tom informovat. K ukládání potřebných údajů slouží *seznam aktualizací vazby*. Ten si musí udržovat každý mobilní uzel a ukládá do něj informace o aktualizacích, které odeslal a jejich životnost dosud nevypršela.

11.5 Přenosy dat

Klíčovou datovou strukturou pro běžný provoz v síti je *cache vazeb*. Do ní se ukládají informace o tom, že některý uzel je momentálně dosažitelný na určité dočasné adrese. Cache vazeb by měl mít každý stroj v IPv6 síti a při odesílání datagramu ji musí konzultovat ještě před cache cílů[7]. Každý záznam v ní by měl obsahovat mimo jiné:

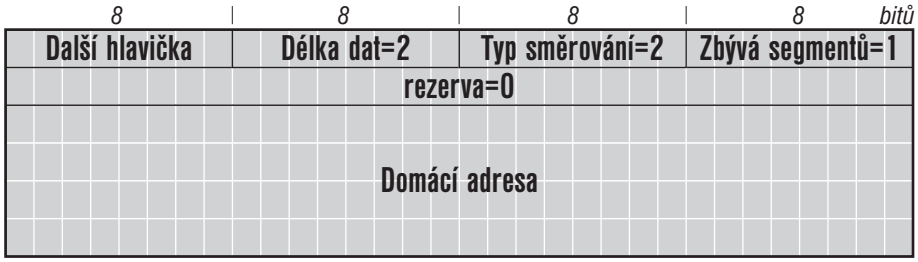
- domácí adresu (podle ní se hledá)
- odpovídající dočasnou adresu
- dobu životnosti (po vypršení musí záznam odstranit)

[6] Pokud se touto dobou ptáte, proč se autentizační informace posílají v hlavičce *Autorizační data*, když autentizace a autorizace nejsou totéž, neptejte se.

[7] Což je logické – než se začne shánět po směrování, bylo by záhodno se podívat, zda se data pro tento uzel nemají posílat na úplně jinou IPv6 adresu.

- příznak, zda se jedná o domácí registraci (zda pro dotyčný mobilní uzel vykonává funkci domácího agenta)
- dosud nejvyšší pořadové číslo z aktualizace vazby

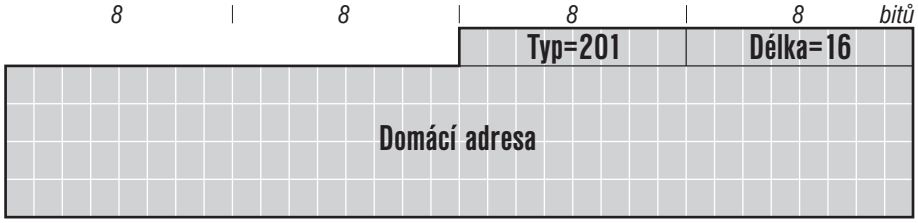
Má-li odeslat datagram, nejprve prohlédne cache vazeb. Pokud obsahuje položku pro daný cíl, připojí k datagramu rozšiřující hlavičku *Směrování*, jejímž prostřednictvím odešle datagram na dočasnou adresu mobilního uzlu. Jako cílovou adresu tedy použije aktuální dočasnou adresu a do hlavičky *Směrování* uloží jedinou adresu, kterou bude domácí adresa mobilního uzlu. Mobilita pro tento účel definuje nový, jednoduchý typ *Směrování*.



Obrázek 11.16: Hlavička *Směrování* typu 2

Když datagram dorazí k cíli, bude hlavička *Směrování* v mobilním uzlu zpracována standardním způsobem a do adresáta v základní IPv6 hlavičce paketu se tak vrátí domácí adresa mobilního uzlu.

Naproti tomu když data odesílá mobilní uzel, používá jako zdrojovou adresu svou dočasnou adresu. Ke každému datagramu však připojuje volbu *Domácí adresa*, kterou adresáta informuje o své domácí adrese. Posílá ji v rámci hlavičky *Volby pro příjemce*.



Obrázek 11.17: Volba *Domácí adresa*

Příjemce datagramu *musí* obsah volby *Domácí adresa* zkopírovat do zdrojové adresy datagramu. Transportní vrstva a její nadřízení tak o mobilitě nemají tušení a vytrvale používají domácí adresu.

správa položek

Přijetím *Aktualizace vazby* se protějšek mobilního uzlu dozvěděl, že jeho

partner má aktuální adresu jinou než domácí. Tato informace je však časově omezená. Její životnost vyjadřuje příslušná položka v aktualizaci, kterou si příjemce zanese do cache vazeb a postupně ji zmenšuje.

Když životnost záznamu v cache vazeb klesne na nulu, měl by být odstraněn. To znamená, že další datagram bude směřovat na domácí adresu a klasickým koloběhem domácí agent–tunel–mobilní uzel vznikne nová aktualizace, díky které bude pro mobilní uzel opět založen záznam v cache vazeb. Pokud však s dotyčným uzlem probíhá čilá komunikace, lze očekávat, že se nic nezměnilo a popsané operace jsou zbytečné. Proto lze na blížící se vypršení platnosti reagovat umírněněji a zaslat *Žádost o vazbu*, jak jsem popsal výše.

11.6 Změny a návrat domů

Prostředí mobilních komunikací je vysoce dynamické. Mobilní uzel může svou adresu změnit během několika minut, což by zúčastněné nemělo vyvést z míry.

detekce změny

Prvním problémem je, aby mobilní uzel vůbec poznal, že se změnila jeho síť. Specifikace mu povoluje používat k tomuto účelu jakékoli dostupné prostředky. Ten základní, který je k dispozici pro všechny počítače bez rozdílu, vychází z objevování sousedů, především jeho části o automatické konfiguraci směrování.

Mobilní uzel naslouchá ohlášení směrovačů, která k němu přicházejí. Z nich se dozví prefixy sítě, v níž se právě nachází, i seznam možných implicitních směrovačů. Vybere si jeden z nich za svůj implicitní směrovač a jeden z jím ohlašovaných prefixů přijme za svůj primární prefix. Na jeho základě si vytvoří primární dočasnou adresu a tu zaregistruje u domácího agenta. Kromě ní může používat i další dočasné adresy s odlišnými prefixy, avšak jako primární může zaregistrovat vždy jen jednu.

Následně mobilní uzel standardním způsobem kontroluje dosažitelnost implicitního směrovače (detekce nedosažitelnosti v objevování sousedů). Jako potvrzení dosažitelnosti přitom bere přijetí libovolného IPv6 datagramu od implicitního směrovače – například ohlášení směrovače, která by měl v určitých intervalech vysílat. Pokud delší dobu nedorazí žádný paket, měl by dosažitelnost implicitního směrovače prověřit výzvou sousedovi.

reakce na změnu

Jakmile vyjde na povrch, že implicitní směrovač je nedosažitelný, mobilní uzel si z toho odvodí, že změnil aktuální síť. Opět tedy zváží informace dostupné v ohlášeních směrovačů (případně o ně aktivně požádá) a vybere si nový implicitní směrovač a novou primární adresu.

Tuto změnu však musí ohlásit – jednak svému domácímu agentovi, jednak všem strojům, se kterými v poslední době komunikoval. Domácímu agentovi zašle klasickou domácí registraci (aktualizaci vazby s nastaveným příznakem *H*). Své ostatní partnery najde v seznamu aktualizací. Ten obsahuje údaje o všech dosud platných (jejich životnost nevypršela) aktualizacích, jež odeslal. Počítačům, které v něm najde, pošle aktualizaci s novou adresou, aby si mohli upravit cache vazeb a uvést ji do souladu se současným stavem.

Kromě toho může poslat žádost o domácí registraci některému z domácích agentů v dočasné síti, kterou právě opustil. Zde jako domácí adresu použije dočasnou adresu z této sítě (která již přestala platit). Pokud ji domácí agent přijme, bude mu tunelem předávat datagramy, které dorazí na jeho předchozí dočasnou adresu (byly odeslány dříve, než dorazila aktualizace vazby ohlašující novou dočasnou adresu).

návrat domů

Návrat do domácí sítě je vlastně speciálním případem přesunu. Mobilní počítač jej objeví stejnou metodou, jako běžnou změnu aktuální adresy. Tentokrát však posílá žádost o zrušení vazby. Ta má podobu běžné *Aktualizace vazby* s nulovou životností. Podle toho příjemce pozná, že záznam pro dotyčnou domácí adresu má odstranit z cache vazeb. Tato informace se posílá domácímu agentovi (s příznakem *H*), aby zrušil domácí registraci, i ostatním uzlům uvedeným v seznamu aktualizací.

Kromě toho musí rozeslat několik ohlášení souseda, aby se opět chopil svých datagramů. Zruší tak účinek předchozích ohlášení svého bývalého domácího agenta.

zrušení vazby

Zatím jsem se zabýval případem, kdy rozhodující aktivity vycházejí od mobilního uzlu. RFC 5846: *Binding Revocation for IPv6 Mobility* doplnilo i opačný případ, kdy službu chce ukončit domácí agent. Slouží k tomu zpráva *Zrušení vazby (Binding revocation)*, jejímž zasláním domácí agent sdělí mobilnímu uzlu, že jeho vazba už není platná. Ten zprávu potvrdí a následně se pokusí obvyklým způsobem vytvořit novou, případně u jiného domácího agenta.

11.7 Rychlé předání

Výše jsem popsal standardní přístup k přesunu mobilního uzlu z jedné sítě do druhé. Jeho nepříjemnou vlastností je, že způsobí krátkodobý výpadek spojení mobilního uzlu, který začne pracovat na změně až po zjištění nedosažitelnosti aktuálně používaného směrovače. Tento výpadek uživatel často vůbec nezaznamená, ale pokud zrovna provozuje aplikaci komunikující v reálném čase (třeba telefonuje po síti nebo hraje on-line hru), může pro něj být velmi citelný.

RFC 5568: *Mobile IPv6 Fast Handovers* proto přišlo s protokolem podporujícím rychlé předání. Jeho základní myšlenkou je, že si mobilní uzel připraví vše předem, aby změna síťových parametrů proběhla rychle a hladce. Za jednoduchým principem se bohužel skrývá nezanedbatelná mašinérie, zahrnující několik nově definovaných zpráv.

získání informací

Spouštěcí mechanismus celé procedury není pevně definován. Typicky bude vycházet z podnětu linkové vrstvy, která ohlásí přítomnost nové potenciální sítě – například když Wi-Fi karta zaznamená signál nového AP slibné úrovně. V této situaci mobilní uzel osloví přístupový směrovač, který právě používá. Dále mu budu říkat starý směrovač. Prostřednictvím ICMPv6 mu pošle *Výzvu směrovači k proxy ohlášení (Router Solicitation for Proxy Advertisement, RtSolPr)* s cílem dozvědět se něco o potenciální síti. Její součástí je linková adresa prostředku, k němuž zvažuje přechod[8].

Odpovědí bude *Proxy ohlášení směrovače (Proxy Router Advertisement, PrRtAdv)*, v němž oslovený směrovač sdělí mobilnímu uzlu informace o poptávaném zařízení. Předpokládá se, že směrovač zná síťovou situaci svých sousedů. RFC 5568 však nijak neřeší, jak ji získá a případně udržuje. Existují tři základní alternativy odpovědi:

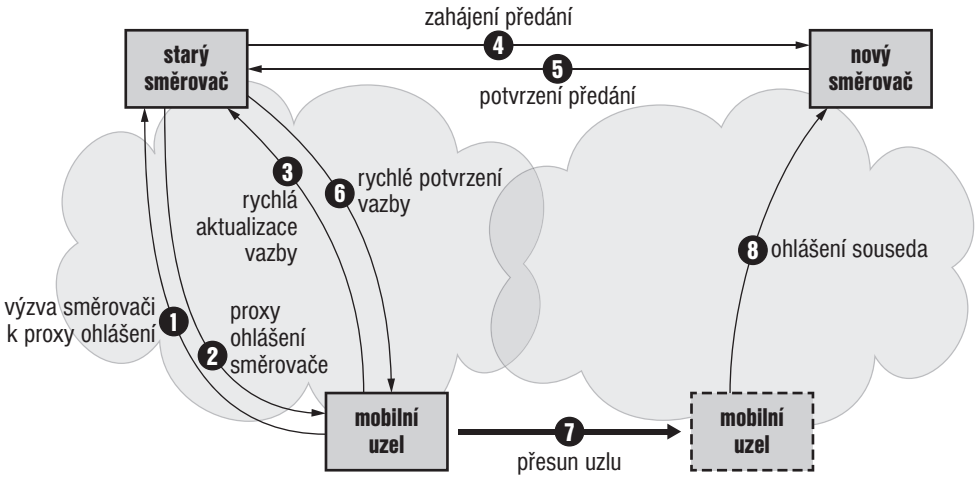
- Směrovač dotazovanou linkovou adresu nezná. V takovém případě rychlé předání končí a mobilní uzel bude muset případně přejít na novou linku obvyklým postupem.
- Směrovač adresu zná a je připojena ke stejnému rozhraní jako linkový prvek, k němuž je uzel aktuálně připojen. V tomto případě algoritmus také končí, protože z hlediska síťové vrstvy se přechodem na novou linku nic nezmění – adresy, prefixy i implicitní směrovač zůstávají v platnosti.
- Směrovač adresu zná a je připojena k jinému jeho rozhraní. V tomto případě by se situace přechodem uzlu změnila. Oslovený směrovač proto v ohlášení poskytne informace o směrovači v cílové síti (nový směrovač): jeho linkovou a IP adresu a prefix(y) příslušné sítě. Žolíkový dotaz je speciálním případem této varianty, v odpovědi směrovač uvede všechny známé sousedy.

Až do tohoto okamžiku je veškerá komunikace ryze informativní. Mobilní uzel se zeptal, co může v potenciální nové síti čekat, a směrovač mu to sdělil. Jestliže se mobilní uzel k přesunu neodhodlá, může vše zase zapomenout a nic se nestane.

přesun uzlu

Pokud se ale mobilní uzel rozhodne přejít do sítě, o níž získal informace, bude protokol pokračovat. Připraví si novou dočasnou adresu odpovídající

[8] Alternativou tohoto přístupu je, že pošle výzvu se „žolíkovou“ adresou a požádá tak o informace o všech směrovači známých sousedech.



Obrázek 11.18: Rychlé předání

prefixu nové sítě a pošle svému stávajícímu přístupovému směrovači *Rychlou aktualizaci vazby (Fast binding update, FBU)*, jejímž smyslem je oznámit mu „hodlám přejít na tuto adresu". K aktualizaci jsou přiloženy autentizační údaje, aby směrovač mohl ověřit, že o změnu žádá skutečně mobilní uzel.

Přípustnost nové adresy si starý směrovač ověří u nového – pošle mu *Zahájení předání (Handover initiate, HI)* s linkovou, starou IPv6 a novou IPv6 adresou mobilního uzlu[9]. Může také požádat, aby nový směrovač ukládal pakety přicházející na novou dočasnou adresu do doby, než se mobilní uzel do nové sítě skutečně přepojí.

Nový směrovač vše prověří a pošle zpět *Potvrzení předání (Handover Acknowledge, HAck)*, kde sdělí, zda je ochoten mobilní uzel přijmout, a případně změní jeho novou přechodnou adresu – například pokud uzlem zvolená je již používána. Od tohoto okamžiku začne starý směrovač předávat tunelem datagramy přicházející na starou dočasnou adresu mobilního uzlu novému směrovači. Zároveň pošle mobilnímu uzlu *Rychlé potvrzení vazby (Fast Binding Acknowledgment, FBAck)*, kterým jej informuje, že je pro přechod vše připraveno.

Uzel se následně přesune do nové sítě a okamžitě pošle nevyžádané *Ohlášení souseda*, aby se nový směrovač dozvěděl o jeho přítomnosti a doručil mu případné uložené datagramy. Následuje pak aktualizace vazeb u domácího agenta i strojů, s nimiž mobilní uzel komunikuje. Díky popsanému postupu je přerušení komunikace omezeno na minimum. Navíc mobilní uzel

[9] Protokol pamatuje i na variantu, že adresu direktivně přidělí nový směrovač. V tom případě *Zahájení předání* představuje žádost o ni a oslovený směrovač ji sdělí v příslušném potvrzení.

o nic nepřijde, protože pakety přicházející v době přechodu či krátce po něm na jeho starou adresu předává starý směrovač novému a ten je doručuje.

RFC 5568 pamatuje i na méně obvyklé situace, jako je odeslání *Rychlé aktualizace vazby* až z nové sítě, pokud to mobilní uzel nestihne ze sítě předchozí, nebo přechod iniciovaný sítí, kdy starý směrovač pošle mobilnímu uzlu *Proxy ohlášení směrovače* z vlastní iniciativy a vyzve jej tak k přestupu do nové sítě.

Sympatická je zpětná kompatibilita rychlého předání. Funguje, jen pokud je podporují obě strany – mobilní uzel i směrovač, který jej obsluhuje. Jestliže jedna strana tento optimalizační mechanismus nezná, nic se neděje a případný přechod uzlu do jiné sítě proběhne klasicky.

11.8 Hierarchická mobilita

Celkem snadno si lze představit následující scénář: vyrazíte si na víkend k babičce a cestu si budete krátit surfováním po Internetu. Jak se budete pohybovat po republice, bude váš mobilní počítač postupně přecházet z jedné podsítě mobilního operátora do jiné. Každý přechod vyvolá změnu adresy a váš počítač tudíž musí rozeslat aktualizace vazby domácímu agentovi a strojům, se kterými komunikuje. Přitom však stále zůstává v síti jednoho operátora a značná část cesty, kterou jeho pakety procházejí, se nemění. Bylo by příjemné, kdyby se tato skutečnost dala nějak využít ke snížení počtu zasílaných aktualizací.

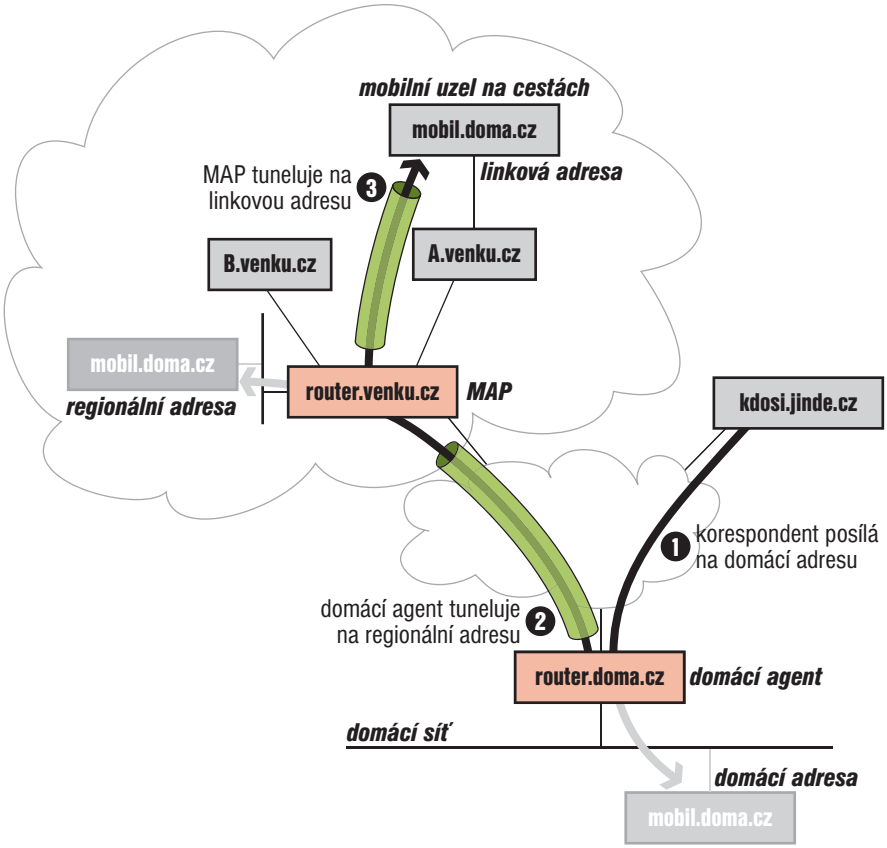
Právě tohle je cílem hierarchické mobility definované v RFC 5380: *Hierarchical Mobile IPv6 Mobility Management (HMIPv6)*. Pro velký úspěch se zde opakuje princip domácího agenta, ovšem na druhé straně. Mobilní uzel získal dalšího zástupce, tentokrát v síti mobilního operátora. Nazývá se *kotevní bod mobility (Mobility Anchor Point, MAP)*. Cílem je, aby mobilní uzel po celou dobu svého pobytu v určité síti (např. v síti jednoho mobilního operátora) používal stále stejnou dočasnou adresu a nezasílal tedy aktualizace vazby. Tato adresa je označována jako *regionální dočasná adresa (Regional care-of address, RCoA)*.

kotevní bod mobility, MAP

Mobilní uzel se při použití hierarchické mobility registruje dvakrát. Nejprve si u kotevního bodu MAP zaregistruje předávání datagramů z regionální adresy na aktuální dočasnou. Následně pak u domácího agenta a všech korespondentů registruje vazbu mezi svou domácí adresou a regionální dočasnou adresou.

Funkce kotevního bodu velmi připomíná domácího agenta. Mobilní uzel se u něj zaregistruje a místo své domácí adresy použije vypočtenou regionální adresu. Kotevní bod se pak chová zcela jako domácí agent – zachytává pakety směřující na regionální adresu mobilního uzlu a předává je na jeho

aktuální dočasnou adresu, pro niž se v hierarchické mobilitě používá označení *linková dočasná adresa (On-link care-of address, LCoA)*.

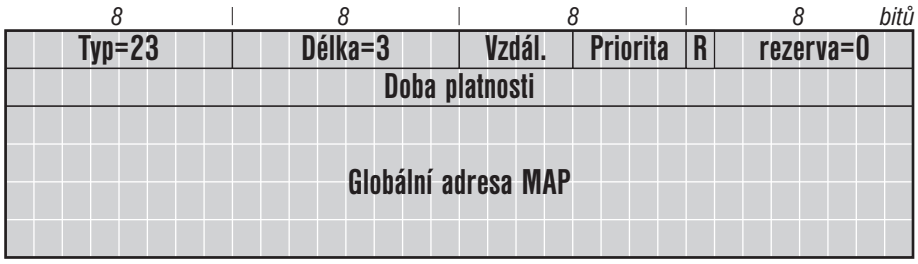


Obrázek 11.19: Hierarchická mobilita

Základní neoptimalizovanou komunikaci s mobilním uzlem podporujícím hierarchickou mobilitu znázorňuje obrázek 11.19. Korespondent pošle datagram na jeho domácí adresu. Domácí agent jej zachytí a podle běžných pravidel pro mobilitu pošle tunelem na adresu, jejíž vazbu má pro danou domácí adresu registrovánu. Tedy na regionální dočasnou adresu. Tuto adresu zachytává kotevní bod a přepošle datagram tunelem na aktuální linkovou dočasnou adresu mobilního uzlu.

volba MAP

Vzniká pochopitelně otázka, jak mobilní uzel najde svůj kotevní bod. Jako obvykle přichází ke slovu *Ohlášení směrovače*, do něhož směrovač přidá informace o kotevních bodech pomocí nové volby *Mobility anchor point (MAP)* znázorněné na obrázku 11.20.



Obrázek 11.20: Volba *MAP* pro *Ohlášení směrovače*

Jejími nejvýznamnějšími položkami jsou pochopitelně *Globální adresa MAP (Global IP Address for MAP)* a *Doba platnosti (Valid Lifetime)*. Operátor sítě může provozovat několik kotevních bodů v různých místech a různých hierarchických úrovních sítě. Je otázkou konfigurace směrovačů, které z nich budou svým klientům ohlašovat. Představte si třeba, že v mobilní síti na obrázku 11.19 budou oba směrovače označené jako *A.venku.cz*, *B.venku.cz* a *router.venku.cz* zároveň pracovat jako kotevní body. První z nich proto bude do sítí připojených k sobě ohlašovat existenci dvou kotevních bodů – sám sebe a hierarchicky vyšší *router.venku.cz*. Ohlašovat zde *B.venku.cz* nemá valný smysl, protože případné doručování datagramů do zdejší sítě přes *B.venku.cz* je přes ruku.

Mobilní uzel tedy může dostat nabídku několika kotevních bodů. Pro výběr nejvhodnějšího slouží položky *Priorita (Pref)* a *Vzdálenost (Dist)* ve volbě *MAP*. První představuje obecnou míru dostupnosti daného kotevního bodu. Čím vyšší hodnota, tím lépe. Je-li priorita nulová, měl by se mobilní uzel tomuto kotevnímu bodu vyhnout. *Vzdálenost* pak obsahuje počet síťových skoků z dané sítě k tomuto kotevnímu bodu. Vzdálenosti se tedy měří podobně jako v protokolu RIP. Na rozdíl od směrování zde ale neplatí, že nejlepší je nejbližší hodnota. Spíše naopak, vzdálenější kotevní bod leží ve vyšší úrovni hierarchie sítě a pravděpodobně jej bude možné využívat po delší dobu.

RFC 5380 popisuje dva možné přístupy k volbě MAP ze strany mobilního uzlu. Jedním je vybrat kotevní bod s nejvyšší *Prioritou*, druhým zvolit nejvzdálenější, který má nenulovou *Prioritu*. Dokument také připouští registraci uzlu u několika kotevních bodů. Pokud je *Priorita* nulová, uzel se nesmí k tomuto kotevnímu bodu nově registrovat, ale smí u něj prodloužit svou stávající registraci. Zcela fatální je nulová *Doba platnosti*, jež signalizuje selhání MAP. Takový kotevní bod si mobilní uzel nesmí zvolit a všechny existující vazby k němu může považovat za ztracené.

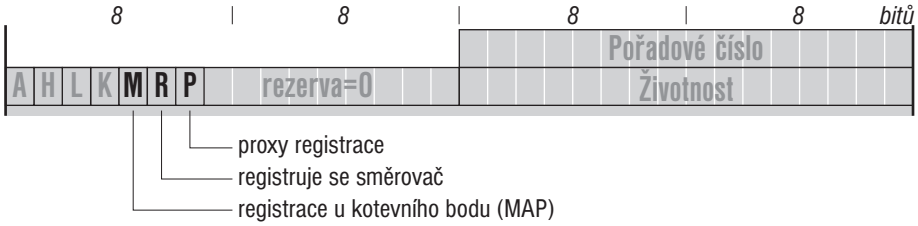
regionální adresa RCoA

Když má mobilní uzel vybrán kotevní bod, vypočte si svou regionální dočasnou adresu. Udělá to velmi jednoduše – spojí prefix (počátečních 64 b)

z globální adresy kotevního bodu se svým identifikátorem rozhraní. Jelikož se regionální adresa nachází ve stejné podsíti jako kotevní bod, je zaručeno, že pakety na ni směřující dorazí do podsítě přímo připojené ke kotevnímu bodu a ten je dokáže obvyklým trikem s objevováním sousedů stáhnout na sebe.

registrace

Pro účely registrace u kotevního bodu byla lehce rozšířena *Aktualizace vazby*, konkrétně do ní přibyl příznak *M (registrace MAP, MAP registration)*. Na obrázku 11.21 jej vidíte společně s příznaky *R* a *P*, které do aktualizace vazby přidávají proxy mobilita a podpora mobilních sítí popsané v následujících částech. Mobilní uzel jej nastaví, když se registruje u kotevního bodu. Aktualizaci vazby mu pošle ze své aktuální linkové dočasné adresy a ve volbě *Domácí adresa* uvede svou regionální adresu. Tím kotevní bod informuje o svých adresách ve venkovní síti.



Obrázek 11.21: Rozšíření příznaků v *Aktualizaci vazby*

Jakmile kotevní bod potvrdí registraci, vytvoří si s mobilním uzlem obousměrný tunel, kterým si budou předávat datagramy směřující na regionální adresu a přicházející z ní. Mobilní uzel následně registruje u svého domácího agenta regionální adresu jako svou dočasnou. Tím je nastaveno vše, aby data pro něj mohla být doručována podle obrázku 11.19. Také při optimalizaci cesty bude svým korespondentům ohlašovat regionální adresu jako svou dočasnou.

Jestliže se mobilní uzel přesune a změní aktuální linkovou adresu, ale zůstane v působnosti svého kotevního bodu, stačí mu změnit svou registraci u kotevního bodu. Všechno ostatní zůstává beze změny a datagramy nadále najdou svého adresáta. Teprve když dorazí do části sítě, v níž není jím používaný kotevní bod ohlašován, musí přejít na jiný a odpovídajícím způsobem změnit i svou registraci u domácího agenta a vazby u všech korespondentů.

bezpečnost

Nabízí se otázka, zda se zavedením hierarchické mobility neotevírají nové možnosti pro kradení datagramů či jiné neplechy. Není tomu tak při výměně dat mezi mobilním uzlem a domácím agentem (ta je chráněna ESP stejně jako předtím), ani mezi mobilním uzlem a jeho komunikačními partnery

(mobilní uzel stále musí prokázat, že dokáže přijímat data i na své domácí adrese).

Zbývá nový prvek, komunikace mezi mobilním uzlem a kotevním bodem. Oba účastníci mohou chtít ověřit totožnost svého protějšku – zda je mobilní uzel oprávněn využívat tuto službu a zda je kotevní bod skutečně ten, za koho se vydává. Budou jím procházet data směřující na domácí i regionální adresu mobilního uzlu, takže má příležitost páchat různé neplechy. Kromě toho by se kdokoli mohl snažit prohlásit se za novou lokální adresu pro registrovanou regionální a ukrást tak data určená mobilnímu uzlu.

K ověření totožnosti účastníků lze použít například certifikáty ověřené autoritou, jíž účastníci důvěřují. Vzájemná komunikace by pak měla být chráněna IPsec. Jak mobilní uzel, tak kotevní bod musí podporovat IKEv2, aby mohli navázat bezpečnostní asociaci a chránit data, jež si vyměňují.

Nejpříjemnější vlastností hierarchizace je, že se jedná o nepovinné rozšíření mobilního IPv6. Domácí agent i korespondent zůstávají beze změny. Pokud mobilní operátor nebo mobilní uzel nepodporují hierarchickou mobilitu, nic se neděje a mobilní uzel bude používat klasické mobilní IPv6. Jen při té šťastné kombinaci, kdy mobilní uzel umí hierarchickou mobilitu a z ohlášení směrovače zjistí, že je k dispozici kotevní bod, může (ale nemusí) se u něj zaregistrovat a chovat se podle hierarchických pravidel.

11.9 Proxy mobilita

Další variantou podpory mobilních zařízení v IPv6 síti je proxy mobilita definovaná v RFC 5213: *Proxy Mobile IPv6*. Jejím cílem je umožnit pohyb v síti i strojům, které nepodporují mobilní IPv6. Vše za ně obstará síťová infrastruktura, jež udržuje mobilní uzel v iluzi, že neopustil svou domácí síť. Jeho domácí síť totiž cestuje společně s ním – vzdálené aktivní prvky ji založí vždy tam, kde se cestující uzel připojí k infrastruktuře.

Vlastní mobilní uzel není účastníkem signalizace a nekladou se na něj žádné požadavky. Komunikuje úplně normálně a ani se nedozví, že se v infrastruktuře někam posunul. Veškeré mobilní záležitosti za něj vyřizuje aktivní prvek, k němuž je na cestách momentálně připojen – ve zdejší terminologii *mobilní přístupová brána (Mobile Access Gateway, MAG)*.

MAG a LMA

Jakmile MAG zjistí, že do jím obsluhované sítě vstoupil mobilní uzel, obrátí se na jeho domácí prvek a zaregistruje u něj vazbu. Úmyslně jsem nenapsal domácího agenta, protože činnost tohoto prvku je při proxy mobilitě poněkud rozšířena. Oficiálně se jmenuje *lokální kotva mobility (Local Mobility Anchor, LMA)* a dokáže fungovat jako obyčejný domácí agent pro standardní mobilní uzly i rozšířený agent pro uzly využívající proxy mobilitu.

přístupový profil

Vzniká samozřejmě otázka, jak MAG zjistí LMA spravující mobilní uzel, který se právě objevil. K tomu slouží tak zvaný *přístupový profil (policy*

profile) mobilního uzlu. Obsahuje jeho základní komunikační parametry – povinně adresu jeho LMA, volitelně prefix domácí sítě či způsob automatické konfigurace.

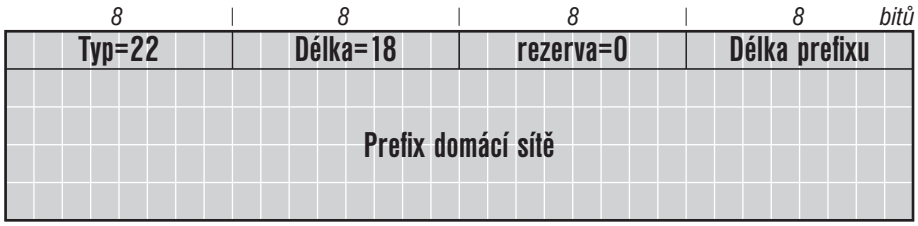
Klíčem k údajům je identifikátor mobilního uzlu, podle nějž jej všechny složky jednoznačně rozpoznají. V této roli nelze použít domácí IPv6 adresu. Jak již bylo řečeno, mobilní uzel netuší, že se pohybuje, svou adresu nezná a teprve ji bude chtít získat vhodným autokonfiguračním mechanismem. Jako jeho identifikátor může sloužit síťový přístupový identifikátor (Network Access Identifier, NAI) podle RFC 4282, MAC adresa, případně i identifikátor uživatele, pokud je jednoznačně spojen s daným uzlem.

MAG zjistí identifikátor uzlu obvykle během autentizační procedury a následně jej použije k získání odpovídajícího přístupového profilu. Specifikace předpokládá, že bude k dispozici úložiště síťových profilů, na něž se mohou obracet MAG a LMA. Nijak podrobněji ovšem tuto složku nedefinuje.

registrace u LMA

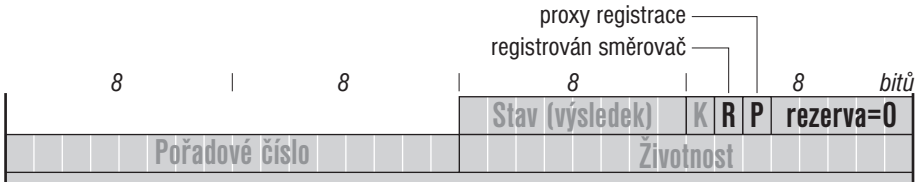
Z přístupového profilu se MAG dozví adresu příslušné LMA a pošle jí žádost o registraci. Proxy mobilita sdílí s běžnou podporou mobility formát zpráv, jen k nim přidává některá rozšíření. MAG tedy pošle *Aktualizaci vazby* se zapnutým příznakem *P*, jenž signalizuje proxy aktualizaci. Její součástí je několik voleb definovaných většinou v RFC 5213:

- *Identifikátor mobilního uzlu (Mobile node identifier)* jednoznačně určující zastupovaný uzel. Formát volby definuje jednoúčelové RFC 4283.
- *Prefix domácí sítě (Home Network Prefix)* může být nulový, pokud jej má přidělit LMA, nebo konkrétní, například když je prefix součástí přístupové politiky, takže jej MAG zná.
- *Typ přístupové technologie (Access technology type)* oznamuje, jakým způsobem je uzel připojen.
- *Indikátor předání (Handoff indicator)* sděluje, zda se jedná o nové připojení, nebo předání mezi dvěma MAG při pohybu uzlu v síti.



Obrázek 11.22: Volba *Prefix domácí sítě*

Dalšími volbami lze oznámit lokální linkovou adresu uzlu, jeho linkový identifikátor či časovou značku pro řazení zpráv. Informace z *Aktualizace vazby* LMA využije k posouzení, jestli je žádost daného mobilního uzlu v pořádku. Je-li nakloněna jí vyhovět, přidělí (nebo potvrdí) uzlu prefix jeho domácí sítě a pošle MAG zpět *Potvrzení vazby*, v němž oznámí příslušný prefix domácí sítě. Také do potvrzení přibyl příznak *P*, jímž LMA signalizuje, že se jedná o proxy registraci. Na obrázku 11.22 jej vidíte společně s příznakem *R* definovaným pro mobilní sítě (viz následující část).



Obrázek 11.23: Rozšířené příznaky v *Potvrzení vazby*

RFC 5213 počítá s tím, že prefix sítě je pro každý mobilní uzel jednoznačný a že se v dané síti nacházejí jen dva stroje – mobilní uzel a MAG. Pokud linka, k níž se mobilní uzel připojí, není dvoubodová, musí takové chování alespoň emulovat.

Současně s přenosem *Potvrzení vazby* dojde i k vytvoření tunelu mezi LMA a MAG, jímž bude následně procházet provoz mobilního uzlu. Zatímco pro vzájemnou signalizaci je ochrana IPsec povinná, datový tunel být chráněn může, ale nemusí. Pro aktualizace a potvrzování vazeb však musí mezi MAG a LMA existovat bezpečnostní asociace. K jejímu vytvoření je doporučován protokol IKEv2, jehož podpora je u obou prvků povinná.

datové přenosy

MAG následně zařadí obdržený prefix do *Ohlášení směrovače* zaslaného mobilnímu uzlu a zároveň se v něm označí jako implicitní směrovač. Díky tomu si mobilní uzel vytvoří adresu s odpovídajícím prefixem a své pakety bude následně odesílat do světa prostřednictvím MAG. Ten je tunelem předává LMA, protože adresa odesilatele patří do adresního prostoru LMA a při přímém odeslání do Internetu by pakety nemusely projít filtry střežícími korektnost zdrojové adresy.

Klíčovým bodem komunikace mobilního uzlu je proto LMA, která odesílá i přijímá jeho pakety. Pro příjem tentokrát není třeba švindlovat s ohlašováním sousedů. Stačí, když LMA standardními směrovacími protokoly ohlašuje, že jeho prostřednictvím je připojena domácí síť mobilního uzlu. Směrování je samozřejmě neefektivní – pakety dělají odbočku přes LMA – a na rozdíl od běžné mobility tentokrát neexistuje žádný optimalizační mechanismus. Jedinou výjimkou je, že pokud MAG pozná, že mobilní uzel posílá datagram do některé ze sítí přímo připojených k MAG, může jej předat přímo.

Nelze očekávat, že by tato služba byla poskytována globálně. Spíše ji najdete v části sítě provozované jedním operátorem (nebo skupinou několika spolupracujících operátorů). Mluví se o ní jako o *doméně proxy mobility* a pouze v rámci této domény bude pohyb mobilního uzlu transparentní.

K základní definici proxy mobility v RFC 5213 existuje několik rozšiřujících doplňků. Umožňují zapojit do hry IPv4, a to jak na straně domácí adresy mobilního uzlu, tak při signalizaci (RFC 5844), autentizační protokol Diameter k poskytování přístupových profilů (RFC 5779), kontrolu spojení mezi MAG a LMA (RFC 5847) či rychlé předávání mobilního uzlu (RFC 5949).

11.10 Mobilní sítě (NEMO)

Klasická podpora mobility počítá s jedním pohybujícím se uzlem, například bezdrátově komunikujícím notebookem. Z představ vizionářů se však postupně přesouvají do reality celé mobilní sítě. Pěkným aktuálním příkladem je poskytování Internetu ve veřejném dopravním prostředku – letadle, vlaku či autobusu. Ve vozidle je vnitřní síť, do níž jsou zapojeny počítače pasažérů. Celá tato síť má nějaké adresy, určitou formu automatické konfigurace a implicitní směrovač, který jí zprostředkovává spojení s Internetem. Budeme mu říkat mobilní směrovač. Situace ve vnitřní síti je stálá, zdejší prefixy se nemění.

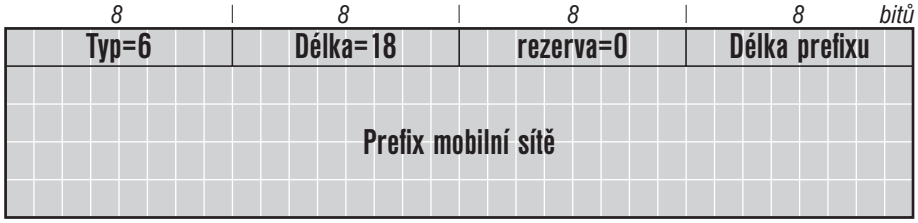
Ovšem celá tato síť se pohybuje, což se projevuje změnou připojení mezi jejím mobilním směrovačem a okolím. Mění se adresa vnějšího rozhraní mobilního směrovače a s ní i cesta mezi vnitřní sítí a Internetem. Jiným, v současnosti dosud vizionářským příkladem jsou osobní sítě, které nás prý dříve či později čekají. Mobil se stane přístupovým směrovačem, který bude poskytovat připojení ostatním komunikujícím zařízením – diáři, hodinkám, hudebnímu přehrávači, kardiostimulátoru ... Člověk se bude pohybovat i se svými hračkami a zase platí totéž co v předchozím případě. Situace uvnitř osobní sítě zůstává poklidná, zatímco její poloha a s ní připojení k Internetu prodělává změny.

Síťová mobilita (NEMO, NEtwork MObility) poskytuje prostředky, jimiž lze tuto situaci ošetřit. Mobilita sítě se díky ní stává zcela transparentní, zařízení v mobilní síti dokonce sama ani nemusí mobilitu podporovat. Definici její základní podoby najdete v RFC 3963: *Network Mobility (NEMO) Basic Support Protocol*.

Základní myšlenka je velmi prostá: mobilní směrovač nebude do aktuální sítě, do níž je právě připojen, šířit směrovacími protokoly informace o síti, kterou připojuje. Pravděpodobně by to stejně nemělo smysl, její adresy neodpovídají zdejším, takže by propagace o jejich dostupnosti dříve či později narazila na filtr směrovacích informací. Místo toho mobilní směrovač udržuje obousměrný tunel se svým domácím agentem. Veškerá komunikace se stroji v mobilní síti prochází tímto tunelem (a tedy domácím agentem).

Mobilní směrovač se podle NEMO chová podobně jako běžný mobilní uzel, ovšem ve své registraci dočasné adresy u domácího agenta nastaví v *Aktualizaci vazby* nově definovaný příznak *R (Router)*. Jím domácímu agentovi sděluje, že se ve skutečnosti jedná o směrovač a za ním je připojena mobilní síť. Příznak *R* byl doplněn i do *Potvrzení vazby*. Domácí agent jím sděluje mobilnímu směrovači, že podporuje NEMO a že z jeho strany je vše připraveno pro spolupráci s mobilní sítí. Následně spolu vytvoří tunel, do nějž bude domácí agent předávat datagramy směřující do mobilní sítě a naopak vybalovat pakety přicházející z ní a předávat je dál do Internetu.

K této činnosti ovšem domácí agent potřebuje znát prefix(y) adres používané v lokální síti. Mobilní směrovač mu je může předat hned na začátku v aktualizaci vazby pomocí nově definované volby *Prefix mobilní sítě (Mobile Network Prefix)*. Její formát vidíte na obrázku 11.24, klíčovými položkami jsou *Prefix mobilní sítě (Mobile Network Prefix)* a jeho *Délka (Prefix length)*.



Obrázek 11.24: Volba *Prefix mobilní sítě* pro hlavičku *Mobilita*

Alternativní možností je během registrace u domácího agenta neřešit prefix a nastavit jej jinými prostředky. Nabízí se třeba manuální konfigurace na straně domácího agenta nebo směrovací protokol mezi mobilním směrovačem a jeho agentem. Podobně se RFC 3963 nezabývá otázkou, jakým způsobem mobilní směrovač zjistí prefixy, jež jsou mobilní síti přiřazeny (tady lze nejspíše očekávat manuální konfiguraci, podobně jako u běžných nepohyblivých směrovačů).

Pro domácího agenta se nabízejí dvě možné cesty, jak na sebe stáhnout datagramy směřující do mobilní sítě, aby je mohl předávat tunelem jejímu směrovači. Může to udělat čistě a prostřednictvím směrovacích protokolů začít ohlašovat, že cesta do mobilní sítě nyní vede přes něj. Druhou možností je, že ponechá směrování v původním stavu, kdy cesta do mobilní sítě vede přes domácí adresu mobilního směrovače. Ovšem pro ni hraje roli domácího agenta, takže prostřednictvím objevování sousedů získá datagramy předávané mobilnímu směrovači a může mu je tunelem předat.

V každém případě z hlediska směrování se mobilní síť tváří, jako by stále zůstávala doma. To je dobré pro agregaci prefixů (je možné ji zahrnout do globálního prefixu pro domácí síť), nikoli však pro vlastní směrování. To

bude mít k ideálu daleko, protože všechna data do a z mobilní sítě protékají domácím agentem. Zde platíme daň za jednoduchost. Stroje v mobilní síti ani jejich komunikační partneři nemusí nic dělat, dokonce se ani nedozvědí o tom, že síť je zrovna na cestách. Na druhé straně ale cesta mezi nimi může být dost krkolomná.

RFC 3963 ve svém názvu nese slovo „základní“. Je možné, že se časem objeví nějaká pokročilá podpora mobilních sítí, která poskytne i optimalizaci směrování. Zatím však žádná taková specifikace nevznikla.

12 Kudy tam

Od samého začátku počítali autoři IPv6 s velmi důležitým problémem: jak přejít ze stávajícího IPv4 na nový protokol. Je zřejmé, že při současné velikosti a stavu Internetu nelze prostě prohlásit „Dámy a pánové do konce roku X používáme IPv4 a počínaje 1. lednem X+1 Internet přechází na IPv6." Jsou potřeba prostředky, které umožní současný provoz obou protokolů, spolupráci mezi nimi a postupný přechod od jednoho k druhému.

Obecná představa tedy zní, že podíl IPv6 na Internetu, který je v současnosti prakticky zanedbatelný, se bude postupně zvětšovat a IPv4 bude naopak mizet, až možná zmizí docela. Ale třeba také ne a některé ostrůvky zůstanou starší verzi věrny.

Pro koexistenci a vzájemnou spolupráci obou protokolů existuje celá řada návrhů. Lze je rozdělit do tří základních skupin:

Dvojí zásobník znamená, že příslušné zařízení podporuje jak IPv4, tak IPv6. To mu umožňuje bavit se s partnery z obou světů. Tento princip sám o sobě není nijak zvlášť lákavý, protože vyžaduje zachování IPv4, ke kterému se jen přidá IPv6. Nicméně slouží jako východisko pro zbývající dvě skupiny. Všechny další způsoby totiž vyžadují, aby alespoň některá zařízení podporovala oba protokoly.

Tunelování je ověřená metoda, jak „protlačit" informaci infrastrukturou, která nemá potřebné vlastnosti. Datagram se prostě zabalí jako data do jiného datagramu, který daná síť dokáže přepravit. V případě spolupráce IPv4 a IPv6 se používají dva režimy tunelování. Konfigurované (manuální) tunely jsou explicitně nastaveny správcem, zatímco automatické se navazují samočinně na základě informací obsažených v adresách. Obecně se tunely používají ke spojení dvou počítačů hovořících stejným protokolem (řekněme IPv6), které ovšem musí procházet sítí vyžadující protokol opačný (IPv4).

Translátory čili překladače se snaží o zprostředkování styku oněch dvou světů. Například pokud váš počítač hovoří pouze IPv6 a jste nuceni získat data od serveru, který podporuje výlučně IPv4, žádný tunel vám nepomůže. Potřebujete zařízení, které by přeložilo vaše datagramy do IPv4 a odpovědi serveru naopak do IPv6. Upřímně řečeno, tady tlačí bota nejvíc. Bylo sice navrženo několik překladových služeb, žádná se však nedočkala širšího nasazení a ta nejnadějnější (NAT-PT) byla později zavržena. Po čase se objevila slibná specifikace NAT64, jejíž implementeace však teprve dozrávají. Nekompatibilita IPv6 s IPv4 a vleklé problémy překladu mezi nimi jsou zřejmě největší brzdou při pronikání nového protokolu do života.

Stručné shrnutí metod, jejichž podrobnější popis najdete dále v textu, uvádí tabulka 12.1.

tunelování		
tunel server/broker	RFC 3053	strana 258
6to4	RFC 3056	strana 261
6rd	RFC 5569	strana 265
6over4	RFC 2529	strana 267
ISATAP	RFC 5214	strana 268
Teredo	RFC 4380	strana 270
Dual-Stack Lite	RFC 6333	strana 275
translátory		
SIIT	RFC 6145	strana 277
NAT64	RFC 6146	strana 284
TRT	RFC 3142	strana 287
BIH	draft-ietf-behave-v4v6-bih	strana 288
SOCKS64	RFC 3089	

Tabulka 12.1: Metody pro přechod k IPv6

Jakousi základní sadu nástrojů pro přechod k nové verzi IP tvoří dvojí zásobník a princip tunelování, protože tyto prvky v té či oné podobě využívá řada ostatních mechanismů. Podrobně je rozebírá RFC 4213: *Basic Transition Mechanisms for IPv6 Hosts and Routers*.

12.1 Dvojí zásobník

Název dvojí zásobník (v originále dual stack) je poněkud zavádějící. Vnucuje člověku představu, že dotyčné zařízení má dva zcela oddělené zásobníky protokolů – jeden pro IPv4 a druhý pro IPv6. Ve skutečnosti se často jedná o jeden hybridní zásobník. Klíčové však je, že zařízení musí podporovat oba dva protokoly a mít jak IPv4, tak IPv6 adresu.

K jejich získání se používají obvyklé postupy. Pro IPv4 ruční konfigurace nebo DHCP, pro IPv6 připadá v úvahu vedle ruční konfigurace některá z automatických forem – buď bezstavová nebo DHCPv6. Také zdejší DNS klient musí znát oba světy. Konkrétně při získávání adres to znamená, že se musí vyznat jak v A záznamech pro IPv4, tak v AAAA záznamech pro IPv6.

Kooperace mezi oběma protokoly se odehrává (pokud je potřebná) zpravidla až v aplikační vrstvě. Odpovídající aplikace si vyzvedne data, která dorazila jedním protokolem, a v závislosti na svých vlastnostech a případné konfiguraci je upraví a odešle druhým protokolem danému příjemci.

Zařízení podporující oba protokoly bývají označována jako IPv4/IPv6 uzly.

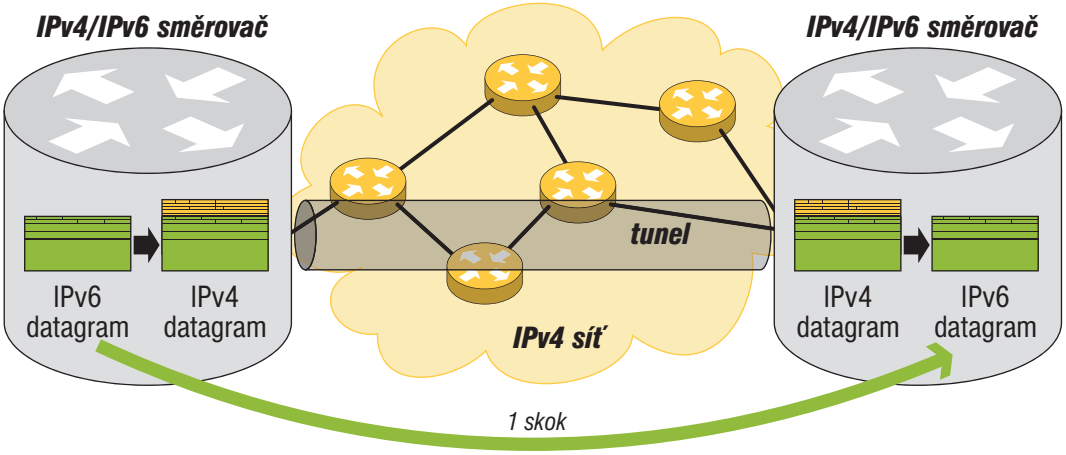
12.2 Obecně o tunelování

Princip tunelování, tedy balení jednoho protokolu do druhého, není žádnou novinkou. V počítačových sítích se s ním setkáte celkem běžně. V počátečních fázích je především třeba tunelovat IPv6 datagramy pro průchod IPv4 sítí, takže se podíváme právě na tento případ. Nicméně myslitelný je i opačný případ – bude o něm řeč v části 12.8 na straně 275.

základní princip

Tunel má dva konce, z nichž každý má svou IPv4 adresu. Když se zařízení na jednom konci rozhodne[1], že daný IPv6 datagram musí odeslat tunelem, vezme jej a vloží jako data do nově vytvořeného IPv4 datagramu. Jeho cílovou adresou bude IPv4 adresa druhého konce tunelu a jako odesilatel bude figurovat zdejší IPv4 adresa tunelu. Skutečnost, že se jedná o tunelovaný IPv6 datagram vyznačí hodnotou 41 v položce *Protokol* obalujícího IPv4 datagramu. Tento způsob přepravy bývá také označován jako *6in4*, nejedná se však o oficiální název.

Datagram se pak odešle běžnou IPv4 sítí. Když dorazí do cíle (na druhý konec tunelu), příjemce podle protokolu 41 pozná, že obdržel tunelovaný paket. Vybalí z něj původní IPv6 datagram a ten pak dále zpracuje podle jeho cílové adresy a svých směrovacích tabulek pro IPv6. Například jej může přijmout, pokud je určen jemu, nebo jej třeba odeslat dalším tunelem někam dál.



Obrázek 12.1: Mechanismus tunelování

Během průchodu tunelem hraje původní IPv6 datagram roli nesených dat a nijak se nemění. Jeho hlavičkami se zabývá až druhý konec tunelu po vybalení. Mimo jiné to znamená, že z hlediska IPv6 je celý průchod tunelem

[1] K tomuto rozhodnutí může dospět buď na základě směrovací tabulky nebo podle speciální adresy příjemce, kterou IPv6 datagram nese.

počítán za jeden skok a vybalující směrovač zmenší položku *Max skoků* v IPv6 hlavičce o jedničku.

konfigurované tunely

Existují dva základní režimy tunelování: konfigurovaný a automatický. Konfigurovaný režim znamená, že na dotyčném zařízení byly provedeny příkazy, kterými vznikl tunel vedoucí kamsi. Vlastně tak vzniklo nové síťové rozhraní (byť virtuální), se kterým se zachází podobně, jako s ostatními. Mimo jiné to znamená, že o odesílání datagramů tímto rozhraním rozhoduje směrovací tabulka. V ní je řečeno, do kterých IPv6 sítí vede cesta daným tunelem. Má-li být paket odeslán tímto rozhraním, zabalí se do IPv4 datagramu s cílovou adresou druhého konce a následuje zpracování podle směrovací tabulky pro IPv4.

Konfigurované tunely se nejčastěji používají k trvalému propojování sítí nebo ke vzdálenému zapojení počítače do IPv6 sítě. Výborně poslouží pro experimenty, když si chce někdo relativně bezbolestně osahat IPv6. Tyto tunely nejsou nijak svázány s fyzickou topologií IPv4 sítě, kterou procházejí. Z hlediska praktického je však velmi záhodno k ní přihlédnout. Jinak hrozí případy, kdy stejnou linkou projde tentýž paket několikrát prostě proto, že tunely, kterými je přepravován, tudy vedou tam a zase zpátky. Pokud se budete připojovat tunelem k nějaké IPv6 síti, vyberte si ten z jejích bodů přítomnosti, který je pro vás nejvhodnější z hlediska topologie IPv4 sítě.

tunel servery

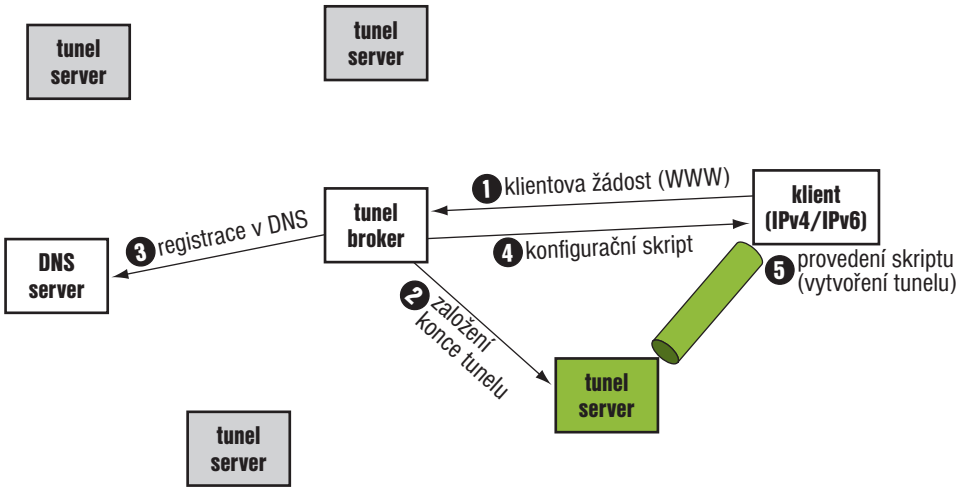
Přestože se to může na první pohled zdát podivné, do kategorie konfigurovaných tunelů spadají i tak zvané tunel servery. Jedná se vlastně o automatickou nadstavbu nad explicitní konfigurací. Tunel server je zařízení, které je zapojeno do IPv4 Internetu i do IPv6 sítě a je ochotno posloužit jako protější konec tunelu pro kohokoli, kdo má zájem.

Uživatel se musí nejprve zaregistrovat (zpravidla vyplněním údajů ve formuláři přístupném prostřednictvím WWW). Robot na základě registrace vytvoří protější konec tunelu a pošle vám konfigurační skript, který musíte spustit. Když to provedete, založíte tak vlastní konec tunelu a připojíte se k IPv6 síti.

Postupem času došlo k dalšímu rozdělení funkcí. Vlastní *tunel server* zpravidla bývá směrovač, kterému nečiní problémy akceptovat značné množství tunelů. Naproti tomu proces registrace a generování konfiguračních skriptů vyžaduje WWW server spolupracující s databází a nejsnadněji se realizuje na běžném počítači. Proto byl zaveden *tunel broker*, který zprostředkovává styk mezi tunel servery a uživateli. Jeden tunel broker může obsluhovat několik tunel serverů. V takovém případě ještě vybere ten z nich, který vzhledem k vámi zadaným údajům považuje za nejvhodnější.

TSP

Výměna informací mezi klientem a tunel serverem či brokerem představuje zajímavý problém. Může být postavena na existujících mechanismech (WWW, E-mail), jak bylo popsáno výše. Jejich nevýhodou je, že vyžadují



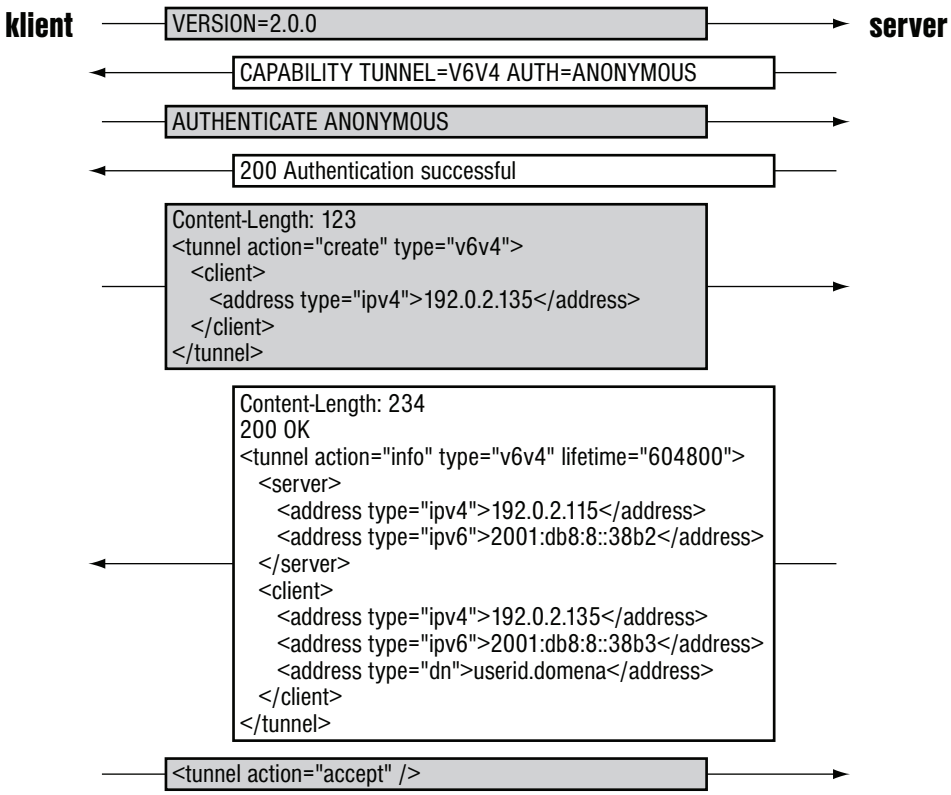
Obrázek 12.2: Tunel server a tunel broker

poměrně kvalifikovaného uživatele, který navíc může být na pochybách, zda zaslaný konfigurační skript dělá skutečně to, co slibuje.

Proto se někteří autoři snaží vymyslet specializované nástroje, které by zajistily výměnu konfiguračních informací a vytvoření tunelu. Asi nejzajímavější je *Tunnel Setup Protocol (TSP)*. Podle něj komunikace probíhá následovně:

1. Klient naváže s tunel brokerem (nebo serverem, TSP lze použít pro oba protějšky) TCP spojení.
2. Broker mu sdělí sortiment nabízených služeb.
3. Pokud si klient vybere, musí se autentizovat. K prokazování totožnosti bude použit SASL (RFC 4422).
4. Po úspěšné autentizaci klient zašle informace o sobě a o svých požadavcích na tunel.
5. Broker buď zajistí založení tunelu a pošle klientovi odpovídající parametry, nebo požadavek odmítne (a případně doporučí klientovi jiné servery, u nichž by mohl uspět).

Výměna informací probíhá ve formátu XML. Použitý jazyk je velmi jednoduchý, používá asi deset prvků (tunnel, client, server, broker, address apod.) a jeho DTD zabere půl stránky. Popis TSP najdete v RFC 5572: *IPv6 Tunnel Broker with the Tunnel Setup Protocol (TSP)* a implementaci například ve veřejné tunelovací službě *Freenet6* (*www.freenet6.net*). Zde si můžete stáhnout TSP klienta pro svůj počítač a začít experimentovat.



Obrázek 12.3: Výměna informací o tunelu pomocí TSP

Základní definice tunelování v RFC 4213 popisuje pouze explicitně konfigurované tunely. V jeho předchůdci, jímž bylo RFC 2893, najdete i automatické tunely využívající speciální IPv6 adresy vytvořené z IPv4. Trpěly však řadou omezení, takže je novější dokument opustil ve prospěch propracovanějších způsobů automatického tunelování, konkrétně 6to4.

problémy tunelování

Princip tunelování přináší několik víceméně nevyhnutelných problémů, které komplikují jeho nasazení. Prvním z nich je velikost paketů. Jestliže datagram vezmeme jako data a zabalíme jej do jiného datagramu, celková velikost paketu naroste, protože přibyla hlavička obalujícího datagramu. Konkrétně balíme-li do IPv4, zvětší se datagram o 20 B.

Tím ale můžeme narazit na omezení linkové vrstvy a bude třeba rozložit vytvořený datagram na fragmenty, aby se vešel. Ve výsledku klesá efektivita a dostavují se různé problémy. Například firewally fragmentovaným datagramům příliš neholdují, protože se často rozhodují také podle portů transportní vrstvy, ovšem transportní hlavička je obsažena jen v prvním

fragmentu a zbývající není jak posoudit. Přísně nastavené firewally také často blokují protokol 41 signalizující tunelovaný datagram.

Další okruh problémů plyne z nedokonalé topologie. Tunely často nevedou nejkratší cestou k cíli, ale zanesou datagramy stranou, někdy i dost drasticky. Tímto neduhem trpí zejména statické tunely, jejichž druhý konec je pevně dán, bez ohledu na cílovou adresu datagramu. Lépe se chovají automatické tunely (jako např. 6to4) využívající pro druhý konec různé cílové IPv4 adresy, obvykle automaticky odvozené z IPv6 adres.

Aby svět nebyl tak jednoduchý, obě základní varianty tunelování se z hlediska popsaných problémů doplňují. Zatímco statické tunely jsou předvídatelnější a dají se vyladit tak, aby pokud možno netrpěly problémy s firewally a velikostí paketů, topologicky jsou na tom špatně. Automatické tunely se naopak lépe chovají vzhledem k topologii, ovšem díky svému proměnlivému charakteru více trpí různými formami zakazování a zahazování paketů.

Obecně platí a bylo opakovaně potvrzeno různými měřeními, že tunelované datagramy častěji nedorazí k cíli. Služby, jejichž zprávy procházejí tunelem, pak vykazují nižší spolehlivost.

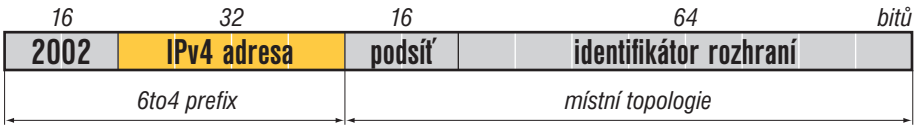
12.3 6to4

Zajímavou a dnes nepochybně nejrozšířenější variantou automatického tunelování je návrh, pro který se vžilo označení *6to4*. Oficiálně se jmenuje „propojování IPv6 domén IPv4 sítěmi“ a jeho definici najdete v RFC 3056: *Connection of IPv6 Domains via IPv4 Clouds*.

Hlavním cílem 6to4 je umožnit koncovým IPv6 sítím vzájemnou komunikaci procházející IPv4 Internetem s minimální konfigurací. Požaduje, aby připojená síť měla k dispozici alespoň jednu veřejnou IPv4 adresu. Tu má přiřazenu 6to4 směrovač, který musí být připojen jak k IPv4 Internetu, tak ke koncové IPv6 síti a procházejí jím veškerá data přepravovaná 6to4. Typické uspořádání vypadá tak, že 6to4 realizuje přístupový směrovač sítě a dotyčnou IPv4 adresou je adresa tohoto směrovače. Není to povinné, roli 6to4 směrovače může hrát celkem libovolný stroj v síti, ale přístupový směrovač bývá pro tuto úlohu ideálním kandidátem.

6to4 prefix

6to4 na základě dané IPv4 adresy vytvoří *IPv6 prefix* délky 48 b pro celou síť. Začíná hodnotou 2002::/16, podle níž lze poznat, že se jedná o prefix pro 6to4. Dalších 32 bitů tvoří IPv4 adresa přístupového směrovače. Vznikne tak prefix standardní délky, který umožňuje adresovat počítače v síti obvyklým způsobem. Strukturu adres používajících 6to4 znázorňuje obrázek 12.4. Kdyby například 6to4 směrovač měl IPv4 adresu 147.230.7.23, vytvořil by pro svou IPv6 síť prefix 2002:93e6:717::/48.



Obrázek 12.4: Struktura 6to4 adresy

Dotyčná IPv6 síť používá 6to4 prefixy jako jedny ze svých IPv6 prefixů (vedle nich může, ale nemusí mít i „seriózní" IPv6 prefix či prefixy) a zdejší počítače se hlásí k dotyčným adresám. Jsou s nimi také uvedeny v DNS.

6to4 směrovač

Směrovač implementující 6to4 pak do vnitřní sítě ohlašuje směrovací informaci, že přes něj vede cesta k síti 2002::/16. Datagramy adresované do jiné 6to4 sítě budou proto předány k doručení jemu. Když se tak stane, vezme si z cílové IPv6 adresy IPv4 adresu 6to4 směrovače vzdálené sítě a datagram mu automaticky tuneluje. Jako zdrojovou adresu použije IPv4 adresu svého odchozího rozhraní (tedy adresu, ze které vychází zdejší 6to4 prefix). Tunel není zakládán trvale, IPv6 paket se podle tunelovacích pravidel jednoduše zabalí do IPv4 datagramu a odešle do Internetu, 6to4 směrovač si pro něj nemusí ukládat žádné stavové informace.

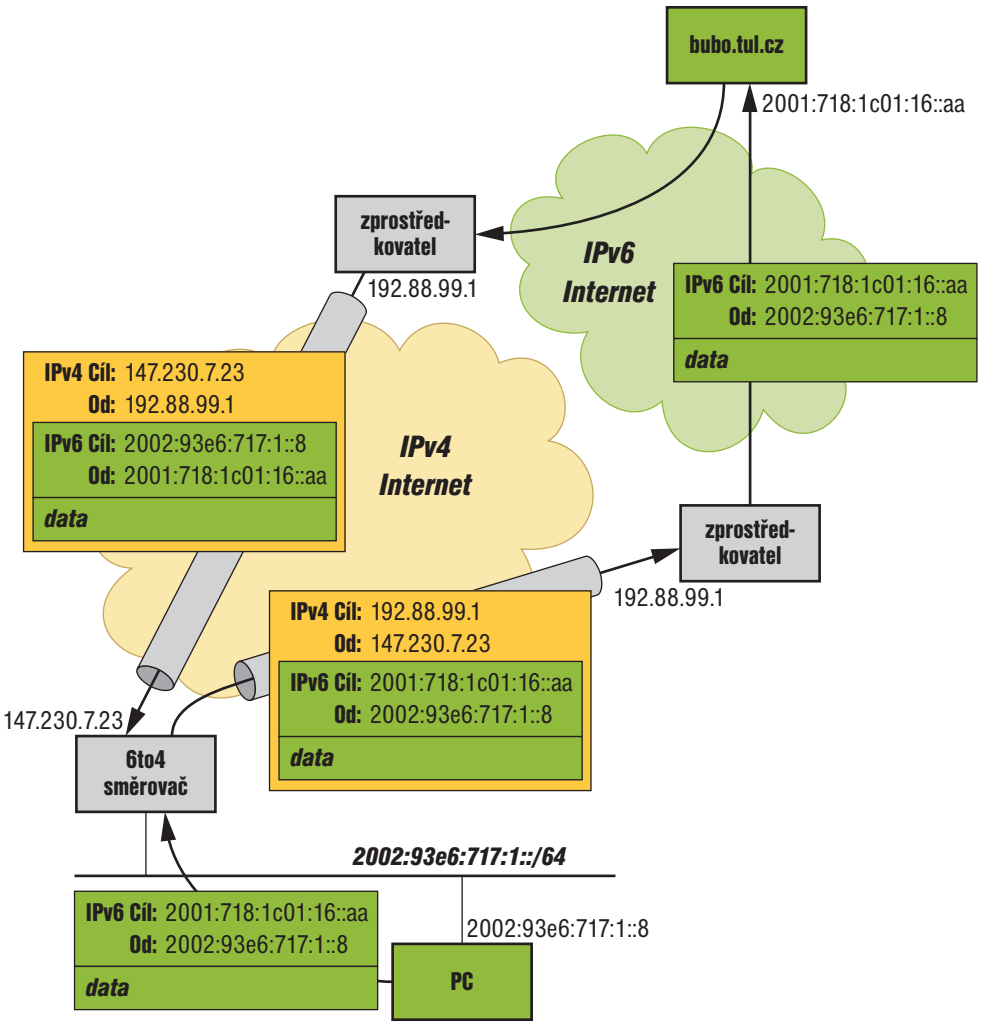
Významnou předností 6to4 je jeho minimální náročnost. Nepotřebuje žádné zásahy do IPv4 směrování a jen minimální do IPv6 směrování (ohlašování 2002::/16 do lokální sítě, zbytek se odehrává na bázi automatických tunelů). Nevyžaduje žádné konfigurované tunely, k životu mu stačí jen přístupový směrovač podporující 6to4 a odpovídající záznamy v DNS.

6to4 a nativní IPv6

Největší problém vzniká při styku s přirozeným IPv6 světem – když spolu potřebují komunikovat stroje, z nichž jeden má pouze 6to4 adresu a druhý jen nativní IPv6 adresu. V takovém případě musí v Internetu existovat alespoň jeden *zprostředkovatel (relay router)*. Potřebuje alespoň jedno 6to4 rozhraní a alespoň jedno nativní IPv6 rozhraní s plnohodnotným připojením.

Jeho pozice vůči nativnímu IPv6 světu je celkem jednoduchá – obvyklými směrovacími mechanismy ohlašuje, že jeho prostřednictvím je dosažitelná síť s prefixem 2002::/16. Je-li takových směrovačů více, standardním způsobem se posoudí, který z nich je pro daný IPv6 uzel nejvýhodnější.

Směrování ze 6to4 sítí do světa nativního IPv6 původní specifikace neřešila příliš dobře. Představte si 6to4 směrovač umístěný v koncové IPv6 síti bez IPv6 konektivity. Potřebuje se dozvědět adresu nejvhodnějšího zprostředkovatele, a to navíc adresu v 6to4 tvaru, aby byl schopen mu automatickým tunelem předávat datagramy od zdejších počítačů. Jednou z možností je, že bude provozovat externí směrovací protokol (čili BGP) a z něj se vše dozví. Jenže takové řešení má velkou režii a klade nezanedbatelné nároky na směrovač i jeho správce, takže je nevhodné pro menší sítě.



Obrázek 12.5: Výměna dat mezi 6to4 a nativním IPv6

RFC 3068: *An Anycast Prefix for 6to4 Relay Routers* přišlo s geniálně jednoduchým řešením. Definovalo fixní výběrovou adresu pro zprostředkovatele. 6to4 směrovač si podle něj nastaví implicitní cestu pro IPv6 na adresu 2002:c058:6301::. Ta podle pravidel 6to4 vede na IPv4 adresu 192.88.99.1, což je výběrová adresa pro všechny 6to4 zprostředkovatele. Standardními mechanismy pro výběrové adresy bude datagram doručen nejbližšímu zprostředkovateli, aniž by bylo nutné páchat nějaké konfigurační násilí.

Výměna dat mezi 6to4 a nativním IPv6 má bohužel silný sklon k asymetrii. Datagramy směřující do nativní IPv6 sítě bude předávat držitel výběrové adresy 192.88.99.1 nejbližší 6to4 odesilateli, zatímco datagramy v protisměru budou předány zprostředkovateli, který je nejblíže protějšímu stroji. Situaci ilustruje obrázek 12.5.

DNS Aby se schopnosti 6to4 využily co nejlépe, je záhodno zařadit jeho adresy do DNS. Mohou fungovat jako jediné IPv6 adresy strojů z koncové sítě, nebo mohou představovat alternativu k regulérním IPv6 adresám získaným standardním způsobem. Převažovat bude zřejmě první případ, protože pokud má koncová síť IPv6 připojení a může využívat globální adresy přidělené lokálním registrem, není valný důvod přidávat k nim 6to4. Ty se hodí především pro sítě, jejichž poskytovatel nepodporuje IPv6 a 6to4 pro ně představuje přímočarou cestu, jak se připojit k IPv6 Internetu.

Záznamy typu AAAA nepředstavují žádný problém, jednoduše v nich bude uvedena 6to4 adresa. Například

```
pc      AAAA  2002:93e6:1707:1:123:45ff:fe67:89ab
```

Větší svízel lze očekávat s reverzním DNS, konkrétně s delegací zodpovědnosti za příslušnou část doménového prostoru na provozovatele koncové sítě. Její 6to4 adresa je odvozena od IPv4 adresy, která pochází od poskytovatele Internetu. Reverzní domény pro IPv4 mají své správce, nabízí se tedy dát odpovídající 6to4 domény do správy stejným subjektům a umožnit jim delegovat poddomény dalším institucím. Problém je v tom, že 6to4 má sloužit zejména sítím, jejichž poskytovatel Internetu se nezná k IPv6 a tudíž těžko bude obhospodařovat reverzní domény pro 6to4. V řetězci delegací tím vypadává článek a celá přirozená konstrukce se hroutí.

RFC 5158: *6to4 Reverse DNS Delegation Specification* zkoumá několik alternativních přístupů k reverzním záznamům pro 6to4, dochází však k závěru, že ten přirozený je přece jen nejlepší. Přichází ale se zcela odlišným systémem delegace poddomén. Vlastník IPv4 adresy si může přímo na centrální adrese

www▸ *https://6to4.nro.net/*

delegovat doménu odpovídající jeho 48bitovému prefixu. Pokud na web přistupujete ze 6to4 adresy, můžete nastavit delegaci jí odpovídajícího prefixu.

Takže například uživatel, který se připojí ze stroje *pc* uvedeného výše, může nastavit autoritativní servery pro doménu *7.0.7.1.6.e.3.9.2.0.0.2.ip6.arpa*.

Šest z pěti bezpečnostních expertů omdlí, když jim popíšete způsob zabezpečení této služby. Nicméně cílem bylo maximálně usnadnit delegace reverzních 6to4 adres. Předpokládá se, že budou sloužit především pro malé sítě, kde jsou všichni jedna rodina a nedělají si naschvály. Sami autoři upozorňují na nedostatky tohoto návrhu, nicméně v rámci možností zatím nikdo nevymyslel nic lepšího. Ostatně, 6to4 by mělo sloužit jen po omezenou dobu. Jakmile bude IPv6 připojení běžně k mání, stane se zbytečným a bude postupně opouštěno.

6to4 v praxi

6to4 se hojně rozšířilo, některé aktivní prvky a operační systémy je dokonce aktivují automaticky, pokud nemají k dispozici nativní IPv6 a disponují veřejnou IPv4 adresou. Reálné zkušenosti bohužel odhalily, že tento mechanismus trpí řadou problémů. Alarmující je jeho špatná spolehlivost – podle měření, která prováděl Geoff Huston či RIPE NCC, se kolem 10 až 15 % požadavků přicházejících na servery po 6to4 nedočká odpovědi.

Příčin je celá řada. Velmi často stojí v pozadí firewally, ať už jim vadí asymetrické směrování mezi 6to4 a nativním IPv6, nebo samotný tunelovací protokol 41. Svůj podíl na problémech má i překračování maximální velikosti paketů způsobované obalující IPv4 hlavičkou, či neveřejné IPv4 adresy, které se navzdory specifikacím občas v 6to4 adresách objevují. Z pozice poskytovatele Internetu je řada těchto problémů bohužel špatně řešitelná. Může sice pro své 6to4 klienty zajistit vlastního zprostředkovatele na výběrové adrese 192.88.99.1, ale nijak neovlivní jeho výběr pro pakety přicházející v protisměru. RFC 6343: *Advisory Guidelines for 6to4 Deployment* obsahuje doporučení – cílené zejména na provozovatele sítí a služeb – jak minimalizovat tyto problémy. Úplně je však odstranit nelze.

6to4 získalo pověst jednoho z nejvýznamnějších zdrojů nespolehlivosti IPv6. Objevují se dokonce snahy o jeho zamítnutí a převedení RFC 3056 a 3068 do historického stavu (*draft-ietf-v6ops-6to4-to-historic*).

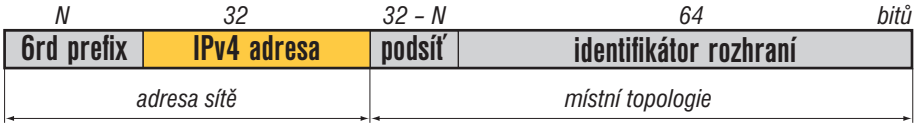
12.4 IPv6 Rapid Deployment (6rd)

Padne-li 6to4, jeho prapor nejspíš převezme 6rd definované v RFC 5569: *IPv6 Rapid Deployment on IPv4 Infrastructures (6rd)*. Stojí na stejných principech jako 6to4, ale odstraňuje jeho provozní nedostatky, protože veškeré klíčové prvky spravuje jeden subjekt. Jeho cílem je, aby poskytovatel Internetu mohl svým zákazníkům nabídnout IPv6, přestože jeho páteřní síť podporuje pouze IPv4.

adresy

6rd také používá speciální adresy, jež v sobě obsahují IPv4 adresy pro tunelování. Místo 6to4 prefixu ale začínají prefixem, který definuje poskytovatel

Internetu a pochází z jeho adresního prostoru. Tento prefix má délku nanejvýš 32 b a je společný pro celou 6rd síť daného poskytovatele. Za ním následuje IPv4 adresa zákazníkova domácího směrovače, na kterou se mají tunelovat data. Jedná se o adresy v rámci poskytovatelovy sítě, takže mohou být i neveřejné. Pokud byl úvodní 6rd prefix kratší než 32 b, zbývá ještě několik bitů pro adresování podsítí.



Obrázek 12.6: Struktura 6rd adresy

Poskytovatel provozuje jednu či několik bran mezi 6rd a nativním IPv6. Směrem dovnitř mohou být adresovány individuálně či výběrově, vše se odehrává v adresním poskytovatelově prostoru, takže záleží jen na jeho rozhodnutí. Do okolního IPv6 Internetu šíří standardními směrovacími protokoly informaci, že jejich prostřednictvím je dostupná síť se 6rd prefixem. Příchozí datagramy pak tunelují a doručují na IPv4 adresy zákaznických směrovačů v nich obsažené.

Zákaznické směrovače musí podporovat 6rd. Jsou konfigurovány tak, aby IPv6 datagramy směřující na adresy začínající 6rd prefixem poslaly tunelem na IPv4 adresu v nich obsaženou, zatímco datagramy adresované kamkoli jinam předávaly – opět tunelem – na některou z bran.

srovnání se 6to4

6rd používá principy 6to4 s odlišným organizačním obalem. Jeho hlavní výhodou je, že vše má pod kontrolou poskytovatel a dokáže tudíž zaručit kvalitu služby pro své zákazníky. Veškerá komunikace mezi jeho 6rd sítí a nativním IPv6 prochází jeho branou (branami). Díky tomu je chování sítě daleko konzistentnější. Nasazení je zároveň velmi snadné, RFC 5569 popisuje příklad francouzského operátora Free, který implementoval 6rd během měsíce. Technologie je v současné době dost populární a řada poskytovatelů po ní mlsně pokukuje, pokud ji už nenasadili.

Zákazník je ovšem na svém poskytovateli závislý. U 6to4 žádnou jeho podporu nepotřebuje. Má-li pevnou veřejnou IPv4 adresu, jednoduše si nakonfiguruje svůj směrovač a může fungovat. To v případě 6rd není možné. Pokud poskytovatel nedefinoval svůj 6rd prefix a neprovozuje příslušnou bránu, nemůže je zákazník nasadit.

Má také k dispozici menší počet adres. 6to4 mu dá k dispozici standardní 48b adresu sítě se 16 b pro adresaci podsítí. U 6rd je takový adresní prostor krajně nepravděpodobný. Poskytovatelé obvykle používají 6rd prefix délky 32 b (což jsou ostatně často všechny IPv6 adresy, jež mají k dispozici), takže už nezbývá místo na adresu podsítě. V případě domácích uživatelů

omezení na jednu podsíť nejspíš nebude vadit, u firemních už by to mohl být problém.

Tady se ovšem otevírá možnost ušetřit na IPv4 adrese. Jelikož se tunelování odehrává v síti jednoho poskytovatele s omezeným adresním prostorem, bylo by možné nevkládat do IPv6 celou IPv4 adresu, ale jen její konec. Pokud například poskytovatelova páteřní síť disponuje prefixem 10.9.0.0/16, stačí do adresy vložit poslední dva bajty a prefix k nim automaticky přidávat. Vzhledem k tomu, že omezené adresování je často diskutovanou vlastností 6rd, lze očekávat, že implementace takové schopnosti nabídnou.

12.5 6over4

Technologie nazvaná *6over4* má podstatně skromnější ambice než obě předchozí. Je zaměřena na izolované počítače podporující IPv6, které se nacházejí uvnitř IPv4 infrastruktury. Cílem 6over4 je umožnit jim plnohodnotnou komunikaci s okolním IPv6 světem, a to opět při co nejmenší míře ruční práce a explicitního konfigurování. Mechanismus je definován v RFC 2529: *Transmission of IPv6 over IPv4 Domains without Explicit Tunnels*. De facto se zde IPv4 používá jako virtuální Ethernet.

princip

Dotyčné počítače musí podporovat jak IPv6, tak IPv4, protože tunelují své IPv6 datagramy do IPv4. Tunelem je posílají směrovači podporujícímu 6over4, který je pak předává dál do IPv6 sítě. Každý počítač využívající 6over4 má svou IPv4 adresu. Jeho IPv6 adresa vznikne tak, že se k prefixu podsítě[2] připojí identifikátor rozhraní, jehož první čtyři bajty jsou nulové a následující čtveřice bajtů obsahuje IPv4 adresu příslušného rozhraní.

Ke své činnosti počítač používá běžné mechanismy IPv6, jako je například objevování sousedů či automatická konfigurace. Řada z nich využívá skupinově adresované datagramy a 6over4 proto musí zajistit, aby fungovalo skupinové adresování.

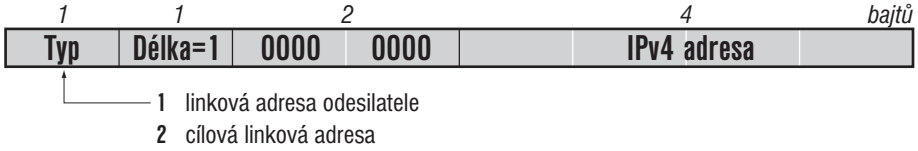
mapování adres

Provádí to velmi jednoduše – prostřednictvím skupinových adres IPv4. Skupinovou IPv6 adresu přímočaře mapuje na IPv4 adresu 239.192.*X*.*Y*, kde *X* a *Y* jsou poslední dva bajty mapované IPv6 adresy. Řada IPv6 skupin tak pochopitelně splyne do jedné IPv4 skupiny a přijímající počítače si musí přebrat, zda jim daný datagram skutečně patří nebo ne. To je však zcela obvyklé a při přenosu skupinově adresovaných datagramů běžným Ethernetem k tomu dochází také.

Důsledkem je, že 6over4 ke své činnosti potřebuje, aby daná IPv4 síť podporovala skupinově adresované datagramy (IPv4 multicast). To v současné době není samozřejmostí a proto je tento požadavek považován za jednu z nevýhod 6over4.

[2] Prefix musí být 64 bitů dlouhý a počítač se jej může dozvědět například z ohlášení směrovače pro bezstavovou automatickou konfiguraci.

Pro mapování individuálních IPv6 adres na IPv4 (počítač se nachází v čisté IPv4 infrastruktuře, takže každý odesílaný IPv6 datagram musí tunelovat a proto potřebuje mapovat cílovou IPv6 adresu na vhodnou IPv4) se používá běžné objevování sousedů. Jedinou specialitou je formát volby pro lokální linkovou adresu, která se přibaluje k výzvě sousedovi a ohlášení souseda. Její podobu vidíte na obrázku 12.7.



Obrázek 12.7: Volba *Linková adresa* pro objevování sousedů při 6over4

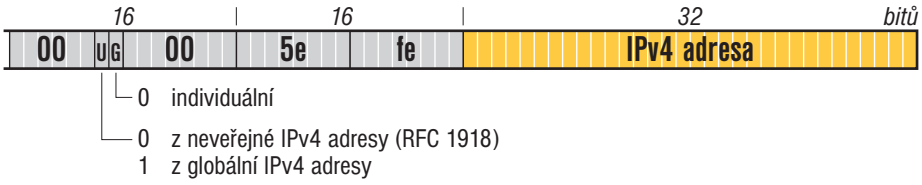
12.6 ISATAP

Vzhledem k požadavku na fungující skupinové adresování v IPv4 jsou výskyty 6over4 dost vzácné. Výrazně častěji se k připojení IPv6 strojů uvnitř IPv4 infrastruktury používá jiný mechanismus, nazvaný *Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)*. Cíl je zde stejný jako u 6over4: umožnit vzájemnou komunikaci IPv6 uzlů, které se nacházejí uvnitř IPv4 sítě, a použít k tomu IPv4 jako linkový protokol.

Již samotný název intra-site naznačuje, že protokol má sloužit uvnitř zákaznické sítě, zatímco komunikaci mezi jednotlivými sítěmi ponechává na jiných. Tvoří tak ideální doplněk například k 6to4. Na rozdíl od 6over4 neklade na IPv4 žádné speciální nároky. Jejich základní princip je ale podobný – IPv6 adresa v sobě obsahuje IPv4 adresu a zařízení podporující ISATAP zabalí datagram podle standardní tunelovací procedury do IPv4 a odešle jej na adresu získanou z cílové ISATAP adresy. Styk s nativním světem zajišťuje ISATAP směrovač, který by měl mít jak IPv4 tak IPv6 připojení.

adresy

ISATAP zavádí speciální formát pro adresu rozhraní, která je vytvořena na základě IPv4 adresy dotyčného stroje. V počátečních 32 bitech obsahuje konstantu 0000:5efe, za níž pak následuje 32 bitů s IPv4 adresou. Přesněji řečeno počátečních 16 b nemusí být nulových, protože osmý bit má obvyklý význam a rozlišuje globálně/lokálně jednoznačný identifikátor (bit *U*). Pokud je IPv4 adresa v identifikátoru rozhraní lokální podle RFC 1918, třeba 10.1.2.3, je identifikátor jednoznačný pouze lokálně, bit U má hodnotu 0 a identifikátor rozhraní tvar 0:5efe:10.1.2.3. Jestliže je ale adresa rozhraní odvozena od globální IPv4 adresy, řekněme 147.230.7.5, je bit U nastaven na jedničku, což vede k identifikátoru 200:5efe:147.230.7.5. Celou strukturu identifikátoru rozhraní představuje obrázek 12.8.



Obrázek 12.8: Identifikátor rozhraní pro ISATAP

Identifikátor rozhraní se připojí k prefixu IPv6 podsítě, který může být nastaven pevně, nebo jej lze získat bezstavovou automatickou konfigurací (hnedle toto téma rozpitvám). Navíc si ISATAP uzel přiřadí obvyklým způsobem lokální linkovou adresu – připojí vytvořený identifikátor rozhraní za prefix fe80::/10. Předpokládejme, že dotyčná síť je připojena pomocí 6to4 a má jen jednu podsíť s prefixem 2002:93e6:1707:1::/64. Uvnitř sítě se používají lokální IPv4 adresy. ISATAP rozhraní našeho počítače by pak mělo adresy

- 2002:93e6:1707:1:0:5efe:10.1.2.3
- fe80::5efe:10.1.2.3

Mimochodem, kombinace 6to4 pro externí tunelování a zajištění komunikace s IPv6 Internetem a ISATAP pro tunelování uvnitř koncové IPv4 sítě je docela silná dvojka, která dovede přivést IPv6 na lecjaká místa.

Určitý problém představuje objevování sousedů. Jelikož na IPv4 nemají být kladeny žádné speciální nároky, znamená to, že po něm nemůžeme chtít doručování skupinově adresovaných datagramů. Objevování sousedů proto nemůže využívat obvyklé skupinové adresy.

V případě hledání linkové adresy některého ze sousedů se tento požadavek odstraní triviálně. Jeho „linkovou adresou" je odpovídající IPv4 adresa, kterou si odesilatel vyzvedne z posledních čtyř bajtů cílové adresy. K tomu nepotřebuje s nikým komunikovat.

konfigurace směrování

Horší je situace s automatickou konfigurací adres a směrovačů. Zde není jak posílat obecné výzvy směrovači a jeho ohlášení. ISATAP proto zavádí novou datovou strukturu – *seznam potenciálních směrovačů (potential router list, PRL)*. Obsahuje IPv4 adresy směrovačů podporujících ISATAP a časovače. Uzel pak periodicky posílá *Výzvu směrovači* cíleně na adresy uvedené v seznamu potenciálních směrovačů. Také jejich ohlášení jsou posílána individuálně na adresu klienta, který se zeptal. Z příchozích ohlášení se ISATAP uzel dozví platné prefixy, nastaví si adresy a implicitní směrování.

Otázku naplnění seznamu potenciálních směrovačů ponechává definice ISATAPu lišácky stranou. Zmiňuje se o možnostech využít manuální konfiguraci, DHCPv4 nebo DNS. Většina implementací sází právě na DNS, kterého se ISATAP klient zkusí dotázat na A záznam pro jméno *isatap* ve své

doméně[3]. Kdyby v síti z našeho příkladu fungovaly dva ISATAP směrovače na adresách 10.1.2.100 a 10.1.2.200, vložil by správce do DNS zóny pro svou doménu záznamy

```
isatap     A  10.1.2.100
           A  10.1.2.200
```

a informoval tak zdejší ISATAP uzly o jejich existenci.

Všední den ISATAP pak probíhá celkem normálně. Když dostane k doručení datagram směřující na adresu s ISATAP prefixem ze stejné podsítě, tuneluje jej po IPv4 přímo adresátovi. Pro ostatní použije běžnou směrovací tabulku – čili zpravidla je předá (opět ISATAP tunelem) některému z implicitních směrovačů. Aby se udržoval v obraze, musí proces se zjišťováním prefixu a implicitních směrovačů pravidelně opakovat a případně se přizpůsobit změněné situaci.

Veškeré detaily najdete v RFC 5214: *Intra-Site Automatic Tunnel Addressing Protocol (ISATAP)*. Citelným omezením protokolu je, že nedokáže přepravovat skupinově adresované datagramy.

12.7 Teredo

Dost nepříjemnou překážku pro řadu tunelovacích mechanismů představuje NAT. Mění adresy a čísla portů, je problém adresovat počítače za ním (zvenčí viditelná adresa se vytvoří, až když vnitřní počítač odesílá data) a navíc řada z nich z bezpečnostních důvodů propustí dovnitř jen data z adresy, na kterou dotyčný stroj nedávno něco odeslal.

Cílem mechanismu nazvaného Teredo je vypořádat se s touto nelichotivou situací. Mimochodem – původně se tento návrh jmenoval shipworm (což je anglické jméno pro sášeň lodní). Ale „worm“ má v počítačovém světě velmi špatné jméno, protože jako červi bývají označovány programy, které se rozlézají elektronickou poštou a nejsou k zastavení. Proto se autoři rozhodli změnit jméno na Teredo (což je zase sášeň, ale tentokrát latinsky).

Základní myšlenka vychází z prostého faktu, že komunikaci je třeba zahajovat zevnitř NATované sítě, aby se na NATu vytvořila odpovídající vazba. Navíc je třeba přizpůsobit chování charakteru NATu, jímž datagramy ke klientovi procházejí. Nejjednodušší je trychtýřový NAT (cone NAT), který klientovi přiřadí určitou adresu a port a propustí k němu pak jakýkoli paket. Složitější je vypořádat se s omezeným NATem (restricted NAT), který klientovi také přiřadí jednu adresu a port, ovšem předá mu datagram jen z té adresy a portu, na něž klient dříve něco poslal. Mají-li klientovi dorazit data

[3] Předpokládá se u něj fungující IPv4, takže může řešit DNS a má nějaké své jméno. Z něj odtrhne úvodní část, nahradí ji řetězcem „isatap“ a v této podobě předloží dotaz DNS.

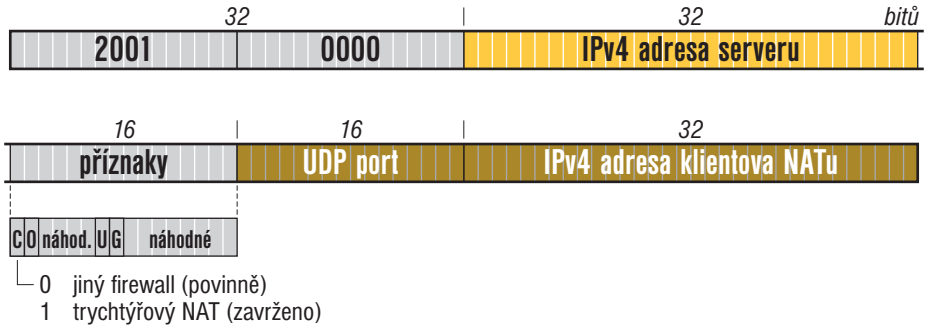
zvenčí, musí nejprve on sám kontaktovat jejich odesilatele. Největší překážkou je symetrický NAT (symmetric NAT), který se chová jako omezený, ale navíc pro data odesílaná k různým cílům přiděluje stejnému klientovi odlišné adresy a porty. S ním se Teredo nedokáže vyrovnat, jedinou šancí je upravit vhodnou konfigurací chování NATu. Pro vlastní přenos dat se používá protokol UDP, který je NATům dobře znám a dovedou s ním zacházet.

adresy Teredo používá dost komplikovaný formát IPv6 adres, které obsahují hned několik kontaktních informací ze světa IPv4. Strukturu adresy vidíte na obrázku 12.9. Začíná pevně daným Teredo prefixem 2001::/32. Za ním následuje IPv4 adresa serveru, který adresu přidělil. Tím je vyčerpáno úvodních 64 bitů, obvyklých pro prefix podsítě. Tato polovina pochází od serveru.

Druhá polovina je ve světě IPv6 interpretována jako identifikátor rozhraní a dává si ji dohromady klient sám. Začíná dvěma bajty příznaků, z nichž význam *U* a *G* je obecný pro všechny identifikátory rozhraní. Původní specifikace (RFC 4380) k nim přidala příznak *C* (Cone), který rozlišoval trychtýřový NAT od jiného a napovídal tak ostatním strojům, jak s klientem komunikovat.

Ukázalo se, že taková informace je k užitku i útočníkům. Proto RFC 5991: *Teredo Security Updates* příznak *C* zavrhlo, nařídilo používat jako jeho hodnotu vždy nulu a chovat se, jako by se stroj nacházel za omezeným NATem. Kromě toho toto RFC zaplnilo 12 bitů v příznacích náhodnými hodnotami, aby omezilo předvídatelnost Teredo adres.

Zbývajících 48 bitů identifikuje venkovní konec NATu, kterým prošla zpráva od klienta. Obsahuje použité číslo UDP portu a veřejnou IPv4 adresu NATu. Vzhledem k tomu, že hrozí určité nebezpečí, že NAT tyto hodnoty v datagramu vyhledá a přepíše, jsou všechny jejich bity invertovány (což naznačuje invertovanými barvami i obrázek 12.9).



Obrázek 12.9: Struktura adresy pro Teredo

klient Počítač, který je v lokální IPv4 síti za NATem a chtěl by získat přístup k IPv6 Internetu, je označován jako Teredo klient. Když zahajuje svůj IPv6 život,

potřebuje nejprve adresu. Obstará si ji ve spolupráci se serverem během tak zvané *kvalifikační procedury (qualification procedure)*. Zahájí ji odesláním obvyklé výzvy směrovači, kterou však zabalí do UDP paketu a odešle prostřednictvím IPv4 na adresu Teredo serveru.

Kde ji vezme, je starostí klienta. Ovšem Teredo serverů je jako šafránu, takže statická konfigurace klienta pravděpodobně postačí. Jedním z nich je *teredo.ipv6.microsoft.com* a pokud používáte některou verzi MS Windows, buďte si jisti, že tohle je váš kůň. Pro Evropu dále připadá v úvahu francouzský *teredo.remlab.net*. Lze očekávat, že servery budou ve vaší implementaci nastaveny a že není třeba na nich nic měnit. Provoz mezi klientem a serverem je malý, takže příliš nezáleží na tom, kam směřuje.

server

Teredo servery sídlí v globálním Internetu a mají připojení k IPv4 i IPv6. Datagram projde NATem, který vytvoří vazbu pro interní adresu odesilatele a přemapuje ji na jinou. Takto upravený paket je následně doručen serveru. Ten se z něj mimo jiné dozví, na jakou veřejnou IPv4 adresu a port byl namapován odesilatel – figurují jako zdroj v hlavičkách IP a UDP.

Server vytvoří ohlášení směrovače, obsahující prefix s výše uvedenou strukturou, a odešle jej tazateli[4]. Své ohlášení opět zabalí do UDP paketu a navíc k němu přidá informaci o odesilateli (origin indication) obsahující IPv4 adresu a UDP port, ze kterého mu dorazila výzva. Jako příjemce uvede adresu a port dosazené NATem.

Součástí kvalifikační procedury původně bylo i zjištění charakteru NATu, za kterým se nachází klient. V počátečních výzvách proto klient ve svém identifikátoru rozhraní nastavil příznak *C* na jedničku a server své ohlášení úmyslně poslal z jiné své IPv4 adresy, než na kterou dorazila výzva. Pokud přesto prošla ke klientovi, je jeho NAT trychtýřový a komunikace mohla být snadnější. Podle RFC 5991 tato fáze odpadá a klient se rovnou chová, jako by neuspěla.

Když klient po několika výzvách nedostane ohlášení (a podle RFC 5991 hned od začátku), nastaví ve své adrese příznak *C* na nulu a pokračuje ve vyzývání. Tentokrát server pošle ohlášení z adresy, na niž mířila výzva. Pro tuto dvojici IPv4 adres a UDP portů existuje na NATu záznam, takže paket bude propuštěn a předán klientovi. Ten vzápětí pošle další výzvu na jinou IPv4 adresu Teredo serveru, příznak *C* ve své adrese ponechá nulový. Porovná pak informaci o odesilateli přibalenou k oběma ohlášením. Je-li shodná, je jeho NAT omezený a Teredo lze použít, i když bude potřeba speciálními pakety otevírat cesty. Pokud se však zdrojové adresy liší, je klientův NAT symetrický a Teredo končí.

[4] 64bitový prefix obsahuje pouze konstantní začátek a IPv4 adresu serveru, je tedy stejný pro všechny klienty obsluhované tímto serverem.

Z *Ohlášení směrovače* a připojené informace o zdrojové adrese získá klient vše potřebné pro vytvoření Teredo adresy – IPv4 adresy serveru i NATu, použitý UDP port i hodnotu příznaku *C*. Vytvořením Teredo adresy na straně klienta končí kvalifikační procedura. Je možné do ní zahrnout i bezpečnostní prvky, ale to s dovolením nechám stranou.

Teredo–Teredo

Pokud spolu komunikují dva Teredo klienti, vyměňují si data přímo. Ideální je, když cílová Teredo adresa má nastaven příznak *C*, tedy adresát se nachází za trychtýřovým NATem přijímajícím data odkudkoli. Protější klient jednoduše zabalí IPv6 datagram do UDP zprávy, kterou pošle na adresu a port uvedené na konci adresy.

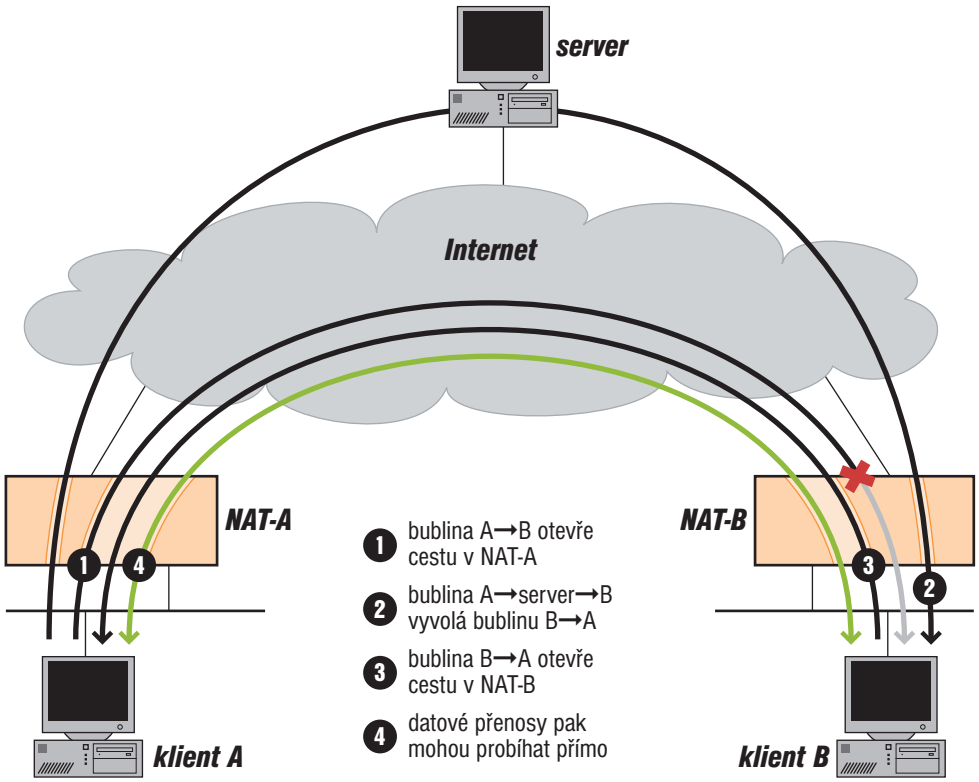
Je-li příznak *C* nulový, znamená to, že NAT cílového stroje je omezený a propustí k lokálnímu stroji jen odpovědi z těch adres a portů, na které sám posílal data. V takovém případě odesilatel pošle dvě tak zvané bubliny. Jedná se o prázdné zprávy, jejichž jediným cílem je projít NATem a vytvořit v něm odpovídající mapování. Jednu bublinu pošle přímo – ta vytvoří záznam ve zdejším NATu (aby mohla přijít odpověď), dorazí k cílovému a tam bude zahozena, protože pro tuto externí adresu zde ještě není mapování.

Druhou bublinu pošle nepřímo na adresu Teredo serveru, který ji pouze předá ze své adresy na cílovou adresu a port. Pro server mapování existuje již od kvalifikační procedury, druhá bublina proto projde k cílovému stroji a ten se dozví, že s ním chce někdo komunikovat. Odpoví proto bublinou adresovanou odesilateli, která otevře cestu v jeho NATu. Nyní mají oba NATy vytvořeno mapování a propustí UDP provoz mezi oběma komunikujícími stroji. Vlastní komunikace už pak může probíhat přímo. Celou proceduru vidíte na obrázku 12.10.

Teredo–IPv6 relay

Pokud si klient potřebuje psát s partnerem obdařeným jinou než Teredo adresou, je situace komplikovanější. Zde vstupuje do hry další z prvků Teredo architektury – *zprostředkovatel, čili relay*. Jedná se o směrovač, který je ochoten předávat datagramy z IPv6 do Teredo světa a naopak. Do IPv6 sítě šíří směrovací informaci, že jeho prostřednictvím je dostupný Teredo prefix 2001::/32.

Teredo klient potřebuje před odesláním paketu do nativní IPv6 sítě nejprve najít vhodného zprostředkovatele. Vytvoří proto ICMPv6 zprávu echo (čtěte ping), zabalí ji do UDP a pošle svému Teredo serveru. Ten ji jednoduše vybalí a odešle IPv6 sítí adresátovi. Odpověď svou cestu začíná také normálně a díky směrovacím protokolům je doručena Teredo zprostředkovateli, který je nejblíže odpovídajícímu stroji. Její předání do Teredo světa opět závisí na příznaku *C* v cílové adrese. Je-li nastaven, pošle zprostředkovatel odpověď přímo adresátovi (přes jeho NAT). V opačném případě ji pošle nepřímo přes adresátův Teredo server. Klient se z ní dozví adresu zprostředkovatele, na jehož IPv4 adresu následně pošle paket, kterým mu otevře cestičku ve svém NATu.



Obrázek 12.10: Zahájení přenosu mezi dvěma Teredo klienty

V obou případech klient získal adresu zprostředkovatele a následně už ví, na kterou IPv4 adresu a UDP port posílat zabalené datagramy pro daného IPv6 adresáta.

seznam partnerů

Bylo by krajně nepraktické provádět popsané tanečky s otevíráním NATů pro každý datagram. Teredo klient si proto udržuje seznam svých aktuálních komunikačních partnerů. V něm si uchovává informace, jak s nimi komunikovat. Pokud pro určitý cíl najde platnou položku v tomto seznamu, naloží s datagramem podle pokynů v ní obsažených.

Autoři návrhu si jsou vědomi, že vyžaduje značnou režii a je poněkud těžkopádný. Své řešení představují jako „krabičku poslední záchrany“ a požadují, aby stroj dal přednost jinému způsobu připojení, pokud je má k dispozici. Když ale nic lepšího není k mání, je tady Teredo.

Jeho specifikaci obsahuje RFC 4380: *Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs)*.

Jeho klíčové klady a zápory mohu pěkně demonstrovat na svém domácím připojení. ADSL modem, NAT v něm ani síť poskytovatele pochopitelně o IPv6 nikdy neslyšeli. Trvalo mi tak pět minut stáhnout si implementaci Teredo pro Linux a nainstalovat ji, konfigurovat se ani nemusela[5]. Jakmile ji teď spustím, okamžitě mám připojení k IPv6. Jednodušší to být nemůže, pomalejší však také ne – *ping* na stejný stroj se po IPv4 vrátí za 16 ms, po IPv6 za 240 ms.

Měření, jež provedl Geoff Huston[6], navíc ukazují neuvěřitelnou nespolehlivost. Bezmála 40 % pokusů o navázání TCP spojení z Teredo klientů skončí neúspěšně. Teredo by skutečně mělo být bráno jako nástroj lehkého experimentování a úplně poslední možnost, jak se snažit o seriózní připojení k IPv6.

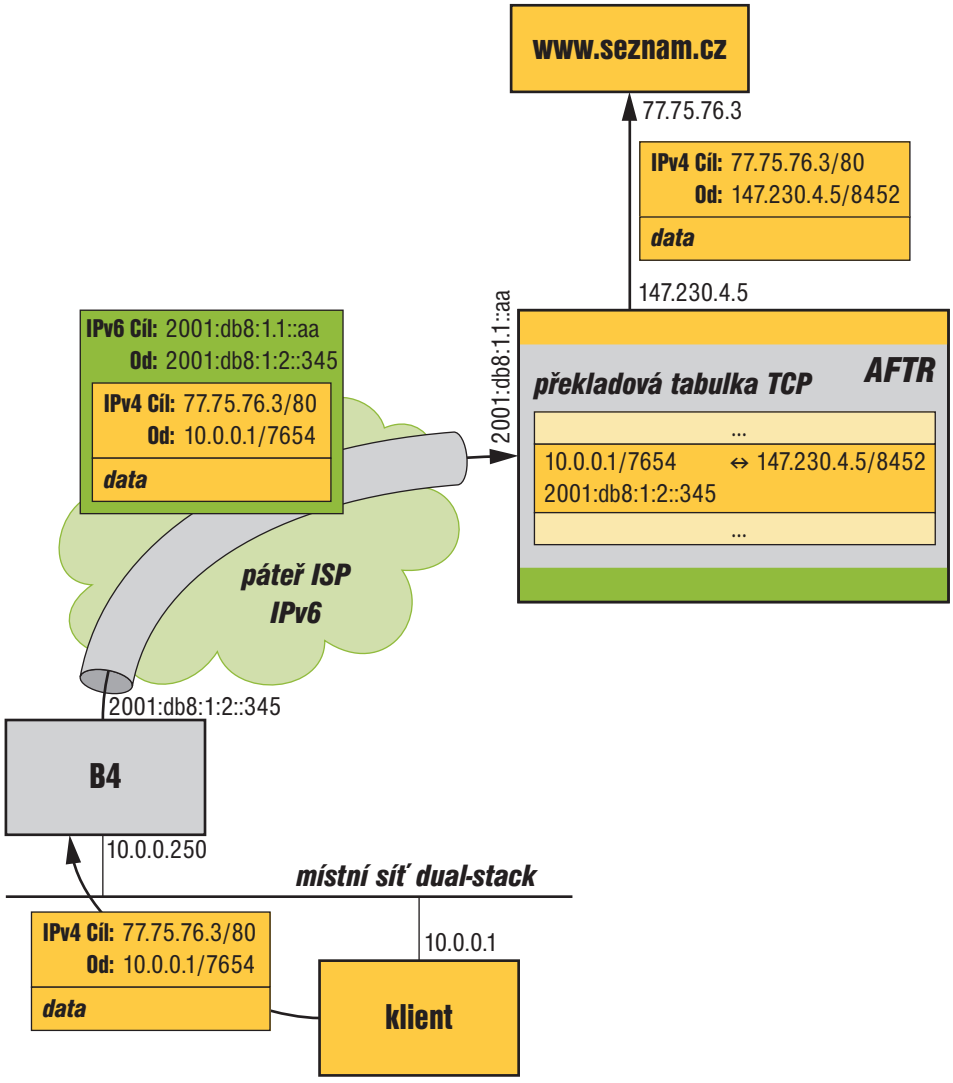
12.8 Dual-Stack Lite

Dual-Stack Lite (též DS Lite) patří mezi nejnovější přechodové mechanismy. Reaguje na nedostatek IPv4 adres a připravuje půdu pro situaci, kdy poskytovatel Internetu vyhodnotí jako jednodušší provozovat svou páteřní síť výlučně protokolem IPv6. Dual-Stack Lite umožní i v takovémto uspořádání poskytovat zákazníkům oba protokoly. Proto se v jeho názvu objevuje spojení dual-stack, přestože se jedná o ryzí tunelovací mechanismus. Tentokrát ovšem funguje obráceně – tuneluje IPv4 páteřní sítí podporující jen IPv6. Definici dual-stack lite najdete v RFC 6333: *Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion*.

princip Základní koncept staví na IPv6 síti, která je provozována nativně a přímočaře. Zákazníci mají veřejné IPv6 adresy, jejich domácí směrovače nic nemění, pouze směrují. S IPv4 je situace složitější. Koncové sítě zákazníků pracují v režimu s dvojím zásobníkem. Je málo IPv4 adres, takže samozřejmě používají privátní adresy podle RFC 1918. Při odchodu z domácí sítě jsou zabaleny do IPv6 a doručeny tunelem na poskytovatelův centrální NAT. Ten je jediným prvkem sítě disponujícím veřejnými IPv4 adresami. Rozbaluje IPv4 datagramy, překládá jejich adresy na své vlastní a odesílá je do veřejného IPv4 Internetu. V opačném směru pak provádí analogický postup.

B4 Zákaznický prvek dual-stack lite sítě autoři pojmenovali *Basic Bridging BroadBand (B4)*. Nabízí obvyklou sadu služeb pro IPv4, neprovádí však NATování. IPv4 datagramy směřující mimo zákaznickou síť jen zabalí do IPv6 a doručí tunelem centrálnímu NATu. Jeho IPv6 adresu (či doménové jméno) může mít pevně nastavenu v konfiguraci, nebo se ji dozví od poskytovatele prostřednictvím DHCPv6. Pro tento účel byla definována nová volba – viz RFC 6334: *Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Option for Dual-Stack Lite*.

[5] S MS Windows Vista není třeba ani to, Teredo je rovnou v systému.
[6] http://www.potaroo.net/ispcol/2011-04/teredo.html



Obrázek 12.11: Dual-stack lite

Vůči zákaznickým počítačům vystupuje jako implicitní brána z jejich sítě. Provozuje DHCPv4 server, jehož prostřednictvím jim dodává IPv4 adresy (z rozsahu podle RFC 1918) i konfigurační parametry. Domácí prvek by také měl provozovat DNS proxy, aby případné klientské dotazy přicházející po IPv4 převedl do IPv6[7].

AFTR Centrální NAT nese název *Address Family Transition Router (AFTR)*. Vytváří druhý konec tunelů vedoucích od zákaznických B4 a implementuje IPv4 NAT (přesněji NAPT), kdy zákaznické privátní adresy a porty překládá na své vlastní veřejné IPv4 adresy a vhodné porty. Proti běžným IPv4 NATům má ovšem jednu významnou odlišnost: k identifikaci zákaznického stroje přidává IPv6 adresu jeho B4. Díky tomu ví, kam poslat tunelem příchozí odpověď, a zároveň mu nevadí, když zákazníci používají konfliktní adresy. Rozsah 10.0.0.0/8 dnes najdete bezmála v každé domácnosti, takže adresu 10.0.0.1 uvidí velmi často. Díky kombinaci s (veřejnou, tedy unikátní) IPv6 adresou B4 daného zákazníka ovšem dokáže rozlišit, o které rozhraní se jedná.

Pokud aplikační protokol používá IP adresy, musí AFTR implementovat příslušný algoritmus pro jeho změnu. RFC 6333 připouští, že z kapacitních důvodů pravděpodobně nebude možné podporovat všechny možné aplikační protokoly a jejich sortiment proto bude více či méně omezený.

Jako každý tunelovací mechanismus, i dual-stack lite má problémy s velikostí datagramů a fragmentací. Autoři doporučují, aby MTU páteřní sítě bylo alespoň o 40 B větší než MTU koncových sítí a k fragmentaci pokud možno nedocházelo. Není-li vyhnutí, musí fragmentace proběhnout na úrovni obalujícího IPv6, tunelované IPv4 datagramy musí zůstat v původní podobě.

Pro přímou komunikaci AFTR s B4 po IPv4 vyhradila IANA adresní rozsah 192.0.0.0/29, přičemž standardní adresou AFTR je 192.0.0.1 a B4 rozhraní 192.0.0.2. Mají všichni stejnou, protože IPv4 provoz je beztak tunelován, takže k rozlišení jednotlivých B4 poslouží jejich IPv6 adresy.

12.9 Stateless IP/ICMP Translation Algorithm (SIIT)

Stejně jako v případě tunelování, i pro překlad paketů mezi IPv4 a IPv6 existuje celá řada mechanismů. V jejich jádru se ale typicky skrývá stejný mechanismus – obecná sada pravidel, jak překládat kterou položku v hlavičkách. Je žádoucí, aby vlastní překlad probíhal ve všech případech stejně a lišil se jen jeho kontext. Proto vzniklo RFC 2765: *Stateless IP/ICMP Translation Algorithm (SIIT)*, které tyto společné principy definovalo.

Později je nahradilo RFC 6145: *IP/ICMP Translation Algorithm*, které věcný obsah příliš nezměnilo. Základní překlad zůstává bezstavový a nevyžaduje

[7] Tím je míněn jen transportní protokol. Typy dotazovaných záznamů nijak nemění, jen přepošle totožný DNS dotaz protokolem IPv6 serveru nabídnutému poskytovatelem.

uchovávání datových struktur s informacemi o historii či aktuálním stavu probíhající komunikace. Každý paket je překládán samostatně, bez vazeb na datagramy předešlé.

mapování adres

Nezbytnou součástí překladu je i mapování adres – pokud se má IPv4 datagram převést na IPv6, je třeba původní IPv4 adresy odesilatele a příjemce převést na vhodné IPv6 adresy. Při převodu opačným směrem je převod adres ještě mnohem těžší, protože adresy z mnohem většího prostoru IPv6 musí být vyjádřeny pomocí IPv4.

Původní SIIT pro tento účel zaváděl dva speciální formáty IPv6 adres, jež v sobě obsahovaly IPv4 adresy: *IPv4-mapované adresy (IPv4-mapped address)* měly tvar ::ffff:a.b.c.d, kde a.b.c.d je IPv4 adresa, a používaly se pro uzly nepodporující IPv6, s nimiž lze komunikovat jen po IPv4. *IPv4-překládané adresy (IPv4-translated address)* byly ve tvaru ::ffff:0:a.b.c.d a obsahovaly dočasnou IPv4 adresu IPv6 uzlů.

Aktuální překladový algoritmus je nahradil *adresami s vloženými IPv4 (IPv4-embedded)*, jež mají vyhrazen prefix – obvykle 96bitový – a za něj přidávají IPv4 adresu. Může se jednat buď o univerzální prefix 64:ff9b::/96, nebo lokální prefix přidělený místním správcem. Jejich podrobný popis najdete na straně 55. Vzhledem k mnohem delším adresám je převod IPv4 na IPv6 velmi snadný a lze jej provádět bezstavově – jednoduše se za daný IPv6 prefix připojí IPv4 adresa.

Opačný směr tak snadný není, protože IPv6 adres je mnohonásobně více. K jejich převedení na IPv4 se proto obvykle používá dynamické mapování podobné převodu privátních IPv4 adresa na veřejné v současných NATech. Problematikou dynamického mapování se ale SIIT nezabývá, ponechává ji na ostatních přechodových mechanismech, jako je například NAT64.

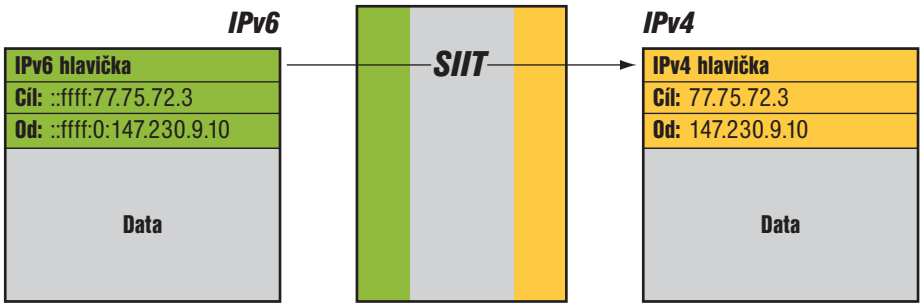
V principu jsou myslitelné i jiné přístupy. Příkladem oboustranného bezstavového mapování je algoritmus IVI používaný v čínské akademické síti CERNET a zdokumentovaný v RFC 6219. Vlastní překlad podle SIIT je na způsobu mapování nezávislý. Jednoduše využívá mapovač jako černou skříňku, které předá IPv4 adresu a dostane odpovídající IPv6 či naopak.

překlad datagramu

Hlavní devizou SIIT jsou jeho přesná pravidla pro vzájemný překlad jednotlivých hlaviček mezi oběma protokoly. Jsou samozřejmě velmi omezená, nepodporují žádná rozšíření – volby ze strany IPv4 a rozšiřující hlavičky ze strany IPv6 SIIT zahazuje. Zapomeňte na mobilitu či IPsec mezi oběma světy, jejichž řešení by byl opravdový oříšek. Na druhé straně je lepší alespoň omezená služba než nic.

IPv4 → IPv6

Základní pravidla pro převod IPv4 datagramu na IPv6 vidíte v tabulce 12.2. Nesená data zůstávají beze změny, jedinou výjimkou je případný výpočet nových kontrolních součtů v transportních hlavičkách, protože do nich jsou zařazeny i IP adresy. Původní IPv4 hlavička je však nahrazena IPv6 hla-



Obrázek 12.12: Průchod datagramu SIIT překladačem

vičkou. Naplnění jejích jednotlivých položek je dost přímočaré, nejsložitější je vypořádat se s fragmentací, jejíž chování se mezi oběma protokoly liší. Je-li zakázáno fragmentovat IPv4 datagram (má nastaven příznak DF), jednoduše se přeloží a nic dalšího není třeba řešit. Případná pozdější ICMPv6 chyba oznamující překročení MTU bude přeložena do ICMPv4 a předána odesilateli. Pokud by se vytvořený IPv6 datagram nevešel hned do odchozího rozhraní, pošle ICMPv4 chybu rovnou sám překladač.

Verze	= 6
Třída povozu	= *Typ služby*
Značka toku	= 0
Délka dat	= *Celková délka* – délka IPv4 hlavičky
Další hlavička	= *Protokol*, hodnotu 1=ICMPv4 změnit na 58=ICMPv6
Max. skoků	= *TTL* – 1
Adresy	= podle mapování

Tabulka 12.2: SIIT – naplnění IPv6 hlaviček při překladu

Při povolené fragmentaci by překladač měl vytvářené IPv6 datagramy fragmentovat tak, aby jejich velikost nepřesáhla 1280 B – minimální velikost, již musí podporovat každá IPv6 linka. Rozšiřující hlavičku *Fragmentace* by měl připojit i v případě, kdy výsledný paket nepřekročí 1280 B, aby zdůraznil, že odesilatel povolil fragmentaci. Takové chování není zrovna optimální, proto autoři doporučují poskytnutí konfiguračních parametrů pro nastavení maximální velikosti vytvářených IPv6 datagramů a vypnutí vkládání nepotřebných fragmentačních hlaviček.

IPv6 → IPv4 Převod opačným směrem probíhá podobně. Pravidla pro naplnění vytvářených IPv4 hlaviček obsahuje tabulka 12.3. Platí pro případ, kdy překládaný IPv6 datagram neobsahuje hlavičku *Fragmentace*. Pokud se jedná o fragment, budou související hlavičky naplněny mírně odlišně: *Celková délka* se proti tabulce zmenší o 8, protože *Délka dat* zahrnuje i osmibajtovou frag-

mentační hlavičku. *Identifikace* převezme spodních 16 b položky *Identifikace* z rozšiřující hlavičky *Fragmentace*. Do příznaku *MF* se zkopíruje hodnota příznaku *M*, příznak *DF* je vynulován. *Posun fragmentu* se převezme z hlavičky *Fragmentace*.

Verze	= 4
Délka hlavičky	= 5
Typ služby	= *Třída provozu*
Celková délka	= *Délka dat* + 20
Identifikace	= 0 nebo vygenerované číslo
Příznaky	= 0
Posun fragmentu	= 0
TTL	= *Max. skoků* – 1
Protokol	= transportní protokol z IPv6, hodnotu 58=ICMPv6 změnit na 1=ICMPv4
Kontrolní součet	= vypočítat standardním algoritmem
Adresy	= podle mapování

Tabulka 12.3: SIIT – naplnění IPv4 hlaviček při překladu

Jelikož je přístup IPv4 k fragmentaci benevolentnější, není třeba se jí tolik zabývat. Je-li zakázána a vytvořený IPv4 datagram je větší než MTU odchozího rozhraní, odešle překladač odesilateli pomocí ICMPv6 chybovou zprávu o překročení přípustné velikosti.

ICMP

Pravidla pro překlad ICMP jsou poněkud složitější, protože ne všechny typy zpráv jsou ve druhém protokolu podporovány (v takovém případě jsou překladačem potichu zahozeny). Jádro je ale stejně přímočaré jako při překladu datagramů. Některé ICMP zprávy obsahují ve svém těle datagram (či jeho část), který zprávu vyvolal. Musí být přeložen stejně jako běžné datagramy.

Koulí na noze SIIT je, že neřeší důležité problémy: vazbu mezi IPv6 a IPv4 adresami a její odraz v DNS. To z něj činí spíše jakýsi polotovar (díky precizně definovaným pravidlům pro překlad datagramů), který využívají jiné překladové mechanismy. Na samotný SIIT však v síti prakticky nenarazíte.

12.10 Network Address Translation – Protocol Translation (NAT-PT)

Z mechanismů, které staví na SIIT a doplňují k němu chybějící součásti, začnu dnes již odmítnutým NAT-PT. Byl sice opuštěn, ale s jeho implementacemi se dosud můžete setkat a za pozornost stojí i jeho snaha o komplexní řešení problému. Snažil se v jednom zařízení kompletně vyřešit problém překladu mezi IPv4 a IPv6 a využít při tom zkušenosti s provozem překladačů adres, jež jsou dnes běžným prvkem IPv4 sítí.

Byl dlouho považován za jeden z klíčových přechodových mechanismů. Velká očekávání však nenaplnil, spíše se v praxi projevily různé jeho nedostatky. Většina z nich plyne ze švindlování s DNS, jež NAT-PT musí provádět, aby se dalo navázat také spojení dovnitř překládané IPv6 sítě. Výsledkem byla jeho poprava v RFC 4966: *Reasons to Move the Network Address Translator – Protocol Translator (NAT-PT) to Historic Status*, které NAT-PT prohlásilo za odmítnutý a historický.

Toto rozhodnutí však vyvolalo ostré kritiky, podle nichž vypovídá spíše o kvalitách IETF než o kvalitách NAT-PT. Faktem je, že za něj dlouho neexistovala rovnocenná náhrada – zařízení, které posadíte mezi koncovou IPv6 síť a IPv4 Internet a ono zajistí transparentní (byť v ledasčem problematickou) komunikaci mezi nimi. Teprve čtyři roky po odmítnutí NAT-PT vyšlo RFC 6146: *Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers* standardizující jeho nástupce. Vzhledem k historickým kořenům a některým zajímavým myšlenkám popíši nejprve NAT-PT a NAT64 se budu věnovat v části 12.11 na straně 284.

Základním stavebním kamenem je směrovač podporující NAT-PT (bývá označován jako NAT-PT překladač, translator). Všechny datagramy tvořící souvislé spojení (například jedna TCP seance) musí procházet stejným NAT-PT překladačem. Proto je pro tuto roli ideální přístupový směrovač jednoduché IPv6 sítě, který je jejím jediným spojením s okolním světem. Ostatně valná většina koncových sítí má takto jednoduchou topologii a poskytuje tudíž vhodné prostředí pro nasazení NAT-PT.

adresování

NAT-PT překladač musí mít k dispozici určitý prostor IPv4 adres, které přiděluje jednotlivým IPv6 strojům ze „své“ sítě. Toto přidělování může být statické (danému IPv6 stroji se přiděluje vždy stejná adresa) či dynamické (adresy se přidělují náhodně podle okamžité potřeby). V obou případech si NAT-PT překladač musí udržovat *stavovou informaci* o přidělených adresách, aby se choval konzistentně. Musí zajistit, aby se mapovaná adresa během komunikace nezměnila.

Kromě sady IPv4 adres potřebuje NAT-PT překladač ještě IPv6 prefix, který bude přidělovat IPv4 adresám z vnějšího světa při překladu do IPv6. Ten může být celkem libovolný, ale musí být v dané síti vyhrazen pro účely NAT-PT. Dále jej budu označovat jako *PrefiX*. NAT-PT překladač inzeruje do lokální IPv6 sítě směrovací informaci, že jeho prostřednictvím je dostupná síť s tímto *PrefiXem*.

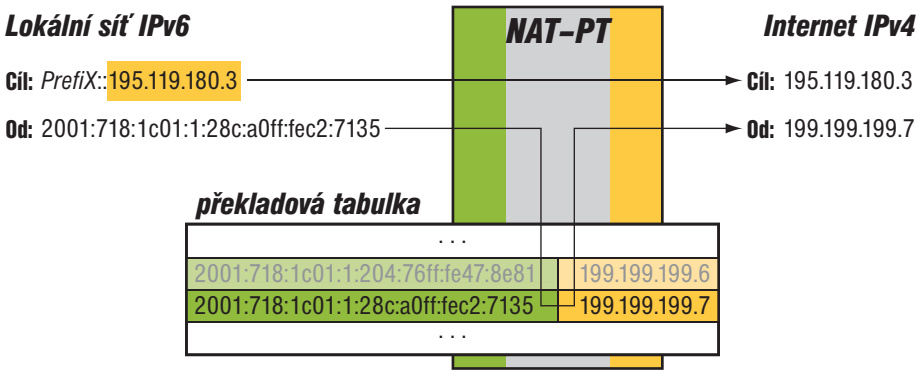
překlad

Popíši nejprve základní chování NAT-PT. Když překladači dorazí IPv6 datagram, jehož cílová adresa začíná *PrefiXem*, provede následující kroky:

1. Pokud datagram zahajuje spojení, přidělí jeho odesilateli IPv4 adresu ze sady, kterou má k dispozici. Informaci o přidělení adresy si uloží. Jestliže datagram není první v daném spojení, má uloženy údaje o mapování jeho adres a pouze je použije.

2. Převede IPv6 datagram na IPv4. Cílovou adresu získá z posledních čtyř bajtů cíle původního datagramu, jako odesilatele dosadí IPv4 adresu získanou v předchozím kroku. Pro převod použije pravidla SIIT.

V opačném směru – když z IPv4 sítě dorazí datagram směřující na jednu z mapovaných IPv4 adres – postupuje analogicky. Převede jej na IPv6 datagram, kde cíl je stanoven podle uložených údajů o mapování a odesilatel je ve tvaru *PrefiX*::*odesilatel_IPv4*, a předá do „své“ IPv6 sítě.



Obrázek 12.13: Mapování adres v NAT-PT

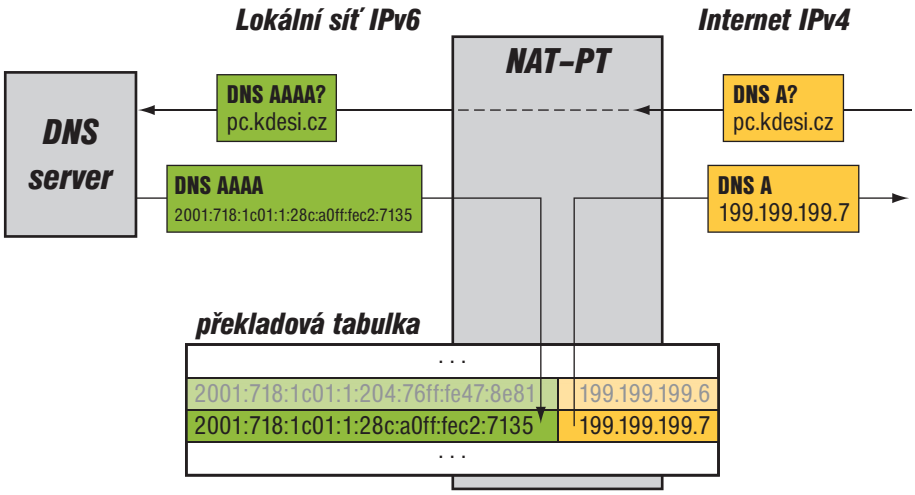
zásahy do DNS

Klasický model NAT má jednu zásadní nevýhodu – umožňuje navazovat spojení jen ve směru z IPv6 sítě ven. Důvodem je, že vnitřní počítače nemají vůči IPv4 světu pevné adresy. Když by přišel datagram na některou z dosud nepřidělených mapovaných IPv4 adres, nevědělo by se, kterému počítači vlastně patří.

Má-li NAT-PT umožnit navazování spojení v obou směrech, musí zasahovat i do DNS. V takovém případě je třeba, aby autoritativní DNS servery pro NATovanou síť byly samy umístěny v této síti a aby tedy všechny dotazy a odpovědi musely procházet stejným NAT-PT překladačem jako provoz z/do dané sítě.

Když NAT-PT překladač obdrží DNS dotaz směřující na některý ze zdejších serverů, změní v něm typ požadavku z A na AAAA. Jakmile dorazí odpověď, provede mapování IPv6 adresy v ní obsažené na IPv4. Původnímu tazateli pak odešle odpověď s A záznamem obsahujícím právě namapovanou IPv4 adresu. Sám si zapamatuje provedené mapování, aby věděl, komu má předávat datagramy na ni přicházející.

Součástí úpravy DNS odpovědi je i změna její životnosti na nulu, aby se neukládala ve vyrovnávacích pamětech a při příštím dotazu se tazatel opět musel obrátit na autoritativní server. To umožňuje, aby se mapování dynamicky měnilo.



Obrázek 12.14: Úprava DNS v NAT-PT zařízení

Změna DNS v opačném směru je poněkud komplikovanější. Když kdosi z vnitřní sítě posílá do vnějšího světa DNS dotaz na adresu k určitému jménu, nelze předem říci, zda odpovědí bude IPv6 nebo IPv4 adresa. Proto NAT-PT předá původní dotaz beze změny a navíc vytvoří jeho kopii, ovšem s dotazem na záznam typu A místo původního AAAA. Odpovědi obsahující AAAA propouští beze změny. Dorazí-li odpověď s A záznamem, změní ji na AAAA a adresu v ní upraví na tvar *PrefiX*::*původní_IPv4_adresa*.

NAPT-PT

Tradiční NAT-PT se zabývá pouze překladem adres. To znamená, že ke každé momentálně komunikující IPv6 adrese musí mít jednu IPv4 adresu, na kterou ji mapuje. O krok dál jde *NAPT-PT (Network Address Port Translation + Protocol Translation)*, který mapuje i čísla portů, ICMP identifikátory a další transportní identifikátory. To umožňuje, aby se například celá IPv6 síť mapovala na jedinou IPv4 adresu. NAPT-PT překladač si pochopitelně musí udržovat o něco více stavových informací (mapují se dvojice adresa+port, nikoli jen adresy). Pojem NAPT-PT se však používá jen výjimečně, obvykle jsou i tato zařízení označována jako NAT-PT a překlad portů se považuje za samozřejmost.

problémy s aplikacemi

Nemalou komplikaci pro NAT-PT představují aplikační protokoly, které přenášejí IP adresy jako součást svých dat. Naštěstí jich není mnoho, ale existují. Jako konkrétní příklad je v RFC 2766 rozebíráno FTP, jehož příkazy PORT a PASV obsahují IPv4 adresu a TCP port. Pro FTP existuje dokonce samostatné RFC 2428: *FTP Extensions for IPv6 and NATs*, které se zabývá řešením tohoto problému. Doporučuje nahradit původní příkazy jejich rozšířenými variantami EPRT a EPSV.

Pokud IPv4 stroj zahajující spojení nepoužívá tyto nové příkazy, musí je NAT-PT překladač nahrazovat. Bohužel toto mapování na aplikační úrovni zpravidla znamená, že se změní délka přenášených dat. NAT-PT překladač tudíž musí upravovat pořadová čísla a potvrzení pro TCP, přepočítávat kontrolní součty a dělat další dost náročné operace, které dále zvyšují počet uchovávaných stavových informací.

Obecně platí, že pokud aplikační protokol přenáší údaje ze síťové vrstvy jako součást svých dat, představuje pro něj NAT-PT překážku. Do překladače je nutno doplnit podporu daného protokolu a měnit data aplikační vrstvy. Pokud NAT-PT překladač nebude provádět tyto úpravy, bude pro danou aplikaci neprůchodný. Přehled o tom, kde všude jsou v různých internetových protokolech a službách zavrtány IPv4 adresy, poskytuje sada RFC 3789–3796: *Survey of IPv4 Addresses in Currently Deployed IETF Standards Track and Experimental Documents*.

12.11 NAT64 a DNS64

Nejvážnější problémy NAT-PT způsobovaly jeho hrátky s DNS, a to především pro dotazy směřující z IPv4 do IPv6 sítě. Ty na jedné straně umožňovaly navázat spojení do IPv6 sítě, ovšem cenou za ně byla řada omezení a hrozících nekonzistencí. Mnoho důvodů pro odmítnutí NAT-PT v RFC 4966 má kořeny v manipulaci s DNS.

NAT64

Ovšem vzhledem ke zjevným výhodám zařízení typu „konvertor mezi IPv4 a IPv6 pro celou síť v jedné krabici“ začal záhy vznikat jeho nástupce nazvaný *NAT64*. Trvalo ovšem čtyři roky, než vyšlo RFC 6146: *Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers* a jeho dporovodné RFC 6147: *DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers*.

Jeho jádro je prakticky shodné s NAT-PT, drobné změny směřují k odstranění problémů, jež se staly osudnými jeho předchůdci. Staví na překladači, který má alespoň po jednom rozhraní do IPv6 i do IPv4 sítě. Překládá datagramy podle pravidel SIIT a mapuje adresy mezi oběma světy. Překladač je určen k tomu, aby koncové IPv6 síti zprostředkoval přístup k IPv4 službám v Internetu. Na rozdíl od NAT-PT je asymetrický – umožňuje volně zahájit komunikaci jen z obsluhované sítě směrem ven.

Pro reprezentaci IPv4 adres v koncové IPv6 síti používá vyhrazený prefix délky 96 bitů. Jedná se o část zdejšího adresního prostoru, vyčleněnou správcem pro potřeby NAT64. V dokumentaci je označován jako Prefix64::/96. Pokud by snad síť používala několik NAT64 překladačů, bude mít každý z nich svůj samostatný prefix. IPv4 adresa se jednoduše připojí za prefix. Pokud například správce pro NAT64 přidělí prefix 2001:db8:a:b:c:d::/96, bude v lokální síti vnější IPv4 adresa 1.2.3.4 reprezentována jako 2001:db8:a:b:c:d:102:304. Mapování IPv4 adres do IPv6 je sta-

tické a bezstavové. Prefix64 a IPv4 adresa se spojí do jednoznačného a stále stejného výsledku.

Odesílání datagramů do IPv4 světa zajistí běžné směrovací tabulky – Prefix64 se podle nich doručí překladači, který provede konverzi a předá datagram ven. Pokud se jedná o první datagram daného odesilatele[8], provede překladač zároveň jeho mapování na IPv4 adresu. To je dynamické, záznamy jsou vytvářeny podle potřeby a uchovávány, dokud se používají. Překladač mívá k dispozici jen omezené množství IPv4 adres (často jen jednu), musí tedy šetřit.

Jakmile jednou překladač namapoval místní IPv6 adresu a port na kombinaci své IPv4 adresy a portu, může být tento záznam využíván v obou směrech. NAT64 se v tomto směru chová stejně jako běžné současné IPv4 NATy.

Překladač by si měl udržovat dvě tabulky: Jedna obsahuje vzájemně si odpovídající dvojice adres a portů (její oficiální název zní Binding Information Base, BIB) a slouží pro vlastní mapování. Ve druhé jsou uloženy aktivní relace a překladač z ní vyvozuje, které položky v BIB jsou dosud potřebné. Kdykoli projde překladačem paket náležející některé z relací, občerství si její záznam v tabulce relací. Jestliže ale relace není delší dobu využívána, bude odstraněna. Zároveň se zkontroluje, zda pro příslušnou položku mapovací tabulky zbyla alespoň jedna aktivní relace. Pokud ne, bude položka z BIB odstraněna a použitý port může být použit pro jiný účel.

Návrh počítá i s možností vkládat do tabulek trvalé položky. Mají zpřístupnit vybrané stroje či služby z vnitřní sítě a učinit je dosažitelnými z Internetu. Na jiné místní stroje se dostat nedá, dokud samy nezahájí komunikaci. V tomto směru se chování NAT64 podobá současným IPv4 NATům.

Celkově je tabulek dokonce šest, protože NAT64 by měl informace ukládat odděleně pro tři podporované protokoly: TCP, UDP a ICMP. Samozřejmě není povinné implementovat překladač přesně takto. Datové struktury jsou pouze konceptuální a popisují chování, jak ho implementace dosáhne, je její věc.

DNS64

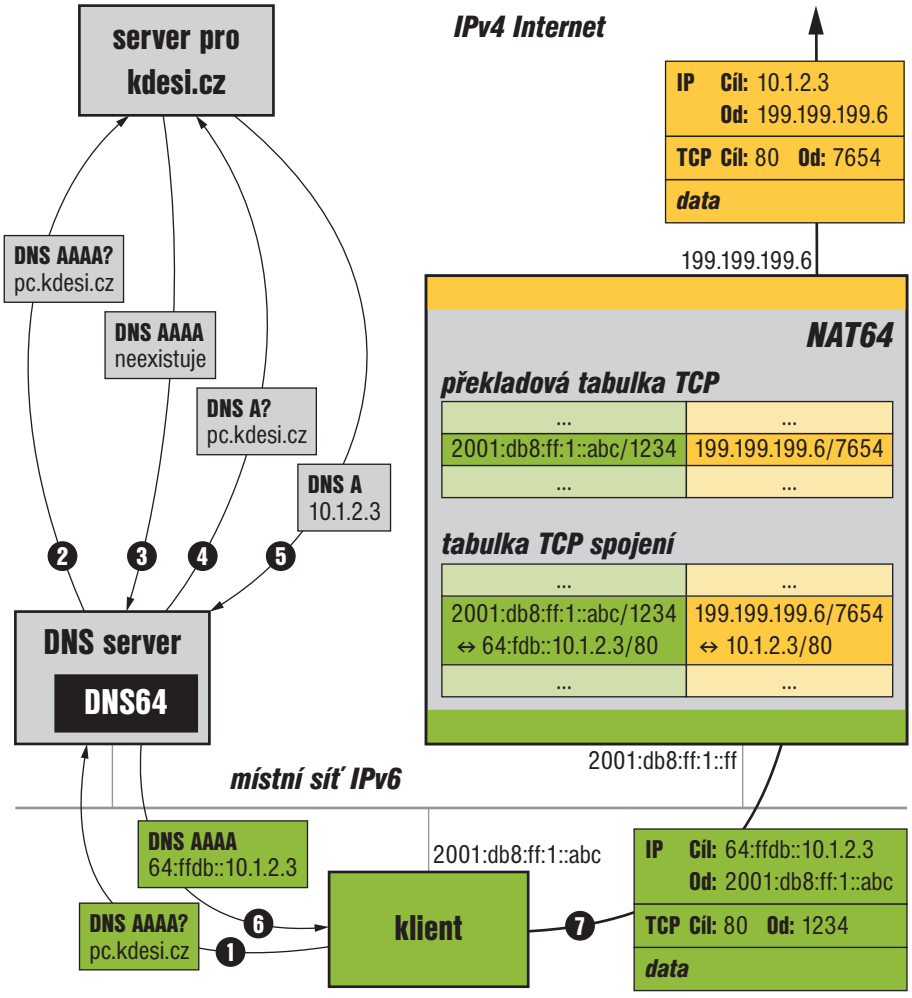
Také NAT64 se neobejde bez machinací s DNS. Proti NAT-PT jsou ovšem výrazně jednodušší – omezují se na dodávání uměle vytvořených záznamů typu AAAA místním strojům, pokud je pro poptávané jméno v DNS jen IPv4 adresa. Algoritmus je označován jako *DNS64* a slouží k tomu, aby IPv6 počítače v místní síti mohly navázat spojení se stroji v IPv4 Internetu.

Vše začíná AAAA dotazem lokálního stroje zaslaným místnímu DNS serveru. Ten nejprve dotaz předá v původní podobě a pokud uspěje, není co řešit, komunikace může proběhnout přímo po IPv6. Jestliže je ale odpověď

[8] Pro danou kombinaci odesilatelovy IPv6 adresy a transportního portu

na AAAA dotaz záporná, zkusí DNS server implementující DNS64 poslat dotaz na záznam typu A pro stejné jméno[9]. Příchozí odpověď s IPv4 adresou pak převede na záznam typu AAAA a adresu upraví výše uvedeným způsobem (připojí za Prefix64::/96).

Klient tak dostane odpověď na svůj DNS dotaz, datagram odeslaný na adresu z ní bude doručen NAT64 překladači, ten provede mapování klientovy adresy na IPv4, datagram přeloží a odešle. Celý proces ilustruje obrázek 12.15.



Obrázek 12.15: Zahájení komunikace s použitím NAT64

[9] V zájmu urychlení může dotazy položit paralelně.

DNS64 zasahuje do obsahu DNS, proto nutně představuje překážku pro DNSSEC. Jestliže si klienti sami údaje neověřují a spoléhají na kontrolu v místním rekurzivním serveru, bude vše fungovat. DNS server ověří odpověď standardním postupem v její původní podobě (záznam typu A) a klientovi pak pošle v odpovědi syntetizovaný záznam typu AAAA, který ovšem vychází z ověřených údajů.

Problém vznikne, pokud si klient chce ověřovat DNSSEC sám, a proto ve svém dotazu nastaví příznak CD. Server s DNS64 nedokáže doprovodit vytvořený AAAA záznam bezpečnostními údaji. V tomto případě předá klientovi původní záznam typu A s doprovodným DNSSEC materiálem. DNS64 pak musí být implementováno na straně klienta.

Proti NAT-PT je valba mezi DNS64 a NAT64 daleko volnější. V DNS se provádí jen bezstavové mapování IPv4 adres do vyhrazené části IPv6 prostoru. To může být prováděno kdekoli (a klidně i na více místech zároveň, jako při ověřování DNSSEC na straně klientů). Jediným požadavkem je, aby se používal stejný Prefix64, který zajistí datagramům odeslaným na mapované adresy doručení do prvku implementujícího NAT64.

Výsledkem je uspořádání, které má sice větší viditelná omezení, než NAT-PT, ovšem výrazně méně vnitřních problémů.

12.12 Transport Relay Translator (TRT)

O podobný úkol jako NAT64, ale v transportní vrstvě, se snaží TRT. Jedná se o směrovač (či jiné zařízení), který hovoří oběma protokoly a předává data mezi oběma komunikujícími partnery. Inspirace TRT pochází ze světa firewallů, kde se využívá podobných myšlenek.

Opět se ocitáme v situaci, kdy počítač v lokální IPv6 síti chce komunikovat s partnerem připojeným pouze po IPv4. Správce lokální sítě vyhradí jeden z jejích prefixů pro reprezentaci IPv4 adres. Požaduje se, aby měl obvyklou délku 64 bitů, a v následujícím textu jej budu opět označovat jako *PrefiX*. IPv4 adresa *a.b.c.d* má ve vnitřní IPv6 síti podobu *PrefiX*::*a.b.c.d*.

TRT zařízení šíří v IPv6 síti směrovací informaci, že jeho prostřednictvím je dostupná síť s *PrefiX*em. Když zdejší počítač *X* pošle paket zahajující TCP spojení s cílem *Q*, bude díky obvyklým směrovacím pravidlům doručen na TRT stroj. Ten odpoví, naváže s *X* TCP spojení a bude mu nadále předstírat, že je *Q*. Toto spojení používá IPv6.

Zároveň však TRT stroj sám naváže TCP spojení s cílem *Q*. Jeho IPv4 adresu si vyzvedne z posledních čtyř bajtů cílové adresy původního IPv6 datagramu a jako zdrojovou adresu použije svou vlastní IPv4 adresu. Pro spojení mezi TRT a *Q* se používá IPv4 a TRT zde pro změnu předstírá, že je odesilatel původní žádosti *X*.

Místo jednoho spojení mezi *X* a *Q* tedy vzniknou spojení dvě: *X*–TRT (IPv6) a TRT–*Q* (IPv4). Následně už TRT pouze vyzvedává data, která mu dorazí jedním z těchto spojení, a promptně je odesílá druhým. Pro UDP se chová analogicky, jen odpadá úvodní navazování spojení.

RFC 3142: *An IPv6-to-IPv4 Transport Relay Translator*, které definuje mechanismy TRT, popisuje jen navázání spojení směrem z IPv6 do IPv4. Stejně by fungovalo i ve směru opačném, tam však vzniká problém s reprezentací IPv6 adres, který TRT samo neřeší. Staví se do pozice, že až někdo přijde s dobrým řešením tohoto problému, TRT je s radostí využije. Pravděpodobně se ale neobejde bez švindlování DNS podobného tomu, za který se sesypala kritika na hlavu NAT-PT.

Zrovna tak se návrh nezabývá otázkou, kde a jak dojde k transformaci IPv4 adres na IPv6 tvar *PrefiX*::*adresa*. Naznačuje tři možné alternativy: statickou tabulku na straně IPv6 uzlu[10], upravený DNS server v lokální síti, který řeší zdejší dotazy, či upravený DNS klient u koncových IPv6 počítačů. Celkově TRT připomíná SIIT – také ponechává některé klíčové problémy stranou.

12.13 Bump-in-the-Host (BIH)

Bump-in-the-Host je soukromý překladač, ukrytý v implementaci síťových protokolů jednoho stroje. Jeho cílem je umožnit aplikacím podporujícím pouze IPv4 komunikovat s IPv6 servery. Předpokládá, že dotyčný stroj je připojen buď oběma protokoly, nebo pouze po IPv6.

Původně existovaly dvě samostatné specifikace: Bump-in-the-Stack (BIS) podle RFC 2767, kde byl překladač součástí síťové vrstvy, a Bump-in-the-API (BIA) podle RFC 3338 s překladačem vloženým do API. BIH nahrazuje obě, protože základní principy jsou podobné. Popisuje tyto dvě alternativy (překlad v síťové vrstvě, nebo v API) jako dvě základní implementační možnosti. V textu autoři spíše doporučují jít cestou API, která je jednodušší a má méně omezení.

princip Výchozím bodem obou variant je zásah do DNS. Jestliže aplikace poptává IPv4 adresy, tedy záznamy typu A, BIH vytvoří paralelní dotaz na stejnojmenné záznamy typu AAAA. Dorazí-li odpověď typu A, BIH se zdrží dalších akcí a komunikace proběhne po IPv4. Jestliže ale získá odpověď jen se záznamy typu AAAA, začne aplikaci předstírat existenci IPv4 adres.

Pro tento účel disponuje konfigurací daným rozsahem privátních adres[11] podle RFC 1918 a mapovací tabulkou obsahující vazby mezi IPv6 a těmito adresami. Protože má BIH k dispozici dostatek IPv4 adres, mapují se jen vlastní adresy a není třeba kouzlit s TCP či UDP porty.

[10] Ta bude použitelná jen ve velmi jednoduchých situacích.

[11] Musí se samozřejmě jednat o adresy, které v dané síti nejsou využívány k jiným účelům. Je-li místní síť adresována z rozsahu 10.0.0.0/8, nabízí se použít pro BIH například 172.16.0.0/12.

Výše uvedený dotaz povede k přidání nových položek do mapovací tabulky pro všechny v ní dosud neobsažené IPv6 adresy z DNS odpovědi. Jim odpovídající privátní IPv4 adresy budou aplikaci vráceny jako výsledek jejího DNS dotazu. Až následně odešle data na některou z těchto IPv4 adres, budou přeložena na IPv6 a odeslána na odpovídající IPv6 adresu. Analogicky pak probíhá i transformace dat přicházejících v protisměru. Aplikace je udržována v iluzi, že komunikuje po IPv4.

BIH funguje i v opačném směru – pokud se někdo pokusí protokolem IPv6 obrátit na zdejší IPv4 aplikaci. V tom případě zůstává DNS stranou, mapování a překlad vyvolá příchozí IPv6 datagram.

součásti

BIH zahrnuje tyto základní součásti:

- *Rozšíření DNS* má na starosti kouzlení s DNS dotazy – vytvoření doplňkového dotazu na AAAA a případné generování umělé odpovědi, pokud se podařilo získat jen IPv6 adresy.
- *Mapovač adres* spravuje mapovací tabulku a zajišťuje vzájemné mapování IPv6 adres a privátních IPv4 adres z rozsahu, který má k dispozici.
- *Překladač protokolů* je součástí varianty BIH implementované v síťové vrstvě. Provádí překlad datagramů mezi oběma protokoly podle principů SIIT (RFC 6145).
- *Mapovač funkcí* je zařazen do varianty BIH umístěné v API. Převádí volané funkce API pro IPv4 na odpovídající univerzální či IPv6 funkce a ze získaných informací sestavuje výsledky.

varianty

BIH ukryté v API se chopí funkcí soketového API pro IPv4, jako například *gethostbyname*, a změní jejich implementaci tak, aby v případě nutnosti docházelo k mapování adres a volání příslušných funkcí API pro IPv6. Vzhledem k tomu, že k překladu zde dochází mezi aplikační a transportní vrstvou, má tato varianta lepší podmínky pro korektní komunikaci. Může například využívat DNSSEC (protože využívá funkce DNS resolveru), IPsec a různá další rozšíření.

BIH implementované v síťové vrstvě má podstatně menší možnosti. Pracuje s již vytvořenými datagramy, které překládá mezi IPv4 a IPv6 podle pravidel SIIT se všemi jejich omezeními. Také DNSSEC na daném stroji nepřipadá v úvahu, protože místní DNS resolver od síťové vrstvy dostává upravené datagramy.

Obě varianty způsobí problém protokolům, které přenášejí IP adresy v aplikační vrstvě. BIH tuto situaci neřeší, pokud mají dotyčné služby pracovat, je třeba jejich aplikační protokol ošetřit jinde.

V době vzniku tohoto textu nebyla standardizace BIH dosud dokončena, text vychází z návrhu *draft-ietf-behave-v4v6-bih*. Podle zkušeností s BIS, je-

hož implementace aby člověk hledal s lucernou v poledne, očekávám, že velkou díru do světa zřejmě neudělá.

12.14 Přechodové nástroje v praxi

V předchozím textu jsem popsal sadu nástrojů, které lze v síti nasadit. Je jich požehnaně (podle bonmotu Randyho Bushe víc než IPv6 datagramů) a člověka může rozbolet hlava, když si má některý vybrat. Podívejme se na jejich nasazení z pohledu zákaznické (domácí či podnikové) sítě.

Obecným požadavkem je, aby stroje z místní sítě měly přístup ke službám poskytovaným oběma protokoly. To se dá zařídit dvěma základními způsoby: buď koncové stroje provozovat v režimu dvojího zásobníku a přivést jim IPv4 i IPv6, nebo na nich provozovat jen jednu verzi IP a komunikaci s druhou zajistit překladačem.

Vhodnější je nepochybně první přístup, který trpí menším množstvím omezení a problémů. Připojit počítač oběma protokoly je asi to nejlepší, co pro něj v současné době můžete udělat. V ideálním případě bude připojení nativní, kdy jak koncová síť, tak poskytovatel Internetu podporují IPv4 i IPv6. Takový stav ale není zrovna obvyklý. Spíše bude nutné sáhnout k tunelování a IPv6 přivést do koncové sítě[12] buď statickým tunelem vedeným k vhodnému poskytovateli, nebo některým z automatických tunelovacích mechanismů. Měl-li bych doporučit konkrétní tunelovací mechanismus, seřadil bych je zhruba následovně:

1. 6rd – pokud jej váš poskytovatel nasadil, jděte do toho. Dostanete IPv4 i IPv6 konektivitu z jednoho zdroje a nebudete závislí na dalších subjektech.
2. Manuální tunel – nabídne konzistentní chování za cenu topologických zacházek. Stabilita a předvídatelnost mi připadají přednější, nicméně budete závislí na provozovateli tunelového serveru.
3. 6to4 – je flexibilní, v podstatě nezávislé[13], ovšem nespolehlivé a poněkud nevypočitatelné.
4. Teredo – zpřístupní IPv6 prakticky kdekoli a z dostupných mechanismů nejlépe prochází NATy a firewally. Efektivita však bývá tragická a spolehlivost je na tom také bídně.

O překladu má smysl uvažovat jen v čisté IPv6 síti, jejíž účastníci potřebují přístup ke službám poskytovaným po IPv4. Takové uspořádání lze považovat za výrazně menšinové, nicméně s ubývajícími IPv4 adresami představi-

[12] Případně i přímo konkrétnímu počítači, ale takové uspořádání představuje spíše experimentální mezistupeň než koncový stav.

[13] Potřebujete sice zprostředkovatele pro přestup do nativní IPv6 sítě, ale o ně není nouze a mají různé provozovatele. I když někteří skončí, stále bude z čeho vybírat.

telné. V běžných případech ale bude i pak pravděpodobně výhodnější poskytnout koncovým stanicím IPv4 s neveřejnými adresami překládanými IPv4-IPv4 NATem (např. prostřednictvím Dual-Stack Lite), než překládat mezi IPv4 a IPv6.

Pravděpodobné scénáře nasazení přechodových nástrojů rozebírá podrobněji RFC 6180: *Guidelines for Using IPv6 Transition Mechanisms during IPv6 Deployment*.

II
IPv6 v praxi

13 IPv6 na vlastní kůži

Snad ve vás předchozí část vzbudila alespoň částečný zájem IPv6 vyzkoušet. Pokud ne, možná jste jen nečetli dostatečně pozorně. Zopakuji proto to nejpodstatnější: Adresy stávajícího IPv4 docházejí. Centrální zásoba IANA byla vyčerpána v únoru 2011, takže regionální registry po rozdělení svých zásob už žádné další adresy nedostanou. Existuje řada prognóz dalšího vývoje Internetu, většinou dost depresivních – obchodování s adresami, masivní používání neveřejných adres s NATem na každém rohu a podobně. IPv6 představuje jediné v současnosti rozvíjené koncepční řešení. Proto se určitě vyplatí věnovat mu nějaký čas a seznámit se intimně.

Nic zásadního tomu nebrání. Protokol je dnes implementován ve valné většině používaných systémů či hardwarových zařízení a také možností pro (alespoň experimentální) připojení k IPv6 Internetu je všude dost. Pokud máte přístup k Internetu a váš systém není příliš exotický, můžete si IPv6 vyzkoušet.

13.1 Lehké oťukávání

Dá se předpokládat, že pro úvodní vyzkoušení nejspíš začnete zprovozněním IPv6 na svém počítači. V zásadě je třeba vyřešit dva problémy:

1. Zda váš operační systém podporuje IPv6, případně jak mu k tomu pomoci. Řešení najdete v následujících kapitolách, kde popisuji stav implementace na různých platformách a související konfigurační příkazy. Stručně řečeno: je pravděpodobné, že váš operační systém IPv6 podporuje, případně jej k tomu lze snadno postrčit.
2. Jak se připojit k IPv6 Internetu. Zde se budu věnovat především této otázce, která je pochopitelně pro všechny systémy stejná.

Pro začátek bude nejjednodušší sáhnout po některé z metod, které kromě připojení k běžnému IPv4 Internetu nevyžadují skoro nic. Nabízejí se dvě: Teredo nebo otevřené tunelovací servery. Oba způsoby mohou být velmi jednoduché z hlediska použití a dobře se vyrovnávají s překážkami současného Internetu v podobě NATů, firewallů a podobných skopičin.

Jestliže používáte MS Windows 7 nebo Vista, máte v podstatě hotovo. Systém ve výchozím nastavení obsahuje klienta pro Teredo, jenž se spojí s Teredo serverem firmy Microsoft a vytvoří vašemu systému IPv6 připojení, aniž byste se museli o něco starat. Stačí si prohlédnout rozhraní svého počítače – vše je popsáno v kapitole 16.1 na straně 331.

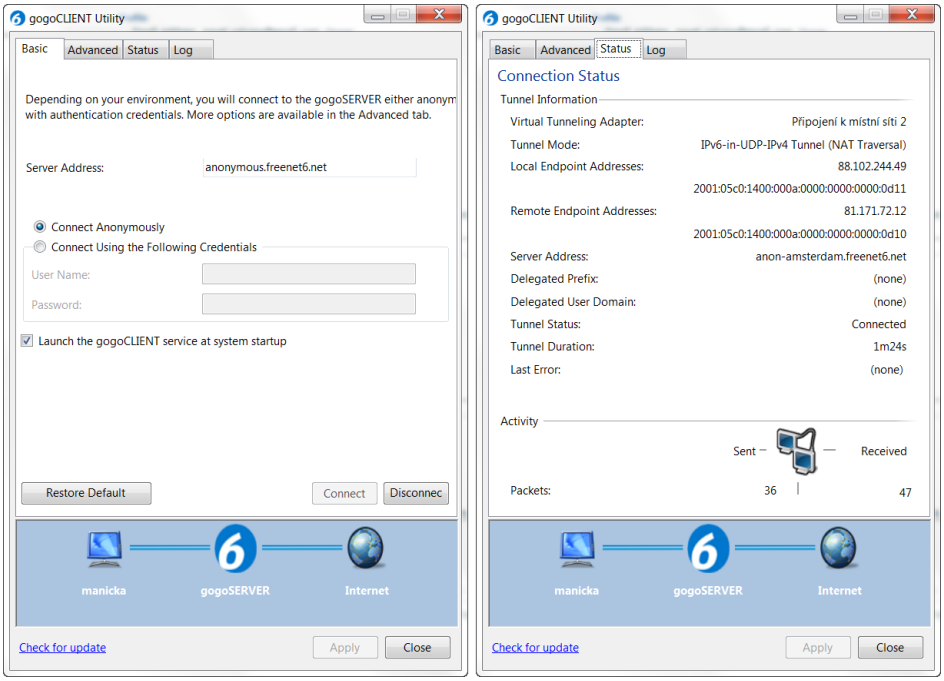
Teredo je k dispozici i pro ostatní systémy, například v podobě programu *Miredo*. Podrobnější informace najdete v příslušných kapitolách.

Jestliže Teredo z nějakého důvodu nechcete nebo nemůžete používat[1], sáhněte po některém z volných tunelovacích serverů. Asi nejjednodušší je použití služby *Freenet6*, kterou najdete na adrese

WWW▶ *http://www.freenet6.net/*

Její vztah k uživatelům je poněkud ambivalentní. Oficiálně je poskytována volně, bez nutnosti jakékoli registrace a přihlašování, ovšem klientský program *gogoCLIENT* ke svému stažení vyžaduje registraci v sociální síti gogoNET. Pokud se vám podaří najít někde klienta, nic vám nebrání v připojení. Nejjednodušší cesta k jeho aktuální verzi samozřejmě vede přes vytvoření účtu v gogoNET.

gogoCLIENT je k dispozici pro řadu platforem – jako instalátor pro MS Windows, nebo jako zdrojový kód přeložitelný v prostředí Linuxu, BSD či MacOS. Způsob použití se lehce liší v závislosti na operačním systému. U graficky orientovaných nejprve spustíte klienta, lehce zkontrolujete nastavení, jež je ve výchozí verzi přepnuto na anonymní připojení a stisknete tlačítko *Connect*.



Obrázek 13.1: *gogoCLIENT* pro MS Windows

[1] Například se mi nepodařilo přinutit k činnosti Teredo klienta ve Windows XP.

Na obrázku 13.1 vlevo vidíte konfigurační panel klienta po úspěšném připojení. Všimněte si, že na něm lze zapnout automatické spuštění klienta při startu systému. V panelu *Status* (zobrazen napravo) se pak dozvíte podrobnosti o svém připojení.

V případě systémů unixového typu je klient textový a jeho instalace dá trochu více práce. Musíte překládat ze zdrojových kódů, navíc bez automatické konfigurace příkazem

```
make target=systém all
```

a poté instalovat opět nestandardně

```
make target=systém installdir=adresář install
```

to vše provedeno v adresáři *gogoc-tsp*. Klienta pak najdete v podadresáři *bin* instalačního adresáře. Tamtéž hledejte jeho konfigurační soubor *gogoc.conf*, němž musíte stanovit server

```
server=anonymous.freenet6.net
```

Klienta pak stačí spustit a tunel je navázán.

Obě popsané varianty – Teredo i anonymní tunel *Freenet6* – mají podobné základní vlastnosti. Mají minimální požadavky, jsou uživatelsky velmi jednoduché a docela dobře fungují i v současném Internetu s hojnými NATy. Ovšem pakety tunelují cestami hodně vzdálenými od optimálních a efektivita je jim vzdálená zhruba stejně jako voda na Marsu. Navíc je spojení tak trochu jednosměrné. Ne snad že by data nemohla protékat oběma směry. Ale tiše se předpokládá, že komunikaci bude navazovat váš stroj. V protisměru to není snadné, protože nejste nikde v DNS, vaše adresy se mohou měnit a představujete proto těžko oslovitelného partnera.

Vytvoření anonymního tunelu dá trochu více práce, ale na druhé straně od něj lze očekávat vyšší spolehlivost a konzistentnější chování.

Teredo či anonymní tunely jsou ideální pro počáteční experimenty. Jakmile zatoužíte po připojení s lepšími vlastnostmi, musíte se poohlédnout po jiných nástrojích.

13.2 Trvalé připojení

IPv6 poskytovatel

Hodláte-li se v IPv6 Internetu pohybovat soustavněji, mělo by vaše připojení k němu mít trvalejší charakter. I zde se nabízí několik variant. V ojedinělých případech váš poskytovatel Internetu podporuje IPv6. Taková situace je sice stále vzácná, nicméně první vlaštovky se už objevily i u nás. Dobrým východiskem pro tyto úvahy je tabulka IPv6 peeringu v centru NIX.CZ na adrese

www▸ *http://www.nix.cz/cz/peering_ivp6*

Pokud v ní najdete svého poskytovatele, zjevně už v oblasti IPv6 není panic a má smysl se u něj informovat, zda a za jakých podmínek nabízí IPv6 připojení svým zákazníkům. Šance na úspěch je v současnosti již nemalá – v polovině roku 2011 měla většina subjektů připojených do NIX.CZ (konkrétně kolem dvou třetin) alespoň jedno IPv6 propojení. U poskytovatelů, kteří nemají IPv6 peering, je pravděpodobnost poskytování služeb novým protokolem mizivá.

Bohužel ani podpora na straně poskytovatele neznamená, že máte vyhráno. Dnes obvykle nebývá připojen přímo koncový počítač, daleko častěji najdeme na konci lokální síť. Typickým příkladem je nejčastější současný způsob připojení, ADSL. Linka od poskytovatele vede do zákazníkova ADSL modemu, jenž funguje zároveň jako směrovač, NAT, DHCP server, firewall a automat na zmrzlinu a zprostředkovává místním počítačům spojení prostřednictvím Ethernetu a/nebo Wi-Fi. A právě běžně používané levné ADSL modemy až příliš často představují výjimku z mého tvrzení, že IPv6 dnes zná valná většina obvyklých systémů. Dají se samozřejmě pořídit exempláře podporující IPv6, ale rozhodně jich není většina.

Když se u vás sejde ta šťastná konstelace, že IPv6 podporuje jak váš poskytovatel, tak zařízení k němu připojené, můžete se připojit nativně. Tedy přenášet přímo IPv6 datagramy a používat individuální globální adresy z rozsahu, jímž disponuje váš poskytovatel.

Obvykle však idyla nenastane a budete i při snaze o serióznější a trvalé IPv6 připojení nuceni svá IPv6 data tunelovat. V Internetu jsou k mání tři velcí poskytovatelé tunelových serverů: *Freenet6*, *SixXS* a *Hurricane Electric*. Jejich služby jsou podobné, pro pevné tunely všichni vyžadují bezplatnou registraci. Podívejme se na ně podrobněji, ale rovnou předešlu, že bez velkého váhání doporučuji první z nich.

autentizovaný Freenet6

Služba *Freenet6*, která kromě výše zmiňovaného anonymního připojení nabízí i autentizovanou variantu, je totiž nejsnadněji použitelná. Proti anonymnímu má její autentizované připojení tři přednosti, jež oceníte především při soustavnějším využívání IPv6:

- Prefix délky 56 bitů. Můžete tedy IPv6 rozvést ve své lokální síti a využívat v ní plnohodnotné globální individuální adresy.
- Pevné adresy, které se nemění v čase ani v závislosti na IPv4 adrese, z níž se připojujete.
- Váš tunel se stává stabilním, proto je zaregistrován do DNS pod jménem *uživatel.broker.freenet6.net*.

IPv6 prostředí pro váš počítač a koncovou síť je stálé a můžete například uvažovat o nabízení služeb protokolem IPv6.

Chcete-li využívat autentizovanou variantu, musíte si vytvořit uživatelský účet. Je nezávislý na účtu pro síť *gogoNET*, jehož pomocí jste získali klienta, musíte tedy mít účty dva.

K připojení poslouží stejný program, *gogoCLIENT*. Pouze na úvodní straně přepněte autentizaci z anonymní na autentizovanou, vyplňte své uživatelské jméno a heslo. Nezapomeňte také změnit jméno serveru z *anonymous.freenet6.net* na *authenticated.freenet6.net*. Používáte-li textového klienta, musí se v konfiguračním souboru *gogoc.conf* objevit

```
server=authenticated.freenet6.net
userid=uživatelské_jméno
passwd=heslo
auth_method=any
```

Jako zajímavost určitě stojí za zmínku, že *Freenet6* nabízí i jiné tunelovací režimy, jež umožňují ryzím IPv6 strojům přístup do IPv4 světa. Patří mezi ně DS-Lite a DSTM, oba postavené na tunelování IPv4 v IPv6.

SixXS Velice podobné služby nabízí i síť *SixXS*. Opět se jedná o systém otevřených tunelových serverů, k nimž se může připojit kdokoli. Hlavní devizou je podstatně rozsáhlejší síť serverů, díky níž lze jsou odchylky od optimálních tras menší, než v případě konkurenčních služeb. *SixXS* nemá anonymní režim, před využíváním služeb se musíte zaregistrovat na adrese

www▸ *http://www.sixxs.net/*

Celá služba je postavena tak, aby oddělila vážné zájemce o připojení od kolemjdoucích zvědavců. První varování se dostaví už při vyplňování žádosti o registraci, kde vás čeká mimo jiné povinné slohové cvičení na téma „proč já, nebohý zemský červ, požaduji na SixXS připojení“. Ne aby vás napadlo je odbýt, protože se vzápětí dozvíte, že vaše žádost bude schvalována ručně. Už si ani nevzpomínám, kdy naposledy jsem se s něčím takovým setkal. Naštěstí vše běželo poměrně svižně a vyrozumění o schválení žádosti dorazilo asi po třech hodinách, byť byla sobota odpoledne.

Dostanete strašlivé uživatelské jméno a ještě strašlivější heslo a po přihlášení jste konfrontováni s místním kreditovým systémem. Komunikace se serverem je totiž podložena virtuální měnou zvanou ISK. Za své požadavky budete muset platit a naopak díky svým aktivitám vydělláváte (např. za registraci dostanete do začátků 25 ISK). Začít musíte žádostí o tunel (stojí 10 ISK), která je opět posuzována manuálně (5 ISK). Kombinace čekání na schvalování žádostí a nutnost hlídat kredity od *SixXS* spolehlivě odradí všechny kromě vážných zájemců.

Ti, kteří vytrvají, si ze serveru stáhnou klientský program nazvaný *AICCU (Automatic IPv6 Connectivity Client Utility)*. Je k dispozici pro řadu operačních systémů, pro některé v grafické verzi, jinde jej musíte ovládat úpra-

vami konfiguračního souboru. Existuje také v podobě distribučních balíčků, takže doporučuji se nejprve poohlédnout, zda jej nemůžete instalovat standardním způsobem. V případě krajního zoufalství je vždy k dispozici překlad ze zdrojových textů, jež jsou také k mání.

Pro Windows je k dispozici grafický klient, v němž vyplníte uživatelské jméno a heslo, za což budete odměněni nabídkou svých tunelů. Vyberete jediný existující a připojíte se.

Pro textovou verzi klienta bude třeba upravit konfigurační soubor */etc/aiccu.conf*. Většině parametrů lze ponechat výchozí hodnoty, jen na začátek vložte své uživatelské jméno a heslo:

```
username franta
password prisnetajne007
```

Vzorový konfigurační soubor je opatřen komentáři, takže jste informováni, jaké dopady příslušný parametr má. Pak už zbývá jen spustit jako superuživatel příkaz `aiccu start`. Tunelu nějakou dobu trvalo, než se rozeběhl. Pokud se krátce po startu ocitnete ve stavu, kdy je IPv6 připojení sice nastaveno, ale reálně nefunguje, nepropadejte panice a zkuste chvíli vydržet.

Hurricane Electric

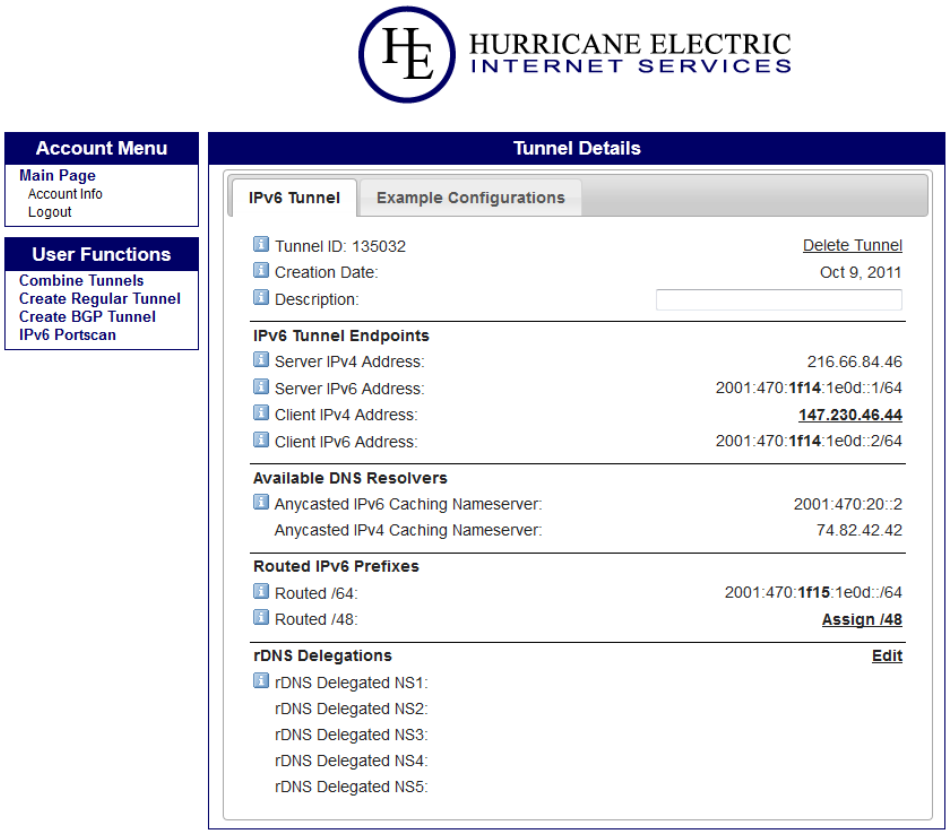
Třetí alternativu představují tunel servery *Hurricane Electric*. Na rozdíl od předchozích dvou se nejedná o virtuálního poskytovatele – firma provozuje svou vlastní páteřní infrastrukturu globálního dosahu a funguje jako poskytovatel konektivity a housingových služeb. Jako výhody svých tunel serverů uvádí profesionální péči a také prefix standardní délky 48 bitů, který od nich dostanete. Služba je dostupná na adrese

www▸ *http://www.tunnelbroker.net/*

Opět není k mání anonymně, musíte si vytvořit účet a následně můžete požádat o založení tunelu (*Create Regular Tunnel*). Na rozdíl od SixXS žádost nepodléhá schvalování, ani se za ni virtuálně neplatí, musíte ale splnit požadavky technické. Přesněji řečeno jeden – dotyčná adresa musí odpovídat na *ping*. Bez toho nelze tunel vytvořit.

Po úspěšném vytvoření tunelu obdržíte základní informace o něm (viz obrázek 13.2). Tentokrát nedostanete žádný specializovaný program, svůj konec tunelu si musíte nakonfigurovat standardními prostředky svého operačního systému. Poznáte je v následujících kapitolách, nicméně HE vám podá pomocnou ruku. Na kartě *Example Configurations* si vyberete svůj operační systém či platformu a obdržíte konfigurační příkazy ušité přímo na míru vašemu konkrétnímu tunelu. Stačí je zadat a permanentně uložit, aby se prováděly při každém startu.

Ve výchozím stavu je do tunelu směrována síť s prefixem /64. Můžete jej povýšit na /48, ovšem je třeba o to požádat – slouží k tomu odkaz *Assign /48* na obrázku 13.2.



Obrázek 13.2: Tunel založený u Hurricane Electric

srovnání Nabízí se pochopitelně srovnání dostupných služeb. Jednoznačnou výhodou *Freenet6* je snadnost použití. Po uživatelské registraci je vám služba okamžitě k dispozici (a v anonymním režimu ještě před ní), při používání jsem nenarazil na žádný zádrhel.

U *SixXS* je rozjezd znatelně pomalejší, odměnou vám ale bude připojení přes server umístěný v České republice[2], což je na výkonu znát. Při přístupu k zahraničním zdrojům nebude rozdíl výrazný, testy proti *ipv6-test.com* nevykazovaly dramatické odlišnosti. Ovšem jakmile zamíříte domů, *SixXS* se vyplatí. Při testech proti serverům CESNETu (*speedtest.cesnet.cz* a *speedtest6.cesnet.cz*) připojení přes *SixXS* vykazuje pro IPv6 přibližně o 10 % nižší přenosovou rychlost, zatímco u *Freenet6* se propadla na méně než polovinu rychlosti IPv4. Je zkrátka znát, jestli pakety běhají oklikou přes Amsterdam, nebo jen procházejí NIX.CZ.

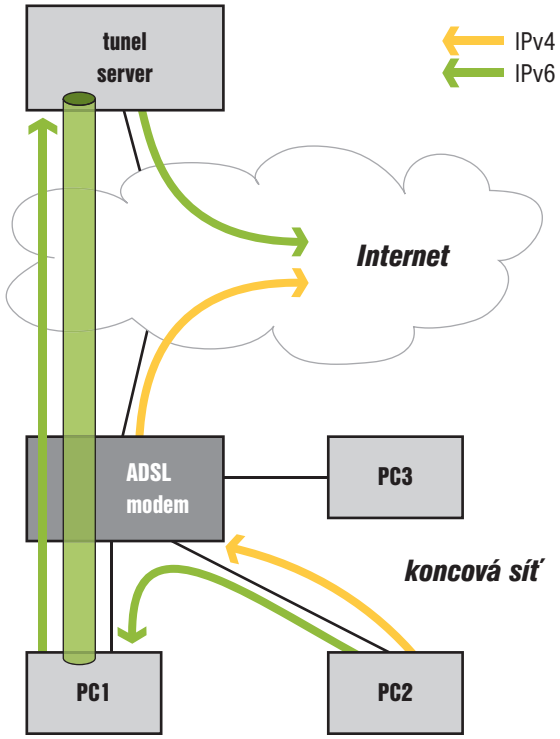
Hustota sítě serverů *Hurricane Electric* převyšuje *Freenet6*, ale na *SixXS* nestačí. V našem prostředí máte na výběr mezi Německem a Nizozemskem, do České republiky nezasahuje. Při přístupu k domácím zdrojům proto nutně očekávejte větší zpoždění. Hlavní devizou této alternativy je standardní přístup k tunelům. Nepotřebujete žádný specializovaný program, používají se běžné konfigurační příkazy. Chcete-li vést tunel nikoli z počítače, ale z hardwarového směrovače, bude HE pravděpodobně jedinou možností.

Co mají všechny varianty společné, je potenciální nekonzistence mezi směrováním IPv4 a IPv6 v koncové síti. Jestliže tunelem obcházíte omezení svého přístupového prvku (řekněme ADSL modemu), budou datagramy obou protokolů směrovány odlišně. Zatímco pro IPv4 zůstane implicitním směrovačem přístupový prvek, pro IPv6 bude tuto roli hrát stroj, ze kterého vychází tunel. Situaci znázorňuje obrázek 13.3. Samozřejmě na ní není nic zásadně špatného, jedná se o různé protokoly a jejich směrování se proto může lišit. Jen to může být poněkud matoucí a svádět k chybným závěrům při řešení problémů s připojením.

6to4 Pokud disponujete alespoň jednou veřejnou IPv4 adresou, můžete uvažovat o automatickém tunelování pomocí 6to4. Díky němu lze získat individuální globální IPv6 prefix, přestože vás obklopuje IPv4, a dokonce vaše data budou přepravována po cestách optimálních či alespoň optimálním blízkých. Popis konfigurace najdete v následujících kapitolách. Není nijak složitý: zapnete 6to4, nastavíte implicitní cestu pro IPv6 na výběrovou adresu 6to4 zprostředkovatelů a do vnitřní sítě ohlašujete 6to4 směrovač jako implicitní cestu do IPv6 Internetu.

Problém je v oné veřejné IPv4 adrese, kterou potřebuje váš 6to4 směrovač a která se může stát příčinou, proč tato cesta nebude schůdná. Někteří zákazníci takovou adresu vůbec nemají a jejich datagramy procházejí něko-

[2] V době vzniku tohoto textu existoval jediný domácí server, který provozovala hostingová společnost IGNUM.



Obrázek 13.3: Nekonzistence směrování IPv4 a IPv6

lika NATy, u jiných je jediná veřejná adresa přidělena přístupovému prvku nepodporujícímu IPv6, tudíž neschopnému pracovat jako 6to4 směrovač. Může se také měnit při každém připojení. V těchto případech zřejmě nezbyde, než tunelovat staticky.

DNS Jakmile bude mít vaše IPv6 připojení trvalejší charakter, pravděpodobně začnete uvažovat o poskytování služeb tímto protokolem. Ruku v ruce s ním kráčí otázka ukládání příslušných informací do DNS a jejich dostupnost. Jestliže si alespoň některé autoritativní DNS servery spravujete sami, pravděpodobně nebude problém naučit je IPv6 a uložit na ně odpovídající AAAA záznamy. Ani v opačném případě ale nemusíte zoufat. Nabídka hostování autoritativních DNS serverů s podporou IPv6 je až překvapivě široká, namátkou:

www▸ *http://www.xname.org/*

www▸ *http://freedns.afraid.org/*

www▸ *http://dns.he.net/*

Pokud byste chtěli zůstat v domácích luzích a hájích, najdete podporu IPv6 například u

www▸ *http://www.hosting90.cz/*

www▸ *http://www.tele3.cz/*

Pokud budete používat 6to4, reverzní doménu pro ně si můžete spravovat na adrese

www▸ *https://6to4.nro.net/*

Po správě reverzních domén sítí připojených statickými tunely je třeba se poohlédnout u poskytovatele tunelů, protože on spravuje reverzní doménu prefixu, z nějž jsou odvozeny jejich adresy. Máte-li regulérní globální prefix, znamená to, že váš poskytovatel zřejmě podporuje IPv6 a s reverzní doménou by proto neměl být problém, deleguje (či spravuje) vám ji stejně jako v případě IPv4.

13.3 Testování a měření

Když zprovozníte IPv6 připojení svého stroje či celé sítě, pravděpodobně vás bude zajímat, zda a jak dobře funguje. Pochopitelně jsou k dispozici základní diagnostické nástroje *ping* (*ping6*) a *traceroute* (*tracert*, *traceroute6*), jimiž lze cíleně ověřovat dostupnost určité konkrétní adresy.

funguje to? Kromě nich ovšem stojí za doporučení některé weby, jež umožňují ověřit si, jak se vaše připojení chová v reálném provozu. Pro úvodní jednoduché ověření, zda u vás IPv6 vůbec stroji rozumně běží, lze doporučit například

www▸ *http://ipv6.google.com/*

Který ověří, zda váš stroj dokáže najít záznam typu AAAA pro zadané jméno a následně se spojit protokolem IPv6. Dotyčný web totiž jinak než po IPv6 není dosažitelný. Druhým elementárním testem může být

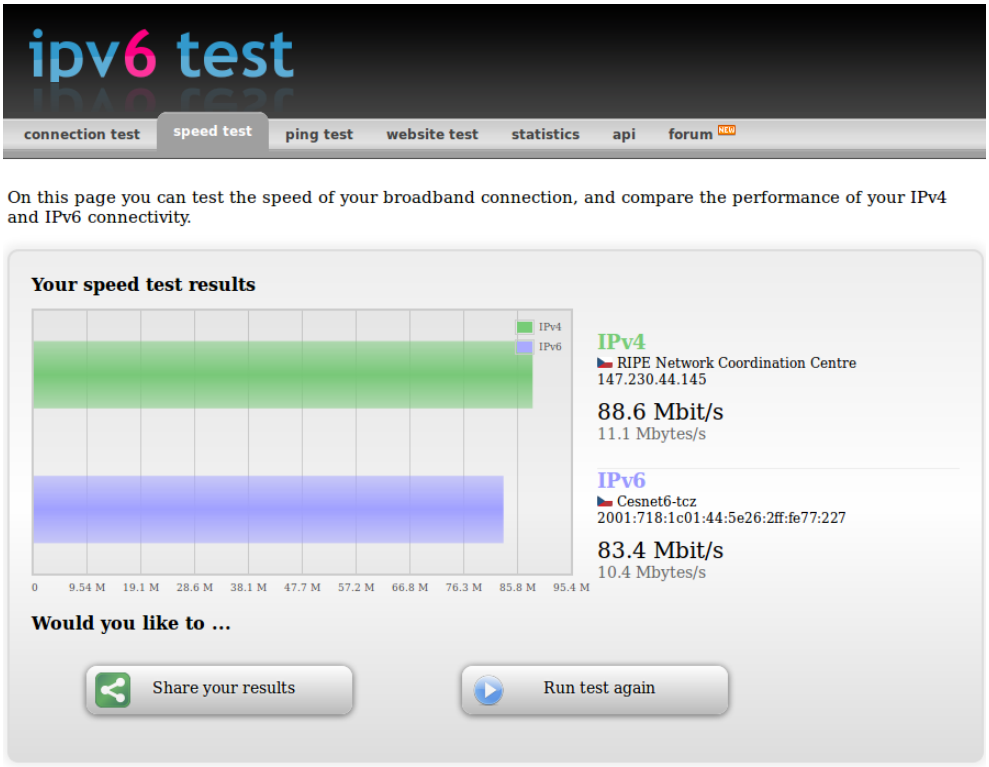
www▸ *http://www.ripe.net/*

V pravém horním rohu zobrazí adresu, z níž jste na web dorazili. Vzhledem k tomu, že *www.ripe.net* má v DNS záznamy typu A i AAAA, dozvíte se, kterému protokolu váš počítač dává přednost, pokud má na výběr oba.

jak rychle? Po úspěšném ověření, že IPv6 skutečně běží, vám pravděpodobně začne vrtat hlavou, jestli je dostatečně rychlé a jak si stojí proti IPv4. Pro hledání odpovědí lze doporučit server

www▸ *http://ipv6-test.com/*

a to především jeho sekci „speed test“. Změří přenosovou rychlost mezi sebou a vaším strojem nejprve protokolem IPv4 a následně IPv6. Výsledek zobrazí, jak číselně, tak v podobě velmi názorného grafu, ze kterého si snadno uděláte představu, jak si oba protokoly stojí.



Obrázek 13.4: Výsledky měření na *ipv6-test.com*

Používáte-li tunelované připojení, bude samozřejmě velmi záležet na poloze druhého konce tunelu. Aneb jak velkou oklikou proti ideální trase musí datagramy putovat. Vzhledem k tomu, že v našich podmínkách lze očekávat především komunikaci s českými zdroji, může použití zahraničního *ipv6-test.com* nahlodat relevanci výsledků.

Proto bych doporučil sáhnout také po testovacích serverech CESNETu. Tentokrát měření není integrováno, musíte testovat IPv4 odděleně od IPv6 a výsledky si dát do souvislosti sami. Dozvíte se ale kromě přenosové rychlosti navíc i zpoždění paketů. K provedení testů se obraťte na adresy

www▸ *http://speedtest.cesnet.cz/*

www▸ *http://speedtest6.cesnet.cz/*

monitoring Pokud jste si pod pojmem „testování“ představili spíše pravidelné ověřování funkčnosti či výkonu síťových prvků a služeb, tedy monitoring, mám pro vás samé dobré zprávy. Nejběžněji používané monitorovací systémy – *Nagios*, *Icinga*, *Zabbix* i *Cacti* – IPv6 podporují. Stačí si jen vybrat, případně upravit konfiguraci svého oblíbence. Přehled existujících programů pro sledování sítě včetně informace o podpoře IPv6 najdete na Wikipedii:

www▸ *http://en.wikipedia.org/wiki/Comparison_of_network_monitoring_systems*

13.4 IPv6 v lokální síti

V předchozí části jsem představil několik cest, jimiž můžete pro koncovou síť získat prefix standardní délky 48 bitů. Otázka je, jak jej využít. Zabývá se jí například RFC 4057: *IPv6 Enterprise Network Scenarios*. Definuje tři základní scénáře pro uspořádání koncové sítě:

- plošná podpora obou protokolů v celé síti
- IPv4 síť s ostrůvky IPv6
- IPv6 síť s ojedinělými uzly podporujícími IPv4

Ovšem neočekávejte od něj návrh řešení. Poslouží spíše jako kontrolní seznam s výčtem možných problémů a otázek, na něž je záhodno znát odpovědi. Ty už ovšem musíte najít sami. Za nejčastější a nejméně problémové lze považovat první řešení, tedy plošnou podporu obou protokolů.

nativně Ideální variantou, jak rozvést IPv6 v koncové síti zahrnující několik podsítí, samozřejmě je nasadit aktivní prvky podporující nový protokol. Můžete pak celou síť či její vybrané části podle potřeby konfigurovat jako IPv4, IPv6 nebo smíšené. Nejobvyklejší bude nepochybně poslední případ, poskytující koncovým zařízením oba protokoly. Konfigurace bývá snadná – jednoduše přidělíte příslušnému rozhraní adresy pro IPv4 i IPv6.

VLAN Jestliže se ale vaše stávající prvky k IPv6 nehlásí a nákup nových je v nedohlednu, možná vám pomohou virtuální lokální sítě, VLAN. Jejich prostřednictvím lze jednotlivé IPv6 podsítě protáhnout až ke směrovačům, které jim rozumějí a dovedou jejich datagramy správně zpracovat. S využitím značkování podle IEEE 802.1Q můžete přenášet spoustu VLAN jedním rozhraním a vytvořit si takové uspořádání, jaké budete potřebovat.

Topologie obou protokolů se může významně lišit. Máte-li L2/L3 přepínač nepodporující IPv6, může se vůči IPv4 chovat jako směrovač a datagramy přicházející z určité VLAN směrovat mimo její hranice. Kromě toho ovšem příslušnou VLAN zavedete pomocí IEEE 802.1Q infrastrukturou dál k (třeba jedinému) IPv6 směrovači, jenž se postará o dopravu IPv6 datagramů. Dokud bude objem IPv6 provozu malý, můžete pro celou síť vystačit s jediným softwarovým směrovačem na Linuxu či některé variantě BSD, který

se postará o směrování IPv6 datagramů z celé sítě a třeba i o její tunelové připojení k vnějšímu Internetu.

Využití VLAN pro přepravu IPv6 datagramů v místní síti rozebírá RFC 4554: *Use of VLANs for IPv4-IPv6 Coexistence in Enterprise Networks*.

tunelování

Když ani virtuální sítě neposkytují pro vaši konkrétní síť řešení, vždy lze sáhnout po starém špatném tunelování a přivést IPv6 ke koncovým strojům pomocí tunelů. Buď staticky (což ale z hlediska správy představuje noční můru), nebo nějakou automatickou formou jako je ISATAP či 6over4. Obecně je ale rozumné a zpravidla i možné se tunelování v místní síti raději vyhnout.

pořízení techniky

Za ideál považuji první variantu, která ovšem vyžaduje podporu IPv6 v aktivních prvcích. To je obecně čím dál tím menší problém, protože výrobci již vzali IPv6 na vědomí a postupně je implementují. S trochou nadsázky se dá prohlásit, že problém chybějící podpory IPv6 v aktivních prvcích se postupně vyřeší samospádem v rámci standardního cyklu inovace síťového vybavení.

V konkrétních případech je ale na místě špetka ostražitosti, protože prospekt snese všechno a pod obecnou deklarací „podpora IPv6“ se může skrývat ledacos. Případný certifikát *IPv6 Ready* samozřejmě hodně napoví, nicméně vždy je záhodno si ještě před koupí vyžádat podrobnější dokumentaci (nejlépe konfigurační manuál) a prolustrovat, zda vyhlédnutý prvek umí vše, co potřebujete.

Jestliže techniku pořizujete na základě formálního výběrového řízení, doporučuji místo obecného požadavku na podporu IPv6 požadovat zcela konkrétní schopnosti. Pomůže vám dokument ripe-501: *Requirements For IPv6 in ICT Equipment*[3], který byl právě pro tyto účely vytvořen. Obsahuje velmi podrobný výčet toho, co byste měli/mohli chtít, včetně odkazů na příslušné specifikace.

13.5 Adresování místní sítě

Pokud jste od poskytovatele dostali prefix délky 64 bitů, tedy jedinou podsíť, máte adresování hotové. V opačném případě se musíte zamyslet, jak navrhnout topologii jednotlivých podsítí a jaké adresy jim přidělit. Dá se to v principu udělat řadou různých způsobů a nijak se nevázat na IPv4. Existuje dokonce celý dokument věnovaný této problematice [15], který propaguje poměrně nezvyklý přístup vytváření podsítí podle charakteru připojených zařízení (ale rozebírá i jiné). Za jeho hlavní výhodu autoři považují snadnější vytváření síťových politik pomocí pravidel ve firewallech a podobných zařízeních.

[3] http://www.ripe.net/ripe/docs/ripe-501

konzistentní IPv4 a IPv6

Valná většina správců ovšem dává přednost topologii totožné s IPv4. Podsítě obou protokolů si odpovídají 1:1 a směrují je stejné aktivní prvky, jejichž příslušná rozhraní mají vždy přiděleny odpovídající IPv4 i IPv6 adresy. Troufám si prohlásit, že z hlediska správy sítě je takové uspořádání nejjednodušší a pokud se máte jen trochu rádi, snažte se k němu dopracovat. Pokud narazíte na technická omezení, bude třeba improvizovat, nicméně doporučuji směřovat ke konzistentní topologii IPv4 a IPv6 jako k cílovému, slovníkem Horsta Fuchse ultimátnímu řešení.

dvoubodové podsítě

Specifickou oblast z hlediska adresování představují podsítě zahrnující jen dvojici vzájemně propojených směrovačů. Typicky se vyskytují v páteřích sítí, mezi budovami, areály, či jinými částmi sítě. Ve světě IPv4 bývá zvykem používat pro ně prefix délky /30 či dokonce /31, který poskytuje jen nezbytné minimum adres a zbytečně neplýtvá. V IPv6 máte na výběr hned čtyři alternativy, jak adresovat dvoubodové podsítě:

- *Použít jen lokální linkové adresy* a globální vůbec nepřidělovat. Pro samotné směrování nejsou globální adresy potřeba – každé rozhraní má svou lokální linkovou adresu a tyto adresy vystupují ve směrovacích tabulkách. Jádro fungování sítě si s nimi vystačí. Drhnout mohou „nadstavbové“ funkce, protože lokální linková adresa je smysluplná a dosažitelná jen v rámci své linky. Může vám vadit, že *traceroute* vypíše nepoužitelnou adresu, že je rozhraní nedosažitelné pro vzdálenou správu (*ping*, dohledový software a podobně), že adresy nemáte pod kontrolou a změní se při změně hardware. V důsledku toho všeho jsou takto adresované páteřní spoje obtížněji spravovatelné a osobně bych touto cestou nešel.
- *Použít místní lokální adresy (ULA)* odpovídá použití privátních adres podle RFC 1918 v IPv4. Adresy jsou směrovány v místní síti, ale nejsou dosažitelné z Internetu. To lze vnímat jako výhodu (obtížněji se na ně zvenčí útočí) i nevýhodu (nedá se zvenčí testovat jejich dosažitelnost). V každém případě budou bez problémů dostupné pro dohledové a správcovské systémy pracující ve vnitřní síti. Jejich použití vás nebude stát ani jednu veřejnou adresu, máte jich k dispozici habaděj (prefix /48), takže si směle můžete dovolit použít prefix podsítě o standardní délce /64. Identifikátor rozhraní lze přiřadit explicitně, takže nebudete odkázáni na šifry odvozené z MAC adres, jako v případě lokálních linkových adres. Z řady pohledů se ULA jeví pro daný účel jako nejvhodnější řešení, jež kombinuje přednosti ostatních alternativ.
- *Použít veřejný prefix standardní délky /64* znamená samozřejmě brutální plýtvání, ovšem při velikosti adresního prostoru si je snadno můžeme dovolit. Připravíte se o jednu z 65 tisíc podsítí (v horším případě z 256), nicméně budete používat nejobvyklejší délku prefixu, se kterou by žádné nástroje neměly mít problém.

- *Použít dlouhý veřejný prefix /127* je varianta, která byla poměrně dlouho nedoporučována. Konkrétně se proti němu postavilo RFC 3627: *Use of /127 Prefix Length Between Routers Considered Harmful*, jehož hlavním důvodem bylo, že prefix této délky připouští jen dva identifikátory rozhraní, 0 a 1. První z nich ovšem koliduje s výběrovou adresou pro všechny směrovače v podsíti. Později byl tento postoj revidován v RFC 6164: *Using 127-Bit IPv6 Prefixes on Inter-Router Links*, který bere prefix /127 na milost a směřuje spíše k opačnému postoji – nepoužívat kratší prefixy. Zmiňuje se o jejich potenciálních bezpečnostních rizicích. Zmiňované útoky, vesměs usilující o různé druhy zahlcení, ovšem nejsou nijak specifické pro dvoubodové linky. Dlouhé prefixy jsou méně obvyklé, ale pokud nepředstavují technický problém[4], lze je označit za variantu mírně lepší proti předchozí. Šetří adresy a omezují prostor pro vznik problémů.

Doporučil bych vybrat si podle osobních preferencí cokoli kromě první varianty. Na TU v Liberci adresujeme páteř veřejnými adresami s prefixem /64, ovšem naše adresní schéma vznikalo v době, kdy ULA ještě nebyly na světě a síťový gentleman by se nezahodil s prefixem /127. Osobně mi na ULA vadí nesouvislost adresního prostoru[5], takže pokud bychom adresy předělávali, asi bychom sáhli po poslední z uvedených alternativ. Nicméně zatím k tomu není důvod.

V každém případě je rozumné vyhnout se u páteřních spojů automatické konfiguraci a používat explicitně přiřazené, jednoduché adresy. Budete je muset poměrně často psát, tak ať se vám píší dobře.

Naopak v koncových podsítích připojujících uživatelská zařízení bývá automatická konfigurace standardem. Vzhledem k řadě pragmatických problémů spojených s DHCPv6 je rozumnější (a rozhodně častější) použít bezstavovou automatickou konfiguraci. Chcete-li udržet své uživatele na uzdě a zabránit chaotickému zapojování všeho, co si zrovna přinesli, nasaďte IEEE 802.1X.

13.6 Aplikace

Jakmile máte IPv6 připojení, můžete provozovat aplikace. Těch existuje nepřeberné množství a jejich přístup k novému protokolu kolísá od naprostého ignorování po bezproblémovou transparentní podporu. Příkladem druhého extrému je třeba WWW server *Apache*. Nemusíte nic konfigurovat, při startu se porozhlédne po vašem počítači a pokud na něm běží IPv6, bude jím automaticky hovořit.

[4] Existují aktivní prvky, které nepřipouštějí prefixy podsítí delší než 64 b.
[5] Což samozřejmě neomezuje funkčnost, je to čistě estetická záležitost.

Jindy musíte IPv6 zapnout v konfiguraci. Třeba poštovní doručovatel *Postfix* vyžaduje ve svém souboru *main.cf* příkaz

```
inet_protocols = all
```

Určitou kuriozitu představují programy, které sice podporují oba protokoly, ale běžící instance zvládá jen jeden. Musíte spustit dva exempláře, z nichž jeden bude komunikovat po IPv4 a druhý po IPv6. Takové chování vykazuje například IMAP server *Dovecot*, *ISC DHCP* (o něm podrobněji v kapitole 21 na straně 379) nebo BIRD (viz část 18.1 na straně 353).

Uvedené příklady slouží jen jako ilustrace přístupů k IPv6, s nimiž se můžete setkat. Není v mých silách rozebírat zde podporu IPv6 v široké nabídce aplikací. Snaží se o to stránka

www▸ *http://www.ipv6-to-standard.org/*

jejíž odezvy při pokusu o hledání ale mají k ideálu opravdu daleko. Nicméně s Googlem v ruce, s ohněm v srdci by neměl být problém vyhledat si pro jakoukoli konkrétní aplikaci aktuální stav podpory IPv6 a zkušenosti uživatelů.

13.7 Život bez NATu

Jedním z přínosů, které se všeobecně očekávají od zavedení IPv6, je masivní opouštění NATů a obnovení přímočaré vzájemné komunikace koncových zařízení. Z toho bude těžit řada aplikací a jejich protokolů, které se obejdou bez komplikovaných obezliček obcházejících nemožnost navazování spojení do NATované sítě.

Na druhé straně ale nelze přehlížet, že NAT má i některé kladné stránky. Jednostranné navazování spojení komplikuje pirátům útok na počítače v koncové síti. NAT sice *není* bezpečnostní prvek a úspěšnému útoku nedokáže zabránit, nicméně komplikuje jej a lokální počítače alespoň částečně chrání.

Kromě toho díky NATu vznikla jednoduchá zařízení typu plug-and-play zajišťující vše potřebné pro připojení domácích sítí k Internetu. Jejich základní složkou bývá ADSL modem, kombinovaný se směrovačem, NATem, firewallem a DHCP serverem. Zařízení zpravidla funguje zcela automaticky – k poskytovateli se připojí po ADSL a protokolem DHCP získá svou adresu. Pro lokální počítače pak používá neveřejné adresy podle RFC 1918, které jim přiděluje také protokolem DHCP. Vůči venkovnímu světu je NATuje na svou vlastní jedinou adresu. Pokud uživatel nemá nadstandardní požadavky, krabičku jednoduše zapojí, spustí a může surfovat.

Je velmi žádoucí, aby se pro IPv6 nabízelo něco podobného, ovšem bez NATu a s globálními adresami v koncové síti. Touto problematikou se zabývá RFC 4864: *Local Network Protection for IPv6*. Jeho název je poněkud

zavádějící, protože se nevěnuje pouze bezpečnosti lokální sítě, ale spíše v něm najdete analýzu předností NATu, jak jsou současným trhem vnímány, a popis prvků IPv6, jimiž lze dosáhnout podobného účinku. Až na malé výjimky to je možné.

Lákavé samočinnosti koncového zařízení a jeho automatické funkce lze dosáhnout poměrně snadno, IPv6 má v tomto směru ledacos připraveno. Svou vlastní adresu může koncový ADSL modem/směrovač získat bezstavovou konfigurací či pomocí DHCPv6. Jeho prostřednictvím lze obstarat i globální prefix pro koncovou síť. Slouží k tomu rozšíření označované jako DHCP-PD (DHCP Prefix Delegation) definované v RFC 3633: *IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6*. Koncové počítače pak opět mohou použít bezstavovou automatickou konfiguraci či DHCPv6.

O základní zabezpečení lokální sítě by se měl postarat firewall integrovaný v koncovém prvku. Autoři dokumentu považují za optimální, aby zařízení tohoto typu obsahovala stavový firewall, který ve výchozím nastavení vpustí dovnitř jen datagramy z adres, na něž nedávno odešel datagram z koncové sítě. Jinými slovy, jejich chování se má velmi podobat dnešnímu NATu – neumožní navázat komunikaci směrem dovnitř, iniciativa vždy musí vzejít z vnitřní sítě. Na rozdíl od NATu bude ale dosaženo jinými prostředky a mělo by být snadné dělat z něj výjimky pro vybrané typy komunikace (IP telefonie, různé komunikátory, síťové hry). Díky nim bude umožněna ona vytoužená přímá komunikace.

Třetím okruhem, kde NAT může být prospěšný, je ochrana soukromí. Neprozradí nic o struktuře sítě za ním, ani o tom, ze kterého konkrétního počítače přišel určitý datagram. Zde IPv6 poslouží jen napůl. Strukturu koncové sítě neskryje, identitu koncových počítačů ano. K tomu poslouží náhodně generované adresy pro ochranu soukromí podle RFC 4941. Navíc díky monumentálnímu rozsahu podsítí je téměř nemyslitelné vyhledávat v nich existující stroje skenováním, jak je běžné ve světě IPv4. Soukromí koncových počítačů je chráněno solidně, zato adresy podsítí zůstávají otevřené a beze změny. To ale nebude nijak palčivý problém, protože struktura lokální sítě bývá triviální. Za ADSL modemem dnes skoro vždy bývá jedna, nanejvýš dvě podsítě (Ethernet a Wi-Fi), složitější topologie je zcela výjimečná.

Kýžené vlastnosti koncového zařízení jsou jasně narýsovány, prostředky k jejich dosažení existují. Teď ještě aby se IPv6 krabičky objevily na trhu za stejně atraktivní ceny, na jaké jsme zvyklí...

13.8 Bezpečnost koncových strojů a sítí

Dost často dnes mívají snahy o nasazení IPv6 charakter experimentu. Pokud se podaří uvést je do provozu, zúčastnění propadnou euforii a příliš se

nestarají o důsledky. Těmi mohou být nové cesty pro útočníky, když IPv4 je pečlivě lustrováno firewallem, zatímco IPv6 se nadšeně poskytuje zcela otevřeně.

Je velmi záhodno hlídat si při testování a nasazování IPv6 záda. Zatím se sice žádné útoky provedené po IPv6 nedočkaly široké publicity, je ale jen otázkou času, kdy se tak stane. Nezapomínejte ve svých firewallech na pravidla pro IPv6. Měla by být v zásadě analogická těm pro IPv4, až na použité adresy, pochopitelně.

filtrování ICMPv6

Netriviální problém představuje filtrování ICMPv6. Zkušenost ze světa IPv4 ukazuje, že ICMP bývalo zneužíváno, proto je někteří správci zcela zakázali. V IPv6 ale takto radikální krok není možný, protože ICMPv6 využívají pro přenos svých zpráv různé doprovodné mechanismy, jako je objevování sousedů či podpora mobility. Totálním zákazem by ledacos přestalo fungovat.

Situaci podrobně analyzuje RFC 4890: *Recommendations for Filtering ICMPv6 Messages in Firewalls*. Popisuje čtyři základní typy útoků, k nimž je ICMPv6 zneužíváno – zahlcení, falšování zpráv, sondování sítě a pašování informací zevnitř ven. Následně pak podle využití a důležitosti jednotlivých typů zpráv definuje velmi konkrétní sadu doporučení, jak s nimi zacházet. Jejich souhrn obsahuje tabulka 13.1, která každému typu zprávy přiděluje jedno z pěti opatření:

- **propustit!** – zpráva musí firewallem projít, jinak bude narušena funkce některých složek IPv6
- **propustit** – pokud správce nemá vážný důvod k opačnému rozhodnutí, firewall by měl zprávu propustit
- **rozhodnout** – zpráva není kritická, její osud závisí na místní policitce
- **zahodit** – zpráva by neměla firewallem projít
- **netřeba** – zpráva bude podle specifikací zpracována či zahozena a není předávána dál, není proto nutné definovat pro ni speciální pravidlo

Zprávy jsou v tabulce uspořádány podle svých typů a kódů. Doporučení navíc rozlišuje, zda je zpráva tranzitní, tedy je určena jinému adresátovi, nebo koncová, čili adresátem je některé z rozhraní stroje implementujícího firewall. Pokud používáte Linux, najdete v RFC 4890 (konkrétně v příloze B) hotový skript, jímž můžete tato pravidla nastavit.

dopady na firewally

Život firewallů ve světě IPv6 bude obecně těžší. Situaci jim komplikují dvě skutečnosti: řetězení hlaviček a předpokládané masivnější používání IPsec. Obvyklá pravidla současných firewallů pracují s IP adresami a transportními porty, podle nichž identifikují aplikace. Ovšem pokud IPv6 datagram

Zpráva	*Typ/Kód*	*Tranzitní*	*Koncová*
Destination Unreachable	1	propustit!	propustit!
Packet Too Big	2	propustit!	propustit!
Time Exceeded	3/0	propustit!	propustit!
	3/1	propustit	propustit
Parameter Problem	4/0	propustit	propustit
	4/1	propustit!	propustit!
	4/2	propustit!	propustit!
nepřiřazené chybové zprávy	5–99	rozhodnout	rozhodnout
experimentální	100–101	zahodit	zahodit
nepřiřazené chybové zprávy	102–126	rozhodnout	rozhodnout
rozšiřující	127	zahodit	zahodit
Echo Request	128	propustit!	propustit!
Echo Response	129	propustit!	propustit!
Listener Query	130	netřeba	propustit!
Listener Report	131	netřeba	propustit!
Listener Done	132	netřeba	propustit!
Router Solicitation	133	netřeba	propustit!
Router Advertisement	134	netřeba	propustit!
Neighbor Solicitation	135	netřeba	propustit!
Neighbor Advertisement	136	netřeba	propustit!
Redirect	137	netřeba	rozhodnout
Router Renumbering	138	zahodit	netřeba
Node Information Query	139	zahodit	rozhodnout
Node Information Response	140	zahodit	rozhodnout
Inverse Neighbor Discovery Solicitation	141	netřeba	propustit!
Inverse Neighbor Discovery Advertisement	142	netřeba	propustit!
Listener Report v2	143	netřeba	propustit!
Home Agent Address Discovery Request	144	propustit	netřeba
Home Agent Address Discovery Reply	145	propustit	netřeba
Mobile Prefix Solicitation	146	propustit	netřeba
Mobile Prefix Advertisement	147	propustit	netřeba
Certificate Path Solicitation	148	netřeba	propustit!
Certificate Path Advertisement	149	netřeba	propustit!
Seamoby Experimental	150	rozhodnout	netřeba
Multicast Router Advertisement	151	netřeba	propustit!
Multicast Router Solicitation	152	netřeba	propustit!
Multicast Router Termination	153	netřeba	propustit!
nepřiřazené informační zprávy	154–199	rozhodnout	zahodit
experimentální	200–201	zahodit	zahodit
nepřiřazené informační zprávy	202–254	rozhodnout	zahodit
rozšiřující	255	zahodit	zahodit

Tabulka 13.1: Filtrovací pravidla pro ICMPv6

používá rozšiřující hlavičky, odsouvá se informace o použitém transportním protokolu a jeho hlavička dál a dál. Firewall musí projít řetězec hlaviček až úplně na konec, aby se k této informaci dostal. Teoreticky může u fragmentovaného datagramu vypadnout z prvního fragmentu a být zcela nedostupná.

Ještě horší je hlavička ESP, která celý zbytek datagramu zašifruje. V tom případě o něm firewall neví vůbec nic, pokud zrovna není místem, kde končí příslušná bezpečnostní asociace a dochází k dešifrování. Nedozví se ani protokol transportní vrstvy, natož jeho porty či další informace. V podobné, i když méně extrémní situaci se ocitne, pokud bude datagram obsahovat neznámou rozšiřující hlavičku. Firewall pak bude muset volit mezi dvěma zly – paket zahodit (a možná zaříznout nové služby) nebo propustit (a riskovat ohrožení chráněné části sítě). Rozhodnutí, zda se přiklonit ke konzervativní či adrenalinové variantě, spočine na jeho správci.

Posílení přímé komunikace mezi koncovými stroji (a jejího případného šifrování) si zřejmě vynutí posun v celkovém pojetí bezpečnosti. Dnes není výjimkou spoléhání na společné firewally mezi koncovou sítí a Internetem, které chrání celou lokální síť. Tento přístup bývá označován „tvrdá slupka, měkké jádro“ a jeho problémem je, že pokud se podaří prorazit tvrdou slupku, další ochrany uvnitř sítě bývají slabé. Pro IPv6 bude zřejmě nutné posílit ochranu na koncových strojích, více ji decentralizovat.

Šifrování bude také házet klacky pod nohy všem stávajícím bezpečnostním nástrojům, jež vycházejí z analýzy obsahu datagramů. Již v současnosti ale vznikají jiné, které zkoumají datový provoz „zvenčí“. Sledují, kdo s kým a jak často komunikuje a vyhledávají v přehledu datagramových přenosů známé vzorce nebezpečných aktivit. Tento přístup je samozřejmě univerzální a bude nepochybně do budoucna dále rozvíjen.

Nelze přehlížet, že některé problémy přináší sám nový protokol. Možnost zneužití rozšiřujících hlaviček či podpůrných mechanismů je reálná a některé části knihy se jí zabývaly. Zevrubnou analýzu rizik souvisejících s IPv6 najdete v RFC 4942: *IPv6 Transition/Coexistence Security Considerations*.

domácí sítě

Předchozí odstavce této sekce byly psány s okem upřeným na sítě netriviální velikosti obhospodařované profesionálními správci. Velmi odlišné jsou z pohledu bezpečnosti domácí sítě či malé firmy spravované laickými uživateli. Typicky jsou provozovány v režimu „koupili jsme si krabici, zapojili ji a dokud vše funguje, je to dobré“. U zařízení určených pro tento segment trhu je proto velmi důležité tovární nastavení, protože často představuje nastavení celoživotní.

IETF proto ve svých textech formulovalo doporučení výrobcům, jak by se všemocné domácí připojovací krabičky měly chovat a jak by měly být zabezpečeny. RFC 6204: *Basic Requirements for IPv6 Customer Edge Routers*

shrnuje obecné požadavky na jejich schopnosti a chování. Speciálně bezpečnosti je věnováno RFC 6092: *Recommended Simple Security Capabilities in Customer Premises Equipment (CPE) for Providing Residential IPv6 Internet Service*. Jejich požadavky by se daly stručně shrnout do věty „udělejte to co nejpodobnější současnému stavu v IPv4". Tedy automatická konfigurace zařízení i koncové sítě, již připojuje. Pro zabezpečení RFC 6092 požaduje stavový firewall, který umožní navázat spojení jen směrem ze sítě do Internetu a připomíná tak asymetrické chování současných domácích prvků, u nich ovšem způsobené NATem.

13.9 IPv6 v páteřní síti

Určitým světélkem v depresivně pomalém nasazování IPv6 jsou páteřní sítě poskytovatelů Internetu. Podpora IPv6 v nich významně překračuje průměrná čísla. Zatím poslední statistiky z roku 2010 ukazují, že nadpoloviční většina tranzitních poskytovatelů má IPv6 nasazeno, v případě peeringových center číslo překračuje 40 % a v páteřích poskytovatelů připojujících koncové zákazníky se zastoupení IPv6 blíží 30 %. Firmy živící se Internetem zkrátka chtějí být připraveny.

Možností pro implementaci v tomto případě není mnoho. Automatické tunelování nepřipadá v úvahu, je třeba volit mezi konfigurovanými tunely a nějakým typem nativní podpory. První varianta se ovšem zpravidla používá jen jako provizorní řešení, tunely komplikují správu sítě a přinášejí pestrou paletu různých problémů. Pro rutinní nasazení je výrazně vhodnější sáhnout po nativním řešení. Nepříjemným důsledkem může být nutnost povýšit páteřní zařízení či alespoň jejich software.

6PE

Rozumným kompromisem pro sítě založené na MPLS může být technologie, kterou vyvinula firma Cisco Systems pod názvem 6PE. Základním principem MPLS je, že datagram se při vstupu do jádra sítě opatří značkou (MPLS label) a následně je přepravován podle této značky předem připravenými trasami, nejedná se o klasické směrování. Zúčastněné směrovače se dělí do dvou kategorií: Provider Edge (PE) jsou na okrajích sítě, jejich prostřednictvím datagramy vstupují do MPLS jádra a zase je opouštějí. Vyměňují si vzájemně informace o dostupných sítích, podle nich značkují datagramy a rozhodují tak o jejich přepravě. Směrovače kategorie Provider (P) se nacházejí uvnitř MPLS sítě. Ve skutečnosti nesměrují, jen předávají pakety sem a tam podle značek. Jednoduše a strašně rychle.

6PE přichází s myšlenkou, že přidání IPv6 do tohoto schématu nemusí znamenat zásadní změny. Jádro sítě (P směrovače) zůstane netknuto, bude nadále předávat datagramy podle značek, aniž by se staralo o jejich obsah. Přizpůsobit se musí jen hraniční PE směrovače, které se kromě IPv4 budou informovat také o dostupnosti IPv6 sítí. Jelikož znají situaci obou protokolů, využívají značky a trasy vytvořené pro IPv4 pro přepravu IPv6. Jednoduše

opatří datagramy takovou značkou, aby byly MPLS jádrem doručeny příslušnému PE směrovači.

Nové směrovače/software pak stačí pořídit jen na hranice páteřní sítě, a to ještě ne na všechny. Jestliže je IPv6 potřeba jen ve třech uzlech, jinde může zůstat vše v původním stavu. Pouze inkriminované tři hraniční směrovače se naučí IPv6 a budou se vzájemně informovat, v ostatních částech sítě bude k dispozici jen starý protokol. Investice lze díky tomu rozložit v čase podle potřeby. V místech podporujících nový protokol je přitom jeho podpora plnohodnotná, plně srovnatelná (i výkonnostně) s IPv4. Tímto způsobem je od roku 2004 implementováno IPv6 v páteřní síti CESNET2 k naprosté spokojenosti všech zúčastněných.

Specifikaci 6PE najdete v RFC 4798: *Connecting IPv6 Islands over IPv4 MPLS Using IPv6 Provider Edge Routers (6PE)*. Je dnes implementována v řadě hardwarových směrovačů (Cisco Systems, Juniper Networks).

směrovací filtry

Páteřní síť zpravidla potřebuje obcovat se svým okolím, obvykle externím protokolem BGP. I zde platí, co jsem napsal v části o bezpečnosti. Radost ze zprovoznění IPv6 občas přezáří skutečnosti, že směrovače do celého světa oznamují nesmysly. Výměna směrovacích informací v IPv4 bývá omezena řadou filtrů a politik, na něž se pro IPv6 občas zapomíná.

Gert Döring pravidelně sleduje dění v globálních směrovacích tabulkách a tepe ve svých prezentacích nejvýznamnější prohřešky. Základní jednoduchá doporučení pro filtrování směrovacích informací najdete na jeho stránce

www▸ *http://www.space.net/~gert/RIPE/ipv6-filters.html*

Pro podstatně zevrubnější popis, rozebírající doporučená pravidla pro jednotlivé body v síti, mohu doporučit

www▸ *http://www.cymru.com/Bogons/ipv6.txt*

14 BSD

Možná vás překvapí, že procházku po existujících implementacích IPv6 začínám systémem, o kterém se až tak často nepíše a nemluví. Důvod je prostý: operační systémy řady BSD sehrály v historii IPv6 velmi významnou roli. Kvalita jejich implementace nového protokolu dlouhá léta převyšovala všechny ostatní minimálně o hlavu a stále patří k naprosté špičce, i když se konkurenti hodně zlepšili. Nepřekvapí, že BSD brzy získalo certifikát *IPv6 Ready* fáze 2.

V raných dobách existovaly hned tři alternativní implementace IPv6 pro tento systém. Poměrně záhy se mezi nimi prosadil japonský projekt *KAME*, jehož kód dnes najdete jako standardní součást jádra. Setkáte se s ním ve FreeBSD od verze 4.0, OpenBSD 2.7 či NetBSD 1.5. Projekt *KAME* skončil v roce 2006 a zanechal po sobě výraznou stopu.

V této kapitole budu vycházet především z NetBSD. Nejsem odborníkem na svět BSD, takže netuším, jak moc či málo se od sebe liší konfigurační mechanismy a soubory jednotlivých variant. Základní příkazy by pochopitelně měly být totožné. Pěkný popis zprovoznění IPv6 v NetBSD najdete ve FAQ na adrese

www▸ *http://www.netbsd.org/docs/network/ipv6/*

14.1 IPv6 v jádře

Je celkem pravděpodobné, že váš systém podporuje IPv6 hned po instalaci. Nejsnadněji to ověříte, když si příkazem

```
ifconfig -a
```

necháte vypsat parametry jednotlivých rozhraní. Pokud vaše jádro podporuje IPv6, objeví se ve výpisu IPv6 adresy (přinejmenším u smyčky, rozhraní `lo0`). Chybí-li, musíte přeložit nové jádro. Jak na to se dočtete v dokumentaci svého systému. Obecně řečeno je třeba opatřit si zdrojové texty jádra, upravit konfiguraci jeho vlastností, přeložit a nainstalovat.

Klíčová volba nese jméno `INET6`. Vedle ní by vás mohly zajímat i volby `IPSEC` a `IPSEC_ESP`, které zapínají podporu IPsec. Chcete-li mít IPv6 s plnou parádou, měl by konfigurační soubor vašeho jádra obsahovat

```
options   INET6
options   IPSEC
options   IPSEC_ESP
```

14.2 Konfigurace rozhraní

automatická konfigurace

BSD samozřejmě podporuje automatickou konfiguraci. Přinejmenším NetBSD ji však neprovádí samo od sebe. Má-li váš stroj být samohybně konfigurovanou stanicí, musíte sáhnout do konfiguračního souboru */etc/rc.conf* a přidat do něj

```
ip6mode="autohost"
rtsol="YES"rtsol_flags="rozhraní"
```

Uveďte identifikátor *rozhraní*, na němž má automatická konfigurace probíhat. Může jich být i více, oddělujte jejich jména mezerami. Dopadne-li vše dobře, měl by příkaz *ifconfig* zobrazit odpovídající adresy, například

```
fxp0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> mtu 1500
        address: 00:c0:9f:04:84L8f
        media: Ethernet autoselect (100baseTX full-duplex)
        status: active
        inet 147.230.16.88 netmask 0xffff00 broadcast 147.230.16.255
        inet6 fe80::2c0:9fff:fe04:848f%fxp0 prefixlen 64 scopeid 0x1
        inet6 2001:db8:1:1:2c0:9fff:fe04:848f prefixlen 64
```

Také směrovací tabulka (viz níže) bude obsahovat implicitní směrovače, které se váš stroj naučil.

ruční konfigurace

Pokud má počítač sloužit jako směrovač, musíte rozhraní konfigurovat ručně. Navíc je třeba, aby sám posílal ohlášení směrovače a ostatní se mohli automaticky konfigurovat podle jeho informací. Do */etc/rc.conf* je proto třeba zapsat

```
ip6mode="router"
rtsol="NO"
rtadvd="YES"rtadvd_flags="rozhraní"
```

Volby `rtsol` a `rtadvd` zde způsobí, že nebude posílat výzvy směrovači a naopak bude posílat ohlášení směrovače do všech *rozhraní*, která uvedete.

O nastavení adresy rozhraní se postará

```
ifconfig rozhraní inet6 adresa prefixlen délka
```

Například poslední z adres obsažených v předchozím výpisu by nastavil příkaz

```
ifconfig fxp0 inet6 2001:db8:1:1:2c0:9fff:fe04:848f prefixlen 64
```

Parametrem `anycast` můžete adresu prohlásit za výběrovou. Konfiguraci rozhraní při startu systému zajistíte úpravou souboru */etc/ifconfig.rozhraní*, do nějž opíšete parametry, avšak bez úvodního *ifconfig*.

tunel

Tunel se v BSD chová jako další síťové rozhraní s názvem `gif`*číslo*. V některých verzích je musíte nejprve vytvořit příkazem

```
ifconfig gif číslo create
```

Poté je, opět prostřednictvím *ifconfig*, prohlásíte za tunel

```
ifconfig gif0 tunnel zdejší_IPv4 protější_IPv4
```

V některých verzích obě funkce zajišťuje příkaz *gifconfig*. Jakmile je tunel založen, zacházíte s ním jako s kterýmkoli jiným rozhraním. Například chcete-li vytvořit tunel ke směrovači s IPv4 adresou 1.2.3.4 a přidělit jeho zdejšímu konci IPv6 adresu 2001:db8:1:ff::1, postupujte následovně:

```
ifconfig gif0 tunnel 147.230.16.88 1.2.3.4
ifconfig gif0 inet6 2001:db8:1:ff::1 prefixlen 64
```

14.3 Konfigurace směrování

Jestliže je váš počítač v roli automaticky konfigurované koncové stanice, nemusíte se o směrování vůbec starat, nastaví se samo. Jen tak ze zvědavosti si můžete prohlédnout směrovací tabulku příkazem

```
route show
```

Výstup má zahuštěnou podobu a bohužel neobsahuje délku prefixu. Chcete-li rozmáchlejší formát s kompletními informacemi, sáhněte po příkazu

```
netstat -rn
```

Hodláte-li tabulku ručně upravovat, poslouží vám obvyklý příkaz *route*, konkrétně v podobě

```
route add -inet6 cíl kudy%rozhraní
```

Vzhledem k tomu, že nejbližší směrovač po cestě (*kudy*) se zadává lokální linkovou adresou, je třeba určit odpovídající rozhraní. Například pokud má vést implicitní cesta rozhraním `fxp0` na směrovač s adresou fe80::201:96ff:fe94:4ee0 vypadá příkaz takto:

```
route add -inet6 default fe80::201:96ff:fe94:4ee0%fxp0
```

O odstranění položky ze směrovací tabulky se postará tentýž příkaz, pokud `add` zaměníte za `delete`.

Dáváte-li přednost dynamickému směrování, můžete nasadit například BIRD (viz část 18.1 na straně 353) nebo jednoduchý *route6d*.

DNS Konfigurace DNS vychází standardně ze souboru */etc/resolv.conf*. Stačí v něm jako *nameserver* uvést IPv6 adresu, například

```
nameserver  2001:db8:1:16::aa
```

testování Jestli IPv6 komunikace funguje si můžete ověřit pomocí programů *ping6* a *traceroute6*. Jedná se o přímé analogie známých programů pro IPv4, upravené pro nový protokol a rozšířené o některé speciality. Pro testování DNS poslouží obvyklý *dig*. Chcete-li mu explicitně nařídit dotaz u některého IPv6 serveru, zadejte mu parametr ve tvaru *@adresa*:

```
dig @2001:db8:1:16::aa www.tul.cz
```

14.4 Přechodové mechanismy

Konfigurovaným tunelům jsem se věnoval již v části o rozhraních a směrování. Zde se podívám na automatické tunelování a možnosti pro překlad datagramů.

6to4 BSD realizuje 6to4 prostřednictvím speciálních pseudorozhraní se jmény `stf`*číslo* (stf jako six-to-four). Jejich podpora však zdaleka ne vždy bývá zapnutá. Pokud při pokusu o použití rozhraní `stf0` narazíte, budete si muset vytvořit nové jádro, v jehož konfiguraci povolíte

```
pseudo-device  stf  1
```

Pak rozhraní `stf0` přiřadíte 6to4 adresu

```
ifconfig stf0 inet6 2002:93e6:1058:1::1 prefixlen 16
```

nastavíte implicitní směrování na výběrovou adresu pro 6to4 směrovače, abyste si vytvořili cestu do nativního IPv6 Internetu[1]

```
route add -inet6 default 2002:c058:6301::
```

a vzhůru do světa 6to4.

Teredo Implementace Teredo pro operační systémy Linux a BSD nese název *Miredo*. Je dostupná buď jako balíček *net/miredo* nebo si ji můžete stáhnout z webu

www ▸ *http://www.remlab.net/miredo/*

Použití je triviální, stačí jednoduše spustit *miredo*. V systému vznikne nové rozhraní `teredo`, dosadí si adresu podle serveru, nastaví směrování a můžete fungovat. Pokud byste chtěli Teredo používat soustavněji, je vhodné v konfiguračním souboru *miredo.conf* nastavit jiný Teredo server. *Miredo* umožňuje provozovat i server a relay, pokud byste měli takové ambice.

NAT64 Problematice přechodových mechanismů se dlouhodobě věnuje kanadská firma Viagénie a z její dílny pochází implementace NAT64 nazvaná *Ecdysis*

[1] Předpokládám, že jste připojeni pouze přes 6to4. Pokud váš stroj sídlí v nativní IPv6 síti a 6to4 potřebujete jen jako doplňkový mechanismus, nemusíte se směrováním podnikat nic speciálního.

www▸ *http://ecdysis.viagenie.ca/*

Je to NAT64 se vším všudy, doplněný DNS64 v několika podobách – jako samostatný program v Perlu nebo jako modifikace serverů BIND či Unbound. NAT64 je realizován úpravou jádra a paketového filtru. Instalujete-li z binární distribuce, stačí spustit přiložený *install.sh*, který přepíše příslušné systémové soubory. Při instalaci ze zdrojových kódů se pomocí *pf_nat64.patch* upraví jejich zdrojové soubory, následuje překlad.

Konfigurace je součástí paketového filtru, do souboru */etc/pf.conf* je třeba přidat řádek, který zařídí mapování mezi NAT64 prefixem (standardně 64:ff9b::/96) a vlastní adresou NATujícího stroje, řekněme 10.1.2.3:

```
nat64 from any to 64:ff9b::/96 -> 10.1.2.3
```

Pro zásahy do DNS můžete použít buď některý z programů nabízených společně s *Ecdysis*, nebo specializovaný *totd*, jehož popis najdete níže.

TRT Pro překlad mezi IPv4 a IPv6 je k dispozici i TRT konvertor. Je realizován pseudorozhraním `faith` a démonem *faithd*. Rozhraní je opět třeba povolit v konfiguraci jádra

```
pseudo-device  faith  1
```

Jelikož je tento mechanismus poměrně nebezpečný, je jeho aktivace komplikovaná. Nejprve musíte TRT povolit jako takové příkazem *sysctl*. Následuje spuštění pseudorozhraní a zásah do směrovacích tabulek, který všechny datagramy mířící na prefix přidělený TRT předá do rozhraní `faith0`.

Závěrečným krokem je spuštění démona *faithd*. Je třeba jej spustit pro každou ze služeb, které má překládat. Pokud se má starat jen o překlad, stačí spustit

```
faithd služba
```

Jestliže ale démon pro tutéž službu běží i na zdejším počítači, je třeba přidat na příkazový řádek cestu ke zdejšímu démonovi a podobu jeho „příkazového řádku“. *faithd* pak částečně nahrazuje *inetd*: Pokud paket míří na zdejší počítač, spustí démona. V opačném případě jej předá tunelovacímu rozhraní k doručení.

Kdybych se rozhodl přidělit TRT prefix 2001:db8:1:eeee::/64 a zapnout překlad pro protokol FTP (které ale má běžet i na tomto počítači), použil bych následující sekvenci příkazů

```
sysctl -w net.inet6.ip6.keepfaith=1
ifconfig faith0 up
route add -inet6 2001:db8:1:eeee::
                  -prefixlen 64 ::1 -ifp faith0
```

```
faithd ftp /usr/libexec/ftpd ftpd -l
```

padělání DNS

Aby počítače z místní IPv6 sítě mohly rozumně spolupracovat se světem IPv4, je třeba upravovat DNS a převádět jejich AAAA dotazy na typ A a v odpovědích příslušně upravovat adresy (viz strana 282). Jednoduchým řešením tohoto problému je *Trick-or-treat daemon* aneb *totd*. Implementuje jednosměrnou změnu DNS – z místní sítě do IPv4 Internetu.

Nejedná se o plnohodnotný DNS server, proto ke své činnosti potřebuje adresu nejlépe místního serveru, jemuž bude předávat dotazy. Kromě toho je třeba vyčlenit mu jeden prefix, na nějž budou převáděny IPv4 adresy. Směrování pak musí zajistit, že datagramy směřující do této sítě budou posílány na stroj s TRT[2]. Konfigurační soubor *totd.conf* si vystačí s touto dvojicí informací:

```
forwarder 2001:db8:1:1::aa
prefix 2001:db8:1:eeee::
```

Klienti jsou konfigurováni tak, aby své dotazy předávali stroji s *totd*. Když je osloven, předá *totd* dotaz serveru uvedenému jako `forwarder`. Pokud ale nedostane odpověď na dotaz typu AAAA, dotáže se na existenci záznamu typu A. Odpověď na něj pak změní na AAAA a IPv4 adresu v ní uvedenou připojí za `prefix`, takže například odpověď

```
A  147.230.16.1
```

předá klientovi ve tvaru

```
AAAA  2001:db8:1:eeee::147.230.16.1.
```

Domácí stránka programu má adresu

www▸ *http://www.dillema.net/software/totd.html*

[2] TRT a *totd* mohou, ale nemusí nutně běžet na stejném stroji.

15 Linux

Linux svým způsobem orámoval vývoj implementací IPv6. Nejprve v roce 1996 přišel jako první s experimentální podporou nového protokolu v jádře verze 2.1.8. A jako poslední přestal svou implementaci prohlašovat za experimentální. Stalo se tak až v polovině roku 2005 u jádra 2.6.12. Po počátečním nadšení totiž následovalo několik let stagnace, kdy se IPv6 kód v jádře nevyvíjel a začal zaostávat za specifikacemi.

V roce 2000 proto v Japonsku vznikl projekt *USAGI*, jehož cílem bylo vyvinout novou, kvalitní implementaci IPv6 pro Linux. Po několika letech dvoukolejnosti, kdy standardní jádro zůstávalo při starém a v rámci USAGI pro ně vznikaly opravy a alternativní verze, se podařilo USAGI kód začlenit do základního jádra. Projekt má své stránky na adrese

www▸ *http://www.linux-ipv6.org/*

ale nedá se říci, že by zrovna hýřily aktuálností.

V současnosti se tedy jádro Linuxu může pochlubit implementací IPv6, která snese srovnání s těmi nejkvalitnějšími. Dokládají to i certifikáty *IPv6 Ready* fáze 2 pro koncový stroj i směrovač, které získal jak v základní kategorii, tak v rozšiřující pro IPsec.

15.1 Distribuce

Pokud jste pohodlní, mám pro vás dobrou zprávu. Váš systém nejspíš podporuje IPv6 rovnou po instalaci. Drtivá většina současných distribucí má ve výchozím nastavení IPv6 zapnuto a pokud jste je aktivně nevypnuli, máte je tam společně se základními nástroji pro práci s ním.

O jednoduchou kontrolu se postará příkaz *ifconfig* – pokud systém podporuje nový protokol, objeví se v jeho výstupu IPv6 adresy jednotlivých rozhraní. O podpoře IPv6 se dokonce můžete přesvědčit i bez síťového připojení, protože lokální smyčka je vždy k mání. Pokud tedy ve výstupu *ifconfig* vidíte zhruba toto

```
lo        Link encap:Místní smyčka
          inet adr:127.0.0.1 Maska:255.0.0.0
          inet6-adr: ::1/128 Rozsah:Počítač
```

váš stroj je IPv6 potentní. Jestliže výpis neobsahuje IPv6 adresu (*inet6-adr*), možná stačí jen vložit modul. Vyzkoušejte

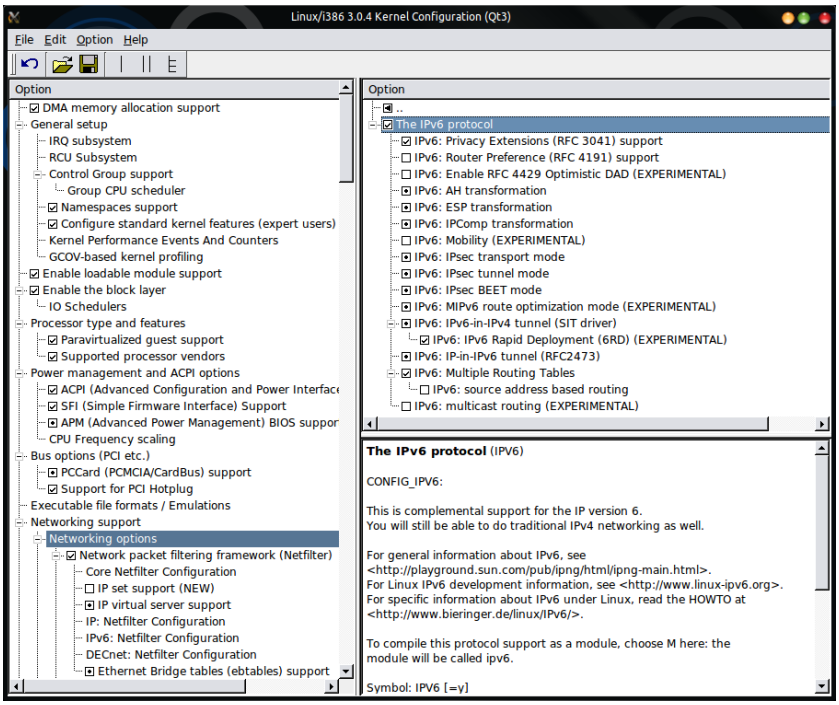
```
modprobe ipv6
```

Skončíte-li s chybovým hlášením, stávající jádro není pro IPv6 připraveno. Budete muset přeložit nové.

Zda a jak se nový protokol protlačil i do různých konfiguračních nadstaveb a grafických nástrojů pro ovládání sítě se nedá říci obecně. Situace se liší v závislosti na distribuci. Nezbývá, než vás odkázat na dokumentaci té vaší. Zde se budu věnovat textovým nástrojům, jež by měly být k dispozici víceméně všude.

15.2 Překlad jádra

Pokud si chcete nebo musíte překládat jádro systému sami, neměli byste mít s IPv6 vážnější problémy. Současné verze obsahují kvalitní kód, který je navíc implicitně zapnutý, takže byste k IPv6 měli přijít bez většího úsilí. Na obrázku 15.1 vidíte výchozí nastavení IPv6 a jeho komponent v jádře verze 3.0.4. Vše je zapnuté, stačí přeložit.



Obrázek 15.1: Konfigurace IPv6 v jádře Linuxu verze 3.0.4

Pokud se chcete sami přesvědčit nebo používáte zdrojové kódy jádra upravené pro určitou distribuci (jejichž výchozí nastavení může být odlišné), nahlédněte při konfiguraci do části *Networking support* ▷ *Networking opti-*

ons ▷ *TCP/IP networking* ▷ *The IPv6 protocol*. Klíčové je zapnutí vlastní položky *The IPv6 protocol*, která pak zpřístupní dílčí parametry. Nemusí být dostupné všechny – to závisí na nastavení ostatních voleb jádra. Po přečtení předchozích kapitol byste s pochopením významu jednotlivých položek neměli mít problémy.

Hodláte-li filtrovat datagramy (a to byste rozhodně měli), zapněte a navštivte ještě sekci *IPv6: Netfilter Configuration*. Najdete ji v sekci *Networking support* ▷ *Network packet filtering framework (Netfilter)*.

15.3 Konfigurace síťových parametrů

Pro nastavování síťových parametrů lze použít dvě alternativní cesty: tradiční dvojici *ifconfig* & *route* nebo novější a vše obalující *ip*. Osobně dávám přednost prvním dvěma, bohužel však u tunelů otevírají některá bezpečnostní rizika a je doporučeno se jim vyhýbat. Proto se budu držet spíše příkazu *ip*. V HOWTO, jehož adresu uvádím v závěru kapitoly, najdete obě varianty.

Pokud má Linux sloužit jako běžná stanice v IPv6 síti, zpravidla není třeba pro konfiguraci rozhraní dělat vůbec nic. Stačí jen počkat a nechat pracovat automatickou konfiguraci.

Chcete-li IPv6 adresu přidělit ručně, použijte příkaz

```
ip -6 addr add IPv6_adresa/délka_prefixu dev rozhraní
```

případně

```
ifconfig rozhraní add IPv6_adresa/délka_prefixu
```

Vezměme si jako příklad adresu 2001:db8:1:1:204:76ff:fe47:c se standardním prefixem podsítě o délce 64 bitů, kterou bych rád přidělil svému počítači. Použiji příkaz

```
ip -6 addr add 2001:db8:1:1:204:76ff:fe47:c/64 dev eth0
```

Stejně jako v případě IPv4 se do směrovací tabulky automaticky přidá cesta do (pod)sítě nově přidané adresy. K prohlížení a úpravám směrovací tabulky používejte buď `ip -6 route` nebo `route -A inet6`.

Čili zobrazení aktuální směrovací tabulky obstará příkaz

```
ip -6 route show
```

případně

```
route -A inet6
```

a nastavení implicitní cesty na směrovač 2001:db8:1:1::1 zajistí

```
ip -6 route add default via 2001:db8:1:1::1
```

nebo

```
route -A inet6 add default gw 2001:db8:1:1::1
```

Směrovat lze samozřejmě i dynamicky. Postará se o to například program BIRD nebo *Quagga* popsaný v kapitole 18 na straně 353.

Má-li rozhraní nebo cíl několik adres, přichází ke slovu algoritmus pro výběr adresy (viz 3.11 na straně 82). Jeho tabulku politik můžete nastavit v souboru */etc/gai.conf* (počínaje *glibc* verze 2.4). Obsahuje samostatné příkazy pro stanovení priorit a značek:

`precedence` *prefix/délka priorita*
`label` *prefix/délka značka*

V jeho manuálové stránce najdete příklad, jak se prostřednictvím těchto příkazů zapíše implicitní tabulka politik uvedená na obrázku 3.16 na straně 83. Nebuďte překvapeni, pokud soubor ve vašem systému chybí. Znamená to jen, že se používá implicitní politika.

tunely

Pro tunely se v Linuxu zřizuje speciální rozhraní s názvem `sit`*číslo* (podle Simple Interface Transition)[1]. Vytvoříte je příkazem

```
ip tunnel add sit1 mode sit remote protější_IPv4
```

kde *protější_IPv4* je IPv4 adresa protějšího konce tunelu. Můžete parametrem `local` omezit místní IPv4 adresu (jinak akceptuje jakoukoli), pomocí `dev` určit rozhraní, z nějž vede, a parametrem `ttl` omezit životnost obalujících IPv4 datagramů. Tunelové rozhraní musíte ručně aktivovat:

```
ip link set sit1 up
```

Pak už s ním můžete zacházet jako s kterýmkoli jiným – přidělit mu IPv6 adresu, zařadit do směrování a podobně. Například tunel s lokální adresou 2001:db8:1:1::baac vedoucí ke vzdálenému stroji s IPv4 adresou 1.2.3.4, který budeme využívat jsko implicitní směrovač pro IPv6, by vznikl příkazy:

```
ip tunnel add sit2 mode sit remote 1.2.3.4
ip link set sit2 up
ip -6 address add 2001:db8:1:1::baac/64 dev sit2
ip -6 route add default dev sit2
```

Přehled o existujících tunelech vám poskytne

```
ip tunnel show
```

[1] Při použití příkazu *ip* může být jméno libovolné, ale je rozumné držet se konvencí.

a pomůže i starý dobrý *ifconfig*, z jehož výstupu je mimo jiné na první pohled patrné, které tunely jste zapomněli aktivovat.

DNS Nastavení DNS nevyžaduje nic speciálního. Jednoduše v */etc/resolv.conf* uvedete v položce *nameserver* IPv6 adresu, například

```
nameserver  2001:db8:1:16::aa
```

testování Pro testování, zda vše funguje jak má, slouží příkazy *ping6* a *traceroute6*. Dělají totéž co jejich starší bratříčci bez šestky na konci, jen pro protokol IPv6. Program *dig* pro zkoumání DNS by měl fungovat bez cavyků podle obsahu *resolv.conf*. Pokud chcete jeho prostřednictvím explicitně otestovat nějaký IPv6 server, můžete použít parametr *@adresa*, třeba takto:

```
dig @2001:db8:1:16::aa www.tul.cz
```

15.4 Přechodové mechanismy

6to4 Automatické tunelování pomocí 6to4 se uvede do pohybu prakticky stejně jako běžný tunel. Řekněme, že je budete nastavovat na stroji s veřejnou IPv4 adresou 1.2.3.4. Odpovídající 6to4 prefix pro místní síť je 2002:102:304::/48. Na rozhraní pro automatické tunelování bude používat IPv6 adresu 2002:102:304::1, na rozhraní vedoucím do vnitřní sítě 2002:102:304:1::baba. Konfigurace musí začít vytvořením tunelovacího rozhraní (tentokrát je protější konec libovolný, při 6to4 jej určuje adresa) a přiřazením adres:

```
ip tunnel add sit1 mode sit remote any
ip link set sit1 up
ip -6 address add 2002:102:304::1/16 dev sit1
ip -6 address add 2002:102:304:1::baba/64 dev eth0
```

Tímto okamžikem by mělo fungovat automatické tunelování k ostatním 6to4 strojům. Abyste mohli komunikovat i s nativním IPv6 Internetem, je třeba nastavit implicitní cestu tímto tunelem k nejbližšímu 6to4 zprostředkovateli. Výběrová adresa pro něj je 2002:c058:6301:: nebo 192.88.99.1. Různí autoři si stěžují na problémy s jednou či s druhou z nich. Vyzkoušejte, která varianta bude fungovat pro vás. Buď

```
ip -6 route add default via 2002:c058:6301:: dev sit1
```

nebo

```
ip -6 route add default via ::192.88.99.1 dev sit1
```

ISATAP S ISATAPem je na Linuxu problém. Jeho podpora sice byla zařazena do standardního jádra (od verze 2.6.25), ale na webu jsou k dohledání jen nefungující návody. Od Luboše Pavlíčka jsem dostal následující tip na konfiguraci klienta (s adresou 10.1.2.3):

```
ip tunnel add is0 mode isatap local 10.1.2.3 ttl 64
ip link set is0 up
ip addr add 2001:db8:1:1::5efe:a01:203/64 dev is0
```

Do DNS je třeba zaregistrovat pod jménem *isatap* IPv4 adresu místního ISATAP směrovače. Má-li být i směrovač realizován na Linuxu, bude pro své ISATAP rozhraní obsahovat analogickou konfiguraci, pochopitelně odvozenou od jeho IPv4 adresy.

Teredo Pokud sídlíte za NATem a chcete pro automatické tunelování využít Teredo, sáhněte stejně jako v případě BSD po programu *Miredo*. Popis najdete na straně 320.

NAT64 Pro překlad mezi IPv4 a IPv6 máte k dispozici dva programy, ani jeden bohužel není ideální. Prvním je *Ecdysis*, který jsem popsal v kapitole o BSD na straně 320. Proti BSD se poněkud liší konfigurace – je třeba upravit a spustit *nat64-config.sh*. Hlavním problémem je závislost na konkrétní (postarší) verzi jádra, která omezuje použitelnost programu.

Alternativou je *TAYGA* dostupná na adrese

www▸ *http://www.litech.org/tayga/*

Nepotřebuje zásahy do jádra, využívá univerzální modul TUN. Trpí však jiným neduhem: pro každý místní IPv6 počítač, jehož datagramy má překládat do IPv4, potřebuje jednu IPv4 adresu. Ovšem pokud máte dostatek IPv4 adres pro všechny místní počítače, proč byste se namáhali s NAT64? Lepší je nasadit dvojí zásobník[2].

`tun-device`	název rozhraní pro komunikaci s jádrem
`ipv4-addr`	vlastní IPv4 adresa
`ipv6-addr`	vlastní IPv6 adresa
`prefix`	NAT64 prefix
`dynamic-pool`	prefix IPv4 adres pro mapování
`map`	statické mapování IPv4 a IPv6 adresy
`data-dir`	adresář s datovými soubory
`strict-frag-hdr`	přidávat vždy do IPv4 fragmentaci?

Tabulka 15.1: Konfigurační parametry *tayga.conf*

Pokud se rozhodnete program nasadit, základní konfigurace je obsažena v souboru *tayga.conf*. Význam jeho parametrů je vysvětlen v tabulce 15.1. Obsah může vypadat například následovně:

```
tun-device nat64
```

[2] TAYGA připouští i použití neveřejných adres podle RFC 1918, které je následně třeba NATovat. Ale i v takovém případě bude jednodušší nasadit dual-stack s neveřejnými IPv4 adresami a IPv4 NATem.

```
ipv4-addr 10.1.6.4       # vlastní adresa
prefix 2001:db8:e:e::/96 # NAT64 prefix
dynamic-pool 10.1.0.0/16 # pro IPv6 stroje
data-dir /var/db/tayga   # stavová data
```

Jeho vlastní adresa je 10.1.6.4, tu ale používá jen u paketů, jež sám odesílá. Pro mapování jsou klíčové parametry `prefix` určující prefix pro mapování IPv4 adres a `dynamic-pool`, z nějž se přidělují IPv4 adres místním strojům. Dynamické přidělení platí lehce přes dvě hodiny, platné vazby si uchovává v místě určeném parametrem `data-dir`.

TAYGA sice při svém startu dokáže vytvořit rozhraní definované parametrem `tun-device`, ale nijak se nestará o přidělení jeho adres na straně jádra, ani o směrování. Vhodný postup je:

1. Vytvořit rozhraní příkazem *tayga --mktun*
2. Nastavit IPv4 a IPv6 adresu jeho jaderné straně.
3. Nastavit směrování NAT64 prefixu a IPv4 adres používaných pro mapování do tohoto rozhraní.
4. Spustit *tayga*.

Druhou složkou je DNS64 upravující DNS dotazy. Tuto činnost zajistí *totd* pospaný v kapitole o BSD na straně 322.

15.5 Další informace

Podrobnější informace ohledně použití IPv6 v prostředí Linuxu najdete v *Linux IPv6 HOWTO* na adrese

www▸ *http://tldp.org/HOWTO/Linux+IPv6-HOWTO/*

Části věnované konfiguraci obsahuje ve dvou variantách – jak pro příkaz *ip*, tak pro mou oblíbenou tradiční dvojici *ifconfig&route*[3]. Správcem dokumentu je Peter Bieringer, na jehož stránkách

www▸ *http://www.bieringer.de/linux/IPv6/*

najdete i mnohé další užitečné informace, i když v poněkud nepřehledném uspořádání.

[3] Já vím, že je to nemoderní, ale starého psa novým kouskům až naprší a uschne. A kromě toho *ifconfig* a *route* jsou k mání v každé odrůdě Unixu.

16 Microsoft Windows

O pozici operačních systémů firmy Microsoft na osobních počítačích se jistě netřeba rozepisovat. Proto je velmi potěšující zprávou, že podpora IPv6 v nich je k dispozici již dlouho a soustavně se zlepšuje. Po několika vývojářských edicích se produkční IPv6 objevilo neprve v Service Packu 1 pro Windows XP v září 2002. O půl roku později následoval v serverové řadě systém Windows Server 2003, opět s oficiálně deklarovanou produkční podporou IPv6.

Nicméně implementace nového protokolu v těchto systémech byla dost nezvyklá. Bylo nutné ji explicitně zapnout a konfigurovat textovými příkazy, zcela odděleně od obvyklých grafických nástrojů. Pokud jste se o IPv6 aktivně nezajímali, neměli jste šanci na ně narazit.

U další generace svých operačních systémů (Windows Vista a Windows Server 2008) Microsoft představil zcela novou implementaci IP. Ta je nyní koncipována jako duální s rovnocennou podporou obou protokolů. Že se IPv6 síťování ve Windows skutečně zlepšilo dokládají i certifikace *IPv6 Ready*, které Microsoft získal.

16.1 Windows 7 a Vista

Z hlediska IPv6 není mezi Windows 7 a Windows Vista žádný významnější rozdíl. Verze 7 převzala implementaci IP od své předchůdkyně, aniž by bylo potřeba cokoli významněji měnit. Pro zjednodušení budu v této části zmiňovat vždy Windows 7, ale cokoli bude řečeno, platí shodně i pro Visty. V uživatelském rozhraní došlo ke kreativnímu přejmenování některých položek[1]. V textu používám vždy názvy z verze 7 a případné odlišnosti uvádím v závorce za nimi.

Systém Windows 7 se může pochlubit velmi slušnou podporou IPv6, jež je ve výchozím nastavení zapnuta. Obsahuje mimo jiné Teredo, takže vám poskytne připojení k IPv6 Internetu, ať jste kde jste. Máte-li na svém počítači tento systém, jste schopni komunikovat po IPv6. Tečka.

Jedním ze zřetelných rozdílů mezi verzemi 7 a Vista jsou síťová rozhraní. Jejich implementace ve Vistách představuje ryzí utrpení:

- Je jich neuvěřitelný počet – na svém koncovém počítači s jedním Ethernetem a jedním Wi-Fi jsem napočítal kolem dvaceti rozhraní.

[1] Například tlačítko *Pokročilé* v dialogu pro nastavení IPv6 už nedokázalo držet krok s tepem doby a bylo proto změněno na svěží a dynamické *Upřesnit*.

- Mají nepoužitelná jména – *Připojení k místní síti* 2* zní mnohem přívětivěji, než linuxové *eth0*, ale pište to na příkazový řádek. Navíc jsou popisy místy vyloženě matoucí (Bluetooth je vydáván za Ethernet). Naštěstí lze alespoň fyzická rozhraní přejmenovat v ovládacím panelu síťových připojení.
- Ve jménech se vyskytují čísla, která nekorespondují s indexy rozhraní – namátkou *Připojení k místní síti* 2* má index 12.

Ve Windows 7 Microsoft odstranil první z problémů. Nabídku rozhraní výrazně prořezal a vyznat se v nich je v nové verzi o řád snazší. Nepříliš použitelná jména i nekorespondující indexy však zůstaly. Kdykoli pracujete v příkazech s rozhraními, dávejte si dobrý pozor a raději vše dvakrát kontrolujte. Riziko chyby je vysoké.

Kompaktní přehled dostupných rozhraní vám poskytne příkaz

```
netsh interface ipv6 show interface
```

Příkaz *netsh* obstará většinu speciálních konfiguračních požadavků. Můžete jej používat ve dvou režimech – interaktivním a řádkovém. V interaktivním režimu jednoduše spustíte *netsh* a následně zadáváte pokyny. Příkaz si uchovává informaci o vašem kontextu. Zadáte-li mu například `interface ipv6`, přepnete se do tohoto kontextu a nemusíte jej už ve svých pokynech opakovat. Zobrazit rozhraní pak lze samotným `show interface`. Veškeré instrukce související s IPv6 patří do kontextu

```
netsh interface ipv6
```

Příjemnou vlastností interaktivního režimu je nápověda, kterou zobrazíte zadáním otazníku. Lze jej kombinovat i s názvy jednotlivých příkazů. Pokud by vás například zajímalo, co všechno si lze v IPv6 kontextu zobrazit, zadejte `show ?`.

V řádkovém režimu můžete pokyny pro *netsh* psát přímo na příkazový řádek ve formě parametrů. V této podobě budu konfigurační příkazy uvádět zde, při reálném použití je pochopitelně jedno, zda zvolíte interaktivní nebo řádkový přístup.

Nemusíte vždy uvádět kompletní instrukce. Stačí začátky jednotlivých slov, pokud jsou jednoznačné. Ke zobrazení seznamu rozhraní vás dovede i

```
netsh i ipv6 sh i
```

Ve své základní podobě jsou konfigurační zásahy provedené pomocí *netsh* dočasné a jejich efekt skončí při restartu systému. Mají-li být změny trvalé, připojte na konec `store=persistent` či jen zkráceně `persistent`. V takovém případě se jejich účinek uloží trvale.

16.1.1 Konfigurace rozhraní

S největší pravděpodobností se o ni nebudete muset vůbec starat. Systém podporuje bezstavovou automatickou konfiguraci i DHCPv6 a řídí se příznaky v ohlášení směrovače, všechno jak má být. Stačí si jen prohlédnout výsledky příkazem

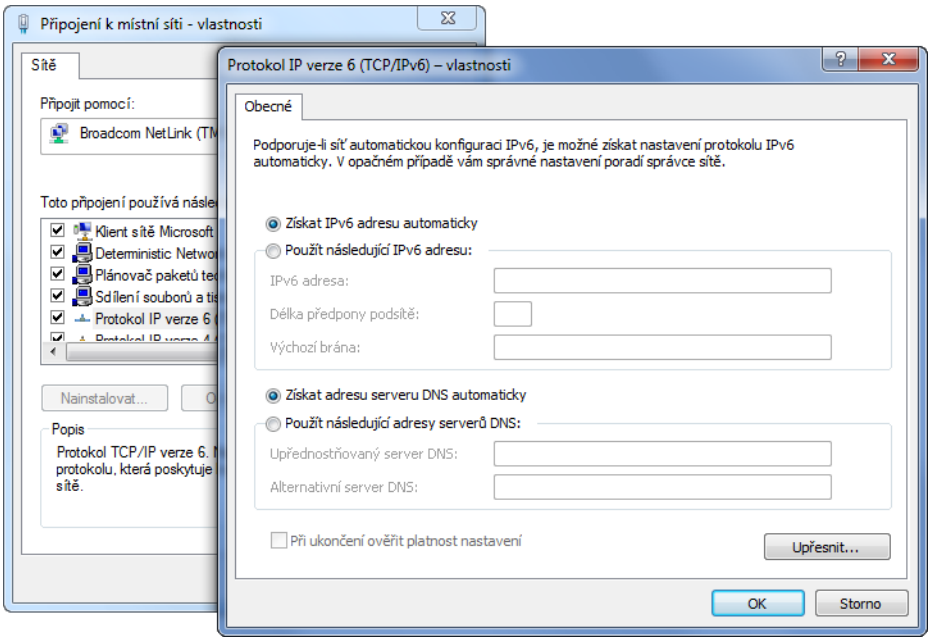
```
ipconfig
```

(případně s volbou `/all`) nebo

```
netsh interface ipv6 show addresses
```

Při bezstavové automatické konfiguraci systém nevytváří identifikátory rozhraní podle EUI-64, ale používá náhodné hodnoty[2]. Pokud by vám toto chování nevyhovovalo, lze je vypnout příkazem

```
netsh interface ipv6 set global randomize-identifiers=disabled
```



Obrázek 16.1: Konfigurace IPv6 vlastností síťového rozhraní

grafické rozhraní

Kdybyste chtěli zcela zrušit automatickou konfiguraci a přejít k manuální, dá se to zvládnout v grafickém uživatelském rozhraní. V *Ovládacích panelech* si vyberte *Síť a Internet* ▷ *Centrum síťových připojení a sdílení* ▷ *Změnit*

[2] I pro trvalé identifikátory. Kromě toho ještě vytváří dočasné náhodné identifikátory pro ochranu soukromí podle RFC 4941.

nastavení adaptéru (Vista: *Síť a Internet* ▷ *Centrum sítí a sdílení* ▷ *Spravovat síťová připojení*). Zvolte odpovídající připojení a otevřete jeho *Vlastnosti*. V nich vyberte *Protokol IP verze 6 (TCP/IPv6)* a opět klepněte na tlačítko *Vlastnosti*. Zde můžete zapnout či vypnout automatickou konfiguraci adres a DNS serverů. Tlačítkem *Upřesnit* (Vista: *Pokročilé*) pak lze mluvit i do směrování a domén automaticky připojovaných k DNS dotazům.

konfigurace adresy

Alternativně můžete konfigurovat vlastnosti rozhraní i z příkazového řádku. O nastavení (resp. přidání) IPv6 adresy se postará příkaz

```
netsh interface ipv6 add address rozhraní adresa
```

K *adrese* můžete připojit tradiční lomítkem oddělenou délku prefixu podsítě. Implicitní hodnotou je 64, takže ji lze většinou vynechat. Dalšími parametry můžete adresu prohlásit za výběrovou (`type=anycast`), omezit její životnost a podobně. Například ethernetovému připojení (rozhraní číslo 12) by se přiřadila adresa 2001:db8:1:5::abc příkazem

```
netsh interface ipv6 add address 12 2001:db8:1:5::abc
```

Windows 7 implementují algoritmus pro výběr adresy popsaný v části 3.11 na straně 82. Jeho tabulku politik vám zobrazí

```
netsh interface ipv6 show prefixpolicy
```

Pro manipulaci s ní slouží `add policy` a `delete policy`. Výchozí nastavení přidává ke standardní tabulce z obrázku 3.16 na straně 83 jeden záznam navíc pro Teredo adresy:

2001::/32 – priorita 5 – značka 5

Jeho důsledkem je, že pokud je váš počítač připojen k IPv6 Teredo tunelem, bude při komunikaci dávat přednost IPv4. Když DNS nabídne pro určité jméno IPv6 i IPv4 adresy, IPv6 při výběru vždy prohraje, nejčastěji kvůli odlišnosti značek. Zdejší konec má značku 5, protějšek obvykle 1 (běžné globální adresy) nebo 2 (6to4). Pokud má i protější strana Teredo adresu, a tedy shodnou značku 5, prohraje kvůli nižší prioritě (všechny IPv4 adresy mají značku 4 a prioritu 10). IPv6 se použije jen v případě, kdy cíl nemá IPv4 adresu. Vzhledem k bídné efektivitě Tereda je to rozumné, objemu IPv6 provozu to ale rozhodně nepřidává.

tunel

Statický konfigurovaný tunel přepravující IPv6 v IPv4 datagramech je realizován dalším rozhraním, jež založíte příkazem

```
netsh interface ipv6
    add v6v4tunnel jméno zdejší_IPv4 protější_IPv4
```

Jako parametry mu zadáváte *jméno* vytvářeného rozhraní a IPv4 adresy zdejšího a protějšího konce tunelu. Následně mu přidělíte IPv6 adresu stejně jako kterémukoli jinému rozhraní a můžete jej začít využívat. Pokud

má zdejší počítač adresu 10.1.2.3 a chci vytvořit tunel vedoucí na 10.9.8.7, jehož zdejší IPv6 adresa bude 2001:db8:1:a::22, použiji příkazy

```
netsh interface ipv6 add v6v4tunnel sit1 10.1.2.3 10.9.8.7
netsh interface ipv6 add address sit1 2001:db8:1:a::22
```

DNS Informaci o místním DNS serveru může systém získat pomocí DHCPv6. Aktuální nastavení vám sdělí příkaz

```
netsh interface ipv6 show dnsservers
```

Ručně mu lze jeho (jejich) adresy nastavit pomocí

```
netsh interface ipv6 add dns adresa_serveru
```

Jako záložní variantu, pokud systém nemá k dispozici ani DHCP ani ručně konfigurované servery, má pevně nastaveny adresy fec0:0:0:ffff::1 až 3. Jedná se o dnes již zavržené adresy lokální pro místo, jež by neměly být podporovány. Je proto záhodno systému raději poskytnout použitelné informace.

testování K ověření, zda IPv6 komunikace funguje jak má, poslouží klasické programy *ping* a *tracert*, které bez problémů akceptují i IPv6 adresy. Při problémech vyzkoušejte raději oba. Situace, kdy uspěje jen jeden z nich nejsou vzácné.

Také *nslookup* pro testování DNS obsahuje vše potřebné. Zná typ záznamů AAAA a nechá si líbit IPv6 adresy dotazovaných serverů. Ty uveďte buď jako druhý parametr příkazového řádku

```
nslookup dotaz dotazovaný_server
```

nebo v interaktivním režimu příkazem `server` *IPv6_adresa*.

sdílení připojení Dost devastující může být pro koncovou síť služba *Sdílení připojení k Internetu*. Jedná se o brontosaura z doby, kdy v domácí síti byl obvykle připojen jediný počítač, jehož prostřednictvím se pak mohly napojovat další. V současnosti, kdy připojení obvykle zajišťuje ADSL modem a zprostředkovává je libovolně velké domácí síti, nemá tato služba valného využití.

Ovšem pokud ji někde zapnete, začne příslušný počítač rázem zasílat ohlášení směrovače a nabízet adresy protokolem DHCP. Výsledkem bude zmatek v síti, jehož důsledkem může být i citelné zpomalení komunikace pro všechny. *Sdílení připojení k Internetu* lze zapnout či vypnout na kartě *Sdílení* ve vlastnostech příslušného síťového rozhraní. Stroje se zapnutým sdílením najdete zachytáváním ohlášení směrovače, například programem *Wireshark* (*www.wireshark.org*) nebo *ramond* (*ramond.sourceforge.net*).

16.1.2 Konfigurace směrování

statické

Pro ruční zásahy do směrovací tabulky lze ve Windows 7 použít dva alternativní programy: *netsh* nebo *route* s parametrem *–6*. Směrovací tabulka je samozřejmě společná, takže je jedno, po kterém z nich sáhnete, výsledek bude tentýž. *route* je kratší, ale zůstanu věrný *netsh*, kterým se konfiguruje všechno ostatní. Aktuální směrovací tabulku vám předvede

```
netsh interface ipv6 show route
```

Novou položku do ní přidáte příkazem

```
netsh interface ipv6 add route cíl rozhraní
```

Pokud není *cíl* přímo připojen, přidejte na konec ještě adresu sousedního směrovače, kterému se mají předávat data. *Cíl* se zadává v obvyklé podobě prefixu, tedy ve tvaru *adresa/délka_prefixu*. Například implicitní cestu vedoucí přes směrovač 2001:db8:1:5::1 rozhraním 12 vložíte do tabulky pomocí

```
netsh interface ipv6 add route ::/0 12 2001:db8:1:5::1
```

Odstranění cesty vypadá stejně, místo `add route` však použijte instrukci `delete route`.

Má-li se stroj chovat jako směrovač, je třeba povolit předávání paketů mezi rozhraními a také rozesílání ohlášení směrovače. Postará se o to příkaz

```
netsh interface ipv6 set interface rozhraní enabled enabled
```

První ze dvojice `enabled` se týká předávání datagramů (v plném tvaru `forwarding=enabled`), druhé pravidelného odesílání ohlášení směrovače (`advertise=enabled`). Tento příkaz je třeba provést pro každé *rozhraní*, pro které má dotyčný stroj působit jako směrovač.

dynamické

Vzhledem k tomu, že již starší systém Windows XP obsahoval podporu dynamického směrování protokolem RIPng, očekával bych i u jeho nástupců podobné schopnosti. Ovšem v nápovědě ani na webu se mi nepodařilo najít nic o dynamickém směrování IPv6 ve Windows 7 ani Vista.

16.1.3 Přechodové mechanismy

Základní přechodové mechanismy umožňující počítači komunikaci s IPv6 Internetem jsou ve Windows 7 implementovány tak, aby pracovaly co nejautomatičtěji. Pravděpodobně nebude potřeba na nic sahat a vše bude fungovat.

6to4

Pokud má počítač globální IPv4 adresu *x.y.z.ž*, založí mu systém automa-

ticky 6to4 rozhraní. Nastaví mu adresu 2002:*xxyy*:*zzžž*::*xxyy*:*zzžž* a do směrovací tabulky přidá odesílání datagramů s prefixem 2002::/16 do tohoto rozhraní.

Postará se také o předávání do světa nativního IPv6. Pomocí DNS si vyhledá adresu 6to4 zprostředkovatele provozovaného Microsoftem pod jménem *6to4.ipv6.microsoft.com* a nastaví jej jako implicitní směrovač. Nevyužívá tedy výběrovou adresu pro 6to4 zprostředkovatele. Chcete-li toto chování změnit, lze příkazem

```
netsh interface 6to4 set relay jméno
```

nastavit jméno stroje, který má 6to4 využívat jako zprostředkovatele pro přestup do nativního IPv6 Internetu.

ISATAP Pro každé LAN rozhraní, které nemá k dispozici nativní IPv6, zkouší systém automaticky vytvořit jedno tunelovací rozhraní typu ISATAP. Adresu směrovače si vyhledá pomocí DNS – ve svém doménovém jméně nahradí první část řetězcem *isatap* a pokusí se najít IPv4 pro vzniklé jméno. Pokud neuspěje, ISATAP rozhraní budou neaktivní. Ke zprovoznění v lokální síti proto stačí uvést do provozu ISATAP směrovač a zaregistrovat jeho adresu do DNS pod jménem *isatap*.

Teredo Také Teredo je standardní součástí Windows 7 a k jeho využití není třeba dělat vůbec nic. Pokud stroj nemá k dispozici nativní IPv6 ani veřejnou IPv4 adresu, systém při startu sám založí rozhraní, kontaktuje Teredo server *teredo.ipv6.microsoft.com*, podle jeho odpovědi přidělí adresu a můžete fungovat.

16.2 Windows XP

Systém Windows XP byl u Microsoftu IPv6 pionýrem. Jednalo se spíše o nakročení ve směru IPv6, protože o plnohodnotné podpoře lze mluvit jen stěží. Protokol si musíte nainstalovat, zapnout a ani pak bohužel systém není schopen rozumně fungovat v síti podporující pouze IPv6. Důvodem je, že své DNS dotazy klade výlučně protokolem IPv4. Přinejmenším kvůli Windows XP tedy musí vaše lokální DNS servery a sítě podporovat starší verzi protokolu, pochopitelně stačí neveřejné adresy podle RFC 1918.

16.2.1 Instalace

Instalace IPv6 je snadná. Přihlásíte se jako uživatel se správcovskými právy, spustíte příkazový řádek a zadáte

```
ipv6 install
```

Vůbec to nebolí a trvá to jen chvilku. Jakmile ohlásí své „OK“, můžete si užívat světa IPv6.

Pro řízení jeho činnosti máte k dispozici dvě alternativní sady konfiguračních příkazů. Jednu z nich představuje *netsh*, popsaný v části věnované Windows 7. Je dostupný i ve starším systému a používá se stejně. Kromě něj lze konfigurační zásahy provádět i příkazem *ipv6*. Tato varianta je popsána v dokumentaci systému věnované IPv6 a lze ji pro XP považovat za klasickou. Proto se jí budu držet i já.

16.2.2 Konfigurace rozhraní

V řadě případů ji vůbec nebudete potřebovat. Pokud je váš počítač zapojen jako běžná stanice v lokální síti, která podporuje IPv6, lze vše ponechat na automatické konfiguraci. Stačí se jen příkazem

```
ipv6 if
```

přesvědčit, zda je vše v pořádku. Zobrazí vám všechna IPv6 rozhraní a jejich parametry. Windows XP identifikují rozhraní pomocí indexů (pořadových čísel). Význam prvních tří indexů rozhraní je pevně přiřazen (viz tabulka 16.2). Teprve od čtyřky začínají skutečná rozhraní – síťové karty, ručně konfigurované tunely a podobně.

1	smyčka (loopback)
2	ISATAP
3	6to4

Obrázek 16.2: Standardní indexy rozhraní Windows XP

Já jsem se například dozvěděl

```
Rozhraní 4: Ethernet: Připojení k~místní síti
  používá protokol Neighbor Discovery
  používá funkci pro zjišťování směrovačů
  adresa linkové vrstvy: 00-03-47-d3-5e-14
    preferred global 2001:db8:1:1:b91c:8336:8c12:dd15,
      životnost 6d23h59m19s/23h56m58s (anonymní)
    preferred global 2001:db8:1:1:203:47ff:fed3:5e14,
      životnost 29d23h59m19s/6d23h59m19s (veřejný)
    preferred link-local fe80::203:47ff:fed3:5e14, životnost infinite
    vícesměrové vysílání interface-local ff01::1,
      1 odkazy, neuvedeno do zprávy
    vícesměrové vysílání link-local ff02::1,
      1 odkazy, neuvedeno do zprávy
    vícesměrové vysílání link-local ff02::1:ffd3:5e14,
      2 odkazy, poslední zpravodaj
    vícesměrové vysílání link-local ff02::1:ff12:dd15,
      1 odkazy, poslední zpravodaj
...
```

Z výpisu je zároveň vidět, že Windows XP mají implementovánu ochranu soukromí podle RFC 4941. Hned první adresa má náhodně vygenerovaný

identifikátor rozhraní a je označena jako anonymní. Pokud by vám to vadilo, lze mechanismus vypnout příkazem

```
netsh interface ipv6 set privacy disabled
```

Zároveň je implementován alespoň základ výběru adres, jak byl popsán v části 3.11 na straně 82. Tabulku politik zobrazí `ipv6 ppt` a upraví `ipv6 ppu`.

konfigurace adresy

Jestliže vám automatická konfigurace nevyhovuje, můžete adresu přidělit ručně. Použijte

```
ipv6 adu rozhraní/adresa
```

kde *rozhraní* je index rozhraní, kterému přidělujete danou *adresu*. Můžete připojit informaci o její životnosti v podobě parametru `life` *sekundy*. Zadáním nulové životnosti lze odstranit existující adresu. Existuje také parametr `anycast`, kterým adresu prohlásíte za výběrovou. Příklad uvedu hnedle.

tunel

Ručně konfigurovaný tunel je třeba vytvořit jako další rozhraní. K tomu slouží příkaz

```
ipv6 ifcr v6v4 zdejší_IPv4 protější_IPv4
```

Jako parametry zadáváte IPv4 adresy zdejšího a protějšího konce tunelu. Připojíte-li parametr `nd`, bude v tunelu prováděno objevování sousedů. Standardně je vypnuto a zdejší IPv6 adresu musíte přidělit ručně.

Předvedu jednoduchý příklad. Řekněme, že náš počítač má adresu 192.168.1.100 a chci jej propojit tunelem se směrovačem na adrese 1.2.3.4. Zdejší konec tunelu má mít IPv6 adresu 2001:db8:1:ff::1. Vše zařídí dvojice příkazů

```
ipv6 ifcr v6v4 192.168.1.100 1.2.3.4
ipv6 adu 5/2001:db8:1:ff::1
```

První ohlásí vytvoření rozhraní číslo 5, kterému následně přidělím adresu. Vše si mohu zkontrolovat pomocí

```
ipv6 if 5
```

16.2.3 Směrování

statické

Pro nastavení statického směrování slouží náš starý známý příkaz `ipv6`, tentokrát s parametrem `rt`. Začněme zobrazením aktuální směrovací tabulky:

```
ipv6 rt
```

Dozvíte se kudy kam, s jakou prioritou a jaká je životnost dotyčné položky. Chcete-li do tabulky zasáhnout, použijte příkaz s následujícími parametry:

```
ipv6 rtu prefix rozhraní[/adresa]
```

Pro přímo připojené cíle postačí uvést jen index *rozhraní*. Pokud se mají datagramy předávat někomu dalšímu, musíte uvést i *adresu* dalšího směrovače, kterou od *rozhraní* oddělíte lomítkem.

Parametrem `life` *sekundy* lze stanovit životnost dané položky a prostřednictvím `preference` *číslo* určíte její prioritu. Stejně jako u adres položku zrušíte zadáním nulové životnosti.

Například bych mohl jako implicitní směrovač nastavit stroj s adresou fe80::201:96ff:fe94:4ee0, který je dosažitelný po lokální síti (rozhraní 4):

```
ipv6 rtu 0::/0 4/fe80::201:96ff:fe94:4ee0
```

Má-li počítač fungovat jako směrovač, je třeba ještě sdělit rozhraní, že jeho prostřednictvím budou procházet i datagramy, které sám neodeslal ani mu nejsou určeny. K tomu musíte použít příkaz pro konfiguraci rozhraní

```
ipv6 ifc rozhraní forwards
```

dynamické

Windows XP obsahují i blíže nedefinovaný protokol pro dynamické směrování. Podle vnějších příznaků soudím, že se jedná o RIPng.

Jeho konfigurace obsahuje dva základní kroky: musíte určit prefixy, které se mají ohlašovat, a musíte nastavit, na kterých rozhraních má počítač vysílat a přijímat ohlášení. Ke konfiguraci ohlašovaných prefixů slouží příkaz

```
ipv6 rtu prefix/délka rozhraní publish
```

Není mi příliš jasné, proč je třeba uvádět identifikátor *rozhraní*, jehož prostřednictvím je dotyčný *prefix* dostupný. Ohlašování RIPng se pro dané rozhraní zapne pomocí

```
ipv6 ifc rozhraní forwards advertises
```

16.2.4 Přechodové mechanismy

6to4

Ohledně 6to4 mám pro vás dvě zprávy, jednu lepší než druhou. Windows XP podporují 6to4 a navíc se snaží, aby tato podpora byla pokud možno bezpracná.

Má-li váš počítač veřejnou IPv4 adresu, systém mu automaticky vytvoří 6to4 rozhraní (číslo 3) s odpovídající adresou a nastaví příslušné směrování. Jestliže je na dotyčném IPv4 rozhraní povoleno sdílení připojení k Internetu, začne se zároveň chovat jako 6to4 směrovač – do lokální sítě bude odesílat ohlášení směrovače s příslušným 6to4 prefixem.

Pro komunikaci s „normálním“ IPv6 světem (čili za hranice 6to4) standardně používá zprostředkující směrovač *6to4.ipv6.microsoft.com*. Nebude

to asi dvakrát efektivní, ale bude to fungovat. Pokud máte k dispozici bližšího zprostředkovatele, nastavte jej příkazem

```
netsh interface ipv6 6to4 set relay adresa
```

ISATAP Také jednoduché použití ISATAP je zcela bez práce. Má-li váš počítač IPv4 adresu, přiřadí se automaticky rozhraní číslo 2 lokální linková ISATAP adresa. Pro komunikaci s ostatními počítači ve vaší IPv4 síti by měla stačit.

Jakmile byste chtěli opustit hranice své sítě, budete potřebovat globální ISATAP prefix a informaci o přístupovém směrovači. Stávající implementace nenabízí žádné prostředky pro jejich automatické zjištění. Budete muset manuálně konfigurovat adresu dotyčného rozhraní a přidat záznam do směrovací tabulky.

Teredo Z Teredo klienta ve Windows XP mám rozpačité pocity. Sice v systému existuje, ale v mé domácí síti nedokázal získat adresu a po chvíli snažení prohlásil zdejší NAT za symetrický. Nepomohly ani experimenty s výběrem serveru. Přitom jak Windows 7, tak *Miredo* v Linuxu fungují na první pokus bez jakýchkoli konfiguračních zásahů. Pokud mu chcete dát šanci, zapněte klienta příkazem

```
netsh interface ipv6 set teredo client
```

16.2.5 Ostatní

Ve Windows XP najdete i omezenou podporu IPsec. Umí AH hlavičku s algoritmem MD5, znají i ESP, ale nedovedou šifrovat. Bezpečnostní asociace musíte konfigurovat manuálně, protože chybí IKE.

Systém obsahuje poměrně bohatou nápovědu k IPv6. Česká terminologie je bohužel dost nekompatibilní s tou mou, ale já zkrátka odmítám nazývat cache sousedů „sousední mezipamětí“, protože to odporuje logice věci a mate čtenáře. Vedle textů zaměřených na konfiguraci najdete v nápovědě i obecné informace o IPv6 jako takovém.

Hezký souhrn všech příkazů podává nápovědná stránka „Nástroje IPv6“. Při hledání „IPv6“ vám ji systém bohužel nenabídne. Nechte si vyhledat celý její název.

16.3 Další informace

Veškeré informace o podpoře IPv6 v produktech Microsoftu najdete pochopitelně na Webu. Adresa je celkem obvyklá:

www▸ *http://www.microsoft.com/ipv6/*

Obsahuje jak přehledové texty uvádějící do problematiky, tak praktické konfigurační pokyny a doporučení.

17 Cisco

Cisco Systems dlouhodobě dominuje mezi výrobci síťových prvků, podpora IPv6 z jeho strany je proto mimořádně důležitá. Naštěstí je na velmi slušné úrovni.

Ve druhém vydání jsem si na tomto místě stěžoval, že musíte sáhnout hodně hluboko do kapsy, protože IPv6 sice je k dispozici, ale je zařazeno až do nejvyšší řady IOS. Tato politika se naštěstí změnila a v současnosti jsou obě verze IP rovnocenné. V základní verzi systému dostanete základní schopnosti a protokoly pro IPv4 i IPv6, zatímco pokročilá verze vás zahrne kompletní nabídkou dovedností.

Stále ještě najdete produkty Cisco Systems, které IPv6 nepodporují. Také sortiment implementovaných vlastností se poněkud liší. Jak je tomu v případě vašich prvků se dozvíte v dokumentaci, případně od prodejce. Dobrým pomocníkem při orientaci ve vlastnostech a schopnostech jednotlivých systémů a zařízení je *Cisco Feature Navigator* na adrese

www ▸ *http://tools.cisco.com/ITDIT/CFN/jsp/index.jsp*

Tuto kapitolu chápejte jako stručný úvod do konfigurace IPv6 v IOSu pro počáteční seznámení, doporučení na detailnější literaturu najdete v jejím závěru. Nebudu zde uvádět obecné informace o konfiguraci Cisco směrovačů. Dovolím si předpokládat, že pokud máte některý produkt tohoto výrobce, umíte s ním zacházet. Proto se soustředím jen na ty prvky konfigurace, které bezprostředně souvisí s IPv6.

Implicitně je směrování IPv6 vypnuto. Takže než se vůbec můžete do něčeho pustit, zahajte konfiguraci příkazem

```
ipv6 unicast-routing
```

17.1 Konfigurace rozhraní

Cisco vyrábí směrovače, takže automatická konfigurace nepřipadá v úvahu. Naopak, směrovač musí poskytovat informace ostatním. K tomu je třeba přidělit adresy jeho rozhraním. Přejděte v konfiguračním dialogu na příslušné rozhraní a zadejte příkaz

```
ipv6 address adresa/délka_prefixu
```

Tímto příkazem zapnete na daném rozhraní IPv6 a zároveň mu přidělíte adresu a definujete zdejší (pod)síť. Současně s aktivací IPv6 se rozhraní přidělí i lokální linková adresa. Pokud se vám nechce vypisovat identifikátor rozhraní v adrese, nabízí Cisco celkem příjemnou zkratku. Zadejte

```
ipv6 address adresa/délka_prefixu eui-64
```

a směrovač si vygeneruje spodní polovinu adresy podle pravidel modifikovaného EUI–64. V *adrese* stačí zadat jen počáteční prefix délky 64 bitů. Takže kdybych chtěl Cisco směrovač připojit Ethernetem do podsítě s prefixem 2001:db8:1:1::/64, použil bych příkazy (vezměme to od podlahy)

```
config term
interface ethernet 0/1
  ipv6 address 2001:db8:1:1::/64 eui-64
```

Připojíte-li na konec slovo `anycast`, deklarujete adresu jako výběrovou. Opakováním `ipv6 address` můžete rozhraní přidělit několik adres[1]. Pro kontrolu poslouží

```
show ipv6 interface
```

Chcete-li některou z adres odstranit, použijte příkaz pro její definici s předřazeným `no`.

tunely

Tunely jsou v IOS představovány speciálními rozhraními `tunnel` *číslo*. Pro každý tunel existuje jedno. Chcete-li je vytvořit, přepněte se na ně jako na kterékoli jiné rozhraní a obvyklým způsobem mu přidělte IPv6 adresu (`ipv6 address`). Poté musíte definovat oba konce tunelu. Poslouží k tomu příkazy

```
tunnel source rozhraní nebo IPv4 adresa
tunnel destination IPv4 adresa
```

Zdejší konec můžete definovat buď IPv4 adresou tohoto směrovače nebo typem a číslem rozhraní. Určuje adresu, kterou bude směrovač používat při tunelování jako zdrojovou. U protějšího konce musíte uvést IPv4 adresu. Na závěr ještě definujte, že se jedná o tunelování IPv6 paketů v IPv4:

```
tunnel mode ipv6ip
```

Řekněme, že chci vytvořit tunel ke směrovači 1.2.3.4, který pro zdejší konec bude využívat IPv4 adresu přidělenou jednomu z ethernetových rozhraní. Definice bude vypadat takto:

```
interface tunnel 0
  ipv6 address 2001:db8:1:fff::/64 eui-64
  tunnel source ethernet 0/3
  tunnel destination 1.2.3.4
  tunnel mode ipv6ip
```

[1] Na rozdíl od IPv4 nemusíte přidávat `secondary`, v IPv6 je několik adres pro jedno rozhraní normální stav.

Tím je zdejší konec tunelu založen.

objevování sousedů

Směrovač musí okolním počítačům poskytovat informace pro bezstavovou automatickou konfiguraci – zasílat ohlášení směrovače. Cisco IOS tuto činnost zajišťuje, aniž byste se museli o něco starat. Směrovač pro každé rozhraní implicitně posílá ohlášení směrovače se všemi prefixy, které jste mu přiřadili. Jako dobu platnosti jim přiděluje 30 dnů a dobu preferování týden.

Pokud chcete upravit chování směrovače při objevování sousedů, použijte příkazy ze skupiny `ipv6 nd`. Nejzajímavější bude nepochybně možnost nastavit seznam a parametry prefixů, které bude do daného rozhraní ohlašovat. Zajistí to příkaz

```
ipv6 nd prefix prefix/délka
```

Lze přidat řadu různých parametrů, jimiž ovlivníte vlastnosti prefixu. Patří mezi ně životnost, doba preferování, či zákaz používání bezstavové automatické konfigurace `no-autoconfig`. Jakmile použijete `ipv6 nd prefix` v konfiguraci rozhraní, zruší se implicitní chování a budou ohlašovány jen prefixy, které jste explicitně uvedli.

Automatickou konfiguraci lze dále ovlivňovat pomocí

```
ipv6 nd managed-config-flag
ipv6 nd other-config-flag
```

První do ohlášení směrovače posílané tímto rozhraním zařadí příznak k použití DHCPv6, druhý slouží pro bezstavové DHCPv6.

Zajímavý může být i příkaz

```
ipv6 nd ra lifetime sekundy
```

který definuje, jak dlouho si může klient ponechat tento směrovač v seznamu implicitních směrovačů. Implicitní hodnotou je půl hodiny, což nejspíš vyhoví ve většině případů. Jestliže ale nechcete, aby přes tento směrovač vedly implicitní cesty zdejších klientů, nastavte `ra lifetime` na nulu.

Lehce rozvinu předchozí příklady a předvedu konfiguraci rozhraní pro dva Ethernety do lokální sítě a tunel kamsi ven. V místní síti (0/1) se používá bezstavová automatická konfigurace doplněná bezstavovým DHCPv6, v síti připojené k 0/2 probíhá automatická konfigurace výlučně pomocí DHCPv6 (bezstavová je zakázána). Do tunelu se ohlášení směrovače neposílá:

```
interface ethernet 0/1
  description Hlavni segment
  ipv6 address 2001:db8:1:1::aaaa/64
  ipv6 nd other-config-flag
  ipv6 nd prefix 2001:db8:1:1::/64
```

```
interface ethernet 0/2
  description Uctarna
  ipv6 address 2001:db8:1:2::bbbb/64
  ipv6 nd managed-config-flag
  ipv6 nd prefix 2001:db8:1:2::/64 no-autoconfig

interface tunnel 0
  description Externi pripojeni
  ipv6 address 2001:a:b:c::2/64
  tunnel source ethernet 0/3
  tunnel destination 1.2.3.4
  tunnel mode ipv6ip
  ipv6 nd ra suppress
```

Poslední konfigurační příkaz je ve skutečnosti zbytečný, protože do tunelů se ohlášení směrovače implicitně neposílá. Naopak pokud je žádoucí, je třeba je zapnout, jak uvidíte na příkladu ISATAP rozhraní.

17.2 Směrování

Statické směrování se konfiguruje zcela standardně, jen místo příkazu `ip route` používáte `ipv6 route`. Jeho tvar je

```
ipv6 route prefix/délka kudy
```

Jako parametr *kudy* poslouží buď IPv6 adresa směrovače, kterému mají být datagramy pro daný *prefix* předány, nebo typ a číslo rozhraní, kterým je má směrovač odeslat (pro dvoubodové spoje). Kdybych tedy chtěl použít výše uvedený tunel jako implicitní cestu ven, zařídil by to příkaz

```
ipv6 route ::/0 tunnel 0
```

17.2.1 RIPng

Nastavení RIPu obsahuje dva nejdůležitější kroky: musíte spustit vlastní proces pro RIPng a následně jej musíte aktivovat na jednotlivých rozhraních. Proces má svůj identifikátor, jehož prostřednictvím se na něj později odkazujete. Spuštění RIPng jako takového zajistí

```
ipv6 router rip identifikátor
```

Tím se zároveň ocitnete v konfiguraci procesu RIPng. Můžete zde třeba nastavit, které cesty z jiných zdrojů se mají předávat do RIPu. Například šíření statických cest zajistí

```
redistribute static
```

Pro každé rozhraní, na kterém chcete zapnout RIP (čili vysílat a přijímat aktualizace směrovací tabulky), musíte v konfiguraci rozhraní uvést příkaz

```
ipv6 rip identifikátor enable
```

Chcete-li, aby se šířila a přijímala i informace o implicitní cestě na některém rozhraní, použijte v jeho konfiguraci

```
ipv6 rip identifikátor default-information
```

Pro kontrolu nastavení RIPng slouží

```
show ipv6 rip
```

Uveďme si příklad konfigurace, který navazuje na výše definovaná rozhraní a tunely. Jako identifikátor pro RIPng proces použijeme „tul“.

```
ipv6 router rip tul
  redistribute static

interface ethernet 0/1
  ipv6 rip tul enable
  ipv6 rip tul default-information

interface ethernet 0/2
  ipv6 rip tul enable
  ipv6 rip tul default-information

interface tunnel 0
  ipv6 rip tul enable
```

17.2.2 OSPFv3

Podobně jako v případě RIPu je třeba i pro OSPF spustit směrovací proces a následně do něj zahrnout jednotlivá rozhraní. O základní spuštění OSPFv3 a vstup do jeho globální konfigurace se postará

```
ipv6 router ospf identifikátor_procesu
```

Místo jména používá OSPF číselný *identifikátor_procesu*. V jednom směrovači může běžet několik nezávislých OSPF procesů a prostřednictvím identifikátorů pak určujete, ke kterému z nich se jednotlivé příkazy vztahují.

Pro svůj vnitřní život OSPF potřebuje přidělit identifikátory také směrovačům. Často si s nimi nemusíte lámat hlavu, pokud některé rozhraní má přiřazenu IPv4 adresu, směrovač ji automaticky použije jako identifikátor (preferuje adresu lokální smyčky). Jestliže ale nikde žádná IPv4 adresa není, vložte do konfigurace OSPF procesu

```
router-id identifikátor
```

Identifikátor se zapisuje ve stejném formátu jako IPv4 adresa. Stanovte si vhodná pravidla pro identifikaci směrovačů a označte je.

Na úrovni OSPF procesu se také nastavuje agregace směrování. Standardně protokol šíří informace o prefixech jednotlivých podsítí, k nimž je směrovač přímo připojen. Můžete však předepsat, že celá oblast má být vůči zbytku světa reprezentována jen jediným prefixem, který obaluje všechny konkrétnější prefixy uvnitř ní. Slouží k tomu příkaz

```
area identifikátor_oblasti range prefix/délka
```

Příkazem `redistribute` lze nařídit předávání informací získaných jinou cestou do OSPF.

Na rozdíl od OSPFv2 musíte OSPFv3 explicitně povolit v konfiguraci rozhraní, jež se má účastnit výměny směrovacích informací. Pro každé zapojené rozhraní zadejte příkaz

```
ipv6 ospf identifikátor_procesu area identifikátor_oblasti
```

Na našem příkladném směrovači budou oba Ethernety náležet do oblasti 0.0.0.100 přidělené zdejší lokální síti. Celou oblast zahrneme do jednoho agregovaného prefixu. Tunel pak zařadíme do páteřní oblasti:

```
ipv6 router ospf 777
  router-id 11.22.33.44
  area 0.0.0.100 range 2001:db8:1::/48
  redistribute static rip connected

interface ethernet 0/1
  ipv6 ospf 777 area 0.0.0.100

interface ethernet 0/2
  ipv6 ospf 777 area 0.0.0.100

interface tunnel 0
  ipv6 ospf 777 area 0.0.0.0
```

Aktuální stav OSPF lze opět zjišťovat pomocí

```
show ipv6 ospf
```

17.3 Mobilita

Směrovače Cisco dovedou pracovat jako domácí agenti. Základní konfigurace je navíc poměrně jednoduchá, stačí v konfiguraci každého rozhraní, pro něž má směrovač tuto úlohu vykonávat, uvést příkaz

```
ipv6 mobile home-agent
```

Lehce matoucí je, že stejnojmenným příkazem, ovšem na globální úrovni, vstupujete do konfigurace domácího agenta. Slouží k nastavení základních parametrů poskytované služby. Asi nejčastějšími příkazy v této sekci budou

```
binding počet_klientů
```

omezující maximální počet mobilních uzlů, které je směrovač ochoten podporovat.

```
binding access jméno_acl
```

odkazuje na přístupový seznam oddělující zrno od plev. Tedy klienty, jimž je povoleno domácího agenta využívat. A konečně

```
authentication spi číslo key ascii klíč
```

určuje parametry pro jejich autentizaci. Příkaz je ve skutečnosti košatější, nicméně základní tvar často postačí. Implicitním autentizačním algoritmem je HMAC-SHA1.

Například domácího agenta, který poskytuje své služby na Ethernetech 0/1 a 0/2 a je ochoten přijmout až 100 klientů s autentizací by definovaly následující příkazy

```
ipv6 mobile home-agent
  binding 100
  authentication spi 100 key ascii topsecret

interface ethernet 0/1
  ipv6 mobile home-agent

interface ethernet 0/2
  ipv6 mobile home-agent
```

17.4 Přechodové mechanismy

6to4

6to4 je v Cisco IOS implementováno jako speciální typ tunelu. Jeho konfigurace se proto podobá konfiguraci běžného tunelu, neuvádí se však adresa protějšího konce, protože je pro každý datagram jiná a 6to4 ji určuje automaticky z cílové adresy. A pochopitelně se liší i režim práce tunelu. Kdybych chtěl pro svůj směrovač aktivovat 6to4 na rozhraní `tunnel 10` a odesílat tunelované datagramy z rozhraní Ethernet 0/3 s IPv4 adresou 1.2.3.4, použil bych následující konfiguraci:

```
interface tunnel 10
  ipv6 address 2002:102:304::1/16
```

```
  tunnel source ethernet 0/3
  tunnel mode ipv6ip 6to4
```

Použitá délka prefixu 16 by měla do směrovací tabulky vložit položku, podle níž se datagramy směřující na 2002::/16 budou posílat do tohoto rozhraní. Je-li 6to4 jediným IPv6 spojením směrovače do světa, je záhodno nastavit implicitní cestu na výběrovou adresu pro 6to4 zprostředkovatele

```
ipv6 route ::/0 2002:c058:6301::
```

ISATAP Hardwarový směrovač bude těžko figurovat jako ISATAP klient, ale naopak by měl být připraven hrát úlohu směrovače pro tento přechodový mechanismus. Opět se jedná o speciální režim tunelu, jehož protější konec zůstane nedefinován, protože je proměnlivý. Rozhraní implementující zdejší konec tunelu (použijeme oblíbený Ethernet 0/3) musí mít nastavenu IPv4 adresu a IPv6 adresa tunelu musí používat identifikátor rozhraní typu EUI-64. ISATAP ke své činnosti potřebuje ohlášení směrovače zasílaná tunelem klientům, je třeba je explicitně povolit. Konfigurace by mohla vypadat následovně:

```
interface tunnel 3
  ipv6 address 2001:db8:1:ee::/64 eui-64
  tunnel source ethernet 0/3
  tunnel mode ipv6ip isatap
  no ipv6 nd ra suppress
```

NAT64 Je to poměrně překvapující, ale směrovače Cisco podporují NAT64 jen částečně v podobě tak zvaného bezstavového NAT64. Znamená to, že sice umí vzájemně překládat IPv4 a IPv6 datagramy, ale nemapují adresy. Bezstavový NAT64 dokáže zpracovávat jen datagramy s IPv6 adresami obsahujícími IPv4, kdy nepotřebuje žádné převodní tabulky a IPv4 adresu si jednoduše vyzvedne z IPv6.

I taková služba může být užitečná. Lze si ji představit například ve spojení s neveřejnými IPv4 adresami podle RFC 1918, z nichž vytvoříte IPv6 adresy obsahující IPv4 pro adresaci místních strojů, jimž bezstavový NAT64 zpřístupní veřejný IPv4 Internet. Nicméně taková konstrukce je dost omezující (bude pravděpodobně znamenat dvojí IPv6 adresy lokálních strojů) a proti plnohodnotnému NAT64 dost neohrabaná.

Výrobce slíbil úplný (stavový) NAT64 na rok 2011. Než svůj slib splní, podívejme se alespoň na bezstavového bratříčka. Jeho konfigurace je triviální – na globální úrovni definujete prefix pro NAT64 a určíte, do kterého IPv6 rozhraní se mají předávat příchozí IPv4 datagramy po překladu:

```
nat64 prefix stateless prefix/délka
nat64 route IPv4prefix/délka rozhraní
```

V konfiguraci každého rozhraní, kde má být uplatňován NAT64, pak jednoduše uveďte

```
nat64 enable
```

17.5 Skupinové adresování

Směrovače Cisco se v současnosti již mohou pochlubit dostatečně kvalitní a spolehlivou podporou pro přepravu skupinově adresovaných datagramů. Stejně jako samotné IPv6 vyžaduje zapnutí, tentokrát příkazem

```
ipv6 multicast-routing
```

MLD Aktivuje vše potřebné, především protokoly MLD a PIM pro všechna rozhraní, na nichž je zapnuto IPv6. Pomocí MLD se směrovač dozvídá o členech jednotlivých skupin. Aktuální verze IOS podporují MLD verze 2. Protokol PIM pak používá k vlastní distribuci skupinově adresovaných dat. Podporovány jsou všechny jeho odrůdy.

PIM Klíčovou informací pro něj jsou adresy shromaždišť jednotlivých skupin, tedy kořenů jejich sdílených stromů. Mohou být obsaženy přímo v adresách (embedded RP, strana 71), pak není co řešit. Tento typ adres je v IOS podporován a automaticky rozpoznáván podle příznaků v adrese. Totéž platí o adresách pro SSM. U ostatních bude třeba napovědět příkazem

```
ipv6 pim rp-address adresa_shromaždiště
```

Celá PIM doména mívá často jediné shromaždiště, takže vystačíte se základním tvarem příkazu, který se vztahuje na všechny skupiny. Ve složitějších případech můžete na konec přidat jméno přístupového seznamu. Pak bude uvedené shromaždiště platit jen pro skupiny, které mu vyhovují. Například chcete-li pro skupiny s prefixem ff05::/64 používat shromaždiště 2001:db8:1::aa, zatímco pro všechny ostatní 2001:db8:a::11, zajistí to následující sada příkazů:

```
ipv6 access-list skupina1
  permit ipv6 any ff05::/64
ipv6 pim rp-address 2001:db8:1::aa skupina1
ipv6 access-list ostatni-skupiny
  deny ipv6 any ff05::/64
  permit ipv6 any any
ipv6 pim rp-address 2001:db8:a::11 ostatni-skupiny
```

Pokud přidáte na konec příkazu `ipv6 pim rp-address` slůvko `bidir`, prohlásíte sdílený strom za obousměrný, tedy BIDIR-PIM.

Ve výchozím stavu je skupinové směrování liberální a je ochotno distribuovat data od kohokoli. K omezení možných zdrojů slouží

```
ipv6 pim accept-register list přístupový_seznam
```

který povolí registrované datagramy, čili předávání skupinových datagramů do distribučního stromu, jen pro stroje vyhovující zadanému *přístupovému seznamu*.

Ke zjišťování informací o aktuálním stavu MLD a PIM poslouží především příkazy

```
sh ipv6 mld interface
sh ipv6 mld groups
sh ipv6 pim
```

L2/L3 přepínače

Pokud máte v síti L2 nebo L3 přepínače firmy Cisco, měli byste podniknout příslušné konfigurační úpravy i na nich. Prvním krokem je nastavit prostřednictvím *Switch Database Management (SDM)* šablonu pro společný IPv4 a IPv6 provoz:

```
sdm prefer dual-ipv4-and-ipv6 šablona
```

Přípustnými hodnotami *šablony* jsou `default` (vyrovnané vlastnosti pro L2 přepínání a L3 směrování), `routing` (rozšířené možnosti směrování) a `vlan` (bez směrování, jen L2 funkce).

Následně zapnete podporu MLD příkazem

```
ipv6 mld snooping
```

17.6 Další informace

Aktuální stav podpory IPv6 ve směrovačích Cisco, kompletní dokumentaci a řadu dalších materiálů najdete na adrese

www▸ *http://www.cisco.com/ipv6/*

Pro detailní studium vlastností a konfigurace skvěle poslouží *Cisco IOS IPv6 Configuration Guide* – kniha o rozsahu 600 stran, která s tradiční Cisco důkladností popisuje teoretické základy a pitvá detaily konfiguračních příkazů. Adresa by vydala na samostatný odstavec, ale Googlem ji snadno najdete. Užitečným doplňkem je pak *Cisco IOS IPv6 Command Reference*. Speciálně pro skupinové adresování se bude hodit *Implementing IPv6 Multicast*.

Dáváte-li přednost tištěné literatuře, za pozornost určitě stojí kniha [12], která se věnuje nasazení IPv6 v sítích různých typů na platformě Cisco.

18 Směrovací programy

Pokud nejsou datové objemy přenášené ve vaší síti příliš velké, můžete uvažovat o softwarovém směrovači realizovaném na PC. V jednoduchém případě vystačíte se statickým nastavením doplněným o ohlašování směrovače. Vše je popsáno v předchozích kapitolách věnovaných jednotlivým operačním systémům a v kapitole 19.1 na straně 369. Pokud ale má váš směrovač podporovat dynamické směrování a hovořit některým ze směrovacích protokolů, potřebujete specializovaný program.

Volně dostupných produktů je k dispozici hned několik. Psí kusy se směrovanými datagramy dokáže dělat vysoce modulární *Click* (*http://read.cs.ucla.edu/click/*). Zajímavým projektem je *XORP (eXtensible Open Router Platform, www.xorp.org)*, který zatím jako jediný podporuje i skupinové směrovací protokoly. Jeho konfigurační jazyk, částečně inspirovaný směrovači Juniper, je bohužel dost krkolomný. Psát adresy v podobě

```
address 2001:db8:1:33::22:11 {
    prefix-length: 64
}
```

je zkrátka proti přírodě.

V této kapitole se přidržím dvou nejznámějších a nejrozšířenějších zástupců této kategorie – programů BIRD a *Quagga*. Díky nim může počítač s operačním systémem typu Unix pracovat jako plnohodnotný směrovač a vyměňovat si s okolím směrovací informace prostřednictvím různých směrovacích protokolů.

Schopnosti obou programů jsou srovnatelné, byly portovány na podobné platformy (*Quagga* navíc nabízí Solaris), hovoří stejnými protokoly: RIPv2, RIPng, OSPFv2, OSPFv3 a BGP4+. *Quagga* může nabídnout především delší tradici a větší rozšíření. BIRD naproti tomu vychází lépe ze srovnávacích testů (nižší zátěž systému, vyšší stabilita) a práce s ním je jednodušší. Pokud nejste zatíženi minulostí, představuje dnes BIRD asi lepší volbu.

18.1 BIRD Internet Routing Daemon

Směrovací démon BIRD potěší vlastence, protože se jedná o výlučně domácí produkt. Původně vznikl jako ročníkový projekt na MFF UK, později upadl do hibernace, ovšem v posledních letech se v péči CZ.NIC opět čile rozvíjí a prosazuje. Dokonce začal ohrožovat pozici programu *Quagga* jako de facto standardu v oblasti softwarových směrovačů. Jeho exempláře čelí například těžké zátěži v nejednom peeringovém centru.

Program je k mání na adrese

www▸ *http://bird.network.cz/*

s licencí GNU GPL. V některých systémech jej lze instalovat v podobě balíčku, jinde budete muset překládat ze zdrojového kódu. Kromě Linuxu, na němž je vyvíjen, byl portován i na systémy NetBSD, FreeBSD a OpenBSD.

BIRD podporuje obě verze IP, nikoli však najednou. Musíte provozovat jednoho démona pro IPv4 a druhého pro IPv6. Tato skutečnost se odráží i v balíčcích, kdy každá verze IP má obvykle svůj balíček – *bird* a *bird6*. Při překladu ze zdrojových kódů musíte u varianty pro IPv6 jeho podporu explicitně zapnout:

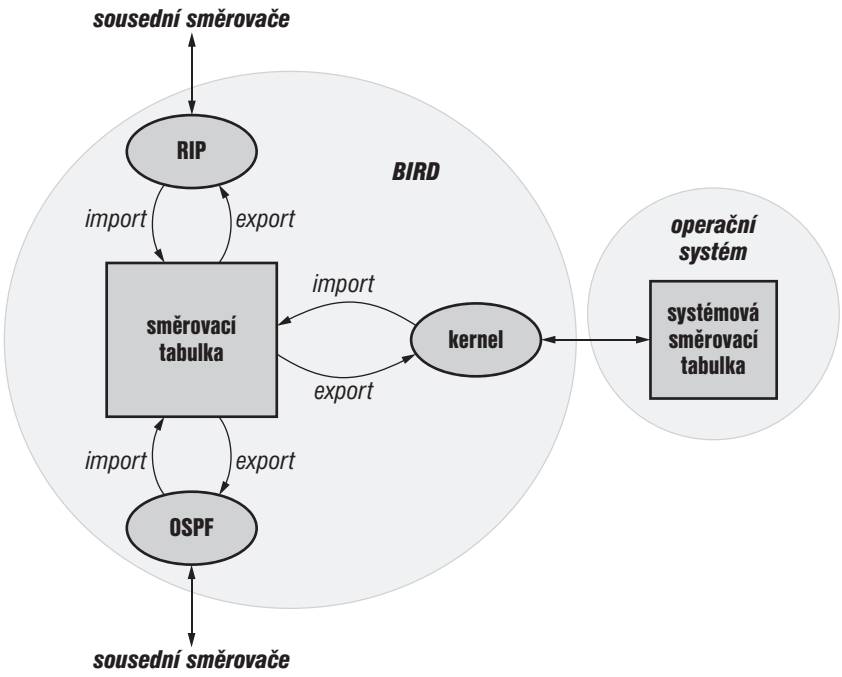
```
configure --enable-ipv6
make
make install
```

BIRD má svou vlastní směrovací tabulku, která může a nemusí být synchronizována se systémovou. Jeho prostřednictvím tedy lze vytvořit směrovač, který si vyměňuje směrovací informace s ostatními, ale nemá vliv na směrování stroje, na němž sídlí. Vnitřních tabulek může být dokonce i víc, včetně pravidel pro přenos informací mezi nimi. BIRD se pak chová jako několik směrovačů v jednom. Tyto schopnosti, stejně jako velmi silný jazyk pro definici pravidel omezujících výměnu směrovacích informací, jsou určeny pro náročnější uživatele, jako jsou poskytovatelé Internetu či peeringová centra. Zde zůstanu jen u velmi základního modelu činnosti, kdy BIRD má zajistit dynamické směrování stroje, na němž je provozován.

18.1.1 Základy konfigurace

Celá konfigurace programu je uložena v souboru *bird6.conf* (pro IPv4 slouží *bird.conf*). Jeho výchozím umístěním je adresář */usr/local/etc*, ovšem při instalaci z balíčku bude pravděpodobně umístěn rovnou v */etc*. V nekomplikovaných situacích bude jeho obsah dost jednoduchý. BIRD se totiž nesnaží obsáhnout celé síťování svého stroje, ale stará se pouze o směrování. Například konfigurace adres jednotlivých rozhraní jde zcela mimo něj, tu je třeba zajistit odpovídajícími příkazy operačního systému, popsanými v předchozích kapitolách.

Základní koncept činnosti programu je takový, že si udržuje svou směrovací tabulku. Dělá to prostřednictvím směrovacích protokolů, přičemž několik protokolů může pracovat se společnou tabulkou. Informace získané jedním z nich jsou uloženy do tabulky a ostatními směrovacími protokoly pak předávány dál. Převzetí směrovacích informací z protokolu do tabulky je označováno jako import, odeslání informací z tabulky některým protokolem je export. Oba směry lze omezovat importními a exportními filtry. Stejným



Obrázek 18.1: Architektura programu BIRD

způsobem je řešena i spolupráce se systémovou směrovací tabulkou, která se chová jako další (fiktivní) směrovací protokol.

Konfigurační soubor obsahuje typicky nastavení globálních vlastností, jako je například úroveň ladicích informací, záznam o činnosti směrovače či jeho identifikátor. Za nimi pak následují popisy parametrů jednotlivých směrovacích protokolů v jednotném tvaru

```
protocol název {
    parametry
}
```

Z globálních voleb stojí za zmínku v první řadě identifikátor směrovače. Je jím IPv4 adresa (i v případě IPv6 směrovače, identifikátory jsou čtyřbajtové) a pro démona obsluhujícího IPv6 je jeho uvedení povinné. Zapisuje se ve tvaru

```
router id IPv4_adresa
```

Globálně se nastavuje také vedení záznamů o činnosti směrovače příkazem

```
log kam co
```

Parametr *kam* určuje, kam mají být příslušné informace zaznamenávány. Můžete použít klíčová slova `syslog` pro záznam systémovým programem *syslog*, `stderr` pro jejich odesílání do standardního chybového výstupu nebo uvést jméno cílového souboru uzavřené do uvozovek. *Co* rozhoduje o tom, kterých informací se dané nastavení týká. Klíčové slovo `all` se vztahuje na všechny typy událostí. Nebo můžete ve složených závorkách uvést čárkami oddělovaný seznam tříd událostí. Definovány jsou třídy `info`, `warning`, `error`, `fatal`, `debug`, `trace`, `remote` (chyby sousedů), `auth` a `bug` (vnitřní chyby BIRDu). Ve výchozím nastavení se vše zaznamenává přes *syslog*. Chcete-li například vše zapisovat do souboru */var/log/bird.log*, použijte

```
log "/var/log/bird.log" all;
```

Na globální úrovni lze také pomocí `debug protocol` nastavit implicitní úroveň ladicích informací. Toto nastavení lze pak přepsat ve vlastnostech každého jednotlivého protokolu. Hodnotou může být `all`, `off`, nebo ve složených závorkách uzavřený čárkami oddělovaný seznam ladicích událostí `states` (zahájení/ukončení protokolu), `routes` (změny ve směrování), `filters` (záznamy o filtrování), `interfaces` (události vyvolané rozhraními), `events` (interní události protokolu) a `packets` (odeslané a přijaté pakety daného protokolu). Pokud by měl BIRD zaznamenávat změny stavu, směrování a rozhraní používaných protokolů, vypadalo by globální nastavení následovně:

```
debug protocol { states, routes, interfaces };
```

18.1.2 Protokoly

Interakce směrovací tabulky BIRDu s okolím probíhá prostřednictvím protokolů. V jejich nastavení se skrývá jádro celé konfigurace. Podívejme se na ně podrobněji.

kernel

Začnu pseudoprotokolem `kernel`, který zprostředkovává styk se systémovou směrovací tabulkou. Má-li BIRD řídit směrování daného stroje, bude se exportovat vše. Podobně by informace měly téci i v protisměru, což znamená instrukcí `learn` vydat BIRDu pokyn, aby se učil cesty ze systémových tabulek, a případně filtrem `include` přitékající informace omezit. Volbou `scan time` určíte, po kolika sekundách je má procházet.

Při ukončení může BIRD „své“ cesty ze systémové tabulky vymazat, nebo je v ní ponechat. Ve druhém případě použijte instrukci `persist`. Konfigurace protokolu `kernel` by mohla vypadat takto:

```
protocol kernel {
    learn yes;
    scan time 10;
```

```
    persist yes;
    import all;
    export all;
}
```

Má-li systém více směrovacích tabulek, může se sekce `kernel` vyskytovat v konfiguračním souboru opakovaně. Instrukcí `kernel table` *číslo* u každé z nich určíte, které systémové tabulky se týká.

device `device` je minimalistický „protokol“, který nemá nic společného se směrováním. Díky němu BIRD získává informace o síťových rozhraních v systému a jejich stavu. Můžete předepsat, jak často má informace zjišťovat. Moderní systémy ale informují o změně stavu rozhraní, takže pravidelná kontrola je prováděna jen pro jistotu. Má-li rozhraní více adres, bude vás zajímat instrukce `primary`, jíž určíte jeho primární adresu. Většinou byste měli vystačit s něčím jako

```
protocol device {
    scan time 15;
}
```

static Blížíme se ke směrování, ale pořád ještě jsme se nedopracovali k opravdovým protokolům. Pomocí `static` lze do směrovací tabulky BIRDu vkládat statické položky. Instrukce tohoto protokolu mají tvar

```
route prefix via kudy
```

Jako parametr *kudy* může vystupovat buď adresa sousedního směrovače, nebo v uvozovkách uzavřený název rozhraní. Ve druhém případě statická položka oznamuje, že stroje s daným prefixem jsou přímo dosažitelné tímto rozhraním. Statické přiřazení prefixu 2001:db8:1:2::/64 rozhraní *eth3* a implicitní cestu vedoucí ke směrovači fe80::abcd dostupnému rozhraním *eth0* by zajistily tyto příkazy

```
protocol static {
    route 2001:db8:1:1/64 via "eth3";
    route ::/0 via fe80::abcd%eth0;
}
```

Statické cesty se hodí také k odmítání některých cílových adres. Jako hodnotu parametru *kudy* můžete použít `drop` (datagram potichu zahodit), `reject` (zahodit a poslat odesilateli ICMP zprávu o nedosažitelnosti cíle) nebo `prohibit` (zahodit se zprávou, že cíl byl správcem zakázán). Má-li váš směrovač potichu ignorovat pakety směřující na dokumentační prefix 2001:db8::/32, zatímco odesilatelům paketů zaslaných na adresy lokální pro místo vracet chybové zprávy, zadejte do konfigurace

```
protocol static {
```

```
    route 2001:db8::/32 drop;
    route fec0::/10 prohibit;
```

rip Konfigurace protokolu RIP je velmi jednoduchá. Můžete sice nastavovat různé parametry a přepisovat tak standardní hodnoty, taková potřeba bude ale dost vzácná. O adresy přímo připojených sítí se starat nemusíte, BIRD se je naučí ze systémové směrovací tabulky. Takže většinou vystačíte s výčtem rozhraní, na nichž má protokol běžet.

Kompletní konfigurace BIRDu, který má provozovat RIP na všech Ethernetech s metrikou 1 a řídit jím systémové směrování může vypadat takto:

```
router id 10.1.2.3;

protocol kernel {
    learn yes;
    import all;
    export all;
}

protocol device {
    scan time 15;
}

protocol rip {
    interface "eth*" { metric 1; };
    import all;
    export all;
}
```

ospf V případě protokolu OSPF je konfigurace složitější – musíte definovat oblasti a zařadit do nich rozhraní. Slouží k tomu příkaz

```
area identifikátor {
    parametry
};
```

Mezi *parametry* nemůže chybět `interface` určující rozhraní patřící do dané oblasti a definující vlastnosti tohoto rozhraní. Z nejčastějších lze zmínit cenu (`cost`) či případný seznam sousedů (`neighbors`). Koncovou oblast vyznačíte parametrem `stub`.

Poměrně běžnou potřebou je agregovat směrovací informace z oblasti a za její hranice posílat místo detailů jen kratší prefixy obsahující všechny místní sítě. Toho lze dosáhnout instrukcí

```
networks { prefix1; prefix2; ... }
```

Pro koncovou oblast lze podobného efektu dosáhnout příkazem

```
stubnet prefix { summary; }
```

Nastal čas na příklad, tentokrát lehce složitější – vytvoříme hraniční směrovač propojující dvě OSPF oblasti. Rozhraní *eth0* patří do páteřní oblasti 0.0.0.0, zatímco rozhraní *eth1* a *eth2* do koncové oblasti 9.8.7.6. Všechny její směrovací informace chceme agregovat do společného prefixu 2001:db8:33::/48:

```
protocol ospf {
    area 0.0.0.0 {
        interface "eth0" { cost 1; };
    };
    area 9.8.7.6 {
        stub yes;
        stubnet 2001:db8:33::/48 { summary; };
        interface "eth1" { cost 1; };
        interface "eth2" { cost 1; };
    };
}
```

radv A opět se dostáváme k části, jež nemá se směrováním mnoho společného a je označována jako protokol jen proto, že v BIRDu je protokolem všechno. `radv` řídí vysílání ohlášení směrovače do připojených rozhraní, což je činnost, která se od směrovače očekává. Uvnitř se mohou vyskytovat jen dva příkazy: `interface` definující vlastnosti rozhraní a `prefix` umožňující změnit vlastnosti rozhraní.

Nejzajímavější jsou samozřejmě parametry ovlivňující automatickou konfiguraci síťových parametrů. Uvnitř `interface` se jedná o přepínače `managed` (použít DHCPv6 pro nastavení adresy) a `other conf` (použít DHCPv6 pro získání ostatních konfiguračních parametrů). Oba jsou implicitně vypnuty.

Příkaz `prefix` se může vyskytnout jak uvnitř `radv` a platí globálně, tak uvnitř `interface` s platností omezenou na dané rozhraní. Vztahuje se na všechny ohlašované prefixy vyhovující jeho adrese a použije se vždy první vyhovující. Do konfigurace proto nejprve zapisujte specifické prefixy, obecnější až po nich.

Nejvýznamnějším parametrem prefixu je `autonomous`, který povoluje či zakazuje bezstavovou autokonfiguraci adres. Implicitně je zapnutý. Kromě něj lze ovlivnit i dobu platnosti prefixu (`valid lifetime`) a preferování z něj odvozených adres (`preferred lifetime`). Hodnoty se zadávají v sekundách, implicitně prefix platí jeden den a adresy jsou preferovány 4 hodiny.

Vzhledem k tomu, že výchozí hodnoty jsou navrženy rozumně, velmi často stačí v protokolu `radv` vyjmenovat rozhraní, do nichž mají být ohlášení po-

sílána, aniž by se musely jakkoli měnit parametry. Konfigurace může být takto jednoduchá:

```
protocol radv {
    interface "*";
}
```

Abych to trochu zkomplikoval přepnu rozhraní *eth0* na konfiguraci pomocí DHCPv6 a adresy s prefixem 2001:db8:1:f00::/56 odvrhnu vynulováním doby jejich preference:

```
protocol radv {
    interface "eth0" {
        managed yes;
        prefix ::/0 { autonomous no; };
    }

    interface "*";

    prefix 2001:db8:1::/56 {
        preferred lifetime 0;
    };
}
```

18.1.3 Řízení běžícího BIRDu

Činnost démona vychází z konfiguračního souboru *bird6.conf*. Pokud se jeho obsah za běhu změnil, musíte běžící program vyzvat, aby si načetl aktuální verzi a přizpůsobil jí své chování. K tomu slouží obvyklý signál HUP, takže použijte

```
kill -s HUP číslo_procesu
```

Stejného efektu a mnoha dalších navíc se ale dá dosáhnout příjemnější cestou – interaktivním programem *birdc6* (pro IPv4 slouží *birdc*). Jedná se o specializovaného klienta, který komunikuje se směrovacím démonem a umožňuje z něj získávat informace a ovlivňovat jeho činnost. Na rozdíl od směrovačů Cisco či dále popsaného programu *Quagga* ale interaktivní rozhraní neumožňuje ručně zasahovat do konfigurace. Chcete-li změnit chování BIRDu, musíte upravit konfigurační soubor a načíst jej.

Spusťte *birdc6*, ohlásí se váš démon a můžete mu interaktivně zadávat příkazy. Kdykoli během práce je vám k dispozici nápovědná klávesa ?, která zobrazí vaše možnosti – seznam příkazů či možná pokračování již rozepsaného příkazu.

K zobrazování aktuálního stavu slouží `show` s dost pestrou paletou možností. Základní informace poskytne `show status`, přehled směrovacích protokolů dodá `show protocols`, rozhraní zobrazí `show interfaces` a směrovací tabulku umožní zkoumat `show route` s možnostmi dalšího upřesnění, jaké cesty vás zajímají.

Z akčních příkazů budete asi nejčastěji používat `configure`, který pobídne běžícího démona, aby si načetl aktuální verzi konfiguračního souboru a upravil své chování podle ní. Má tedy stejný účinek jako výše zmíněný signál HUP. Můžete také zakazovat, povolovat či restartovat jednotlivé protokoly. Slouží k tomu trojice

```
disable  protokol
enable   protokol
restart  protokol
```

Na místě *protokolu* může stát i klíčové slovo `all`, má-li mít příkaz plošný účinek. Pozor na příkaz `reload`, jehož název by mohl svádět k domněnce, že znovu načte konfiguraci (což dělá `configure`). `reload` se týká směrovacího protokolu, jehož jméno uvedete v argumentu, a způsobí nový export a import cest mezi ním a směrovací tabulkou. Přidáním `in` nebo `out` na konec omezíte obnovení jen na import, resp. export.

A pokud byste chtěli démona zcela ukončit, použijte příkaz `down`. Skončí jak BIRD, tak *birdc6*, který už si nemá s kým povídat.

18.2 Quagga

Pověstí nejpoužívanějšího otevřeného směrovacího programu se může pochlubit *Quagga*.

www▸ *http://www.quagga.net/*

Licence je opět GNU GPL. Také podporované systémy jsou podobné: Linux, FreeBSD, NetBDS, OpenBSD a Solaris, pro některé je k dispozici v podobě předem připravených balíčků.

Quagga je pokračovatelem projektu *Zebra*. Původně vznikla jako jeho odnož, když skupina vývojářů požadovala častější aktualizace. Poté, co *Zebra* v roce 2005 zamrzla, představuje *Quagga* jejího přirozeného nástupce a přešla na ni valná většina původních uživatelů.

Klíčové vlastnosti, jako je například neobvyklá struktura programu, zdědila samozřejmě po své předchůdkyni. Ostatní směrovací démoni jsou většinou monolitičtí, kdy se jeden velký program snaží obstarat všechno. *Quagga* naproti tomu přichází s modulární strukturou – každý směrovací protokol v ní má svého vlastního démona.

Středobodem všeho dění je démon *zebra*. Ten v sobě shromažďuje směrovací informace, spolupracuje s jádrem systému a upravuje jeho směrovací tabulky. Ostatní démoni, jako jsou *ripd*, *ripngd*, *ospfd* či *bgpd*, slouží jen jako rozhraní centrálního démona pro daný směrovací protokol.

Tato koncepce je nepochybně zajímavá a stylově čistá. Přináší nové možnosti – můžete například rozprostřít démony jedné *Quaggy* po několika počítačích. Má však i své negativní rysy. Tím nejvýznamnějším je roztříštěnost ovládání. Co démon, to konfigurační soubor. Co démon, to samostatné řídicí rozhraní. Zkrátka spravovat *Quaggu* je o poznání komplikovanější, než je v kraji zvykem. Ještě že alespoň existuje *vtysh* (viz níže).

18.2.1 Základy konfigurace

konfigurační soubory

Quaggu můžete řídit prostřednictvím konfiguračních souborů nebo se k ní připojit a zadávat příkazy interaktivně. Každý démon se ovládá nezávisle. Má svůj konfigurační soubor a své interaktivní rozhraní. Názvy konfiguračních souborů jednotlivých démonů dodržují jednotné schéma *jméno_démona.conf*. Obvykle jsou umístěny v adresáři */etc/quagga/*, ale v konkrétní distribuci se umístění samozřejmě může lišit. Lehce matoucí je, že centrální démon nese historické jméno *zebra* a základní konfiguraci tudíž najdete v souboru *zebra.conf*.

interaktivní rozhraní

Chcete-li činnost *Quaggy* sledovat a ovlivňovat v přímém přenosu, můžete použít její interaktivní rozhraní (v dokumentaci označované jako VTY, Virtual Terminal Ynterface). Aby vůbec fungovalo, musíte do konfiguračního souboru *zebra.conf* zařadit příkazy

```
password heslo1
enable password heslo2
```

Těmi se definují hesla pro přístup k démonovi (každý má svůj soubor a svá hesla) a pro vstup do privilegovaného režimu. Pak už se můžete připojit Telnetem na odpovídající port – například

```
telnet localhost 2601
```

démon	*port*
zebra	2601
ripd	2602
ripngd	2603
ospfd	2604
bgpd	2605
ospf6d	2606

Tabulka 18.1: Porty pro řízení démonů

Každý démon má svůj port, jejich hodnoty shrnuje tabulka 18.1. Po připojení musíte zadat heslo podle příkazu `password` z konfigurace. Bude-li správné, objeví se démonovo řídicí rozhraní. Zatím jste však přihlášeni jen jako běžný uživatel, který se sice smí dívat, ale nemůže do ničeho zasahovat. Můžete si prohlížet statistiky rozhraní či směrovací tabulky.

Hodláte-li aktivně zasahovat do práce *Quaggy*, musíte nejprve povýšit. Použijte příkaz

```
enable
```

a pokud se prokážete správným heslem, přejde rozhraní do privilegovaného režimu. Nyní máte k dispozici veškeré příkazy. Můžete měnit konfiguraci, zapínat/vypínat zobrazování ladicích informací a podobně.

Autoři se snažili, aby příkazy a ovládání *Quaggy* co nejvíce připomínaly směrovače firmy Cisco Systems. Pokud znáte tuto platformu, měli byste být jako doma[1]. Při interaktivní práci oceníte především dva pomocníky: tabulátor a otazník.

Tab Tabulátor automaticky doplňuje příkazy. Napište prvních pár znaků, stiskněte **Tab** a program si sám doplní zbytek příkazu. Pokud je začátek jednoznačný. V opačném případě nabídne možná pokračování.

? Otazník je ještě lepší, protože poskytuje nápovědu. Stiskněte jej kdykoli a dostanete informace o tom, jaké jsou vaše aktuální možnosti. Máte-li prázdný příkazový řádek, **?** zobrazí seznam hlavních příkazů. Vyberete si řekněme `show`. Napište příkaz a znovu **?**. Tentokrát je odpovědí seznam parametrů, kterými lze pokračovat po `show`. A tak dále a tak podobně.

Zajímat by vás mohl i příkaz `list` který vypíše seznam všech příkazů (včetně parametrů) použitelných v dané situaci. Přehled těch nejčastějších uvádí tabulka 18.2. Příkazy nemusíte vypisovat celé, stačí jen začátky slov, pokud jsou jednoznačné. Místo `show interface` stačí jen `sh in`, ulevíte svému tabulátoru. Ostatně i v tabulce některé zkracuji.

Interaktivní zásahy do konfigurace vám umožní příkaz `config term`, jímž vstoupíte do konfiguračního režimu. Pokud si nebudete jisti, jaký je vlastně aktuální stav programu, použijte `show run` (nebo `write term`), který vám vypíše celou momentálně platnou konfiguraci daného démona.

Nezapomínejte, že interaktivně provedené konfigurační kroky ovlivňují aktuální činnost programu, ale s jeho prvním restartem zmizí. Má-li být účinek trvalý, musíte je uložit do konfiguračního souboru. Slouží k tomu příkaz

[1] A zjistíte, že v obývacím pokoji chybí dvě stěny, zatímco uprostřed kuchyně roste strom. Narazíte totiž na řadu odlišností. *Quagga* jednak neimplementuje zdaleka vše, navíc se příkazy směrovačů Cisco postupem času vyvíjejí, ale *Quagga* zůstává věrná těm původním.

`show interface`	zobrazí stav rozhraní
`show ip route`	aktuální směrovací tabulka IPv4
`show ipv6 route`	aktuální směrovací tabulka IPv6
`enable`	přechod do privilegovaného režimu
`disable`	návrat do pozorovacího režimu
`config term`	přechod do konfiguračního režimu
`show running`	zobrazení aktuální konfigurace
`write memory`	uložení aktuální konfigurace do souboru
`reload`	restart démona
`exit`	opuštění konf. režimu či celého démona

Tabulka 18.2: Nejběžnější interaktivní příkazy

`write memory`[2] nebo `write file`. Oba dělají totéž – zapíší aktuální konfiguraci běžícího démona do konfiguračního souboru.

vtysh

Nešikovnost ovládání – především interaktivního – bije do očí. Několik konfiguračních souborů by člověk ještě vydýchal, ale nutnost poskakovat mezi několika Telnety, když se snažíte hovořit s *Quaggou* přímo, je prostě přes ruku. Proto vznikl *vtysh*, čili *VTY shell*.

Jedná se o nástroj pro interaktivní řízení celé *Quaggy*, který kontaktuje a řídí všechny její běžící démony. Konfigurační příkazy *vtysh* zahrnují příkazy jádra i jednotlivých démonů, takže můžete vše ovládat pod jednou střechou. Program může spustit jen superuživatel *root* a veškerá hesla a autorizace zde odpadají[3].

Při ukládání konfigurace do souboru nabízí *vtysh* dvě možné alternativy: buď si každý z běžících démonů uloží svůj vlastní soubor (toto je implicitní chování), nebo se celá konfigurace uloží do jednoho společného konfiguračního souboru *Quagga.conf* (s velkým „Q“). Obsahuje kompletní nastavení vašeho softwarového směrovače. Chcete-li toto chování, je třeba do konfiguračního souboru *vtysh.conf* zařadit příkaz

```
service integrated-vtysh-config
```

Jednotliví démoni při startu nejprve hledají společný *Quagga.conf* a pokud jej nenajdou, načtou konfiguraci ze svého specifického souboru. Je velmi záhodno zvolit si jeden režim konfigurace a ten důsledně využívat, jinak snadno oklamete sami sebe. Dokumentace doporučuje režim se samostatnými soubory pro každého démona.

S *Quaggou* můžete hovořit jak přímo, tak prostřednictvím *vtysh*, a to dokonce současně. Konfigurační změny provedené jednou cestou se okamžitě

[2] Je převzat od směrovačů Cisco, kde ukládá konfiguraci do trvalé paměti. V případě *Quaggy* je trvalou pamětí samozřejmě soubor.

[3] Lze změnit v konfiguračním souboru *vtysh.conf*.

projeví i v ostatních. Systém vám však nedovolí paralelní konfiguraci – jakmile přejdete do konfiguračního režimu, jinde už si můžete jen prohlížet.

Ve zbytku kapitoly popíši konfiguraci vybraných démonů. Budu se tvářit, že je konfiguruji samostatně. Ve skutečnosti to však můžete udělat i prostřednictvím *vtysh*. Volba je na vás.

18.2.2 zebra

Centrální démon slouží především k práci s jednotlivými rozhraními a k nastavování statických cest.

rozhraní

Logika práce s rozhraními odpovídá směrovačům Cisco. Chcete-li upravovat vlastnosti rozhraní, musíte se po vstupu do konfiguračního režimu přepnout na vybrané rozhraní příkazem

```
interface jméno
```

Skutečnost, že konfigurujete rozhraní, se projeví přítomností nápisu `(config-if` ve výzvě příkazového řádku. Následně pak můžete nastavovat jednotlivé parametry, jako třeba stručný popis (příkaz `description`) nebo adresu:

```
ipv6 address adresa/délka_prefixu
```

Toto nastavení se okamžitě promítne do operačního systému, jak si můžete ověřit pomocí *ifconfig*. Existující adresu můžete rozhraní odebrat tak, že příkazu pro její nastavení předřadíte slovo `no`. Toto je obecný princip, `no` na začátku umožňuje negovat libovolný konfigurační příkaz.

Pro řízení ohlášení směrovače slouží příkazy skupiny `ipv6 nd`. Implicitně je vypnuto, musíte je pro příslušná rozhraní aktivovat příkazem

```
no ipv6 nd suppress-ra
```

K dispozici je celá řada parametrů ohlášení. Klíčové jsou samozřejmě prefixy které bude obsahovat:

```
ipv6 nd prefix-advertisement prefix/délka
```

Za adresu prefixu lze ještě připojit informaci o životnosti, zakázat automatickou konfiguraci či posílat kompletní adresu směrovače pro hledání domácích agentů.

Quagga sama o sobě nedovede vytvářet nová rozhraní (například tunely). Musíte je založit na úrovni operačního systému a ve směrovacím programu je pak jen využíváte jako kterékoli jiné rozhraní.

Uvedu příklad konfigurace rozhraní pro počítač, který má dva Ethernety do lokální sítě a je připojen tunelem (rozhraní `sit0`) kamsi ven. V místní síti `eth0` se používá bezstavová automatická konfigurace doplněná bezstavovým DHCPv6, v síti `eth1` probíhá automatická konfigurace výlučně pomocí DHCPv6 (bezstavová je zakázána). Do tunelu se ohlášení směrovače neposílá. Mohla by vypadat třeba takto:

```
interface lo
 ipv6 nd suppress-ra

interface eth0
 description Hlavni segment
 ipv6 address 2001:db8:1:1::aaaa/64
 no ipv6 nd suppress-ra
 ipv6 nd other-config-flag
 ipv6 nd prefix-advertisement 2001:db8:1:1::/64

interface eth0
 description Uctarna
 ipv6 address 2001:db8:1:2::bbbb/64
 no ipv6 nd suppress-ra
 ipv6 nd managed-config-flag
 ipv6 nd prefix-advertisement 2001:db8:1:2::/64
                                              no-autoconfig

interface sit0
 description Externi pripojeni
 ipv6 address 2001:a:b:c::2/64
 ipv6 nd suppress-ra
```

statické cesty

Druhou oblastí zájmu démona *zebra* je definice statických cest. Slouží k ní příkaz

```
ipv6 route prefix/délka kudy
```

Jako parametr *kudy* můžete uvést buď adresu směrovače, kterému se mají příslušné datagramy předávat, nebo název rozhraní, kterým je má *Quagga* odesílat ven.

Řekněme, že datagramy směřující do lokální sítě (2001:db8:1::/48) chceme předávat směrovači 2001:db8:1:1::abcd a jako implicitní cestu budeme používat zmíněný tunel `sit0`. Zařídí to dvojice příkazů

```
ipv6 route 2001:db8:1::/48 2001:db8:1:1::abcd
ipv6 route ::/0 sit0
```

Na konec příkazu `ipv6 route` lze připojit údaj o vzdálenosti či klíčová slova `blackhole` (vyhovující datagramy potichu zahodit) nebo `reject` (zahodit a poslat odesilateli ICMP zprávu).

18.2.3 ripngd

Konfigurace RIPng je velmi jednoduchá. Zahájíte ji příkazem

```
router ripng
```

Hlavní částí konfigurace RIPng je definice rozhraní či sítí, na kterých má běžet. Slouží k tomu příkaz

```
network rozhraní
```

Místo *rozhraní* můžete také uvést síť (v obvyklém tvaru *prefix/délka*), avšak rozhraní jsou podstatně častější.

Existuje několik příkazů, kterými se dá ovlivnit obsah ohlašovaných informací. Prvním z nich je `redistribute`, jenž řídí předávání informací získaných z jiných zdrojů do RIPng. Jako parametr může sloužit název jiného směrovacího protokolu – například

```
redistribute bgp
```

předává do RIPng cesty, které se dozvěděl z BGP. Častými hodnotami jsou též `static` (předává statické cesty z démona *zebra*) a `connected` (ohlašuje přímo připojené sítě). Chcete-li zmenšit počet ohlašovaných položek, můžete použít

```
aggregate-address prefix/délka
```

Quagga pak bude ohlašovat uvedený *prefix* jako jedinou položku a nebude předávat dál žádné jeho podprefixy. To se hodí zejména u hraničního směrovače sítě, který zná podrobně její topologii, ale do vnějšího Internetu ohlašuje jen jediný prefix pro celou zdejší síť.

Třetí příkaz z této skupiny přikazuje směrovači, aby do RIPu zahrnul i implicitní cestu:

```
default-information originate
```

Na závěr opět uvedu příklad pro náš fiktivní směrovač se dvěma Ethernety a jedním tunelem. Spustím RIPng na obou rozhraních, nechám agregovat zdejší síť (prefix délky 48 bitů) a šířit informace o implicitní cestě.

```
router ripng
  network eth0
  network eth1
```

```
  network sit0
  redistribute static
  redistribute connected
  aggregate-address 2001:db8:1::/48
  default-information originate
```

18.2.4 ospf6d

Pro OSPFv3 je třeba především nastavit identifikaci směrovače, rozdělit jednotlivá rozhraní do oblastí a případně nařídit agregaci cest. Identifikátor směrovače nastavíte pomocí

`router-id` *identifikátor*

Identifikátor se zapisuje ve stejném formátu jako IPv4 adresa a pokud váš směrovač nějakou má, je nejlepší použít jako identifikátor právě ji. Nemají-li vaše směrovače IPv4 adresy, je třeba stanovit si pravidla pro jejich rozlišení – můžete například použít IP adresy z některé z neveřejných sítí (třeba 10.0.0.0, ať netroškaříme).

Přidělení rozhraní do oblasti zajistí

`interface` *název_rozhraní* `area` *identifikátor_oblasti*

Pro každou oblast pak můžete stanovit prefix, který obaluje všechny její adresy. *Quagga* bude při předávání informací do jiných oblastí agregovat celou oblast do tohoto jediného prefixu. Odpovídající příkaz zní

`area` *identifikátor_oblasti* `range` *prefix/délka*

Na našem směrovači budou oba Ethernety náležet do oblasti 0.0.0.100 přidělené zdejší lokální síti. Celou oblast zahrneme do jednoho agregovaného prefixu. Tunel pak zařadíme do páteřní oblasti:

```
router ospf6d
  router-id 11.22.33.44
  interface eth0 area 0.0.0.100
  interface eth1 area 0.0.0.100
  interface sit0 area 0.0.0.0
  area 0.0.0.100 range 2001:db8:1::/48
  redistribute static rip connected
```

19 Ohlašování směrovače

V této kapitole se podíváme na ohlašování směrovačů, které je klíčovou součástí bezstavové automatické konfigurace. Podíváme se jak na pozitivní stránku věci (tedy jak ohlášení posílat), ale navštívíme i temnou stranu síly, čili obranu před nechtěnými ohlášeními, jež občas zasílají zmatené stroje.

19.1 Ohlašování – radvd

U hardwarových směrovačů je rozesílání *Ohlášení směrovače* a jeho obsah součástí konfigurace daného zařízení. Chcete-li si postavit směrovač z počítače s operačním systémem typu Unix, musíte jej naučit se ohlašovat. Směrovací programy jako *BIRD* či *Quagga* to dovedou zajistit a nabízejí ve své konfiguraci příslušné příkazy.

Když vám tato cesta nevyhovuje[1], musíte svůj stroj naučit ohlašování jinak. K tomuto účelu poslouží specializovaný program *Router ADVertisement Daemon (radvd)*.

Ve stávajících distribucích jej často nenajdete a pokud by byl problém opatřit si jej v podobě balíčku, poslouží web:

www▸ *http://www.litech.org/radvd/*

Konfigurace programu je uložena zpravidla v souboru */etc/radvd.conf*. Obsahuje definice chování *radvd* pro jednotlivá rozhraní. Každé z nich má tvar

```
interface jméno {
    volby_rozhraní
    prefix prefix/délka {
        volby_prefixu
    };
    případné další prefixy
};
```

Nejdůležitější volby rozhraní shrnuje tabulka 19.1. Vedle nich existuje i řada dalších, například pro nastavení všech potřebných časových intervalů. V běžných případech však zpravidla vystačíte s implicitními hodnotami.

Sortiment voleb prefixu je podstatně užší a až na výjimky vystačíte s jejich implicitním nastavením. Mezi nejvýznamnější patří `AdvOnLink`, která v podstatě říká, že všechny stroje s tímto prefixem jsou dosažitelné na této lince. Druhá důležitá volba je `AdvAutonomous`. Povoluje či zakazuje použití tohoto

[1] Například když usoudíte, že si vystačíte se statickým směrováním.

`AdvSendAdvert`	Má se do daného rozhraní posílat ohlášení směrovače? Přípustnými hodnotami jsou `on` a `off`.
`AdvManagedFlag`	Má se vedle bezstavové použít také stavová automatická konfigurace?
`AdvOtherConfigFlag`	Má se stavová konfigurace použít pro zbývající prvky konfigurace (tedy jiné než adresy)?
`AdvDefaultLifetime`	Doba životnosti oznámení o implicitním směrovači. Udává se v sekundách.
`AdvHomeAgentFlag`	Je směrovač ochoten vykonávat roli domácího agenta?
`AdvHomeAgentInfo`	Má do ohlášení směrovače přidávat volbu s informacemi o domácím agentovi?
`HomeAgentPreference`	Udává preferenci domácího agenta.

Tabulka 19.1: Volby rozhraní v konfiguraci *radvd*

prefixu pro bezstavovou konfiguraci adres. Implicitně jsou obě nastaveny na `on`.

Pro nasazení mobilního IPv6 je důležitá volba `AdvRouterAddr`. Nastavíte-li ji na `on` (implicitně je vypnuta), bude *radvd* posílat v ohlášení celou adresu, ne jen prefix. Zasílání celé adresy požaduje mobilní IPv6.

Velmi jednoduchý *radvd.conf* může vypadat následovně:

```
interface eth0
{
   AdvSendAdvert on;
   prefix 2001:db8:1:35::/64 {};
};
```

Složitější konfigurace se dvěma prefixy, podporou mobility a dvěma rozhraními, z nichž `eth0` používá bezstavovou konfiguraci doplněnou bezstavovým DHCPv6, zatímco `eth1` povoluje jen stavovou konfiguraci protokolem DHCPv6, by mohla obsahovat

```
interface eth0
{
    AdvSendAdvert on;
    AdvOtherConfigFlag on;
    AdvHomeAgentFlag on;
    AdvHomeAgentInfo on;
    HomeAgentPreference 1;

    prefix 2001:db8:1:35::/64
```

```
    {
        AdvRouterAddr on;
    }

    prefix 2002:1001:203:35::/64
    {
        AdvRouterAddr on;
    };
};

interface eth1
{
    AdvSendAdvert on;
    AdvManagedFlag on;
    AdvHomeAgentFlag on;
    AdvHomeAgentInfo on;
    HomeAgentPreference 1;

    prefix 2001:db8:1:99::/64
    {
        AdvRouterAddr on;
        AdvAutonomous off;
    }

    prefix 2002:1001:203:99::/64
    {
        AdvRouterAddr on;
        AdvAutonomous off;
    };
};
```

Máte-li připraven konfigurační soubor, stačí spustit *radvd* a ohlašování směrovače začne fungovat. Až na výjimky nebudete potřebovat žádnou z jeho voleb. Snad jedině `-C` *soubor*, kterou lze předepsat jméno konfiguračního souboru.

19.2 Likvidace „pirátských“ ohlášení – ramond

Stanice, která se chopí iniciativy a začne do sítě rozesílat svá vlastní ohlášení směrovače, může v konfiguraci ostatních napáchat pěknou paseku. Je záhodno takové aktivity přinejmenším sledovat a pokud možno potírat, aby jejich devastující účinky měly co nejmenší dopad.

Tuto službu dovedou zajistit hardwarové směrovače (viz kapitola 17 na straně 343), ale existuje pro ni i softwarové řešení. Nejpoužívanější je program *Router Advert MONitoring Daemon (ramond)*, který sleduje ohlášení

směrovačů a podle svého konfiguračního souboru podniká příslušné akce. Najdete jej na adrese

www▸ *http://ramond.sourceforge.net/*

Pokud pro váš systém neexistuje balíček a budete instalovat ze zdrojových kódů, připravte se na poněkud strohý přístup. Chybí obvyklý konfigurační skript, případnou adaptaci pro vlastní systém musíte provést editací souboru *Makefile*. O překlad se postará obvyklý příkaz

make all

Instalovat výsledný *ramond* pak musíte ručně. Manuálové stránky nejsou součástí distribuce, nicméně dají se dohledat na manpages.ubuntu.com.

konfigurace Základem konfigurace je soubor *ramond.conf*. Implicitně jej program hledá v */etc*, ale parametrem `-c` *soubor* si lze poručit libovolnou cestu k němu. Soubor je ve formátu XML a v distribuci najdete jeho ukázkovou verzi i s vysvětlujícími komentáři.

Celá konfigurace je zabalena do prvku `<ramond>`, jehož obsahem mohou být jen prvky `<mac-list>` definující seznamy MAC adres a `<rule>` s jednotlivými pravidly chování programu.

`<mac-list>` obsahuje seznam MAC adres používaných později k povolení či potlačení ohlášení z nich zasílaných. Má přiděleno jednoznačné jméno v atributu `name`, jednotlivé adresy jsou zabaleny do prvků `<entry>`. Jeho tvar je tedy následující:

```
<mac-list name="jméno">
  <entry>adresa1</entry>
  <entry>adresa2</entry>
  ...
</mac-list>
```

Každé pravidlo má svůj vlastní prvek `<rule>`. Při příchodu jakéhokoli ohlášení směrovače jsou pravidla procházena v pořadí podle konfigurace a první, jemuž ohlášení vyhoví, bude použito. V konfiguraci je proto třeba nejprve uvést výjimky a teprve po nich obecná pravidla.

Podmínky kladené na ohlášení jsou vyjádřeny pomocí atributů prvku `<rule>`. Můžete použít libovolnou kombinaci následujících atributů (každý se ale smí vyskytnout nanejvýš jednou a nese jen jedinou hodnotu):

- `mac` určuje MAC adresu ohlašujícího směrovače. Hodnotou atributu není vlastní adresa, ale jméno prvku `<mac-list>`, který adresy definuje.
- `prefix` omezuje ohlašovaný prefix. Hodnota atributu je interpretována jako prefix ohlášeného prefixu. Pokud prefix v ohlášení začíná

uvedeným prefixem, pravidlo se použije. Hodnota `prefix="::/0"` se tedy vztahuje na libovolný ohlášený prefix.

- `interface` definuje rozhraní, z nějž ohlášení přišlo.
- `lifetime` se týká životnosti. Jedinou podporovanou hodnotou je 0, která způsobí aktivaci pravidla při ohlášení ukončujícím životnost prefixu.

Tělo pravidla definuje akci, která se má provést. Na výběr jsou tři základní možnosti:

- Prázdné tělo znamená, že ohlášení je v pořádku a *ramond* nemá nic podnikat.
- `<clear/>` požaduje likvidaci daného ohlášení. *ramond* proto vzápětí pošle stejné ohlášení, ovšem s nulovou životností, aby si všichni nežádoucí údaje vymazali.
- `<execute>`*cesta*`</execute>` požaduje spuštění skriptu s danou (absolutní) cestou. Pro spouštěný program nastaví několik proměnných prostředí, v nichž mu předá základní informace o ohlášení, jež jeho spuštění vyvolalo. Jejich přehled najdete v tabulce 19.2. Součástí distribuce je jednoduchý skript *demo.pl*, který údaje vypíše (a skončí tedy v logu).

`$PREFIX`	ohlášený prefix
`$PREFIX_LEN`	délka ohlášeného prefixu
`$SOURCE_ADDR`	zdrojová IPv6 adresa paketu
`$SOURCE_MAC`	zdrojová MAC adresa paketu
`$INTERFACE`	rozhraní, z nějž ohlášení dorazilo

Tabulka 19.2: Proměnné prostředí pro skripty spouštěné *ramond*

Každé pravidlo může obsahovat libovolný počet prvků `<execute>`, za nimiž případně může následovat `<clear/>`. Pokud je přítomen, musí být vždy až na konci pravidla.

Jednoduchá konfigurace pro koncovou síť s prefixem 2001:db8:1::/48 a dvěma směrovači s MAC adresami 0a:c3:21:5e:32:a7 a 00:4a:d7:f4:85:d1 oprávněnými ohlašovat prefixy z tohoto prostoru by mohla vypadat takto:

```
<ramond>
  <!-- seznam oprávněných směrovačů -->
  <mac-list name="smerovace">
    <entry>0a:c3:21:5e:32:a7</entry>
    <entry>00:4a:d7:f4:85:d1</entry>
  </mac-list>
```

```
  <!-- logovat zrušení od oprávněných směrovačů -->
  <rule mac="smerovace" lifetime="0">
    <execute>/etc/ramond/log cancelled</execute>
  </rule>

  <!-- povolit korektní ohlášení -->
  <rule mac="smerovace" prefix="2001:db8:1::/48">
  </rule>

  <!-- logovat a zrušit vše ostatní -->
  <rule>
    <execute>/etc/ramond/log bogus-advert</execute>
    <clear/>
  </rule>
</ramond>
```

Druhé a třetí pravidlo je myslím jasné. První se stará o případy, kdy některý z oprávněných směrovačů cosi zruší – tedy pošle ohlášení s nulovou životností. Taková událost je zajímavá, proto stojí za zaznamenání do logu.

20 BIND

Berkeley Internet Name Domain (BIND) je široko daleko nejrozšířenější implementací DNS serveru. Program vyvíjí *Internet Software Consorctium (ISC)* a dává jej k dispozici zdarma v podobě zdrojového kódu. Producenti jednotlivých operačních systémů či distribucí připravují binární verze pro své miláčky. BIND již dlouhou dobu najdete snad v každém Unixu podobném operačním systému a k mání je i portace pro MS Windows.

Jedinou v současnosti podporovanou verzí je *BIND 9* (a pracuje se na verzi 10). Občas se jako s žijící zkamenělinou můžete dodnes setkat s některým z jeho předchůdců, jejich použitelnost pro IPv6 je však velmi omezená. *BIND 4* je nepodporuje vůbec, *BIND 8* sice umí záznamy pro IPv6, ale sám hovoří pouze IPv4. Vzhledem k tomu, že ISC prohlašuje verze 4 a 8 za zastaralé a dále je nepodporuje, postupně z Internetu mizí.

Na podzim roku 2011 nese aktuální verze číslo 9.8 a najdete v ní vše, co je potřeba. Zvládá záznamy typu AAAA i PTR, podporuje dokonce i opuštěné A6 a DNAME, hovoří plynně oběma protokoly, IPv6 adresy se mohou vyskytovat v celé řadě konfiguračních příkazů. Zkrátka mazlíček. Je sice pomalejší než verze 4 a 8, ovšem stále zvládá několik tisíc dotazů za sekundu, což většině z nás jistě vystačí.

překlad a start

BIND má četné automatické mechanismy, které zatím vždy fungovaly k mé spokojenosti. Když jej překládáte, sám si zjistí, zda váš systém obsahuje IPv6 knihovny. Pokud ano, přeloží se s podporou IPv6. Distribuční balíčky tuto podporu mívají také.

Dříve se obdobně choval i při startu. Ověřil si podporu IPv6 v systému a když ji nalezl, připojil se k IPv6 portu a byl připraven naslouchat. Bohužel se ukázalo, že v některých systémech se IPv4 a IPv6 porty prapodivně ovlivňují navzájem. Proto autoři programu vsadili na bezpečnější kartu a program se nyní sám od sebe nepřipojí k IPv6. Musíte mu to poručit v konfiguračním souboru *named.conf* v sekci `options` volbou

```
listen-on-v6 { any; };
```

Místo `any` můžete samozřejmě vyjmenovat konkrétní rozhraní, na nichž chcete provozovat DNS po IPv6. Obvykle však vystačíte s globálním povolením.

zóny

Nejvýznamnější složkou konfigurace *BINDu* jsou zónové soubory jednotlivých domén. Ty jsem dost podrobně (včetně příkladů) popsal v kapitole 9 na straně 191. Základní konfigurační soubor *named.conf* by pro každou doménu měl obsahovat

```
zone "doména"{
    type master;
    file "jméno_souboru";
};
```

Kdyby například konfigurace domény *kdesi.cz*, kterou jsem propíral v kapitole 9, byla uložena v souboru *kdesi.domena*, obsahovala by odpovídající položka *named.conf*

```
zone "kdesi.cz" {
    type master;
    file "kdesi.domena";
};
```

ochrana Zabezpečení BINDu sice přímo nesouvisí s poskytováním DNS pro nebo po IPv6, ale je tak důležité, že mu věnuji pár řádků. V poslední době se totiž objevily nové ošklivé útoky, jimiž vám mohou podstrčit do vyrovnávací paměti DNS serveru falešné údaje. Klíčem jsou rekurzivně řešené dotazy, kdy server převezme dotaz od klienta, najde na něj odpověď, pošle ji klientovi a zároveň si ji uloží do vyrovnávací paměti.

Základem útoků je položit serveru dotaz a vzápětí mu podstrčit falešnou odpověď. Je proto velmi žádoucí dovolit klást dotazy jen místním strojům. Slouží k tomu volba `allow-query`, jejíž hodnotou je přístupový seznam obsahující prefixy vaší sítě. Pokud má například IPv4 prefix 10.1.2.0/24 a IPv6 prefix 2001:db8:1::/48, vypadaly by konfigurační příkazy následovně:

```
acl mistni { 10.1.2.0/24; 2001:db8:1::/48; }
options {
    allow-query { mistni; };
};
```

Pro server vyhrazený jen k řešení dotazů od místních klientů tím může úprava konfigurace skončit. Jestliže server ale zároveň slouží jako autoritativní pro některé z vašich domén, přestal by rázem komunikovat se zbytkem světa. Globální zákaz dotazů zvenčí je u něj třeba doplnit cíleným otevřením domén spravovaných serverem. Ke každé doméně, pro niž má typ serveru hodnotu `master` nebo `slave` je třeba doplnit

```
allow-query { any; };
```

Konkrétně konfigurace domény *kdesi.cz* by vypadala takto:

```
zone "kdesi.cz" {
    type master;
    allow-query { any; };
    file "kdesi.domena";
};
```

Nyní server povolí externím strojům dotazy na domény, u nichž zná autoritativní odpovědi, ale žádné jiné.

server bez IPv4

Vzpomínáte si na doporučení, že místní rekurzivní server řešící klientské dotazy by měl mít k dispozici oba protokoly? I když je umístěn v čisté IPv6 síti, která není připojena k IPv4 Internetu, měli byste mu umožnit řešit dotazy prostřednictvím IPv4. Dá se toho dosáhnout konfigurovaným tunelem mezi DNS serverem a vhodným IPv4/IPv6 směrovačem. Tentokrát se bude tunelovat obráceně – IPv4 datagramy budete balit do IPv6.

Podstatně jednodušším a vhodnějším řešením je však nastavit BINDu zprostředkovatele (forwarder). Musí to být server podporující oba protokoly, jehož konfigurace připustí rekurzivní dotazy od koncového serveru. Spolu budou hovořit IPv6, zprostředkovatel bude v Internetu pátrat oběma protokoly podle potřeby.

U koncového serveru potřebné chování zajistí příkaz `forwarders` v úvodní části `options`. Jeho hodnotou je seznam adres zprostředkovatelů, například

```
options {
    forwarders {
        2001:db8:1:aa::bb;
        2001:db8:1:cc::dd;
    };
};
```

Má-li BIND nastaveny zprostředkovatele, zkusí dotaz nejprve řešit jejich prostřednictvím. Když se nedočká odpovědi, pokusí se najít odpověď sám. Toto chování lze změnit příkazem

```
    forward only;
```

v `options`. Zakážete jím samostatnou aktivitu BINDu a ponecháte zprostředkovatele jako jedinou možnost, jak hledat odpovědi na dotazy. Na straně zprostředkovatele je samozřejmě třeba vhodným nastavením `allow-query` povolit dotazy ze serverů, jimž zprostředkovává přístup ke globálnímu DNS.

21 Server pro DHCPv6

Pomalému vývoji specifikace DHCPv6, o němž jsem psal v části 6.5 na straně 128, odpovídá i velmi vláčné tempo jeho implementací. Dlouhá léta byl jediným univerzálně použitelným a rozumně fungujícím klientem i serverem polský *Dibbler*. Teprve v prosinci 2007 vydalo Internet Systems Consortium (pachatelé BINDu) čtvrtou verzi svého *DHCP* s podporou nového protokolu a v následné verzi 4.1 doplnilo chybějící prvky, například zprostředkovatele (relay). V současnosti již lze ISC DHCP považovat za plnohodnotnou a úplnou implementaci DHCPv6.

Tyto dva programy nejsou jediné, ale považuji je za nejvýznamnější, proto se jim zde budu věnovat. Kromě nich dokáží poskytovat DHCPv6 služby i směrovače Cisco Systems, na nichž lze zprovoznit jak server, tak zprostředkovatele. Pokud ale konfiguraci DHCP serveru vytváříte strojově, bude pravděpodobně snažší provozovat jej na počítači, kde je k dispozici řada doprovodných užitečných programů, než na hardwarovém směrovači.

21.1 Dibbler

Program *Dibbler* je s námi od roku 2004. Inspirací pro jeho jméno se stal nejslavnější prodavač párků na Zeměploše, Kolík Ať sepicnu (v originále Cut Me Own Throat Dibbler). Obsahuje všechny tři základní účastníky protokolu – klienta, server i zprostředkovatele – pro operační systémy MS Windows a Linux. Můžete jej získat na adrese

www▸ *http://klub.com.pl/dhcpv6/*

ale možná to nebude nutné, protože některé distribuce jej již obsahují. Nejznámějším příkladem bude rozhodně *Ubuntu*.

Konfigurační soubory *Dibbleru* se obvykle nacházejí v adresáři */etc/dibbler* (v Linuxu) nebo v aktuálním adresáři (ve Windows). Pro práci serveru je rozhodující *server.conf*. Obvykle obsahuje globální volby platné pro celý server, následované popisem jednotlivých rozhraní. Ten má tvar

```
iface "jméno" {
    vlastnosti rozhraní
}
```

Řadu voleb lze použít jak globálně, tak v popisu rozhraní i v ještě menších jednotkách (v popisu třídy či jednotlivého klienta). Platí obvyklá pravidla – volba je účinná jen v tom rozsahu, v němž byla definována, a lokální definice má přednost před globální.

Nejčastěji používané budou nepochybně volby pro nastavení DNS:

```
option dns-server adresy serverů
option domain doména
```

První sděluje klientům IPv6 adresy místních DNS serverů, na něž se mají obracet se svými dotazy. Je-li adres více, oddělují se čárkami. Druhá volba oznamuje doménu, kterou mají klienti automaticky připojovat k hledaným jménům.

Pro přidělování adres z určitého balíku nejlépe poslouží zavedení třídy:

```
class {
    volby
    pool adresy
}
```

Volby se tentokrát vztahují jen na danou třídu a mohou definovat klienty, kteří do ní jsou zařazeni. K tomu slouží dvojice příkazů `accept-clients` a `reject-clients`. Rozsah poskytovaných adres lze stanovit dvěma způsoby. Buď uvedete první a poslední adresu oddělené pomlčkou, nebo zapíšete první adresu následovanou lomítkem a délkou prefixu. Volné přidělování adres od 2001:db8:1:cc:: do 2001:db8:1:cc::ffff zajistí

```
class {
    pool 2001:db8:1:cc::-2001:db8:1:cc::fff
}
```

nebo ekvivalentní

```
class {
    pool 2001:db8:1:cc::/112
}
```

Jestliže chcete některým klientům poskytovat specifické nastavení (například stálou adresu), použijte příkaz `client`:

```
client identifikace {
    vlastnosti klienta
}
```

Takovéto definice představují výjimky z obecně platných nastavení. Ve *vlastnostech* můžete klientovi předepsat specifické volby nebo příkazem `address` adresu, která bude jen a jen jeho. Nejčastějším způsobem identifikace klienta bude jeho DUID ve tvaru `duid` *hodnota*.

Podívejme se na příklad kompletního souboru *server.conf* pro server s jedním rozhraním `eth0`, který přiděluje klientům adresy z výše uvedeného rozmezí. Oznamuje jim adresy dvou lokálních DNS serverů a implicitní doménu *kdesi.cz*. Dva klienti mají přiděleny pevné adresy:

```
iface "eth0" {
    option dns-server 2001:db8:1:1::aa, 2001:db8:1:1::bb
    option domain kdesi.cz

    class {
        pool 2001:db8:1:cc::/112
    }

    client duid 0x000100010c00a03700065b3c275a {
        address 2001:db8:1:1::33:44
    }

    client duid 0x000100010c00a3c200065ba03c19 {
        address 2001:db8:1:1::55:66
    }

}
```

Volby pro DNS by mohly být zařazeny také na globální úroveň, odkud by se zdědily do všech rozhraní, u nichž není uvedeno jinak. Jelikož má server jen jediné rozhraní, je to jedno.

Pokud by měl server pracovat jako bezstavový a poskytovat klientům jen informace o DNS, konfigurace by se výrazně zjednodušila. Přibude do ní `stateless` a zmizí vše, co souvisí s adresami:

```
stateless

iface "eth0" {
    option dns-server 2001:db8:1:1::aa, 2001:db8:1:1::bb
    option domain kdesi.cz
}
```

Nepříjemným omezením *Dibbleru* je, že se stará pouze o DHCPv6. Pokud chcete provozovat i verzi 4, musíte jej doplnit dalším programem.

21.2 ISC DHCP

Název této části poněkud připomíná písničku Zkratky od Ivana Mládka, ale ten program se skutečně takto krkolomně jmenuje. Jedná se o veterána, který je ve službě už od poloviny devadesátých let. Naučit starého psa novým kouskům asi dalo dost práce, takže s podporou DHCPv6 přišel teprve ve verzi 4.0 koncem roku 2007. Bydlí na adrese

www▸ *http://www.isc.org/sw/dhcp/*

Existuje pouze pro systémy typu Unix (Linux, BSD, Solaris, Aix, ...), z nichž většina jej rovnou obsahuje. Pro ty ostatní si můžete stáhnout a přeložit zdrojové kódy.

ISC DHCP podporuje jak DHCPv4, tak DHCPv6, ale nikoli oba najednou. Pokud chcete provozovat na jednom stroji DHCP server pro oba protokoly, musíte spustit dva exempláře *dhcpd* a odlišit je parametrem `-4` a `-6` na příkazovém řádku. Neuvedete-li ani jeden z nich, bude se implicitně chovat jako DHCPv6 server, což je pozoruhodné.

Obvyklým umístěním konfiguračního souboru je */etc/dhcpd.conf*. Pokud budete provozovat dva servery na jednom stroji, můžete každému z nich předepsat na příkazovém řádku jinou konfiguraci volbou `-cf` *soubor*. Nabízí se použít pro ně jména *dhcpd4.conf* a *dhcpd6.conf*. Podívejme se, jak vypadá konfigurace pro DHCPv6.

Konfigurační soubor se skládá z příkazů dvou typů: parametrů a deklarací. Parametry ovlivňují chování programu – nastavují časové konstanty, zapínají či vypínají různé prvky. Mohou se vyskytovat v různých kontextech, které pak omezují jejich působnost. Lze definovat parametry globální, které změníte pro jednu konkrétní podsíť a ještě jiné hodnoty použijete pro konkrétního klienta. Deklarace popisují klienty a jejich uspořádání.

Na rozdíl od *Dibbleru*, který svou konfiguraci člení podle rozhraní, pro *ISC DHCP* je základní organizační jednotkou podsíť. Pro každou podsíť, kterou má obsluhovat (přímo či přes zprostředkovatele), musí konfigurační soubor obsahovat jednu deklaraci

```
subnet6 prefix/délka {
    parametry
    deklarace
}
```

Ještě před podsítěmi můžete uvést globální parametry. Pravděpodobně mezi ně bude patřit definice lokálních DNS serverů a implicitní domény, kterou mohou klienti automaticky připojovat ke svým dotazům. Postarají se o to

```
option dhcp6.name-servers adresy;
option dhcp6.domain-search doména;
```

Hodnoty, má-li jich parametr víc, oddělujte čárkami. Celou definici pak vždy zakončete středníkem. Mohou být samozřejmě umístěny i uvnitř podsítě, pokud jí chcete definovat odlišné parametry DNS.

Konfiguračně nejsnazší variantou je dynamické přidělování adres z určitého rozsahu. Zajistí je příkaz `range6`, jemuž dostupný rozsah adres sdělíte buď uvedením první a poslední adresy oddělených mezerou, nebo zadáním první adresy, lomítka a délky prefixu. Dejte pozor, aby byly v souladu

s prefixem své podsítě. Chcete-li v podsíti 2001:db8:1:cc::/64 dynamicky přidělovat adresy od 2001:db8:1:cc:: do 2001:db8:1:cc::ffff, použijte deklaraci

```
subnet6 2001:db8:1:cc::/64 {
    range6 2001:db8:1:cc:: 2001:db8:1:cc::fff;
}
```

nebo ekvivalentní

```
subnet6 2001:db8:1:cc::/64 {
    range6 2001:db8:1:cc::/112;
}
```

Deklaracemi `pool` můžete v jedné podsíti vytvořit několik rozsahů s různou politikou, například různě dlouhými dobami zápůjčky nebo s různými omezeními na firewallech. Příkazy `allow` a `deny` pak řídíte, kteří klienti mohou získat adresu z daného rozsahu.

Jednou z používaných variant je `deny unknown-clients`, jímž zakážete přidělování adres strojům, které váš DHCP server nezná. Tedy neobsahuje pro ně deklaraci `host`. Jejím prostřednictvím můžete také klientům nastavovat specifické parametry. Má tvar

```
host jméno {
    parametry
    deklarace
}
```

Jméno musí být v rámci konfiguračního souboru jednoznačné, obvyklou hodnotou je doménové jméno příslušného počítače. Klíčovou otázkou je identifikace příslušného klienta, jež probíhá prostřednictvím DUID. Stanoví ji příkaz `host-identifier` ve tvaru

```
host-identifier option dhcp6.client-id DUID;
```

Například přidělování výše uvedených adres pouze známým klientům by zajistila skupina příkazů (pro ilustraci uvádím záznam pro jeden známý počítač):

```
subnet6 2001:db8:1:cc::/64 {
    deny unknown-clients;
    range6 2001:db8:1:cc::/112;
}

host pc1.kdesi.cz {
    host-identifier option dhcp6.client-id
        00:01:00:01:0c:00:a0:37:00:06:5b:3c:27:5a;
}
```

Deklarace `host` je také cestou k definování pevné adresy pro daného klienta. V takovém případě je vhodnější zařadit ji do podsítě, protože má smysl jen v jejím kontextu. Adresu definujete pomocí

```
fixed-address6 adresa;
```

Následující ukázkový konfigurační soubor zajistí přidělování adres v podsíti cc komukoli, zatímco v podsíti 1 přiděluje jen dvě pevně definované adresy. Všichni klienti obdrží shodné informace o DNS serverech a zdejší doméně:

```
option dhcp6.name-servers
                    2001:db8:1:1::aa, 2001:db8:1:1::bb;
option dhcp6.domain-search "kdesi.cz";

subnet6 2001:db8:1:cc::/64 {
    range6 2001:db8:1:cc::/112;
}

subnet6 2001:db8:1:1::/64 {

    host pc1.kdesi.cz {
        host-identifier option dhcp6.client-id
            00:01:00:01:0c:00:a0:37:00:06:5b:3c:27:5a;
        fixed-address6 2001:db8:1:1::33:44;
    }

    host pc2.kdesi.cz {
        host-identifier option dhcp6.client-id
            00:01:00:01:0c:00:a3:c2:00:06:5b:a0:3c:19;
        fixed-address6 2001:db8:1:1::55:66;
    }
}
```

ISC DHCP bohužel své uživatele nerozmazluje přívětivou dokumentací (*Dibbler* je v tomto směru o třídu lepší). Podrobnosti je třeba vyčíst z manuálových stránek *dhcpd*, *dhcpd.conf* a *dhcp-options*.

21.3 Určení DUID

Zjištění DUID může představovat tvrdý oříšek. Kde jsou ty časy, kdy se počítače identifikovaly podle své snadno zjistitelné MAC adresy. DUID z ní sice může být odvozen, ale nejasností pro něj zbývá dost a dost. Existují tři základní možnosti, jak jej určit:

- Načíst dokumentaci klienta a vyhledat jej v jeho datových strukturách. Klienti si zpravidla DUID někam zapisují, stačí jen zjistit kam.

Například ve Windows 7 se jedná o registr *HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip6\Parameters\Shcpv6-DUID*[1], v Linuxu o soubor */var/lib/dhcpv6/dhcp6c_duid* a podobně. Toto je čistá cesta.

- Odchytit klientovu žádost programem typu *tcpdump* či *Wireshark* a DUID v ní najít. To nemusí být úplně jednoduché, navíc některé programy vypisují obsah paketů v jiném formátu, než vyžaduje konfigurační soubor DHCP. Tuto cestu jistě zvolí každý správný haxxxor.
- Pokud máte povoleno přidělování adres komukoli, můžete si klienta a jeho DUID vyhledat v záznamu o pronajatých adresách, který si server vede v souboru *dhcpd.leases*. Osobně mi tato varianta připadá jako nejpragmatičtější.

[1] Windows 7 a Vista zobrazují DUID také ve výpisu příkazu `ipconfig /all`.

III
Přílohy

A Rezervované adresy a identifikátory

Zde uvádím přehled nejvýznamnějších adres rezervovaných IANA pro specifické účely.

A.1 Skupinové adresy

Aktuální hodnoty najdete na adrese

www▸ *http://www.iana.org/assignments/ipv6-multicast-addresses*

Lokální na rozhraní

ff01::1	všechny uzly
ff01::2	všechny směrovače

Lokální na lince

ff02::1	všechny uzly
ff02::2	všechny směrovače
ff02::4	DVMRP směrovače
ff02::5	všechny OSPF směrovače
ff02::6	všechny pověřené OSPF směrovače
ff02::9	RIP směrovače
ff02::a	EIGRP směrovače
ff02::b	mobilní agenti
ff02::c	SSDP (Simple Service Discovery Protocol)
ff02::d	všechny PIM směrovače
ff02::e	RSVP zapouzdření
ff02::16	všechny MLDv2 směrovače
ff02::6a	objevování skupinových směrovačů
ff02::1:1	jméno linky
ff02::1:2	všichni DHCP agenti
ff02::1:3	lokální jmenná služba
ff02::1:ff*xx*:*xxxx*	vyzývaný uzel
ff02::2:ff*xx*:*xxxx*	informace o uzlu

Lokální v místě (site)

ff05::2	všechny směrovače
ff05::1:3	všechny DHCP servery
ff0*x*::1:1000 až 13ff	hledání síťových služeb (SLP verze 2)

Dosah podle potřeby

ff0*x*::	rezervovaná
ff0*x*::101	NTP (Network Time Protocol)

A.2 Skupinové identifikátory

Aktuální hodnoty najdete na adrese

www▸ *http://www.iana.org/assignments/perm-mcast-groupids*

4000:0000	proxy sítě

A.3 Výběrové adresy

Přehled aktuálních hodnot je k dispozici na adrese

www▸ *http://www.iana.org/assignments/ipv6-anycast-addresses/*

prefix:0:0:0:0	směrovače v podsíti
prefix:fdff:ffff:ffff:ff80–fffd	rezervováno
prefix::fdff:ffff:ffff:fffe	domácí agenti
prefix::fdff:ffff:ffff:ffff	rezervováno

B Specifikace IPv6

V textu jsou porůznu roztroušeny odkazy na dokumenty a definice jednotlivých komponent rozlehlého světa IPv6. Ovšem vyhledávat v textu, které RFC definuje třeba objevování sousedů, nemusí být zrovna jednoduché. Pro usnadnění orientace zde uvádím jejich tématicky uspořádaný přehled.

B.1 Jádro protokolu

základní definice IPv6
RFC 2460

volba *Upozornění směrovače*
RFC 2711

jumbogramy
RFC 2675

ICMPv6
RFC 4443, RFC 4884

B.2 Přenos po linkových technologiích

Ethernet
RFC 2464, RFC 6085

FDDI
RFC 2467

Token Ring
RFC 2470

NBMA sítě (více účastníků, ale bez všesměrového adresování, např. ATM)
RFC 2491

ATM
RFC 2492

Frame Relay
RFC 2590

Firewire (IEEE 1394)
RFC 3146

Fibre Channel
RFC 4338

osobní bezdrátové sítě IEEE 802.15.4
RFC 4944, *RFC 4919*, *RFC 6282*

PPP
RFC 5072

B.3 Adresy

architektura adres
RFC 4291

globální individuální agregovatelné adresy
RFC 3587

skupinové adresy pro IPv6
RFC 2375

skupinové adresy odvozené z rozhraní
RFC 4489

adresy chránící uživatelovo soukromí
RFC 4941

kryptograficky generované adresy
RFC 3972

výběrové adresy
RFC 2526, *RFC 4786*

adresy v IPv4/IPv6 překladačích
RFC 6052

doporučení pro přidělování adres
RFC 6177

multihoming
RFC 3178, *RFC 3582*, *RFC 4218*, *RFC 4219*

B.4 Směrování

RIPng
RFC 2080

OSPF pro IPv6
RFC 5340

IS-IS
RFC 1142, *RFC 1195*, *RFC 5302*, *RFC 5304*

BGP4+
RFC 4271, *RFC 4760*, *RFC 2545*

6PE
RFC 4798

B.5 Skupinově adresovaná data

Multicast Listener Discovery (MLD)
RFC 2710, *RFC 3590*

Multicast Listener Discovery verze 2 (MLDv2)
RFC 3810

PIM-SM
RFC 4601, *RFC 5059*, *RFC 5796*, *RFC 6226*

PIM-DM
RFC 3973

BIDIR-PIM
RFC 5015

PIM-SSM
RFC 4607

B.6 DNS

IPv6 obsah v DNS
RFC 3596

doporučení pro přenos DNS po IPv6
RFC 3901

provozní otázky a problémy IPv6 DNS
RFC 4472

B.7 Automatická konfigurace

objevování sousedů
RFC 4861

inverzní objevování sousedů
RFC 3122

bezstavová automatická konfigurace adres
RFC 4862, *RFC 4941*

zabezpečení bezstavové konfigurace adres
RFC 3971, *RFC 3972*

automatická konfigurace DNS
RFC 6106

DHCPv6
RFC 3315, *RFC 6221*

bezstavové DHCPv6
RFC 3736

přeadresování sítě
RFC 2894, *RFC 4076*, *RFC 4192*

Service Location Protocol verze 2 (SLPv2)
RFC 2608, *RFC 3111*, *RFC 3224*

B.8 IPsec

základní architektura bezpečnostních mechanismů
RFC 4301

Hlavička AH
RFC 4302

Hlavička ESP
RFC 4303

kryptografické algoritmy
RFC 4835

IKEv2
RFC 5996, *RFC 4307*, *RFC 5998*

B.9 Mobilita

podpora mobility v IPv6
RFC 6275, *RFC 3776*

mobilní sítě (NEMO)
RFC 3963

hierarchická mobilita
RFC 5380

proxy mobilita
RFC 5213

mobilní IPv6 a IKEv2
RFC 4877

B.10 Přechodové mechanismy

obecný mechanismus tunelování IPv6
RFC 2473

dvojí zásobník a konfigurované tunely
RFC 4213

tunnel broker
RFC 3053

Tunnel Setup Protocol
RFC 5572

6to4
RFC 3056, *RFC 3068*, *RFC 5158*, *RFC 6343*

6rd
RFC 5569

6over4
RFC 2529

ISATAP
RFC 5214

Teredo
RFC 4380, *RFC 5991*

DS-Lite
RFC 6333, *RFC 6334*

SIIT
RFC 6145

NAT64 a DNS64
RFC 6146, *RFC 6147*

NAT–PT
RFC 2766, *RFC 4966*

TRT
RFC 3142

BIS
RFC 2767

BIA
RFC 3338

IPv4/IPv6 přechod založený na SOCKS (SOCKS64)
RFC 3089

Literatura

[1] Amoss J. J., Minoli D.: *Handbook of IPv4 to IPv6 Transition*
CRC Press, 2007, ISBN 0-8493-8516-4

[2] Beijnum I.: *Running IPv6*
Apres, 2006, ISBN 1-59059-527-0

[3] Blanchet M.: *Migrating to IPv6: A Practical Guide to Implementing IPv6 in Mobile and Fixed Networks*
Wiley, 2006, ISBN 978-0471498926

[4] Dostálek L. a kolektiv: *Velký průvodce protokoly TCP/IP: Bezpečnost*
2. vydání, Computer Press, 2003, ISBN 80-7226-849-X

[5] Hagen S.: *IPv6 Essentials*
O'Reilly, 2006, ISBN 978-0596100582

[6] Hughes L. E.: *The Second Internet*
InfoWeapons, 2010, ISBN 978-0-9828463-0-8
http://ipv6forum.com/dl/books/the_second_internet.pdf

[7] kolektiv autorů (editor Dunmore M.): *An IPv6 Deployment Guide*
projekt 6NET, 2005
http://www.6net.org/book/deployment-guide.pdf

[8] kolektiv autorů: *Enabling efficient and operational mobility in large heterogeneous IP networks*
projekt ENABLE, 2008, ISBN 978-8469106471
http://www.ipv6tf.org/pdf/enablebook.pdf

[9] Koodli R. S., Perkins C. E.: *Mobile Inter-networking with IPv6: Concepts, Principles and Practices*
Wiley-Interscience, 2007, ISBN 978-0471681656

[10] Li Q., Tatuya J., Shima K.: *IPv6 Core Protocols Implementation*
Morgan Kaufmann, 2006, ISBN 978-0124477513

[11] Li Q., Tatuya J., Shima K.: *IPv6 Advanced Protocols Implementation*
Morgan Kaufmann, 2007, ISBN 978-0123704795

[12] McFarland S., Sambi M., Sharma N., Hooda S.: *IPv6 for Enterprise Networks*
Cisco Press, 2011, ISBN 978-1-58714-227-7

[13] Murphy N. R., Malone D.: *IPv6 Network Administration*
O'Reilly, 2005, ISBN 978-0596009342

[14] Hagino J.: *IPv6 Network Programming*
Digital Press, 2004, ISBN 978-1555583187

[15] Sander S. a kol.: *Preparing an IPv6 Addressing Plan*
SURFnet, 2010
http://labs.ripe.net/Members/steffann/preparing-an-ipv6-addressing-plan

[16] Siil K. A.: *IPv6 Mandates: Choosing a Transition Strategy, Preparing Transition Plans, and Executing the Migration of a Network to IPv6*
Wiley, 2008, ISBN 978-0470191194

[17] Stallings W.: *Cryptography and Network Security*
4. vydání, Prentice Hall, 2006, ISBN 978-0131873162

[18] York D.: *Migrating Applications to IPv6*
O'Reilly, 2011, ISBN 978-1-449-30787-5

Publikací o IPv6 stále přibývá. Některé z nich jsou volně ke stažení z webu, do jiných můžete alespoň nahlédnout na *books.google.com*.

Rejstřík

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

CESNET

Sdružení CESNET bylo založeno v roce 1996 vysokými školami a Akademií věd České republiky. V současnosti je financováno především z prostředků vládní Rady pro výzkum a vývoj a z prostředků svých členů. Sdružení se zabývá výzkumem a vývojem informačních a komunikačních technologií, buduje a rozvíjí národní gigabitovou optickou síť CESNET2 určenou pro výzkum a vzdělávání.

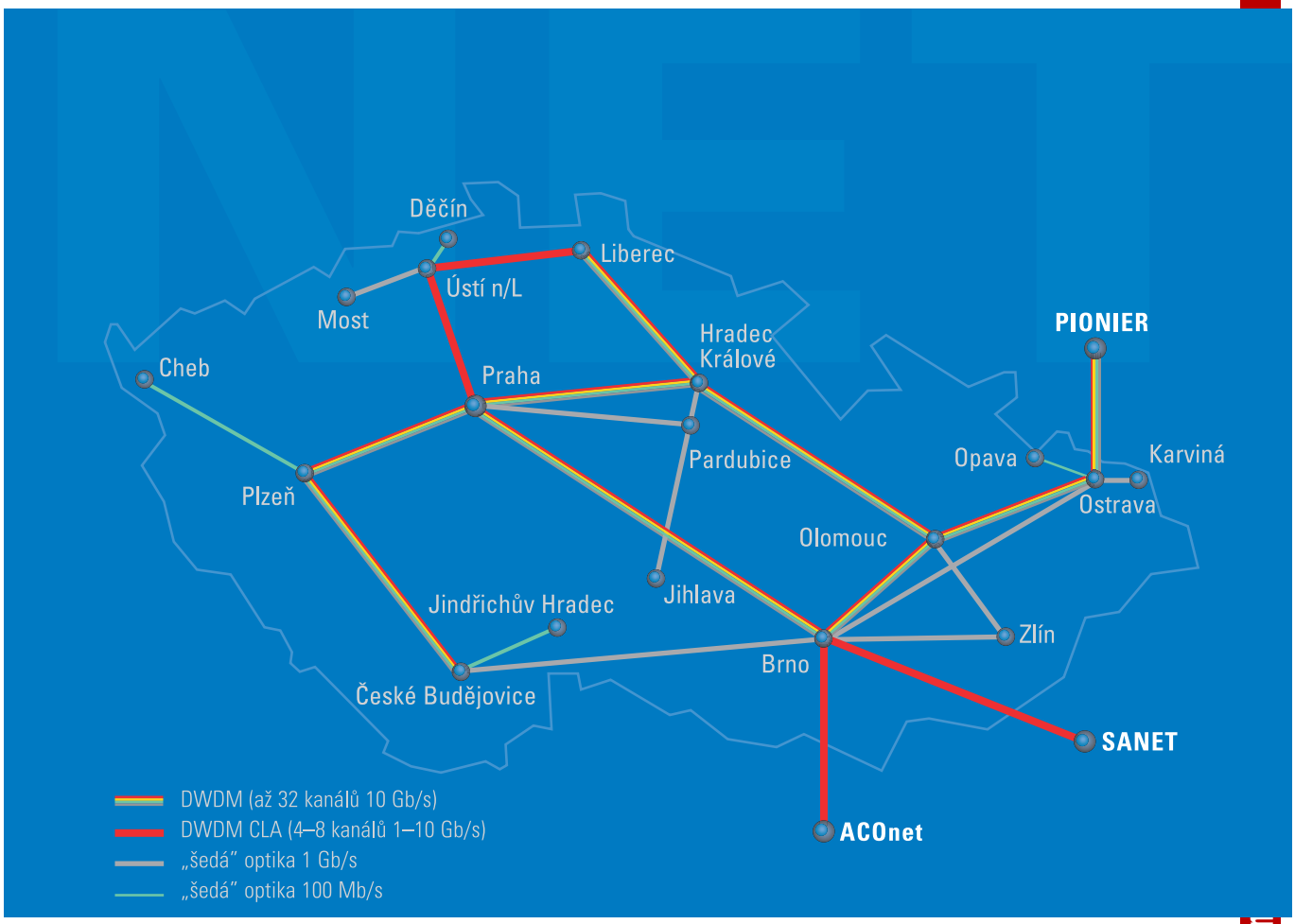
Díky výzkumným aktivitám a dosaženým výsledkům sdružení CESNET reprezentuje Českou republiku v projektu budování panevropské sítě GÉANT2 a v dalších mezinárodních projektech.

Výzkumný záměr

Pro léta 2004 až 2010 získalo sdružení dotaci formou institucionální podpory svého výzkumného záměru Optická síť národního výzkumu a její nové aplikace. Cílem výzkumného záměru je vytvořit prototyp transparentního integrovaného komunikačního prostředí, které vyhoví specifickým požadavkům vědeckovýzkumné komunity, a v reálném provozu ověřit jeho vlastnosti a životaschopnost.

Národní síť CESNET2

Výzkumné a vývojové aktivity sdružení CESNET jsou zpravidla realizovány v prostředí národní akademické sítě CESNET2. Její páteř propojuje největší univerzitní města České republiky okruhy s vysokými přenosovými rychlostmi. Uživateli sítě jsou především vysoké školy, Akademie věd ČR, výzkumná pracoviště, ale i některé střední školy, nemocnice a knihovny. Kromě klasické IP konektivity, tj. standardního připojení k Internetu, je síť CESNET2 schopna uživatelům poskytnout zejména pro vědecké a výzkumné účely vyhrazená optická spojení mezi konkrétními koncovými lokalitami.



Optická infrastruktura sítě CESNET2 a způsoby osazení vláken na konci roku 2007

IPv6

Sdružení se věnuje nové generaci internetového protokolu od roku 1999. IPv6 infrastruktura v síti CESNET2 prošla několika generacemi – od vyhrazených ATM kanálů, přes tunely až po současnou podobu, kdy je realizována nativně technologií 6PE. Podpora obou verzí IP je zcela srovnatelná, včetně přepravy skupinově adresovaných datagramů a externího napojení na mezinárodní (GÉANT2) i národní (NIX.CZ) úrovni.

CESNET