FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Testování implementací NAT64
Student:	Lukáš Vacek
Vedoucí:	Ing. Jiří Smítka
Studijní program:	Informatika
Studijní obor:	Informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2018/19

Pokyny pro vypracování

1) Popište protokoly IPv4, IPv6 a proveďte rešerši přechodových mechanizmů.
2) Najděte možná řešení implementací NAT64.
3) Nakonfigurujte NAT64 na vybraném CISCO routeru a na PC.
4) Stanovte metodologii pro otestování NAT64.
5) Proveďte testy funkčnosti, ztrátovosti, latence, rychlosti přenosů a propustnosti.
6) Analyzujte výsledné testy.

Seznam odborné literatury

Dodá vedoucí práce.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

doc. RNDr. Ing. Marcel Jiřina, Ph.D.
děkan

V Praze dne 5. ledna 2018

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Testování implementací NAT64

Lukáš Vacek

Katedra počítačových systémů
Vedoucí práce: Ing. Jiří Smítka

12. května 2018

Poděkování

Rád bych tímto poděkoval Ing. Jiřímu Smítkovi za jeho ochotu a odbornou pomoc při tvorbě této práce. Také bych rád poděkoval rodině a přátelům za to, že mě při tvorbě práce podporovali.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 12. května 2018 .....

Odkaz na tuto práci

Vacek, Lukáš Vacek. *Testování implementací NAT64*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2018.

Abstrakt

Bakalářská práce se ve své teoretické části zabývá internetovými protokoly a přechodovými metodami mezi nimi. Rozebírá metody dvojího zásobníku, tunelování a NAT64. Posledním zmíněným mechanizmem se potom zabývá praktická část, která zkoumá jeho tři různé implementace.

Klíčová slova internetové protokoly, IPv4, IPv6, přechodové mechanizmy, NAT64, Tayga, Jool, Cisco

Abstract

The bachalor thesis theoretically deals with Internet protocols and transition methods among them. It analyses methods dual-stack, tunneling and NAT64. The practical part deals with the last mentioned mechanism and examines its three different implementations.

Keywords internet protocols, IPv4, IPv6, transition mechanisms, NAT64, Tayga, Jool, Cisco

Obsah

Úvod **1**
Struktura práce . . . 1
Cíl práce . . . 2

1 Internetové protokoly **3**
1.1 IPv4 . . . 3
1.2 IPv6 . . . 7

2 Přechodové mechanizmy **13**
2.1 Dvojí zásobník . . . 13
2.2 Tunely . . . 13
2.3 Stateless IP/ICMP Translation Algorithm . . . 17
2.4 Network address translator . . . 19

3 Implementace a konfigurace NAT64 **25**
3.1 Tayga . . . 25
3.2 Jool . . . 26
3.3 CISCO . . . 28

4 Testování vybraných implementací **29**
4.1 Funkčnost . . . 30
4.2 Ztrátovost . . . 31
4.3 Latence . . . 32
4.4 Rychlost přenosů . . . 34
4.5 Propustnost . . . 34
4.6 Shrnutí testů . . . 36

Závěr **37**

Literatura **39**

A Seznam použitých zkratek **43**

B Obsah přiloženého CD **45**

Seznam obrázků

1.1 Hlavička protokolu IPv4 (převzato z [1]) 6
1.2 Google statistiky využití jejich serverů po IPv6 (převzato z [2] . . 8
1.3 DNS dotazy (převzato z [3]) 9
1.4 Hlavička protokolu IPv6 (převzato z [4]) 11
1.5 Porovnání hlaviček protokolů (převzato z [5]) 12

2.1 Výměna dat mezi 6to4 a nativním IPv6 (převzato z [4]) 14
2.2 Zahájení komunikace s použitím NAT64 (převzato z [4]) 24

3.1 Uchycení Jool na prerouting řetěz (převzato z [6]) 27

4.1 Konfigurace sítě - latence 30
4.2 Konfigurace sítě - ztrátovost, rychlost přenosů, propustnost 30
4.3 Procentuální ztráta paketů 32
4.4 Úspěšně přenesené pakety 33
4.5 Průměrná latence 33
4.6 Rychlost přenosů 34
4.7 Šířka pásma 35

Seznam tabulek

4.1 29

Úvod

Internetové protokoly jsou jedny ze základních stavebních kamenů dnešního Internetu. Mají na starost adresaci paketů a jsou tedy takovým kurýrem či pošťákem sítí.

V dnešní době se používají výhradně dva protokoly IPv4 a IPv6. Když se první zmíněný protokol vyvíjel na začátku osmdesátých let, předpokládalo se, že čtyři miliardy adres budou stačit. Po několika letech se zjistilo, že tento počet je nedostatečný a byly provedeny kroky, které jejich vyčerpání oddálily. Zároveň počátkem devadesátých let se začal vyvíjet nový protokol, označený jako IPv6. Ten měl odstranit problém s nedostatkem adres a zároveň přinést nové vlastnosti.

V silných regionech (Evropa, Severní Amerika a Asia—Pacifik) jsou v současnosti veřejné IPv4 adresy téměř vypotřebované. To je důvod, proč se pomalu začalo přecházet na novější protokol IPv6. Problémem je, že tyto dva protokoly jsou nekompatibilní. Nelze tedy komunikovat ze stroje používající pouze IPv6 protokol na stroj, který má pouze protokol IPv4 a naopak. Aby taková komunikace byla možná, využívá se přechodových mechanizmů.

Struktura práce

Práce se na začátku věnuje jednotlivě internetovým protokolům, jejich řešení fragmentace datagramů, formátu a obsahu hlavičky datagramu. Dále se zaměří na přechodové mechanizmy. Nejdříve bude popsán mechanizmus duálního zásobníku (dual-stack). Následují tunelovací mechanizmy jako jsou Teredo, 6over4 či 6to4. Práce popisuje SIIT algoritmus, kterého využívá další přechodový mechanizmus a to NAT64. Mechanizmus NAT64 je rozebrán více do hloubky, neboť o něm pojednává celá praktická část práce.

V praktické části jsou rozebrány tři různé implementace NAT64. Nejdříve se autor zaměří na jejich konfiguraci, a jakým způsobem překlad řeší. Poté

budou podrobeny testům zaměřené na ztrátovost, latenci, šířku pásma a rychlostipřenosu. Následně proběhne jejich analýza.

Cíl práce

Tato práce může být přínosem pro administrátory sítí při rozhodování, jakou implementaci překladače NAT64 provozovat. Také jim pomůže vybrané implementace nasadit a pochopit jejich fungování. Důvodem, proč se autor tomuto tématu věnuje je, že žádné podobné výsledky naměřené neexistují.

Kapitola 1

Internetové protokoly

V dnešní době se používají výhradně dva protokoly IPv4 a IPv6, jejichž základní funkcí je zajištění logického adresování v internetové síti. Jsou tedy jedny z nejvytíženějších protokolů z rodiny TCP/IP. Internetové protokoly, stejně jako ICMP, náleží jak do síťové vrstvy ISO/OSI modelu, tak i do síťové vrstvy TCP/IP modelu. [7]

Internetové protokoly zajišťují správné doručování v síti. Nedisponují žádnými mechanizmy na zajištění spolehlivé a spojované komunikace. [8] Protokoly k paketům přidají logické adresy a očekávají, že dorazí na správné místo. Pokud během přenosu nastane chyba, odesílatel je informován pomocí protokolu ICMP, který oznámí, o jakou chybu se jedná. Spolehlivost a spojovanou komunikaci přenechávají na protokolech vyšší vrstvy. V rodině protokolů TCP/IP jsou protokoly transportní vrstvy TCP a UDP.

1.1 IPv4

Celé adresování dříve v síti obstarával sám protokol TCP, který zajišťoval spolehlivý přenos, i za cenu vyšší režie a pomalejší rychlosti. [9] Čas ukázal, že toto řešení není ideální. Komplikace se objevovaly především při přenosu hlasu. Následkem toho došlo k odtržení směrování od protokolu TCP, a tak vznikl internetový protokol. První RFC dokument o internetovém protokolu vyšel v roce 1980. [10] Zároveň s tímto odtržením vznikl i další protokol – UDP. [11]

1.1.1 Logické adresy

Logická adresa je 32bitová a skládá se ze 4 bloků po 8 bitech. Adresa se zapisuje pomocí decimálního zápisu odděleného tečkou. Blok tedy může nabývat hodnot od 0 do 255. Adresa může vypadat například takto: 12.34.56.78.

Některé adresy jsou vyhrazené pro speciální účely. Většina těchto adres se nepoužívá ve veřejné síti, avšak mají mají účel pro testování a experimenty, nebo jsou přímo vyhrazeny pro privátní sítě. [12]

- 127.0.0.0/8, jinými slovy localhost. Jedná se o blok adres, které se přiřazují loopback rozhraním. Pomocí těchto adres lze zjistit stav TCP/IP stacku na vlastním počítači.
- 169.254.0.0/16 blok je tzv. „link local“. Pokud adresa není přiřazena například z dhcp, je na rozhraní přiřazená adresa z tohoto bloku.
- 224.0.0.0/4 je blok rezervovaný pro multicast. Umožňuje poslat stejná data jen na vybrané stroje.

1.1.2 Vývoj adresování

V počátcích protokolu se nepředpokládalo s tak širokým využitím jako dnes. Prognóza uváděla pouze tři varianty sítí, ve kterých se velikost masky neměnila. [8]

- Třída A: první bit oktetu začíná 0. Dalších 7 bitů oktetů reprezentuje adresu sítě a zbylých 24 bitů náleží pro lokální adresy.
- Třída B: v tomto případě první bit je 1, druhý je 0 a následujících 14 bitů patří adrese síťě. Posledních 16 bitů je pro lokální adresy.
- Třída C: nyní jsou rezervovány první tři bity. První dva mají hodnotu 1, třetí zase 0. Poté následuje 21 bitů pro adresy sítí a zbylých 8 je pro lokální adresy.

Při takovém rozdělení docházelo k velmi častému plýtvání. Názornou ukázkou je například propojení dvou routerů v internetu. Nejmenší možný použitelný rozsah je třída C s celkem 256 adresami. Jsou použity pouze dvě adresy pro zprovoznění komunikace a zbylé zůstanou bez využití.

V důsledku tohoto plýtvání se zrušilo třídní rozdělení sítí a zavedly se podsítě. Škálování sítí se provádí pomocí masky sítě. Maska, stejně jako adresa sítě je 32bitová. Jedná se o číslo určující počet bitových jedniček zleva. Maska se zapisuje dvěma způsoby, a to jako adresa sítě čtyřmi osmi bitovými oktety nebo jako suma jedniček oddělená lomítkem od konce adresy (CIDR formát).[13] Může se to zapsat tedy jako 12.34.128.0/23 nebo 12.34.128.0 s maskou 255.255.254.0.

Dalším krokem, který vedl ke snížení plýtvání adres, bylo vyhrazení třech bloků sítí pro privátní sítě. Těmi jsou 10.0.0.0/8, 172.16.0.0/12 a 192.168.0.0/16. Tyto sítě se používají pro adresování uvnitř firem, škol a domácností. Ve veřejném internetu se nepoužívají. To znamená, že žádná veřejně dostupná služba neběží na privátních adresách. [14]

Jedním z posledních kroků, který ušetřil několik bloků adres, je metoda Network Address Translation (dále jen NAT). Díky této metodě je stále možné používat internetový protokol IPv4 a mapovat logické adresy na jiné.[15] Pokud chce stroj nacházející se v NATované privátní síti, komunikovat s Internetem, musí svojí komunikaci směřovat přes NAT. NAT změní zdrojovou logickou adresu v datagramech a tuto změnu si zapamatuje. Na základě namapování, NAT změní v příchozí odpovědi logickou adresu na adresáta z privátní sítě.

Tato technologie omezuje transparentnost internetu, u kterého pozměnila jeho směřování na architekturu klient/server na rozdíl od peer-to-peer. Názornou ukázkou jsou online hry. Aby se hráči z jiných sítí mohli ve hře propojit, musí existovat veřejně dostupný server, jenž tuto konektivitu umožní.

Sami autoři tuto technologii prezentovali jako krátkodobé řešení problému nedostatku adres a škálování internetu. Byli si vědomi problémů, které NAT přinese. [16]

1.1.3 Fragmentace

K fragmentaci u IPv4 dochází na routrech, které škálují podle hodnoty MTU. MTU (Maximum Transmission Unit) označuje maximální velikost paketů, které mohou po daném spojí být přeneseny. Pokud k fragmentaci dojde, vytvoří se pakety se stejnou hlavičkou a povolenou velikostí MTU. Do těchto nových paketů se napíše offset, který určuje pořadí fragmentu paketu. [8]

1.1.4 Zjišťování adres

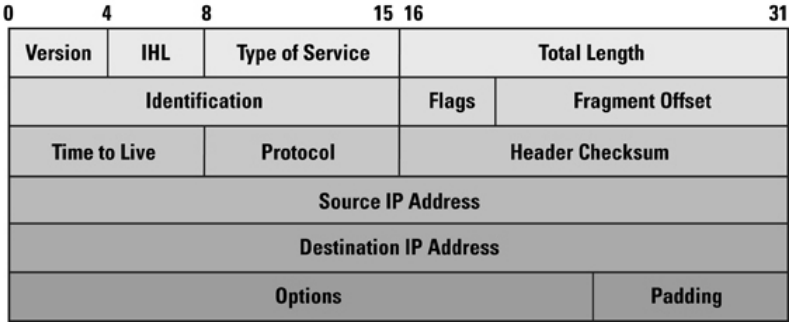
Odesílání paketů v rámci jedné sítě zajišťuje protokol ARP, který není součástí IPv4. ARP provádí překlad logických adres na adresy hardwarové. Odešle broadcastový dotaz s dotazem, kdo vlastní danou logickou adresu. Pokud se někdo s takovou logickou adresou nachází v síti, odpoví tázajícímu a ten nyní ví, kam směřovat svůj dotaz. [7]

1.1.5 Hlavička protokolu

Hlavička protokolu IPv4 má nejméně 20 bajtů. V případě, že budou použity přídavné volby, velikost se zvětší. Hlavička je vyobrazována po 32bitových slovech i z důvodu, že výsledná velikost hlavičky musí být násobkem 32. Rozložení jednotlivých polí v hlavičce protokolu popisuje obrázek 1.1.

Version: Čtyřbitové pole určuje **verzi** IP protokolu. Hodnota je vždy u IPv4 4.

Internet Header Length (IHL): Délka hlavičky pole říká, jak velká je hlavička protokolu. Velikost je udávána ve čtyřbajtových slovech, kde minimální hodnotou je 5. Pole má délku 4 bitů.



Obrázek 1.1: Hlavička protokolu IPv4 (převzato z [1])

Type of Service (ToS: Osmi bity reprezentovaný **typ služby** popisuje, jaká data protokol zapouzdřuje. Pomocí jednotlivých bitů lze nastavit priority a vlastnosti směrovačů, jak se má k paketu zachovat. První tří popisují, například zda se jedná o běžnou informaci, kriticky důležitou informaci nebo řízení sítě. Další bitu nastavují zpoždění, propustnost či spolehlivost.

Total length: Celková délka udává velikost paketu včetně přenášených dat uvnitř datagramu a hlavičky IPv4 protokolu. Pro tuto hodnotu je vyhrazeno 16bitů.

Identification: Každý datagram obsahuje unikátní identifikátor v rámci datagramu nebo proudu datagramů. Identifikátor je přidělován odesílatelem. Pokud dojde k fragmentaci při přenosu mají jednotlivé fragmenty stejné ID. Cílové zařízení pak pomocí ID pozná, že jde o datagramy patřící k sobě a na základě offsetu je složí dohromady.

Flags: Příznaky jsou tvořeny pouze třemi bity. První bit je rezervován a musí být vždy nula. Druhý bit je označován jako DF (Don't Fragment) a je-li hodnota 1, znamená to, že paket nemá být dělen na fragmenty. Opačně tomu tak je v případě 0. Třetí a poslední bit oznamuje, zda se jedná o poslední kus fragmentu či následují za tímto datagramem ještě další. Bit se také nazývá MF (More Fragments). Více fragmentů je očekáváno má-li bit honotu 1.

Fragment Offset: 13 bitové pole navazuje na předešlé. Toto pole určuje pořadí fragmentu mezi datagrami se stejným identifikátorem. První fragment má offset 0, další jsou pak odvozeny od MTU.

Time To Live (TTL): Doba platnosti určuje délku života datagramu a je udávána v sekundách. Pokud je tedy hodnota 0, musí být datagram zničen. Změna se provádí vždy, když je hlavička zpracovávána. Každý směrovač, který mění obsah hlavičky, musí zmenšit hodnotu aspoň o 1.

Protocol: Pole **protokol** o velikosti jednoho bajtu uvádí, jaký protokol je použit na další vrstvě datagramu. Jednotlivé hodnoty jsou nyní popsány na stránkách organizace IANA pod nadpisem „Protocol Numbers".

Header Checksum: Kontrolní součet hlavičky je jeden z mechanizmů

pro korekci chyb vzniklé při přenosu. Každé zařízení pracující s hlavičkou IPv4 vypočítá kontrolní součet datagramu a porovná to s hodnotou uvedenou v poli. Pokud hodnoty při kontrole nejsou správné, je datagram zahozen bez odeslání zprávy. Při každé manipulaci s datagramem musí být hodnota kontrolního součtu změněna. Důvodem je, že se některé hodnoty polí mění (například TTL). Pole je veliké 2 bajty.

Source Address: 32-bitová zdrojová adresa uzlu.

Destination Address: 32-bitová adresa cílového uzlu.

Options: Volby jsou nepovinnými parametry, které se v hlavičce mohou objevit. Podmínkou je, že použité parametry musí znát všechna zařízení na cestě přenosu, aby je mohli zpracovat.

Padding Výplň slouží k doplnění volby na 32bitové slovo. Jeho délka je proměnná.

[7, 8]

1.2 IPv6

Na začátku devadesátých let už bylo zřejmé, že adresy IPv4 budou brzy vypotřebovány. V důsledku toho IETF (organizace která vyvíjí a podporuje internetové standardy) rozhodla, že je potřeba přijít s něčím novým, co nepřinese jen rozšíření adresního prostoru ale i nové vlastnosti a vylepšení.

Požadavek na větší rozsah adresního prostoru vedl k nemalým debatám o optimální délce adresy. Nakonec byla stanovena na 128 bitů, tedy čtyřnásobek délky použité u IPv4. To znamená, že k dispozici je 3.4×10^{38} adres. To je jen těžko představitelné číslo. Povrch zemeěkoule činí přibližně půl miliardy kilometrů čtverečních. To znamená, že na jeden čtvereční milimetr zemského povrchu připadá 667×10^{15} adres.

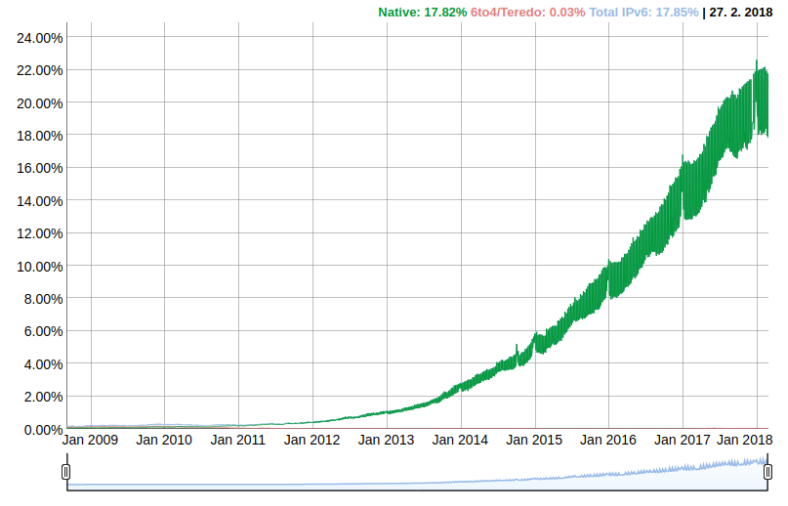
Došlo k zrevidování hlavičky a byly odebrány zbytečné položky, například kontrolní součet. Dalším vylepšením je odebrání zjišťování adres pomocí protokolu ARP. Zjišťování adres nyní provádí protokol IPv6 za pomocí ICMP.[4]

1.2.1 Aktuální situace

V posledních letech nárůst dotazů po IPv6 narostl. Společnost Google se rozhodla IPv6 prosazovat a na základě toho na svých serverech dává přednost dotazům jdoucím po IPv6.[4] Na obrázku 1.2 je vidět graf četnosti těchto dotazů.

Z českých firem, které pomáhájí prosazovat protokol IPv6 je společnost CZ.NIC. Tato společnost spravuje registr českých domén. Obrázek 1.3 ukazuje procentuální zastoupení DNS dotazů po internetových protokolech.

Dotazy na Google, i na DNS servery, se pohybují okolo 20 %. Českých domén, které mají i IPv6 adresu, je skoro 400 000, což je 30 %.



Obrázek 1.2: Google statistiky využití jejich serverů po IPv6 (převzato z [2]

Česká vláda v roce 2009 přijala ustanovení o přechodu na internetový protokol verze 6. To znamená, že v rámci pravidelné obměny hardwaru musí síťové prvky podporovat IPv6. Toto z větší části již většina splnila. [17]

1.2.2 Logické adresy

Logická adresa má 128 bitů a je reprezentována pomocí osmi skupin po čtyřech bajtech. Každý bajt je pak reprezentován jako dvojice hexadecimálních čísel. Adresy vypadají například takto:

```
2001:0db8:85a3:0000:0000:8a2e:0000:0370
```

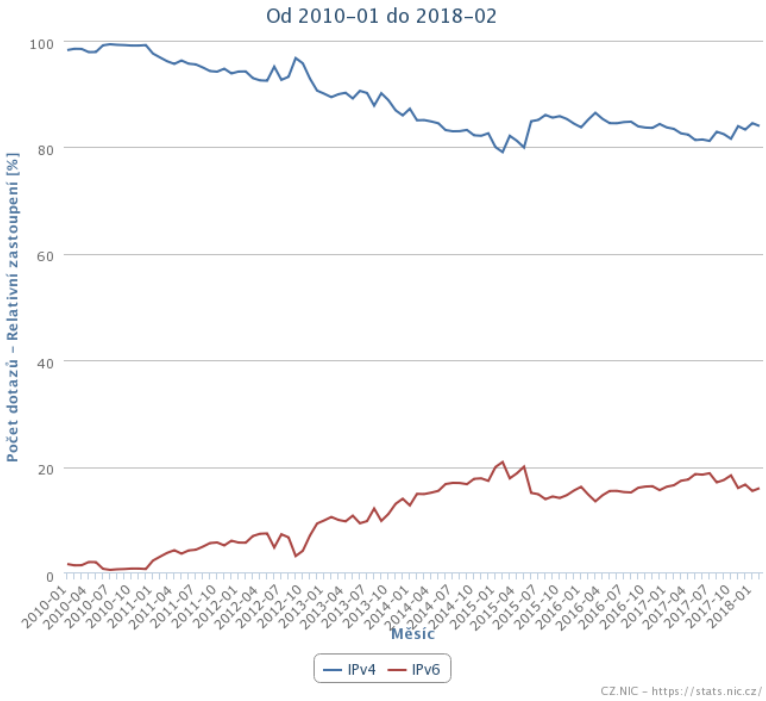
Velmi častý výskyt nul v zápisech nabízí dvě možnosti jak zápis zkrátit.

1. Je možné zkrátit jeden či více bloků obsahující pouze samé nuly. Blok je následně nahrazen zápisem „::“. Takové to zkrácení se dá použít pouze jednou, aby bylo zřejmé jaká část adresy je zkrácena. Výsledek u příkladu výše může vypadat dvěma způsoby.

   ```
   2001:0db8:85a3::8a2e:0000:0370
   2001:0db8:85a3:0000:0000:8a2e::0370
   ```

2. Nuly na začátku jednotlivých bloků se také mohou vynechat, ty na konci bloku nikoli. Celý blok nul se tedy může zkrátit pouze na jedinou. Výsledek po použití pouze tohoto pravidla vypadá následovně:

   ```
   2001:db8:85a3:0:0:8a2e:0:370
   ```



Obrázek 1.3: DNS dotazy (převzato z [3])

Pravidla zkrácení můžeme kombinovat. V tomto případě nejkratší možná varianta je následující.

```
2001:db8:85a3::8a2e:0:370
```

1.2.3 Typy logických adres

Stejně jako u IPv4, jsou bloky IPv6 adres rozděleny do několika skupin, které jsou identifikovatelné pomocí svého prefixu.

- ::/128 je nedefinovanou adresou. Ta je přiřazena rozhraním, kterým dosud nebyla přidělena IPv6 adresa.
- ::1/128 reprezentuje u IPv6 localhost.
- fc00::/7 je blok pro unikátní lokální adresy.
- fe80::/10 je prefix pro lokální linkové adresy, za kterým následuje 54 nulových bitů. Adresa končí identifikátorem rozhraním, které může být náhodně vygenerováno nebo vyvozeno z MAC adresy.
- ff00::/8 je blok skupinových adres. Prefix se například využívá při procesu objevování sousedů.

- 64:ff9b::/96 je určen pro algoritmy mapování mezi internetovými protokoly.
- 2001::/32. Tento blok využívá přechodový mechanizmus Teredo.
- 2001:db8::/32. Tento blok je rezervován pro účely dokumentace a testování.
- 2002::/16 blok je využit tunelem 6to4.
- **Globální individuální adresy** jsou skupinou adres, které jednoznačně identifikují svého nositele v Internetu, jsou synonymem pojmu veřejné adresy u IPv4. Dnes tyto adresy mají na prvních třech místech binární konstantu `001`. Za nimi následuje globální směrovací prefix o velikosti 45 bitů, který identifikuje koncovou síť. Ten je přidělován lokálním internetovým registrem. Dalších 16 bitů adresy tvoří identifikátor podsítě a zbylých 64 bitů je určeno pro identifikaci rozhraní.

1.2.4 Fragmentace

Narozdíl od IPv4, kde mohl fragmentovat každý směrovač u protokolu IPv6 to je jiné. Pokud by po cestě mělo dojít k fragmentaci, pakety se zahodí a odesílateli je zaslán ICMP paket se zprávou, že IPv6 pakety překročily velikost MTU. Ve zprávě se nachází i povolená hodnota MTU, aby na stejném místě nedošlo opět k zahození.

1.2.5 Objevování sousedů

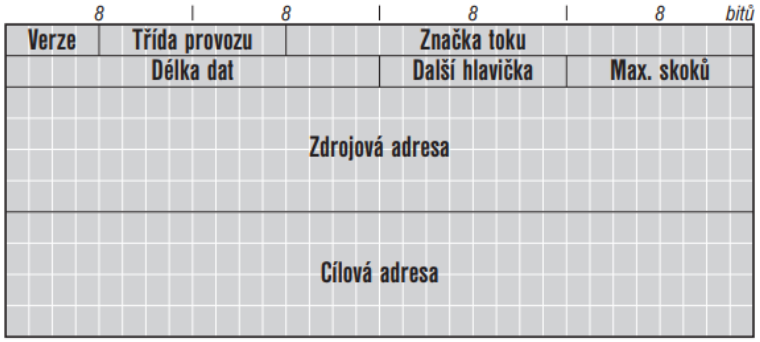
Ke zjišťování adres v rámci jedné sítě není u IPv6 potřeba další protokol, jako tomu bylo u IPv4 s ARP. Tento proces je zakomponován přímo v IPv6 a je označován jako objevování sousedů. V podstatě se jedná o to samé, jen se změnily názvy a adresa.

Pro překlad se využívá skupinových adres se společným prefixem, kterým je ff02:0:0:0:0:1:ff00::/104. Za tento prefix se připojí posledních 24 bitů IPv6 adresy, pro kterou se hledá adresa linková. Na takto složenou adresu je tazatelem odeslána výzva sousedovi. Pokud soused s takovou adresou existuje, tazateli odpoví, jakou linkovou adresu vlastní. Na těchto skupinových adresách vyzývaného uzlu musí poslouchat každý IPv6 uzel, který byl do takové skupiny přiřazen. Pokud by tak neučinil, nemohl by dostávat výzvy, a tedy ani odpovídat.

1.2.6 Hlavička protokolu

Formát datagramu je obvyklý: hlavička následována daty. Velikost hlavičky je pevně daná a činí 40 B. Oproti IPv4 je velikost větší a to i přesto, že došlo k velké změně jejího obsahu. Mnoho polí bylo zrušeno, změněly pořadí nebo

název. Důvodem, proč je i přes to hlavička větší, než minimální 20B velikost IPv4, je velikost adres. Zdrojová i cílová adresa zabírají 75 % celé hlavičky, což lze vidět na obrázku 1.4 (co čtvereček to bit).



Obrázek 1.4: Hlavička protokolu IPv6 (převzato z [4])

Verze: 4bitové pole indetifikující verzi protokolu. Tato hodnota je zde konstantně 6.

Třída provozu: Položka o velikosti jednoho bajtu vyjadřuje rozdílné třídy a priority datagramu. Umožňuje datagramu být přednostně zpracován směrovačem a dříve tak odeslán dál. Také je to navržené pro zaručení určité přenosové rychlosti, zpoždění či rozptyl, což ale v praxi zatím moc nefunguje.

Značka toku: Následujících 20 bitů slouží jako identifikátor. Směrovač datagramy se stejným vlastnostmi (identifikátorem, odesílatelem, adresátem a vlastnostmi spojení) bude směrovat po stejné cestě.

Délka dat: Na určení délky dat je vyhrazeno 16 bitů. Hlavička se do velikosti nepočítá, jde čistě o data nacházející se za ní. Pokud 64 KB je pro data malá velikost a je potřeba vytvořit větší datagram, lze použít rozšiřující hlavičku *Jumbo obsah*.

Další hlavička: Další hlavička uvozuje v 8 bitech, jaká hlavička či jaký druh dat se nachází za standardní hlavičkou. Rozšiřující hlavičky jsou zřetězené přímo za sebou a jejich pořadí je přesně určeno. Na konci řetězu se nachází data.

Max. skoků: V 8 bitech se určuje životnost datagramu. Zamezují se tím tak cykly, které mohou vzniknout. Každý směrovač, který přijde s datagramem do styku, sníží hodnotu maximální počet skoků o 1.

Zdrojová adresa: 128bitová adresa zdrojového uzlu.

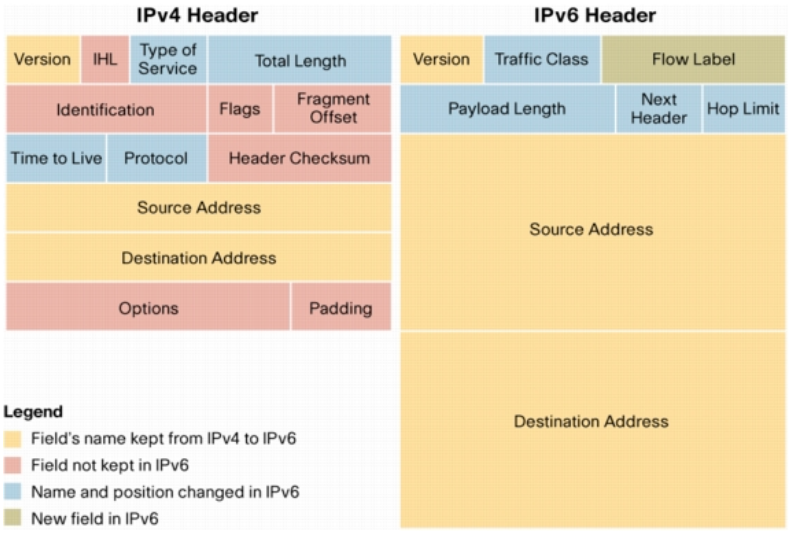
Cílová adresa: 128bitová adresa cílového uzlu. [4, 18]

1.2.7 Kompatibilita s IPv4

IPv6 není kompatibilní s IPv4, jelikož se jedná o dva odlišné protokoly. Důsledkem této nekompatibility vzniklo několik mechanizmů, které tento problém

řeší. Tyto mechanizmy jsou popsány v další kapitole.

Porovnají-li se hlavičky protokolů IPv4 a IPv6 (obrázek 1.2.7), lze vidět, jak se rozvržení položek a položky samotné liší. Mnoho nepotřebných položek bylo odstraněno, další změnily jméno a pouze jedna byla přidána. Kromě toho je i jiné řešení přídavných voleb. U IPv4 se rozšiřující volby přidávají do jedné hlavičky za adresy, a tím hlavička nabírá na velikost. Kdežto u IPv6 mají hlavičky jak základní, tak rozšiřující pevnou velikost a řetězí se za sebou.



Obrázek 1.5: Porovnání hlaviček protokolů (převzato z [5])

Kapitola 2

Přechodové mechanizmy

2.1 Dvojí zásobník

Jedná se o mechanizmus, kde koexistují oba protokoly zároveň. To znamená, že síťové rozhraní má adresy IPv4 i IPv6. Na rozhraních je upřednostňována komunikace IPv6 nejen pomocí systému DNS. Dotaz je nejdříve veden po IPv6 a pokud nepřijde kladná opověď, dotaz se opakuje po IPv4.

Toto řešení je dobré zavést do zaběhnutých sítí. Poté, co administrátoři zajistí úplnou funkčnost po IPv6, bude moct být protokol IPv4 odstraňován, aniž by uživatelé cokoliv postřehli.[4]

2.2 Tunely

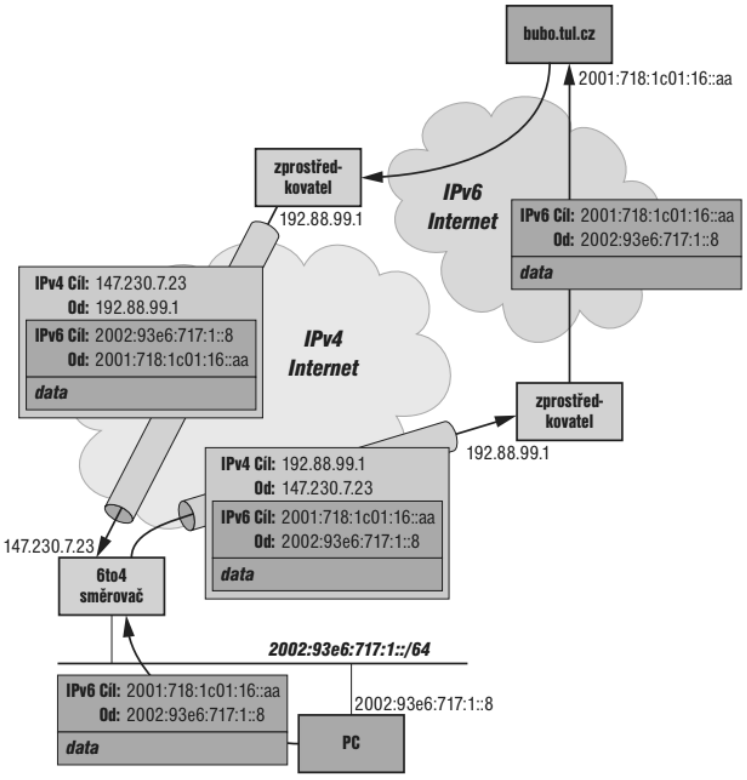
Mechanizmus tunelování funguje na principu zapouzdření obsahu jednoho protokolu do druhého. Tunel má dva konce. K jednomu konci tunelu připutuje paket, který je následně obalen jiným, tak aby v tunelu mohl fungovat a mohl být poslán na druhý konec. Na druhém konci je obal paketu odstraněn a poslán dále do sítě. Tento princip je využíván i u jiných tunelovaných, služeb jako je VPN či HTTP tunel.

V případě internetových protokolů je nejčastěji tunelována IPv4 síť. Tunel má na každém konci IPv4 adresu. Doputuje-li k jednomu konci tunelu IPv6 datagram a zařízení rozhodne, že jeho následující cesta vede skrz tunel, obalí ho IPv4 datagramem. V hlavičce, jenž obaluje IPv4 datagram, bude v položce protokol hodnota 41 značící zapouzdření.[19] Cílová adresa bude nastavena na druhý konec tunelu. V tunelu bude datagram směrován sdandardním způsobem. Poté, až dorazí na konec tunelu, se příjemce podívá do hlavičky datagramu. Uvidí-li hodnotu protokolu 41, rozbalí obsah a následně směřuje dále do sítě již IPv6 datagram. Tunel je pro IPv6 datagram jeden skok, i když se v tunelu se může provést skoků několik.[4]

2.2.1 6to4

Umožňuje IPv6 sítím komunikovat přes IPv4 síť. Směrovač spojující tyto sítě musí vlastnit alespoň jednu veřejnou IPv4 adresu. Na základě čtyřkové adresy je vytvořen 48bitů dlouhý prefix, který je propagován do sítě. Prefix se skládá ze dvou částí. Začíná hodnotou 2002::/16, což je adresa určující, že se jedná o 6to4 tunel. Druhá část, tedy zbylých 32bitů, je právě veřejná IPv4 adresa směrovače. Je-li tedy veřejná IPv4 adresa 147.32.232.248, prefix bude vypadat takto: 2002:9320:E8F8::/48.

Komunikace od nativního IPv6 zařízení směřuje na nejbližší 6to4 směrovač, který ji tunelem přepošle na cílový 6to4 směrovač. V případě že koncové sítě nemají přístup do globálního IPv6 Internetu, nastaví jako implicitní cestu anycastovou adresu 192.88.99.1 (v IPv6 2002:c058:6301::). Tuto adresu využívají 6to4 zprostředkovatelé, kteří datagramy následně předávají do IPv6 sítí (obrázek 2.1).[4]



Obrázek 2.1: Výměna dat mezi 6to4 a nativním IPv6 (převzato z [4])

2.2.2 IPv6 Rapid Deployment (6rd)

Jedná se o obdobu 6to4. Funkčnost je stejná jen prefix a jeho správu má na starosti jeden subjekt. V tomto případě jde o poskytovatele Internetu, který definuje obsah a velikost prefixu a také provozuje několik bran mezi 6rd a nativním IPv6. Adresa sítě zde začíná 6rd prefixem a následuje IPv4 adresa. Velikost prefixu může nabývat až 32 bitů. Zbylé bity z prefixu se využijí pro podsítě, které pokračují za IPv4 adresou. [4]

2.2.3 6over4

6over4 je metoda, která umožňuje formátovat přenos IPv6 datagramů přes IPv4 infrastrukturu. Zaměřuje se na izolované počítače podporující IPv6. Technologie pro fungování potřebuje adresy obou internetových protokolů. Adresa IPv6 je odvozena z IPv4 adresy, která může být jak globální tak i privátní. Tato odvozenina vznikne z prefixu podsítě, který musí být 64 bitů dlouhý, následována čtyřmi nulovými bajty a konče IPv4 adresou.

IPv6 protokol u této metody neztrácí mechanizmy jako objevování sousedů či bezstavovou automatickou konfiguraci. Pro tyto funkčnosti jsou zde využity standardně link-local, adresy tedy prefix FE80::/64. To ovšem znamená, že 6over4 musí umožnit používání multicastu. Toho je dosaženo tak, že skupinová IPv6 adresa je mapována na IPv4 adresu 239.192.X.Y, kde X a Y jsou poslední dva bajty mapované IPv6. Problémem tohoto řešení je, že ne každá IPv4 síť multicast podporuje.[4]

2.2.4 ISATAP

ISATAP nebo-li *Intra-Site Automatic Tunnel Addressing Protocol* také cílí na izolované stroje, akorát odstraňuje nevýhodu v podobě multicastu. Protokol slouží hlavně pro potřeby v zákaznických sítích a nezaměřuje se na komunikaci mezi dílčími sítěmi. Tu přenechává jiným metodám, jako je 6to4.

Datagram je standardním způsobem pomocí ISATAP zařízení zabalen do IPv4 a poslán tunelem na druhý konec. Tyto zařízení mají speciální formát adresy, který opět vychází z IPv4 adresy. Prvních 64 bitů je určeno pro prefix sítě, za kterým následuje 6 nulových bitů. 7 bit říká, zda IPv4 adresa je lokální nebo globální. Bit má hodnotu jedna v případně globální adresy v opačném případě nula. 8 bit je individuální. Může být nastaven na jakoukoliv hodnotu. Logická adresa dále pokračuje konstantou 005efe a končí IPv4 adresou. [4]

2.2.5 Teredo

Většina tunelů naráží na problém, v případě kdy mají komunikovat s počítači nacházejícími se za NATem. Teredo je mechanizmus, který tento problém řeší.

Prvotní komunikace pochází z NATované sítě, aby v NATu došlo k namapování adres a portů. Teredo rozlišuje, o jaký typ NATu se jedná. Pokud se

jedná o trychtýřový NAT (cone NAT) nebo omezený NAT (restricted NAT), je schopen si s nimi poradit. Narazí-li ale na symetrický NAT (symmetric NAT), neumí se s ním vypořádat. V takovém případě je doporučeno změnit konfiguraci na omezený NAT.

Struktura adresy je zde komplikovanější. Skládá se z pěti částí, kde první část je teredo prefix 2001::/32. Prefix podsítě uzavírá IPv4 adresa Teredo serveru, který ji přidělil. Třetí část má 16 bitů a je vyhrazena pro příznaky. První bit v této části, pokud je nastaven na 1, říká, že se jedná o trychtýřový NAT. 0 značí jiné varianty. Tato informace v adresách byla zneužita útočníky, proto RFC 5991 hodnotu bitu nařídilo nastavovat vždy jako 0. Zda je adresa globální a unikátní, reprezentuje 7 a 8 bit. Zbylé bity v poli příznaků jsou nastaveny na náhodnou hodnotu kromě 2. bitu, který má vždy hodnotu 0. V následujících 16 bitech je uložena hodnota UDP portu, která byla použila. Poslední a pátou částí je IPv4 veřejná adresa NATu. Hodnoty 4. a 5. části jsou invertovány.

Stroj nacházející se v NATované IPv4 síti a který by chtěl do IPv6 Internetu, označujeme jako klienta. Server je zase zařízení, které je ve veřejném Internetu a má jak IPv4 adresu tak IPv6. Klient standardně komunikuje se serverem, jenž kromě překladu adres, informuje klienta o použitém UDP portu a IPv4 adrese.

Komunikují-li dva Teredo klienti mezi sebou, probíhá to na přímo. Spojení je dosaženo ve čtyřech krocích.

1. První klient zahájí komunikaci a tím otevře cestu v NATu na své straně. Pokud narazí na NAT druhého klienta, přejde k 2. kroku.

2. Aby první klient navázal spojení skrze NAT protějšku, využije k tomu Teredo server. Stejný dotaz, který první klient poslal druhému, se nyní pošle ale přes Teredo server, jenž ho přepošle cílovému stroji.

3. Druhý klient nyní ví, že s ním chce první komunikovat, a tak mu odešle zprávu, čímž otevře cestu skrze NAT na jeho straně.

4. Teď již oba ví, jak s druhým komunikovat skrze NAT.

Pokud chce klient navázat spojení se strojem, který nemá Teredo adresu, je zde potřeba zprostředkovatele. Zprostředkovatel je směrovač, který předává datagramy z IPv6 do Teredo světa a vice versa. Teredo prefix 2001::/32 je tímto směrovačem šířen do IPv6 světa.

Režie komunikace se značně projevuje na funkčnosti. Testy provedené Geoffem Hustonem ukazují, že kromě rychlosti je problém i se spolehlivostí spojení. [20]

2.2.6 Dual-Stack Lite

Mechanizmus Dual-Stack Lite, oproti předchozím tuneluje naopak IPv4 skrze IPv6 sítě. Reaguje na situaci, kdy provideři budou provozovat páteřní síť jen pomocí IPv6 protokolu. Může je vést k tomu nedostatek adres nebo zjištění, že je to pro ně mnohem jednodušší (nemusí provozovat dva protokoly zároveň – konfigurace dynamického směrování, DHCP atd.).

Tunel se skládá ze dvou prvků z *Basic Bridging BroadBand* (B4) a *Address Family Transition Router* (AFTR). První zmíněný prvek se nachází na zákaznické straně tunelu a pro zákaznické počítače funguje jako defaultní brána ze sítě. Jeho činností je balit IPv4 pakety do IPv6. Z toho vyplývá, že musí vlastnit asdresy obou protokolů. Odesílaným IPv6 datagramům se jako cílová adresa zapíše adresa AFTR.

AFTR je NATem ležícím na druhém konci tunelu. AFTR rozbalí příchozí IPv6 datagram a namapuje zdrojovou IPv4 adresu na veřejnou adresu AFTR. Jelikož je často zdrojovou adresou privátní adresa je k namapovanému záznamu ještě uložena IPv6 adresa B4, ze které byly datagramy poslány. Díky tomu je zajištěno správné odeslání odpovědi. Je-li záznam namapován, IPv4 datagram je směrován dále do Internetu. [4, 21]

2.3 Stateless IP/ICMP Translation Algorithm

Stateless IP/ICMP Translation Algorithm neboli SIIT je bezestavový algoritmus, který překládá hlavičky protokolů mezi IPv4 a IPv6. SIIT neuchovává žádné záznamy o překladu, ani žádné informace o stavu. Jsou využita pouze přesně daná pravidla pro překlad. Důsledkem je, že veškeré rozšiřující možnosti protokolů jsou zahazovány. U IPv4 jde hlavně o položku options, která je ignorována. V případě IPv6 jsou zahazovány zase rozšiřující hlavičky.

2.3.1 IPv4 → IPv6

Fragmentace u obou protokolů se řeší rozdílnými způsoby. Existují dva případy, které mohou nastat. Provede-li IPv4 uzel před odesláním paketů MTU discovery (zjištění maximální možné velikosti paketu), komunikace proběhne bez fragmentace. Překladač bez problémů přeloží ICMPv6 odpovědi od IPv6 směrovačů a pošle IPv4 uzlu, který podle toho nastavý MTU hodnotu. Pokud MTU discovery neproběhne, fragmentaci provede sám překladač. Řeší se tím problém u IPv6, kde směrovače nefragmentují pakety. Velikost fragmentů paketů je 1280B, což je minimální velikost MTU u IPv6.

Překladač vytvoří IPv6 hlavičku následujícím způsobem v případě, pokud se nejedná o fragment a byl proveden MTU discovery.

Verze	6
Třída provozu	Je defaultně překopírována z hlavičky IPv4 bit po bitu z pole *Typ služby*.
Značka toku	0
Délka dat	Celková délka z hlavičky IPv4 mínus velikost hlavičky IPv4.
Další hlavička	Jedná-li se o ICMPv4, je hodnota změněna na ICMPv6. Tedy 1 je změněna na 58. Jinak je hodnota zkopírována z IPv4 hlavičky.
Max. skoků	Hodnota z IPv4 hlavičky odečtená o jedničku.
Zdrojová adresa	Nejčastěji se vezme 96bitů dlouhý prefix a ten se doplní IPv4 adresou.
Cílová adresa	Stejný způsob jako u zdrojové adresy

V případě, že jde o fragmentovaný paket, liší se hlavička v položkách *délka dat* a *další hlavička*. *Délka dat* se zvětší o 8 kvůli rozšiřující hlavičce fragmentace. Stejný důvod je u *další hlavičky*, který bude nabývat hodnoty 44.

Pole hlavičky fragmentace budou mít hodnoty.

Další hlavička	Stejné jako IPv6 hlavičky
Posun	Hodnota se zkopíruje z IPv4 hlavičky.
Příznak M	Bit se také překopíruje z IPv4 hlavičky.
Identifikace	Dolních 16 bitů se zkopíruje z IPv4. Zbytek se doplní nulami.

2.3.2 IPv6 → IPv4

Fragmentace v tomto směru nepředstavuje takovou překážku. Pakety z IPv6 sítě přijdou nafragmentované již od odesílatele. Pokud má dojít po překladu na IPv4 k fragmetaci, řeší si to samotné směrovače. Je-li fragmentace zakázana a MTU je větší než přípustná hodnota, odešle překladač odesílateli ICMPv6 chybovou zprávu se správnou velikosti MTU.

Naplnění hodnot hlavičky IPv4 je velice podobné jako v předchozím příkladě. Takto je to v případě nefragmentovaného IPv6 paketu.

Verze	4
Délka hlavičky	5
Typ služby	Zkopírovaný obsah *třídy provozu* z hlavičky IPv6.
Celková délka	Délka dat plus velikost IPv4 hlavičky.
Identifikace	Pouze nuly nebo vygenerované číslo.
Příznaky	MF nastaven na 0 a DF na 0, pokud velikost paketu leží mezi 88 a 1280 bajty.
Posun fragmentu	Vše nuly.
TTL	Hodnota *max. skoků* z hlavičky IPv6 mínus 1.
Protokol	Hodnota 58 (ICMPv6) se změní na 1 (ICMPv4). Hodnoty 0 (HOPOPT), 43 (IPv6-Route) a 60 (IPv6-Opts) jsou přeskočeny, protože nemají pro IPv4 žádný význam. Ostatní jsou zkopírovány.
Kontrolní součet	Vypočítán při vytváření.
Zdrojová adresa	K namapování 128bitové adresy na 32bitovou se používá dynamické mapování podobné tomu u NATů.
Cílová adresa	Posledních 32bitů z cílové IPv6 adresy je cílová adresa pro IPv4.

Když dorazí IPv6 paket obsahující hlavičku fragmentace budou položky IPv4 paketu pozměněny.

Celková délka	Hodnota bude menší o 8 (velikost hlavičky fragmentace).
Identifikace	Zkopírovaných spodních 16 bitů z pole *identifikace* z IPv6 hlavičky fragmentace.
Příznaky	MF je zkopírovan z *M* příznaku fragmentující hlavičky.
Posun	Zkopírován z *posunu* z IPv6 hlavičky fragmentace.
Protokol	Hodnota 58 (ICMPv6) se změní na 1 (ICMPv4). Ostatní rozšiřující hlavičky jsou přeskočeny a hodnota *další hlavičky* je zkopírována z poslední IPv6 hlavičky.

[22, 23]

2.4 Network address translator

Traditional Network Address Translator dále jen NAT je jeden z mechanizmů, který prodloužil IPv4 život. NAT se skládá ze dvou částí basic NAT a NAPT.

Basic NAT je metoda, která mapuje skupinu adres Internetového protokolu na jinou. Mapování je prováděno dynamicky – adresa na adresu (jedna vnitřní adresa na jednu veřejně přístupnou adresu). Má-li basic NAT více veřejných adres, než je uzlů ve vnitřní síti, má každý uzel zajištěnou komunikaci do Internetu. V opačném případě každý uzel tento přístup mít nebude. Je zde také možnost provést statické mapování adresy, tím bude tato adresa zpřístupněna z Internetu.

Vlastnit více veřejných adres je velice drahá záležitost. Většinou firmy musí hospořadit pouze s jednou veřejnou adresou, proto *basic NAT* není pro ně řešením. NAPT (*Network Address Port Translation*) mapování provádí pomocí dvojice takzvaných tuplů, datových struktur. První se skládá z vnitřní IP adresy a portu transportní vrstvy. Druhý má také dvě položky, a to veřejnou IP adresu a přiřazený port transportní vrstvy. Pro přiřazení čísla portu se prochází záznamy dvojic tuplů, které mají stejné vnitřní a veřejné IP adresy. Vybere se následně takové číslo portu, které není v záznamech použito. NAPT tedy umožňuje mapovat vše pouze na jednu veřejnou adresu.

Na NAT může být nahlíženo jako na bezpečnostní mechanizmus, kde prvky za NATem nejsou viditelné zvenčí. Této charakteristiky může být ale zneužito. NAT může fungovat jako anonymizátor a útočník je potom složitě dohledáván.[24]

NAT, jako překladový mechanizmus mezi IPv4 a IPv6, je navržen ve dvou provedeních, jako NAT-PT a nebo NAT64. Oba stavějí na překladači SIIT (kapitola 2.3) a doplňují k němu chybějící součásti.

2.4.1 Network Address Translation - Protocol Translation

Network Address Translation - Protocol Translation (NAT-PT) vychází z NATu popsaného v RFC 3022, ale tentokrát slouží k převodu adres mezi IPv4 a IPv6 protokolem. K převodu mezi protokoly se využívá SIIT algoritmus. NAT-PT se snaží být mechanizmem, který zajistí veškerou komunikaci s IPv4 sítí. Řeší i DNS dotazy. Je nasazen na hraničním směrovači mezi lokální sítí a Internetem. Stejně jako NAT se NAT-PT skládá z obdobných částí, kterými jsou Basic-NAT-PT a NAPT-PT.

Po vzoru basic NATu mapuje Basic-NAT-PT adresy jedna ku jedné s tím rozdílem, že se mapuje IPv6 na IPv4 a opačně. Chce-li uzel v IPv6 síti komunikovat s IPv4 uzlem, nastaví cílovou adresu na PREFIX::IPv4. PREFIX je prefix sítě s maskou /96, která je propagována do IPv6 sítě, jejíž výchozí brána je adresa NAT-PT. IPv4 adresa následující za PREFIXEM je adresou cílového uzlu.

NAPT-PT je zkratkou pro *Network Address Port Translation - Protocol Translation*. Po vzoru NAPT je schopen mapovat provoz jen na jednu adresu za využití protokolů transportní vrtvy. Pro komunikaci s IPv4 světem využívá, stejně jako Basic-NAT-PT, adresu složenou z PREFIXu a IPv4 adresy.

Řešení DNS dotazů je rozděleno na odchozí a příchozí. A dotaz jdoucí na DNS server v IPv6 síti jde přes NAT-PT, kde je pomocí algoritmu přeměněn na AAAA dotaz. Odpověď od DNS server jde rovněž přes NAT-PT, kde je dotaz přeložen zpět na A záznam. Při dotazování v opačném směru, tedy odchozím, můžou nastat dvě situace.

1. DNS server nacházející se v IPv6 síti může znát doménový záznam pro dotaz. Tím pádem dotaz nepůjde přes NAT-PT, ale zůstane v IPv6 síti a odpověď se doručí přímo tazateli. Jinými slovy, dotaz a odpověď nebude měněna z A na AAAA a opačně. Ten následně naváže komunikaci s cílovým uzlem v IPv4 sítí přes NAT-PT.

2. Dotaz musí jít mimo IPv6 síť, protože odpověď zná DNS server v IPv4 sítí. AAAA dotaz jde přes NAT-PT, kde je přeměněn na A dotaz a doručen DNS serveru. Odpověď jde zpět přes NAT-PT, kde je přeměněna zpátky a doručena tazateli. Následná komunikace probíhá stejně jako v prvním případě.

Dnes se již NAT-PT nevyužívá. Autoři v RFC 4966 doporučují IETF, aby NAT-PT byl zakázán, což také bylo nakonec učiněno. V dokumentu autoři problémy rozdělují do tří skupin.

První skupinou jsou problémy spojené s IP adresami. Některé aplikační protokoly vkládají do dat paketů IP adresy. Ty jsou v IPv4 a IPv6 jinak dlouhé takže může dojít k přetečení maximální délky paketu. Problém s tím spjatý je ten, že se mění obsah paketu, což rozbíjí bezpečnostní mechanizmy IPsecu.

Další skupinou jsou problémy, které se přímo netýkají DNS. Jelikož veškerý provoz musí jít přes NAT-PT, jak datový tak i DNS dotazy je to velice náchylné místo na útoky. Při DDOS útoku na NAT-PT bude vyřazena celá IPv6 síť, protože NAT-PT nebude zvládat mapovat tak velké množství a zhroutí se. Tento problém by vyřešilo více překladačů a následně jejich společná synchronizace, to ale nebylo nikdy uspokojivě vyřešeno.

Poslední, třetí skupinou jsou problémy přímo týkající se DNS. Vzhledem k tomu, že se DNS dotazy a odpovědi mění, nelze použít zabezpečení DNSSEC. Tazatel tak neví, jestli adresa kterou dostal od DNS je ta správná nebo je pod útokem *Man in the middle*. [25, 26, 27]

2.4.2 NAT64

Až po 4 letech od zakázání NAT-PT přišel na svět v roce 2011 nový mechanizmus, který ho nahradil. Jelikož největším kamenem úrazu bylo DNS, v novém provedení se překladač stará jen o to, co mu jde nejlíp, překládat. DNS dotazy má na starosti DNS64, který zase řeší pouze DNS dotazy. Fungují na sobě nezávisle (můžou běžet na jiných strojích) jediné, co mají společného

je prefix, ale o tom později. Oba se tedy společně starají o převod a překlad komunikace mezi IPv6 a IPv4 sítí.

Stavový NAT64 je mechanizmus, který překládá IPv6 pakety na IPv4 pakety a zpět. Překlad je prováděn pomocí SIIT algoritmu. Momentální specifikace NAT64 pouze definuje unicast komunikaci. Jinými slovy, podporuje jen protokoly TCP, UDP a ICMP. Multicast, streamovaci protokoly či IPsec vyřešen není.

Navázaní komunikace přes NAT64 může standardně iniciovat jen IPv6 uzel. Je zde ovšem možnost provést nastavení statického mapování a dovolit tak navázat IPv4 uzlům komunikaci s IPv6 stroji. NAT64 musí mít na svém odchozím rozhraní veřejně adresovatelnou IPv4 adresu, kterou po vzoru NAPT umožňuje sdílet pro přístup k Internetu. Aby toto mapování bylo umožněno, překladač si vytváří tabulky, do kterých si překlad ukládá. Pro každý protokol (TCP, UDP a ICMP) si vytváří dvě tabulky – jednu pro mapování zdrojové adresy s cílovou (Binding Information Bases - BIB) a druhou pro mapování vytvořených relací (tou je Session table).

Pro komunikaci uzlů v IPv6 síti s IPv4 stroji se využívá IPv6 prefixu, kde překladač je výchozí branou. Ten může mít jeden ale i více takových prefixů. Jeden takový prefix bude dále označován jako Pref64::/n. Každý Pref64::/n má velikost maximálně 96 bitů a to z toho důvodu, že je za každý jednoduše zřetězena IPv4 adresa. Mám-li Pref64::/n 64:ff9b::/96 bude za něj IPv4 adresa 147.32.232.248 zřetězena jako 64:ff9b::147:32:232:248. Díky tomu potom NAT64 snadno ví, jaká je cílová destinace paketu.

Struktury a počet tabulek, které si NAT64 vytváří pro udržování dat, nemusí jednotlivé implementace dodržovat. Musí ale uchovávat všechny potřebné informace pro překlad. Následný popis je takto dělen kvůli jednoduššímu popsání.

Záznamy do BIB tabulek jsou pro každý protokol (UDP, TCP, ICMP) provedeny stejným způsobem.

$$(X',\mathrm{x}) \longleftrightarrow (\mathrm{T},\mathrm{t})$$

První dvojce náleží zdrojovému stroji. X' je jeho IPv6 adresou a x zase protokolem transportní vrstvy. Druhou dvojici tvoří potom zvolená veřejná IPv4 adresa NAT64 a port tranportní vrstvy. Adresa je označena jako T a port písmenem t.

Pokud je překládán UDP nebo TCP protokol, do tabulek relací se vytváří záznam následujícího formátu.

$$(X',\mathrm{x}),(Y',\mathrm{y}) \longleftrightarrow (\mathrm{T},\mathrm{t}),(\mathrm{Z},\mathrm{z})$$

První dvě dvojce říkají, jaké hodnoty nabýval paket v IPv6 síti. X' je zdrojovou adresou, kdežto Y' je adresou cílovou. Druhé dvě dvojce jsou oproti tomu hodnoty, jaké překladač zapsal do IPv4 paketu po překladu. T je ‚stejně jako u BIB, zvolenou veřejnou IPv4 adresou NAT64. Adresu cílového stroje

patří označení Z. Malá písmena x,y,t a z označují porty, kde x a t jsou porty zdrojovými. Zbylé dva porty y a z označují zase cílové porty.

Při překladu ICMP protokolu se do tabulek relací nezapisují záznamy stejného formátu, jako u UDP a TCP z toho důvodu, že ICMP je protokol síťové vrstvy. Nezapouzdřuje oproti UDP a TCP další aplikační vrtvu, ale přenáší rovnou data spjaté s ICMP protokolem. Jde o různé zprávy informující například o důvodu chyby. Proto záznamy v tabulce relací vypadají následovně.

$$(X',Y',\text{i1}) \longleftrightarrow (\text{T,Z,i2})$$

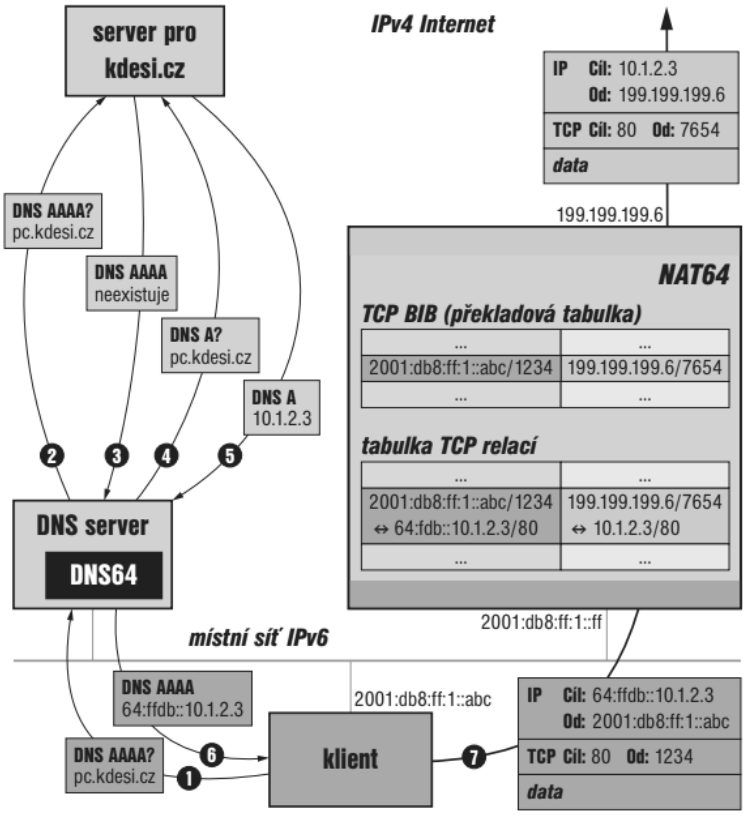
První trojce obsahuje informace z IPv6 sítě. X' je opět zdrojovou IPv6 adresou a Y' adresou cílovou. Označení i1 je identifikátorem ICMPv6 paketu. Určuje, zda se jedná o ping, chybovou zprávu nebo jiné. T je jednou z veřejných IPv4 adres NAT64 a Z IPv4 adresou cílového stroje. Identifikátor ICMPv4 zde potom zastupuje i2.

Přijde-li na NAT64 paket, překladač se koukne do BIB tabulky, jestli už nemá pro toto spojení vytvořený záznam. Nemá-li, vytvoří ho, a poté zapíše nový záznam v relační tabulce. Pokud by záznam v BIB již existoval, vytvoří se záznam pouze v relační tabulce. Po skončení relace je záznam smazán. Nenachází-li se žádný záznam v relační tabulce, který vychází z BIB záznamu, je smazán i ten v BIB. [4, 28, 23]

Společně s překladačem vyšel samostatný RFC dokument o implementaci DNS nesoucí jméno DNS64. Narozdíl od NAT-PT nemusí být DNS64 na stejném stroji, není totiž navržen jako součást NAT64. Je zde možnost mít více redundantních DNS64 serverů, takže zátěž může být rozložena. Uzel v IPv6 síti odešle na DNS64 AAAA dotaz. Ten ho předá dál a pokud mu přijde kladná AAAA odpověd, přepošle ji tazateli. Ten tak může komunikovat s cílovým zařízením přímo po IPv6. V případě, že přijde odpověď záporná DNS64, odešle dotaz na stejné jméno, ale tentokrát pro A záznam. Příchozí odpověď pak převede na AAAA záznam tím, že za Pref64::/n příslušného NAT64 připojí IPv4 adresu z obdržené odpovědi. Následně ji předá tazateli, který potom komunikuje s cílovým uzlem pomocí překladače.

Ověřování DNSSECem přímo na DNS64 není problémem, protože A i AAAA dotazy nejsou měněny při přenosu. Problém nastane pokud validace probíhá na lokálním stroji. V případě, že existuje jen A záznam, DNS64 tento záznam překládá na AAAA záznam a to je problémem pro DNSSEC. Pokud chce uživatel validovat lokálně, musí si v takovém případě zprovoznit i DNS64. [4, 29]

Jak probíhá proces komunikace a činnosti NAT64 a DNS64 nejlépe ilustruje obrázek 2.2.



Obrázek 2.2: Zahájení komunikace s použitím NAT64 (převzato z [4])

Kapitola 3

Implementace a konfigurace NAT64

Implementace lze dohledat v hardwarovém i v softwarovém provedení. Softwarové implementace pak mohou být ve formě aplikace nebo v podobě modulu linuxového jádra. Autor práce vybral od každé možnosti jednu a ty poté vystavil stejným testům.

3.1 Tayga

Tayga je softwarovou implementací NAT64 určenou pro operační systém Linux. Implementace je provedena ve formě aplikace a není tedy nutné zasahovat jakkoliv do jádra operačního systému. Tayga vytváří rozhraní typu TUN (síťový tunel), pomocí kterého si vyměňuje pakety s jádrem systému.

Tayga je primárně navržena jako bezestavová implementace, ale v případě potřeby může být provozována stavově. U bezestavové implementace nedochází k přepisování portu transportní vrstvy a sledování relací. Jsou zde stejně jako u basic NATu adresy mapovány 1:1 s tím rozdílem, že je překládána IPv6 adresa na IPv4 adresu. Při nedostatečného počtu IPv4 adres Taygu použijeme jako stavový NAT64 za pomoci tradičního NATu.

Tayga pro fungování potřebuje dva síťové rozsahy. První se využívá jako Pref64::/n. Jde tedy o IPv6 rozsah pomocí kterého stroje v síti komunikuji s IPv4 zařízeními. Druhý je rozsah, na který se IPv6 adresy překládají. Velikost tohoto rozsahu se odvíjí od počtu uzlů v IPv6 síti, které potřebují komunikovat s IPv4 Internetem. Místo, kde nejpravděpodobněji chcete provozovat NAT64, je na hraničním směrovači. To znamená, že potřebujete poměrně velký rozsah veřejných IPv4 adres. Vzhledem k tomu, že vlastnit jednu veřejnou IPv4 adresu je vzácnost, využití Taygy ve stavovém provedení je častější variantou. Na stroji s překladačem se to vyřeší pomocí dynamického překladu. Vytvoří se NAT, který bude překládat již přeložené IPv4 adresy na jednu veřejnou IPv4

adresu. Na linuxovém stroji lze toho dosáhnout pomocí příkazu iptables a maškarády. Příkaz přesněji vypadá takto:

iptables -t nat -A POSTROUTING -o <rozhraní> -j MASQUERADE

Tím se nastaví tradiční NAT, kde uvedené rozhraní vlastní veřejně adresovatelnou adresu.

Tayga umožňuje dle RFC 6146 mapování z IPv4 do IPv6. Mapování se nastavuje v jediném konfiguračním souboru, který Tayga potřebuje ke svému běhu. Ten najdeme v případě instalace ze zdrojových sobourů v `/usr/local/etc`, kde by měl být uložen pod jménem `tayga.conf`. Nové linuxové operační systémy mají již ve svých repozitářích balíčky Taygy, takže je možné provést instalaci pomocí instalátoru balíčků. V takovém případě se konfigurační soubor nachází přímo v `/etc` pod jménem `tayga.conf`.

Aby bylo potom IPv6 zařízení dostupné i z IPv4 sítě, povolí se statické mapování v konfiguračním souboru pomocí direktivy `map`. Jako první argument očekává IPv4 adresu z IPv4 rozsahu přidělený Tayze. Druhým argumentem je IPv6 adresa uzlu, který má být přístupný.

Kromě statického mapování v konfiguračním souboru nastavujeme, jak se bude jmenovat TUN rozhraní Taygy, které bude obstarávat překlad. Také tam volíme IPv4 adresu Taygy (volitelně i IPv6), prefix (Pref64::/n), uložiště souboru pro dynamický překlad či IPv4 rozsah. Po nastavení direktiv v konfiguračním souboru je potřeba vytvořit TUN rozhraní a přiřadit mu jak IPv4 tak i IPv6 adresu. Protože na směrovač bude směřován provoz s Pref64::/n, potřebuje vědět, co s takovým prefixem má udělat. Proto je potřeba přidat cesty směřující na TUN rozhraní. To samé je potřeba udělat i u IPv4 rozsahu. Posledním krokem je zapnutí překladu. K tomu slouží příkaz `tayga`. Více možností příkazu je tradičně možné zjistit z manuálových stránek. [30]

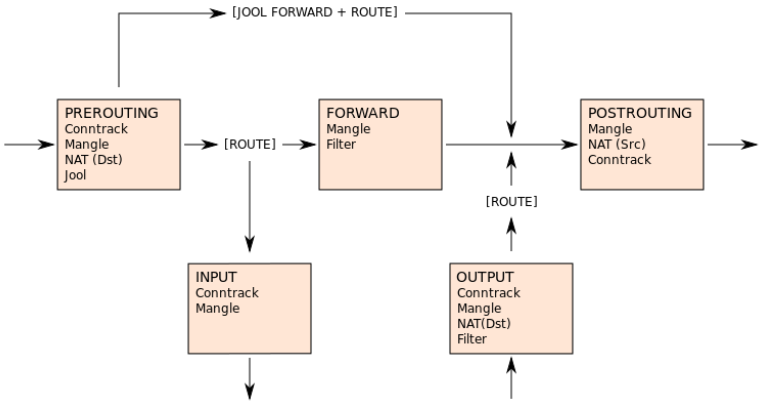
Jedná se o velice snadno konfigurovatelnou a nasaditelnou aplikaci. K nasazení není třeba doinstalovávat žádné další balíčky. Tayga je závislá jenom na balíčku libc6, který obsahuje standardní knihovny používané téměř všemi programy v systému.

3.2 Jool

Jool se řadí mezi softwarové open-sourcové implementace překladače NAT64. Implementace je určená pro Linux ve formě přídavného modulu do jádra systému. Jool podporuje linuxová jádra od verze 3.2.0 a vyšší, takže starší linuxové operační systémy nejsou s tímto řešením kompatibilní. Nejnovější jádro, na kterém Jool byl testován dle dokumentace, je 4.12.0-041200-generic.

Přesněji se jedná o Netfilter module, který je uchycen na prerouting řetěz. Jelikož Netfilter samotný není vhodný pro změnu paketů protokolů nacházející se na třetí vrstvě ISO/OSI modelu, proto Jool má svoji vlastní forwarding

pipeline. Jak je tato pipeline zakomponována do procesu filtrů a adresace popisuje obrázek 3.1.



Obrázek 3.1: Uchycení Jool na prerouting řetěz (převzato z [6])

Implementace Joolu se skládá ze dvou modulů jádra, dvou uživatelských aplikací a jednoho uživatelského démona. Jeden modul jádra je implementace SIIT a druhý je stavový NAT64. Ke každému modulu náleží právě jedna uživatelská aplikace, které slouží k jejich konfiguraci. Uživatelský démon se používá k synchronizaci relací mezi moduly SIIT a NAT64.

Instalace modulů a aplikací probíhá vzlášť, kde první se instalují moduly. Instalace se distribuce od distribuce liší pouze použitím jiného baličkovacího instalátoru a instalovaných balíčků. Nejdříve je potřeba nainstalovat základy v podobě C a C++ kompilátoru s utilitou Make. Poté je potřeba doinstalovat závislosti, které moduly pro svůj běh potřebují příkazem `apt-get install linux-headers-$(uname -r)`. Posledním krokem před instalací Joolu je nainstalovaní frameworku DKMS pro ovládání modulů jádra, který se nachází ve stejnojmenném balíčku. Nyní stačí jen stáhnout nejnovější verzi Joolu rozbalit a nainstalovat pomocí DKMS.

Uživatelské aplikace také potřebují doinstalovat pár balíčků před jejich instalací. Balíček `pkg-config` je jedním z nich. Slouží jako správce knihovních kompilací a propojení příznaků, které pracují s programy automake a autoconf. Ty je také potřeba doinstalovat. Oba se nacházejí v balíčku `autoconf`, který pomáhá s automatickou konfigurací. Balíček `libnl-genl-3-dev` je také potřeba nainstalovat. Jde o knihovnu pro aplikace zabývající se síťovou komunikací mezi jádrem a uživatelskou aplikací. Po nainstalování všech prerekvizit je potřeba přejít do rozbalené implementace, přesněji do `Jool-<version>/usr`. Tam provést trojici příkazů v pořadí `./configure`, `make` a `make install`, což nainstaluje uživatelské aplikace.

Po dokončení instalací stačí modul NAT64 přidat do jádra a nastavit překladači Pref64::/n. K tomu postačí jeden příkaz `modprobe -first-time jool pool6=<Pref64::/n>`. [6]

3.3 CISCO

Velkou výhodou tohoto řešení je, že není potřeba nic instalovat. Jediné, co je potřeba mít příslušný směrovač, který NAT64 umí. Stačí si vybrat řešení překladače a následně nakonfigurovat. Je možné nakonfigurovat statický stavový, dynamický stavový či překladač s dynamickými porty. K zprovoznění překladu potom potřebujeme pár jednoduchých kroků.

Prvním krokem je povolení přesměrovávání (forwarding) IPv6 unicastových datagramů. Rozhraní, které bude sloužit pro kontakt s IPv6 sítí, povolit, nastavit mu IPv6 adresu a umožnit mu překlad. U rozhraní jdoucího do Internetu je potřeba udělat to samé s tím rozdílem, že mu bude přidělena IPv4 adresa. Po nastavení rozhraní vytvoříme IPv6 *access list*, ve kterém povolíme Pref64::/n. Dalším krokem nastavíme NAT64, jaký Pref64::/n má používat. Následně je potřeba určit překladači, na jaké IPv4 adresy může překládat. Posledním krokem před uložením konfigurace je přiřadit překladači *access list* společně s IPv4 adresami. [31, 32]

Kapitola 4

Testování vybraných implementací

Testování vybraných implementací probíhalo v síťové laboratoři na Fakultě informačních technologií ČVUT, jenž je určena pro výuku síťových předmětů či provádění podobných testů. Testy byly prováděny na místních počítačích, které mají 4GB RAM a jako procesor jim slouží Intel Celeron CPU G540. Tyto počítači sloužily jako NAT64 a cílový server. Autor také využil místního Cisco směrovače 2901, který byl otestován jako NAT64. Dále byl jako klient použit notebook HP Probook G2 s 4GB RAM, procesorem i3-5010U a gigabitovým ethernetem.

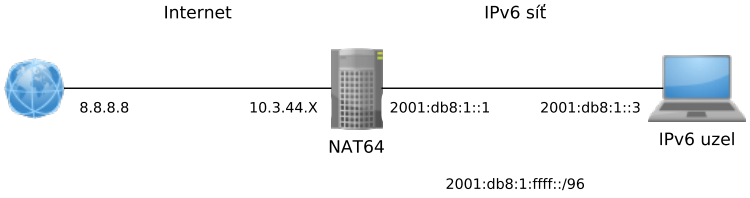
Jako linuxový operační systém byla zvolena distribuce Debian a to ve třech jejích různých verzích s kódovými označeními Wheezy, Jessie a Stretch. Důvodem, proč bylo zvoleno více verzí, bylo prozkoumání vlivu jádra operačního systému na linuxové implementace. Tabulka 4.1 poté popisuje jádra jednotlivých verzí. Ta byla v dostatečné verzi, aby na každém operačním systému byly nasazeny obě implementace překladače.

Konfigurace sítě při testech měla dvě podoby. První (diagram 4.1) ukazuje, jak byla síť nastavena pro testy latence. Druhou podobu potom vykresluje diagram 4.2, který posloužil pro testy ztrátovosti, rychlosti přenosů a propustnosti.

Testy probíhaly za pomocí dvou programů, kterými byly ping a Iperf. Nástroj ping inspirovaný sonary a echolokací vznikl již v osmdesátých letech 20.

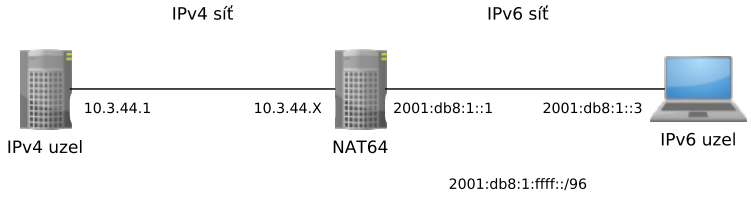
Kódové označení	**Linuxové jádro**
Wheezy	3.2.04
Jessie	3.16.0
Stretch	4.9.0

Tabulka 4.1: Jádra jednotlivých verzí distribuce Debian



Obrázek 4.1: Konfigurace sítě - latence

století.[33] Tento nástroj vytváří echo request datagramy protokolu ICMP a očekává echo request datagramy. Jinými slovy, zkoumá dostupnost cílového uzlu. Program Iperf je oproti pingu složitější. Funguje na architektuře klient/server, kdy klient nastaví test, jaký chce provést, a poté naváže komunikaci se serverem. Na serveru stačí tuto aplikaci pouze pustit příkazem `iperf3 -s`.



Obrázek 4.2: Konfigurace sítě - ztrátovost, rychlost přenosů, propustnost

4.1 Funkčnost

Všechny tři implementace jsou funkční za dodržení potřebných prerekvizit. Jool potřebuje linuxové jádro 3.2.0 a vyšší, Tayga 2.4 a výšší a NAT64 na Ciscu potřebuje jeho podporu.

Jediné řešení, které výrazně zasahuje do standardního zpracování paketů, je Jool. Jool obchází tabulku forward zabudovanou v jádře procesoru (obr. 3.1) a tím i nastavená pravidla nástrojem `iptables`. To znamená, že pokud má být provoz filtrován, pravidla musí být umístěna na preroutingu nebo postroutingu. Jool mimo jiné obchází i zakázání přeposílání paketů nastavené v jádře. V operačním systému se systemd to jsou nastaveni `net.ipv4.ip_forward` a `net.ipv6.conf.all.forwarding`.

Ohledně konfigurovatelnosti je na tom Jool velmi dobře. Nabízí velkou škálu nastavení či vypsání jednolivých tabulek. Je možno vidět jak BIB zá-

znamy, tak i navázané relace definované podle RFC. To v Tayze je možno vidět akorát tabulku BIB, která je uložena na disku ve formě souboru. Umístění tohoto souboro je specifikováno v konfiguračním souboru. U překladače na Cisco směrovači je situace podobná Joolu. Umožňuje, jak je u Cisca zařízeních obvyklé, velké množství konfigurovatelnosti a výpisů.

Linuxové implementace jsou čistě překladači IP protokolu a neumějí překládat DNS dotazy. Mají-li být dostupné i stroje pouze s IPv6 pomocí DNS, musí být nainstalován a nakonfigurován DNS64. Obě řešení Jool i Tayga podporují Bind, který je i plnohodnotným DNS serverem. Cisco má v sobě zabudovaný DNS-ALG, který umí překládat DNS dotazy, ale neumožňuje DNSSEC. Chceme-li i u Cisca validovat dotazy je potřeba nainstalovat DNS64. Tím může být například již výše zmíněný Bind.

4.2 Ztrátovost

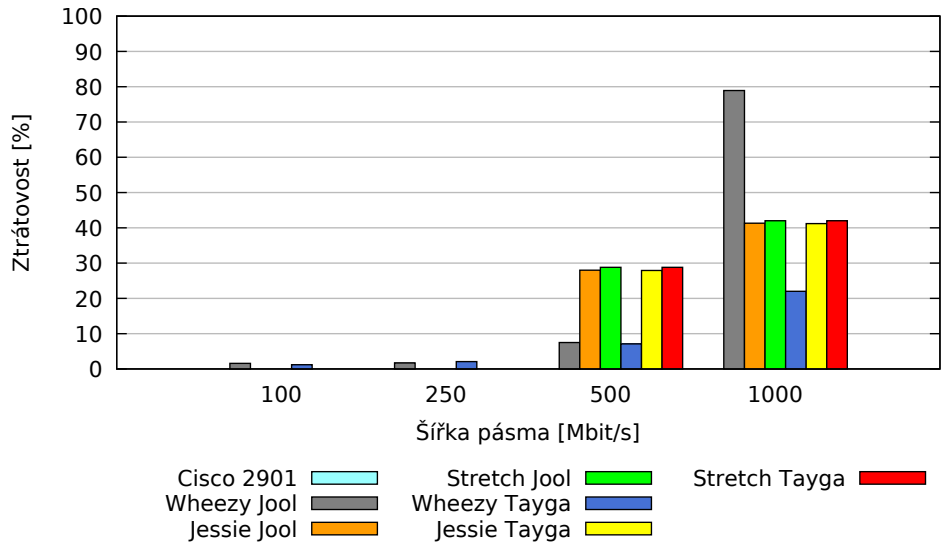
Ztrátovostí je myšlen počet ztracených paketů během přenosu, pakety tedy nedorazí do cílové destinace. Ztráta paketů může být způsobena i problémy přenosu samotného, což se stává například u beztrátových sítí jako je Wi-Fi. Wi-Fi používá metodu CSMA/CA (Carrier Sense Multiple Access/with Collision Avoidance), která se snaží kolizím a tedy i potencionální ztrátám vyhýbat. To ale neznamená, že k nějakým ztrátám nedojde. Vzhledem k tomu, že konfigurace sítě byla pouze po gigabitovém ethernetu, ztráty vzniklé při testech byly způsobené pouze překladačem.

Testy byly prováděny programem Iperf. Ty probíhaly po protokolu UDP, který nepotvrzuje doručení a tedy ideální pro otestování kolik paketů bude zahozeno. Chování překladačů se ke zvolené šířce pásma může lišit, proto byly provodeny testy i s rozdílnou šířkou pásma. Testy byly provedeny pro 100, 250, 500, 1000 Mbit/sec a každý trval 60 sekund. Klientský příkaz programu Iperf vypadal následovně:

```
iperf3 -c <Pref64::/n>::10.3.44.1 -u -t 60 -b <šířka pásma>m
```

Procentuální ztrátovost popsaná grafem 4.3 ukazuje několik zajímavostí. První je, že ztrátovost řešení u Cisca je nulová. Testy byly opakovány pro podezření na chybu měření, ale i výsledek byl stále stejný. Druhá věc, která nastala při testu se šířkou pásma 1000 Mbit/s byla, že distrubuce Wheezy s implementací Jool zamrzla a PC přestal odpovídat a byl neovladatelný. To je také důvodem velké ztrátovosti, kterou v grafu je možné pozorovat. Dále z grafu vyplývá, že Tayga na distribuci Wheezy má nejmenší ztrátovost při šířce pásma 1000 Mbit/s a to i přesto, že ztratí nepatrné procento paketů při menší šířce, kde ostatní testovaní mají nulové ztráty.

Zdálo by se, že překladač na směrovači Cisco je jednoznačně nejlepší, co se týká ztrátovosti. Tento dojem ale vyvrací graf 4.4 ukazující počty správně



Obrázek 4.3: Procentuální ztráta paketů

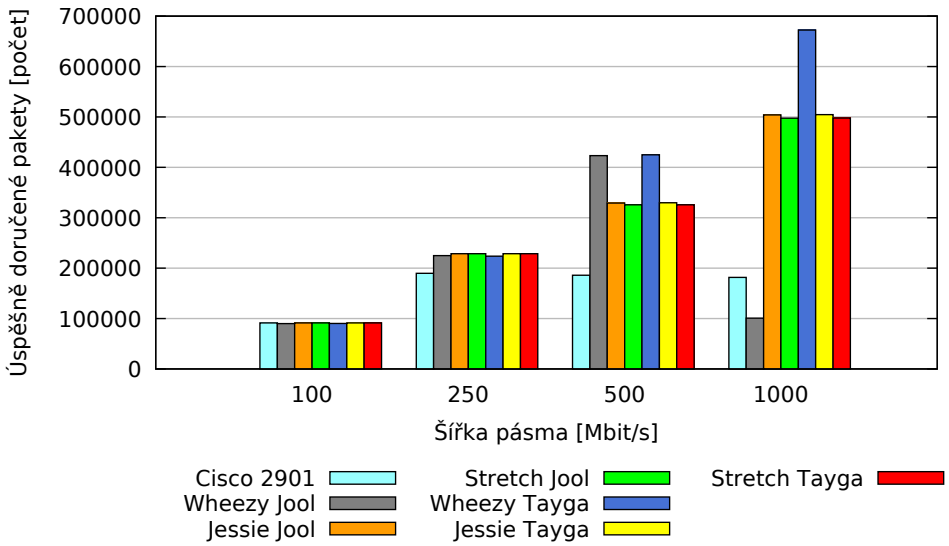
doručených paketů. Správně doručené pakety jsou takové, které se během přenosu neztratily, nebyly zahozeny a dorazily ve správném pořadí. Cisco při šířce pásma 1000 Mbit/s odeslalo zhruba stejně paketů jako když šířka pásma byla 250 nebo 500 Mbit/s. V těchto třech případech se jednalo o circa 180 tísíc paketů, což je nejméně ze všech. Není brán v potaz test implementace Jool na distibuci Wheezy, kde zařízení přestalo odpovídat, což reflektuje i tento graf. Ostatní testovaní odeslaly paketů rovnoceně při šířce 250 Mbit/s přes 220 tisíc paketů , při 500 Mbit/s 320 ticíc paketů a při 1000 Mbit/s úspěšně dorazilo 500 ticíc paketů. U Cisca stejný počet paketů při různé šířce autor nedokáže vysvětlit důvod. Zařízení bylo konfigurováno po načtení základní konfigurace a při pročítání dokumentace, kterou společnost Cisco poskytuje online, nenašel, jak je překladač implementován.

4.3 Latence

Latence nebo-li odezva se měřila pomocí příkazu ping pro IPv6 protokol. Jako cílová destinace byla adresa `8.8.8.8`, která patří společnosti Google. Z klienta nacházejícího se v IPv6 síti bylo odesláno 10000 paketů s intervalem 0.01 sekundy. Příkaz tedy přesně vypadl takto:

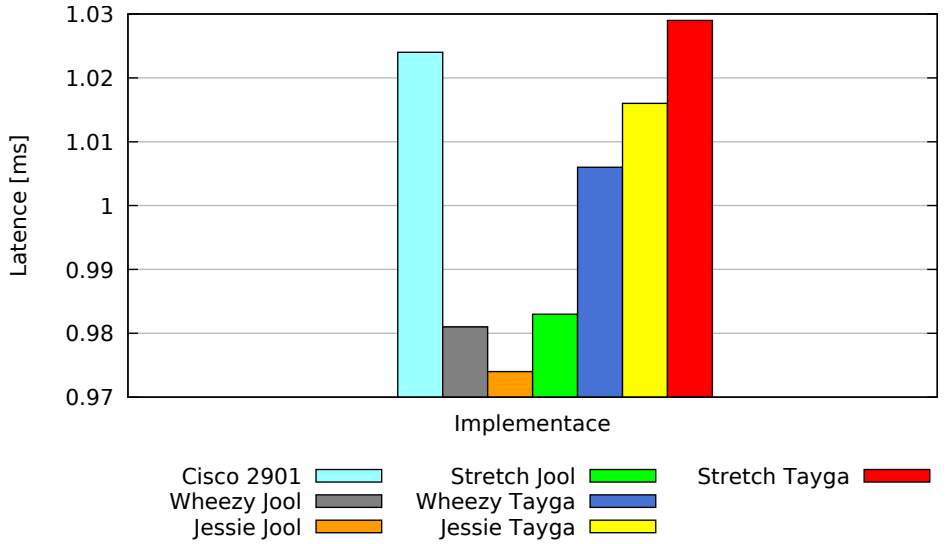
```
ping6 -c 10000 -i 0.01 <Pref64::/n>::8.8.8.8
```

Graf 4.5 ukazuje průměrnou hodnotu latence. Ačkoliv je rozdíl řádově v setinách milisekund, nejlépe z testu vyšla implementace Jool skrze všechny dis-



Obrázek 4.4: Úspěšně přenesené pakety

tribuce.



Obrázek 4.5: Průměrná latence

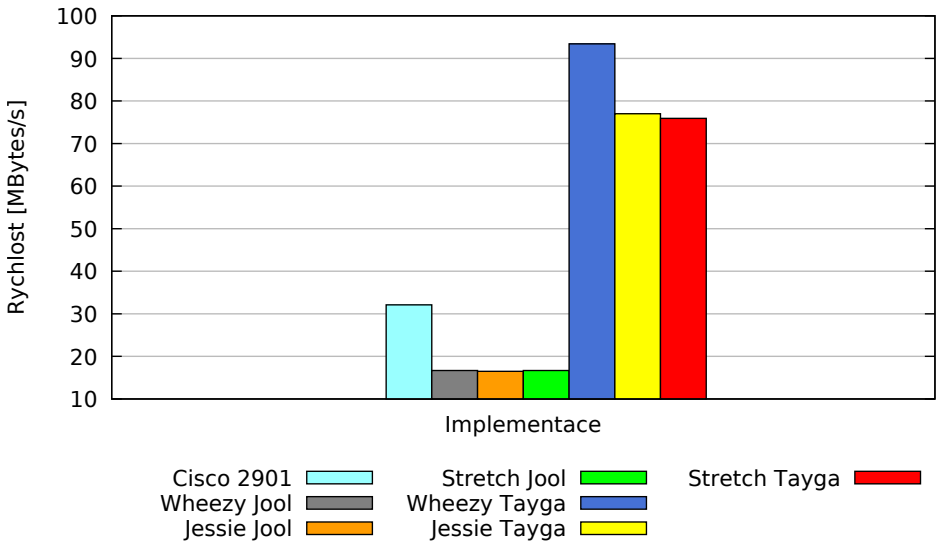
4.4 Rychlost přenosů

Rychlost přenosu označuje kolik dat se za časovou jednotku průměrně podaří přenést mezi dvěma body.

K testu opět posloužil program Iperf a jako síťová konfigurace byla použita ta s lokální serverem (diagram 4.2). Autor zjišťoval, kolik dat bude přeneseno za 60 sekund. Interval mezi odesláním paketů byla nastavená na 1 sekundu. Situace tedy podobná stahování či nahrávání dat na Internetu. Příkaz byl zadán v této podobě:

```
iperf3 -c <Pref64::/n>::10.3.44.1 -p 5201 -i 1 -t 60
```

Tento test vyšel ve prospěch Taygy. V každé distribuci rychlost přenosu přenosu překročila hodnotu 70 MBajtů/s, což lze vidět v grafu 4.6. Cisco překročilo 30 MBajtů/s a Jool ve všech třech distribucích dosáhl pouze 16 MBajtů/s, což je zhruba 4krát méně než Tayga. Jako nejlepší distribuce pro Taygu potom pozoruhodně vyšla nejstarší z nich a to Wheezy. Ta průměrně pakety přenášela rychlostí 90 MBajtů/s.



Obrázek 4.6: Rychlost přenosů

4.5 Propustnost

Propustnost velmi úsce souvisí se šířkou pásma, která určuje fyzickou velikost spoje. Propustnost je skutečná velikost takového spoje. Použití analogie s dálnicí je šířkou pásma možný počet aut který daným úsekem může projet.

Propustnost je potom skutečný počet aut, který tím úsekem projede i přes veškeré zúžení a nehody.

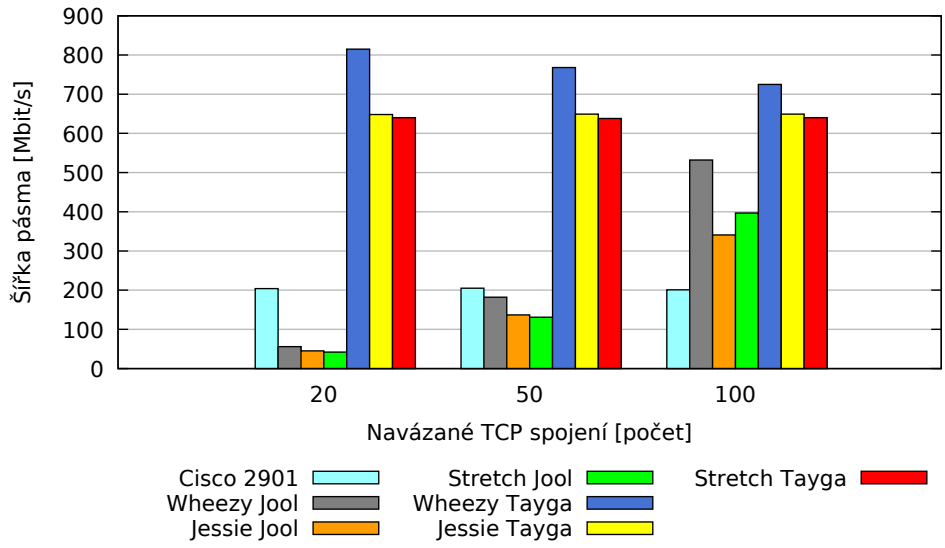
Test probíhal na síťové konfiguraci popsaný diagramem 4.2 a za použití programu Iperf. Bylo otestováno více paralelních TCP spojení zároveň konkrétně 20, 50 a 100. Příkaz, který byl použit, vypadal takto:

```
iperf3 -c <Pref64::/n>::10.3.44.1 -P <počet spojení>
```

Výsledky lze vidět v grafu 4.7. Ty tak trochu obkreslují výsledky u rychlosti přenosu, což je pochopitelné. Test měření rychlosti paketů ukazuje výsledky pouze jednoho navázaného spojení.

Cisco podle výsledků rozděluje šířku pásma mezi všechny spojení stejným dílem. U 20 paralelních spojení je šířka pásma jednoho spoje 10 Mbitů/s u 100 spojů to činní zase 2 Mbitů/s. Stejným způsobem funguje i Tayga. Ta má ovšem šířku pásma mnohem vyšší. U distribuce Jessie a Stretch je šířka pásma zhruba stejná a přesahuje 600 Mbitů/s, což jedno spojení při navázání 20 paralelních spojení má šířku pásma 30 Mbitů/s a 1 spojení při 100 je 6 Mbitů/s. Distribuce Wheezy měla u 20 spojů nejvíce ze všech a to 800 Mbitů/s. S navyšováním spojů se velikost pásma potom blížila ke stejným hodnotám jako měly distribuce Jessie a Stretch.

Opačným způsobem na to jde Jool. Ten každému spojení přidělí šířku pásma velikou circa 3,5 MBitů/s a celková velikost je pak odvozená od počtu navázaných spojení. Zvyšování celkové šířky pásma lze vidět v grafu.



Obrázek 4.7: Šířka pásma

4.6 Shrnutí testů

Každé řešení se vždy minimálně v jednom testu vyjímalo nad ostatními. Ať Cisco u ztrátovosti, kdy neztratilo v podstatě ani jeden paket či Jool u testu latence, kde dosahovalo jednoznačně nejnižších hodnot. Má-li být vybrána na základě těchto testů jedna implementace jako nejlepší, byla by to Tayga. V propustnosti a rychlosti přenosů byla jednoznačně nejlepší implementací a v ostatních testech nijak nezaostala. Samozřejmě také záleží do jaké sítě a s jakým provozem bude implementace nasazována. Pokud je kladen důraz na to, aby v síti nedocházelo ke ztrátám paketů, je nejvhodnější Cisco.

Závěr

V práci byly nejdříve popsány internetové protokoly IPv4 a IPv6 a přechodové mechanizmy mezi nimi. Autor vybral a popsal tři různé implementace překladače NAT64, který je jedním z přechodových mechanizmů. Jako implementace byly vybrány linuxová aplikace Tayga, modul linuxového jádra Jool a řešení překladače společnosti Cisco. Tyto tři řešení byly nakonfigurovány a řádně otestovány stanovenou metologií. Provedené testy zaměřené na funkčnost, ztrátovost, latenci, rychlost přenosů a propustnost byly následně analyzovány.

Autor se domnívá, že se mu práce vydařila, neboť výsledky měření mají vypovídající hodnotu o jednotlivých implementacích a jejich důrazu na vlastnosti překladače. Z testů vyplývá, že Cisco je zaměřené na ztrátovost, Jool na rychlost překladu a Tayga zase na propustnost.

Práce jistě usnadnila administrátorům rozhodnutí, jaká z těchto tří implementací je pro ně nejvhodnější při nasazování IPv6 only sítě. Sám autor se problematikou IPv6 a jejího rozšíření chce dál zabývat. Rád by podnítil a pomohl zavedení IPv6 v Národní technické knihovně.

Literatura

[1] Ali, F.: IP Spoofing. Červen 2011, [Online; Navštíveno: 5. 4. 2018]. Dostupné z: https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-38/104-ip-spoofing.html

[2] Google: Statistics. [Online; Navštíveno: 16. 4. 2018]. Dostupné z: https://www.google.com/intl/en/ipv6/statistics.html

[3] CZ.NIC: IPv6 DNS dotazy. [Online; Navštíveno: 16. 4. 2018]. Dostupné z: https://stats.nic.cz/stats/ipv6_queries/

[4] Satrapa, P.: *Internetový průvodce verze 6*. CZ.NIC, z. s. p. o., třetí vydání, 2011, ISBN 978-80-904248-4-5.

[5] Cisco: IPv6 Extension Headers Review and Considerations. Říjen 2006, [Online; Navštíveno: 12. 4. 2018]. Dostupné z: https://www.cisco.com/en/US/technologies/tk648/tk872/technologies_white_paper0900aecd8054d37d.html

[6] de Monterrey, T.: Documentation. [Online; Navštíveno 16. 4. 2018]. Dostupné z: https://www.jool.mx/en/documentation.html

[7] Osterloh, H.: *TCP/IP kompletní průvodce*. Softpress, první vydání, 2003, ISBN 80-86497-34-8.

[8] Postel, J.: Internet Protocol. STD 5, RFC Editor, September 1981, http://www.rfc-editor.org/rfc/rfc791.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc791.txt

[9] Cerf, V.; Dalal, Y.; Sunshine, C.: Specification of Internet Transmission Control Program. RFC 675, RFC Editor, December 1974, http://www.rfc-editor.org/rfc/rfc675.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc675.txt

[10] Postel, J.: DoD standard Internet Protocol. RFC 760, RFC Editor, January 1980.

[11] Postel, J.: User Datagram Protocol. STD 6, RFC Editor, August 1980, http://www.rfc-editor.org/rfc/rfc768.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc768.txt

[12] Cotton, M.; Vegoda, L.: Special Use IPv4 Addresses. RFC 5735, RFC Editor, January 2010.

[13] Pummill, T.; Manning, B.: Variable Length Subnet Table For IPv4. RFC 1878, RFC Editor, December 1995.

[14] Rekhter, Y.; Moskowitz, R. G.; Karrenberg, D.; aj.: Address Allocation for Private Internets. BCP 5, RFC Editor, February 1996, http://www.rfc-editor.org/rfc/rfc1918.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc1918.txt

[15] Hain, T.: Architectural Implications of NAT. RFC 2993, RFC Editor, November 2000.

[16] Egevang, K. B.; Francis, P.: The IP Network Address Translator (NAT). RFC 1631, RFC Editor, May 1994, http://www.rfc-editor.org/rfc/rfc1631.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc1631.txt

[17] Průša, J.: Jak jsou na tom weby českých úřadů s DNSSEC a IPv6. Srpen 2015, [Online; Navštíveno: 10. 4. 2018]. Dostupné z: https://www.lupa.cz/clanky/jak-jsou-na-tom-weby-ceskych-uradu-s-dnssec-a-ipv6/

[18] Deering, S. E.; Hinden, R. M.: Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, RFC Editor, December 1998, http://www.rfc-editor.org/rfc/rfc2460.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc2460.txt

[19] Conta, A.; Deering, S.: Generic Packet Tunneling in IPv6 Specification. RFC 2473, RFC Editor, December 1998, http://www.rfc-editor.org/rfc/rfc2473.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc2473.txt

[20] Huston, G.: Testing Teredo. Duben 2011, [Online; Navštíveno: 3. 4. 2018]. Dostupné z: http://www.potaroo.net/ispcol/2011-04/teredo.html

[21] Durand, A.; Droms, R.; Woodyatt, J.; aj.: Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion. RFC 6333, RFC Editor, August 2011, http://www.rfc-editor.org/rfc/rfc6333.txt. Dostupné z: http://www.rfc-editor.org/rfc/rfc6333.txt

[22] Bao, C.; Li, X.; Baker, F.; aj.: IP/ICMP Translation Algorithm. RFC 7915, RFC Editor, June 2016.

[23] Bao, C.; Huitema, C.; Bagnulo, M.; aj.: IPv6 Addressing of IPv4/IPv6 Translators. RFC 6052, RFC Editor, October 2010.

[24] Srisuresh, P.; Egevang, K.: Traditional IP Network Address Translator (Traditional NAT). RFC 3022, RFC Editor, January 2001.

[25] Tsirtsis, G.; Srisuresh, P.: Network Address Translation - Protocol Translation (NAT-PT). RFC 2766, RFC Editor, February 2000.

[26] Aoun, C.; Davies, E.: Reasons to Move the Network Address Translator - Protocol Translator (NAT-PT) to Historic Status. RFC 4966, RFC Editor, July 2007.

[27] Satrapa, P.: Tak nám zabili NAT-PT. Říjen 2007, [Online; Navštíveno 13. 4. 2018]. Dostupné z: https://www.lupa.cz/clanky/tak-nam-zabili-nat-pt/

[28] Bagnulo, M.; Matthews, P.; van Beijnum, I.: Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers. RFC 6146, RFC Editor, April 2011.

[29] Bagnulo, M.; Sullivan, A.; Matthews, P.; aj.: DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers. RFC 6147, RFC Editor, April 2011.

[30] Lutchansky, N.: TAYGA Simple, no-fuss NAT64 for Linux. Prosinec 2010, [Online; Navštíveno 23. 4. 2018]. Dostupné z: http://www.litech.org/tayga/

[31] Cisco: Stateful Network Address Translation 64. Leden 2018, [Online; Navštíveno: 20. 4. 2018]. Dostupné z: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ipaddr_nat/configuration/15-mt/nat-15-mt-book/iadnat64-stateful.html

[32] Cisco: NAT64—Stateless versus Stateful. Červen 2011, [Online; Navštíveno: 20. 4. 2018]. Dostupné z: https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/enterprise-ipv6-solution/white_paper_c11-676277.html

[33] Muuss, M.: The Story of the PING Program. [Online; Navštíveno: 16. 4. 2018]. Dostupné z: http://ftp.arl.army.mil/mike/ping.html

Příloha A

Seznam použitých zkratek

ARP Address Resolution Protocol

NAT Network Address Translation

ICMP Internet Control Message Protocol

ICMPv6 Internet Control Message Protocol version 6

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

ISO/OSI International Standards Organization Open Systems Interconnection

TCP/IP Transmission Control Protocol/Internet Protocol

TCP Transmission Control Protocol

UDP User Datagram Protocol

RFC Request For Comment

CIDR Classless Inter-Domain Routing

MTU Maximum Transmission Unit

MF More Fragments

DF Don't Fragment

TTL Time To Live

CSMA/CA Carrier Sense Multiple Access/with Collision Avoidance

PC Personal Computer

VPN Virtual Private Network

HTTP Hypertext Transfer Protocol

SIIT Stateless IP/ICMP Translation

DNS Domain Name System

DNSSEC Domain Name System Security Extensions

RAM Random Access Memory

DKMS Dynamic Kernel Module Support

BIB Binding Information Bases

DDOS Distributed Denial Of Service

AFTR Addres Family Transition Router

B4 Basic Bridging BroadBand

DHCP Dynamic Host Configuration Protocol

ISATAP Intra-Site Automatic Tunnel Addressing Protocol

IETF Internet Engineering Task Force

MAC Media Access Control

Příloha B

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `text` ..... text práce
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `src` ..... zdrojové soubory k PDF