Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Kristian Zupan

Mehanizmi prehoda na IPv6

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentorica: doc. dr. Mojca Ciglarič

Ljubljana, 2012

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Univerza *v Ljubljani*
Fakulteta *za računalništvo in informatiko*

Tržaška 25
1001 Ljubljana, Slovenija
telefon: 01 476 84 11
01 476 83 87
faks: 01 426 46 47
01 476 87 11
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 00031/2012
Datum: 12.04.2012

Univerza v Ljubljani, Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Kandidat: **KRISTIAN ZUPAN**

Naslov: **MEHANIZMI PREHODA NA IPV6**
TRANSITION MECHANISMS TO IPV6

Vrsta naloge: Diplomsko delo univerzitetnega študija prve stopnje

Tematika naloge:

Preučite problematiko postopnega prehoda s protokola IPv4 na IPv6. Pojasnite težave, ki jih prinaša sobivanje obeh protokolov in raziščite načine, kako se lahko zagotavlja medsebojna povezljivost omrežij in naprav na različnih verzijah protokola IP. V nadaljevanju se osredotočite na mehanizme, ki uporabljajo različne načine prevajanja omrežnih naslovov (NAT). Izberite značilne implementacije prevajanja s stanji in brez stanj ter na testnem poligonu primerjajte njuno učinkovitost. Preizkusite pravilnost prevajanja najpogostejših aplikacijskih protokolov in komentirajte morebitne težave. Ze enega izmed protokolov, ki se slabo prevajajo, raziščite možnosti izvedbe dodatnega aplikacijskega prehoda. Če bo možno, aplikacijski prehod tudi implementirajte ter testirajte njegovo delovanje. V zaključku kritično ovrednotite svoje delo in opišite možnosti prektične uporabe narejenega.

Mentor:
M. Ciglarič
doc. dr. Mojca Ciglarič

Dekan:
prof. dr. Nikolaj Zimic

Fakulteta za računalništvo in informatiko • UNIVERZA V LJUBLJANI • 2

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Kristian Zupan, z vpisno številko **63090064**, sem avtor diplomskega dela z naslovom:

Mehanizmi prehoda na IPv6

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Mojce Ciglarič,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela
- soglašam z javno objavo elektronske oblike diplomskega dela v zbirki "Dela FRI".

V Ljubljani, 3. septembra 2012 Podpis avtorja:

Zahvaljujem se mentorici doc. dr. Mojci Ciglarič za vodenje, spodbujanje in pomoč pri izdelavi diplomskega dela.

Zahvaljujem se Nejcu Škobernetu za nesebično pomoč pri izdelavi aplikacijskega prehoda in posredovanju znanja o prehodnih mehanizmih.

Zahvaljujem pa se tudi svoji družini in Karmen za podporo in razumevanje v času mojega študija.

Kazalo

Povzetek

Cilj diplomske naloge je bila primerjava mehanizmov prehoda na IPv6. Primerjali smo družine mehanizmov, ki uporabljajo NAT64 brez stanj in NAT64 s stanji. Primerjali smo dve implementaciji: Taygo, ki je NAT64 brez stanj, in Ecdysis, ki je NAT64 s stanji. Primerjava je bila narejena tako, da smo preizkusili delovanje komunikacije z vozlišča v omrežju IPv6 do vozlišča v omrežju IPv4 z različnimi pogosto uporabljenimi aplikacijskimi protokoli. Glede na kakovost prevajanja smo protokole razvrstili v tri skupine: dobro, pogojno in slabo prevedljive. Narejena je bila tudi primerjava učinkovitosti prevajanja obeh izbranih implementacij. Narejen je bil aplikacijski prehod, ki omogoča delovanje, sicer le pogojno prevedljivega protokola FTP skupaj z NAT64 brez stanj. Implementiran je bil kot nadgradnja namestniškega strežnika, ki prevaja ukaze FTP. Aplikacijski prehod je bil ustrezno testiran in z njegovo vključitvijo v testni poligon smo dokazali, da je možno tudi s protokolom FTP dostopati iz omrežja IPv6 do strežnika v omrežju IPv4. V sklepu smo pokazali na pomen pravilnega razumevanja in pretehtane izbire prehodnih mehanizmov v omrežjih ponudnikov dostopa do interneta.

Ključne besede

prehodni mehanizmi, IPv6, aplikacijski prehod, NAT64, preslikovanje mrežnih naslovov

Abstract

The aim of this bachelor thesis was the comparison between IPv6 transition mechanisms. The comparison was made between the two families of mechanisms: stateless NAT64 and stateful NAT64. We have compared two implementations: Tayga, which is a stateless NAT64 and Ecdysis, which is a stateful NAT64. Communication using diffrent application layer protocols from a node in IPv6 network to a node in IPv4 network was tested. Considering the quality of the transmision the protocols were arranged in three groups: well, conditionally and badly translatable. Additionally, effectiveness of address translation was measured and the differences were discused . An application level gateway, which enables us to use non translatable protocol FTP with stateless NAT64, was also implemented. It was designed as a supplement to a proxy server, performing the translation of FTP commands. Application level gateway was adequately tested and with its inclusion in the testing ground, we have proved that it is aslo possible to communicate from IPv6 network to IPv4 network with FTP protocol. In conclusion we have shown the importance of proper understanding and weighted choice of transition mechanisms in the networks of Internet service providers.

Keywords

Transition mechanisms, IPv6, Application-level gateway , NAT64, Network address translation

Poglavje 1

Uvod

1.1 Opis problematike

Leta 1998 je bil predstavljen internetni protokol IPv6, ki naj bi nadomestil trenutni IPv4. Razlog za njegov razvoj je bilo primanjkovanje mrežnih naslovov IPv4, saj je na svetu vedno več naprav, ki potrebujejo povezavo z internetom [30]. Trenutno se še vedno največ uporablja protokol IPv4, vendar lahko pričakujemo porast uporabe IPv6, saj so z izčrpanjem javnih naslovov organizacije IANA (31. januar 2011) in RIR APNIC (15. april 2011) nekateri deli sveta že ostali brez javnih naslovov IPv4 [31]. Ker se danes več uporablja IPv4 in še vedno le relativno malo Pv6, moramo danes uporabljati mehanizme, ki omogočajo IPv6 znotraj omrežij IPv4. Vendar pa delež IPv6 z izčrpavanjem naslovov IPv4 vztrajno narašča in kmalu se bo razmerje obrnilo. Zato se bomo v delu osredotočili na obratno situacijo, torej na dostop do vozlišč IPv4 iz omrežja IPv6.

1.2 Motivacija in opis diplome

Za raziskovanje te problematike smo se odločili zato, ker bo v prihodnosti v večini uporabljen protokol IPv6, še vedno pa bodo obstajali strežniki, ki se jih zaradi administrativnih razlogov ne bo dalo nadgraditi na protokol

IPv6 [34]. Vozlišč IPv4 ne bi bilo ustrezno kar odrezati od ostalega sveta. Zato potrebujemo prehodne mehanizme, ki bodo omogočali dostop do teh vozlišč. Med mehanizme, ki to problematiko rešujejo, spadata tudi NAT64 s stanji in NAT64 brez stanj. Vendar pa še ni raziskano, kateri izmed teh dveh mehanizmov je bolj učinkovit in kateri bolje prevaja določeno vrsto protokolov.

Primerjava prevajanja mehanizmov bo koristila zlasti ponudnikom dostopa do interneta, ki bi želeli takšen mehanizem namestiti v svoje omrežje. Na podlagi tega, kakšne storitve želijo nuditi, bi se lahko odločili, ali je za njih bolj primeren NAT64 s stanji ali NAT64 brez stanj.

V uvodu diplomske naloge je pojasnjena problematika obravnavane teme, navedena motivacija za diplomsko delo in podan opis naloge. Sledi navedba metodologije ter pregled področja in sorodne literature.

V drugem poglavju so predstavljeni nekateri prehodni mehanizmi, ki rešujejo obravnavano problematiko. Za vsak prehodni mehanizem je opisano njegovo delovanje, nato so navedene njegove prednosti in slabosti.

V tretjem poglavju je narejena primerjava delovanja NAT64 s stanji in brez stanj. V tem poglavju smo postavili tudi hipoteze, ki smo jih poskušali v naslednjih poglavjih potrditi. Primerjava koristi zlasti skrbnikom omrežij in ponudnikom dostopa do interneta, da se lahko odločijo, katera vrsta NAT64 je bolj primerna za njihov tip omrežja.

Četrto poglavje je namenjeno primerjavi teh mehanizmov v praksi. Primerjava je bila narejena med Ecdysisom, ki je NAT64, s stanji in Taygo, ki je brez stanj. Opisan je tudi testni poligon, ki smo ga uporabili za testiranje posameznega mehanizma. Predstavljeni so rezultati testiranja učinkovitosti in delovanja prevajanja protokolov. Navedeno je, zakaj se nekateri protokoli ne prevedejo in kaj potrebujejo drugi, da se prevod lahko zgodi.

V petem poglavju je načrtovana, narejena in opisana prva implementacija aplikacijskega prehoda FTP64. FTP je eden izmed tistih protokolov, ki uporablja omrežne naslove v podatkovnem delu aplikacijske plasti. Da ga lahko uporabljamo z NAT64, potrebujemo aplikacijski prehod ali ALG

(ang. Application-level gateway). Glavna funkcionalnost ALG-ja je prevajanje mrežnih naslovov, ki se nahajajo znotraj podatkovnega dela aplikacijske plasti, da lahko vozlišče na drugi strani NAT-a normalno komunicira z nami [21].

V sklepnih ugotovitvah, ki so na koncu naloge, so navedeni glavni prispevki diplomske naloge in komu lahko ti prispevki koristijo.

1.3 Metodologija

Metodologija, ki smo jo izbrali pri našem delu, je sledeča:

1. Pregled obstoječega dela in literature, ki se nanaša na to področje.
2. Izbira implementacije NAT64 s stanji in brez stanj.
3. Postavitev poligona oziroma testnega omrežja za vsako implementacijo.
4. Izvedbe testiranja in pregled rezultatov.
5. Izbira slabo prevedenega protokola, za katerega bomo implementirali ALG.
6. Načrt, implementacija in testiranje ALG-ja.

1.4 Pregled področja in sorodna literatura

Organizacija IETF je izdala več dokumentov in osnutkov, ki se nanašajo na tematiko prehodnih mehanizmov. Nishitani in ostali v osnutku [17] predlagajo pogoste funkcije, ki jih mora imeti CGN. V tem osnutku so tudi našteti prehodni mehanizmi, ki uporabljajo ta pristop. Prehodni mehanizem NAT444 kot ga predlaga IETF, je opisan v osnutku [20], mehanizem DS-Lite pa v RFC 6333 [13]. Mehanizem NAT64 s stanji je definiran v RFC 6146 [4], delovanje strežnika DNS64 pa je opisano v RFC 6147 [5]. Algoritem, ki ga uporablja NAT64 brez stanj, je opisan v RFC 6145 in RFC 6052 [14, 6].

Pristop A+P je opisan v RFC 6346 [9]. Vsi trije mehanizmi, ki ta pristop uporabljajo, pa v osnutkih [11, 12, 7]. Načrt za izdelavo aplikacijskega prehoda smo dobili iz RFC 6384 [8].

Raziskav s področja prehodnih mehanizmov ni veliko. Che in Lewis [10] primerjata tranzicijske mehanizme in njihovo učinkovitost v simulacijskem okolju. Avtorja navajata določene težave, ki nastanejo pri tranzicijskih mehanizmih, vendar se ti ne tičejo NAT64/DNS64. Martin [16] poudarja, da je strategija IPv4-v-IPv6 nepopolna in bomo imeli veliko večje težave pri migraciji na IPv6-v-IPv4. Zaradi tega nastajajo potrebe po dodatnih tranzicijskih mehanizmih. Al Ja'afreh in ostali [1, 2, 3] primerjajo učinkovitost uporabe dvojnega sklada in tuneliranja za povezave med omrežjem IPv4 in omrežjem IPv6. Wing [24] ocenjuje in primerja različne znane pristope NAT-a v omrežjih IPv4, IPv6 in mešanih omrežjih.

Ciso na svojih spletnih straneh nudi poglobljeno primerjavo NAT64 s stanji in brez stanj [35, 36]. I. Pepeljakpa v svojem blogu strokovno utemelji, zakaj se je bolje NAT64 brez stanj izogniti [33].

Diplomsko delo se v veliki meri nanaša na članek Nejca Škoberneta in Mojce Ciglarič [22], kjer je narejena raziskava mehanizma NAT64 s stanji v praksi. V članku je raziskano, kateri aplikacijski protokoli se prevajajo, kadar uporabljamo ta mehanizem. Po poligonu, opisanem v članku, je povzeta tudi zasnova lastnega testnega poligona. Razvidno je tudi, kateri aplikacijski protokoli potrebujejo ALG-je, vendar v članku ni opisa nobene konkretne implementacije. Prav tako ni narejenega testa učinkovitosti tega prehodnega mehanizma.

Poglavje 2

Pregled prehodnih mehanizmov

Prehod na IPv6 ne bo nastal čez noč, saj bi nenaden prehod lahko povzročil odpoved številnih storitev, ki so sedaj na voljo. Prav tako bi od ostalega sveta odrezali tista vozlišča, ki se jih ne da nadgraditi na protokol IPv6. Zaradi tega bosta morala oba protokola še nekaj časa sobivati skupaj. Posledica tega je, da morajo znati vozlišča IPv4 komunicirati z vozlišči IPv6 ter obratno. Da je to lahko mogoče, pa potrebujemo prehodne mehanizme. Različni prehodni mehanizmi opisujejo različne recepte, kako zagotavljati navzkrižno komunikacijo med napravami, ki delujejo navadno le v določenih okoliščinah.

V tem poglavju so opisane osnove prehodnih mehanizmov, ki so danes aktualni, kadar želimo komunicirati z omrežjem IPv4 preko omrežja IPv6. Cilj tega poglavja je, da bralec izve, kateri prehodni mehanizmi obstajajo, in da se seznani s tem, kako delujejo. Hkrati pa dobi občutek, s katerimi problemi se ti mehanizmi soočajo.

Poznamo tri vrste metod, ki jih uporabljajo prehodni mehanizmi. To so tuneliranje, prevajanje in dvojni sklad. Vse tri so opisane v Waddington in Chang [23]. Tuneliranje deluje tako, da se naredi enkapsulacija datagrama IPv4 znotraj datagrama IPv6. Glava datagrama IPv4 ostane nespremenjena, kateremu pa se doda novo glavo IPv6.

Prevajanje je metoda, pri kateri se podatke iz stare glave (npr. IPv4) uporabi za to, da se napolni podatke v novi glavi, v obliki, ki jo zahteva

drugi protokol (npr. IPv6). Paketu se nato doda novo glavo.

V primeru dvojnega sklada imajo vozlišča v omrežju implementiran tako protokol IPv4 kot tudi IPv6. Zaradi tega lahko vozlišča komunicirajo tako z omrežjem IPv4 kot z IPv6.

2.1 Pristop "Carrier-grade-NAT (CGN)"

NAT444, DS-Lite in NAT64 uporabljajo tako imenovan pristop CGN (Carrier-grade-nat) ali LSN (Large Scale NAT) [17]. Za ta pristop je značilno, da je potreben poleg NAT-a v lokalnih omrežjih tudi NAT v omrežju ponudnika dostopa do interneta (ang. Internet Service provider). Njihova slabost je, da jih naročniki ne morejo prilagajati svojim potrebam, kot je recimo odpiranje vrat.

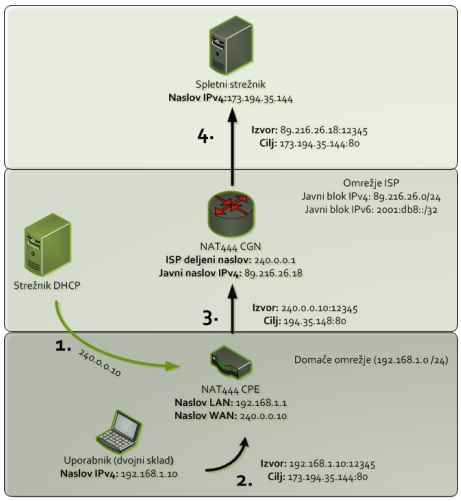
CGN pomeni Carrier Grade NAT, torej preslikovanje mrežnih naslovov, ki je namenjeno ponudniku dostopa do interneta in ne končnim uporabnikom (kot smo morda bili do sedaj vajeni). Potreben je zato, ker ne moremo zagotoviti naročniku lasnega fiksnega naslova IPv4.

2.1.1 NAT444

NAT444 je dobil ime po dejstvu, da je uporabljena dvojna preslikava naslovov IPv4 [20]. Pri tej tehniki gre v bistvu za uporabo dveh NAT44 zaporedoma, torej preslikave iz IPv4 v IPv4. Prvega v lokalnem omrežju, drugega pa v omrežju ponudnika dostopa do interneta, ki mora biti veliko zmogljivejši. Je najenostavnejša različica CGN-ja, saj so potrebne minimalne spremembe v obstoječem omrežju. Komponente NAT444 in funkcije posamezne komponente so predstavljene na sliki 2.1.

Delovanje NAT444 lahko opišemo z naslednjimi koraki:

1. Strežnik DHCP, ki je v lasti ponudnika dostopa do interneta, dodeli CPE-ju (Customer-premises equipment) zasebni naslov IPv4 znotraj omrežja ponudnika dostopa do interneta.



Slika 2.1: NAT444

2. Odjemalec naslovi strežnik IPv4. To lahko naredi, saj ima nameščen dvojni sklad.

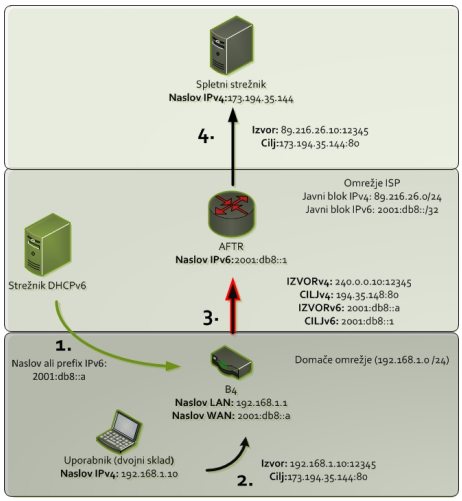
3. Ko prispe paket do prevajalnika mrežnih naslovov v omrežju stranke, mu ta preslika privatni naslov v lokalnem omrežju, v privatnega znotraj omrežja ponudnika dostopa do interneta.

4. Ko prispe paket do NAT444 CGN-ja, mu ta znova preslika privatni naslov, le da tokrat v javnega.

Glavna prednost NAT444 je, da so za njegovo postavitev potrebne minimalne spremembe v omrežju in da uporablja obstoječo tehnologijo. Zaradi tega je NAT444 priporočljiv zlasti takrat, ko hočemo, da so stroški za prehod majhni.

Slabost NAT444 je, da uporabniki ne morejo po svoje nastaviti NAT-a v omrežju ponudnika dostopa do interneta. Zaradi tega odpovejo nekatere aplikacije, ki morajo imeti za normalno delovanje odprta vrata proti odjemalcu.

Težava NAT444 so tudi privatni naslovi znotraj omrežja ponudnika dostopa do interneta, saj se lahko zgodi, da se lokalno omrežje in omrežje



Slika 2.2: DS-Lite

ponudnika dostopa do interneta prekrivata. Zaradi tega se lahko uporabi blok naslovov E (240.0.0.0/4). Težava teh naslovov je, da so neuborabni, saj jih nekateri usmerjevalniki zavračajo. Druga rešitev so tako imenovani "ISP shared addresses" [25], ki naj bi jih odstopila organizacija IANA.

2.1.2 DS-Lite

Glavna razlika med NAT444 in DS-Lite je, da namesto preslikave v omrežju stranke uporabi enkapsulacijo paketa IPv4 znotraj paketa IPv6 [13]. Prednost tega je, da se uporabi en NAT manj in da odpade problem prekrivanja blokov privatnih naslovov, ki je prisoten pri NAT444. Pri tem mehanizmu je uporabljena kombinacija dvojnega sklada in tuneliranja.

Delovanje DS-Lite lahko opišemo z naslednjimi koraki:

1. Strežnik DHCPv6 dodeli iniciatorju (B4 na sliki 2.2) naslov IPv6 ali predpono.

2. Odjemalec naslovi strežnik IPv4. To lahko naredi, saj ima nameščen dvojni sklad.

3. Ko prispe paket do iniciatorja, naredi ta enkapsulacijo paketa IPv4 znotraj paketa IPv6 in pošlje paket proti koncentratorju (AFTR na sliki 2.2).

4. Koncentrator nato dekapsulira paket IPv6, naredi preslikavo izvornega naslova in vrat ter pošlje paket naprej do strežnika.

Slabost DS-Lite je, da je poleg postavitve CGN-ja potrebna tudi nadgradnja obstoječe opreme pri naročnikih.

2.1.3 NAT64/DNS64

Kadar odjemalci nimajo nameščenega dvojnega sklada, temveč uporabljajo samo IPv6, je smiselno uporabiti NAT64 [4]. NAT64 potrebuje za delovanje poseben strežnik DNS64, ki poskrbi za pretvorbo DNS odgovorov tipa A (za vozlišča IPv4) v odgovore tipa AAAA (za vozlišča IPv6) [5]. NAT64 uporablja translacijo.

Koraki pri delovanju NAT64 so sledeči:

1. Odjemalec pošlje strežniku DNS64 poizvedbo za naslov IPv6 od www.domena.test.

2. DNS64 pošlje strežniku DNS, ki je odgovoren za www.domena.test, poizvedbo za naslov IPv6.

3. Ta mu odgovori, da nima naslova IPv6 za www.domena.test

4. DNS64 pošlje strežniku DNS poizvedbo za naslov IPv4 od www.domena.test.

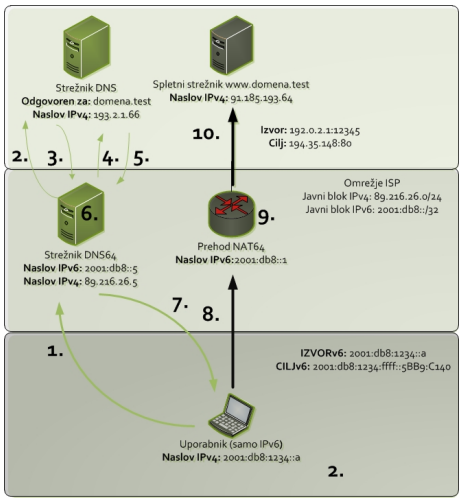
5. DNS ima naslov IPv4 za www.domena.test, zato mu vrne odgovor.

6. DNS64 doda predpono naslovu IPv4, da dobi naslov IPv6.

7. DNS64 vrne odjemalcu odgovor za www.domena.test.

8. Odjemalec pošlje paket po omrežju IPv6 proti strežniku.

9. Prehod NAT64 prestreže pakete ter jim spremeni ciljni in izvorni naslov.



Slika 2.3: Primer delovanja NAT64 s stanji

10. Prehod NAT64 pošlje po omrežju IPv6 paket proti strežniku.

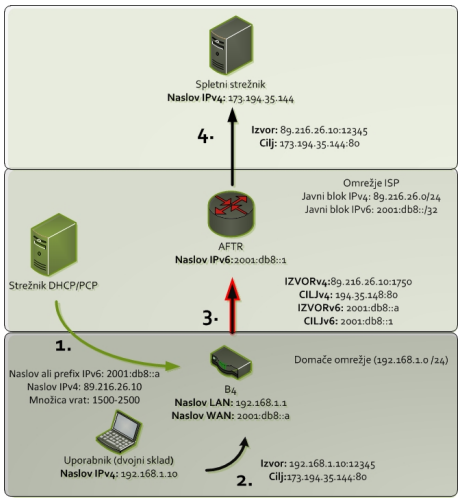
Glede na to, kako se preslika izvorni naslov vozlišča IPv6, poznamo dva načina. Prvi način je brez stanj (ang. stateless), drugi pa s stanji (ang. stateful). Primer implementacije prvega je Tayga, drugega pa Ecdysis. Razlike med NAT64 s stanji in brez stanj so opisane v poglavju 3.

Prednost tega mehanizma je, da v lokalnem omrežju naročnika ne potrebujemo CPE-ja, ki bi opravljal tuneliranje ali prevajal naslove.

2.2 Pristop "Address-plus-port (A+P)"

Lightweight 4over6, 4rd in dIVI uporabljajo pristop imenovan"Address-plus-Port". Za ta pristop je značilno, da več uporabnikov uporablja isti naslov, a različno množico vrat. Za razliko od CGN-ja A+P ne uporablja NAT-a v omrežju ponudnika dostopa do interneta.

Naveden pristop je koristen za ponudnike dostopa do interneta takrat, kadar imamo v večini naročnike, ki potrebujejo povezljivost z omrežjem IPv4, ne bodo pa uporabljali svojih naprav za strežnike. Kot primer lahko vzamemo veliko množico mobilnih naprav.



Slika 2.4: Lightweight 4over6

2.2.1 Lightweight 4over6

Lightweight 4over6 je vrsta nadgradnje DS-Lite. Izboljšava je, da se preslikovanje ne dogaja več na strani ponudnika dostopa do interneta, temveč na iniciatorju v lokalnem omrežju. S tem se bistveno zmanjša obremenitev omrežja, prepustnost pa se poveča.

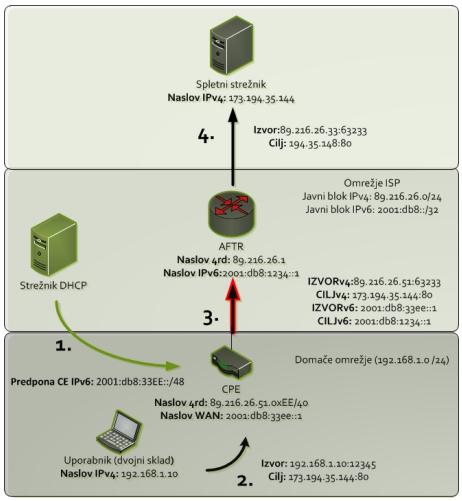
Delovanje je sledeče:

1. Iniciator pridobi naslov IPv6, naslov IPv4 in množico vrat, ki jih sme uporabiti.

2. Odjemalec pošlje zahtevo proti strežniku.

3. Iniciator preslika izvorni naslov in izbere naključna vrata iz množice vrat. Naredi še enkapsulacijo paketa IPv4 znotraj IPv6.

4. Koncentrator dekapsulira paket IPv6 in pošlje paket IPv4 proti strežniku.



Slika 2.5: 4rd

2.2.2 4rd

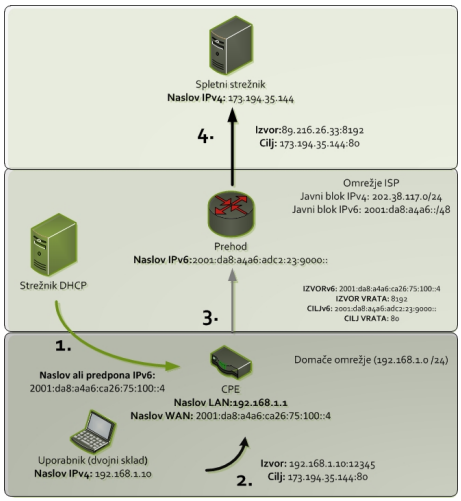
4rd se glede na Leightweight 4over6 razlikuje v tem, da si ne zapomni, v kateri paket IPv6 se enkapsulira paket IPv4, temveč naredi to algoritmično. S tem naj bi se prepustnost še povečala.

Delovanje je sledeče:

1. Strežnik DHCP dodeli CPE-ju predpono CE. Iz nje dobi CPE naslov IPv6, naslov IPv4 in množico vrat, ki jih sme uporabiti.
2. Odjemalec pošlje zahtevo proti strežniku.
3. CPE preslika naslov IPv4 v naslov 4rd in algoritmično enkapsulira paket IPv4.
4. Koncentrator algoritmično dekapsulira paket IPv6 in pošlje paket IPv4 proti strežniku.

2.2.3 dIVI

Mehanizem dIVI uporablja dvojno preslikavo brez stanj. Je nadgradnja protokola IVI [15] in omogoča deljenje enega naslova IPv4, kljub temu da gre za



Slika 2.6: dIVI

prehodni mehanizem brez stanj.

Delovanje mehanizma je sledeče:

1. Strežnik DHCP dodeli CPE-ju naslov IPv6 ali predpono.
2. Odjemalec pošlje zahtevo proti strežniku.
3. CPE uporabi naslov IPv6, da preslika vrata. Ko to naredi, pošlje proti prehodu z naslovom IPv6. Ta naslov je narejen iz predpone in strežnikovega naslova IPv4.
4. Prehod nato dobi naslov IPv4 iz naslova IPv6. Izvorna vrata ostanejo ista. Preslikava je možna tudi ena proti mnogo.

Poglavje 3

Primerjava NAT64 s stanji in NAT64 brez stanj

V tem poglavju je narejena podrobnejša primerjava prehodnih mehanizmov NAT64 s stanji in NAT64 brez stanj. Osnovno delovanje NAT64 skupaj z DNS64 je predstavljeno v razdelku 2.1.3, zato bomo na tem mestu opredelili razlike med NAT64 brez stanji in s stanji. Opisan je postopek delovanja prevajanja naslovov, kadar uporabljamo en ali drug način. Iz povzetka nato izhaja, kdaj je bolj smiselno uporabiti NAT64 s stanji in kdaj brez stanj.

Obe vrsti NAT64 se razlikujeta predvsem v tem, kako preslikujeta naslove IPv6 v naslove IPv4. Primerjava glavnih lastnosti je podana v tabeli 3.1.

3.1 NAT64 brez stanj

3.1.1 Delovanje

V primeru NAT64 brez stanj je informacija o preslikavi naslova vsebovana kar znotraj naslova IPv6. To pomeni, da NAT64 uporabi algoritem za pridobitev naslova IPv4 iz naslova IPv6 [6]. Preslikava naslovov je ena proti ena, torej se vsak naslov IPv6 preslika v svoj naslov IPv4 [35].

Pri NAT64 brez stanj je potrebno ločiti dve vrsti naslovov. To so vgrajeni IPv4 znotraj IPv6 (ang. IPv4-embedded IPv6 addresses) in pretvorjeni

IPv4 znotraj IPv6 (ang. IPv4-converted IPv6 addresses). Prvi naslovi so naslovi IPv6, ki znotraj sebe vsebujejo naslov IPv4. To so naslovi tistih vozlišč, ki pošljejo paket iz omrežja IPv6 proti omrežju IPv4. Drugi naslovi predstavljajo vozlišča IPv4 znotraj omrežja IPv6. Ta vrsta naslovov se uporablja zato, da lahko vozlišča IPv6 naslavljajo vozlišča IPv4. Struktura obeh naslovov je ista in jo lahko vidimo na sliki 3.1.

Da lahko NAT64 brez stanj preslikuje eno ali drugo vrsto naslovov, mora imeti znano predpono NAT64 in njeno dolžino (ang. prefix length). Predpono NAT64 potrebuje zato, da lahko na podlagi nje izlušči naslov IPv4 iz naslova IPv6 ali pa obda IPv4, da dobi naslov IPv6. Od dolžine predpone pa je odvisno, kako oziroma kam se naslov IPv4 pripne predponi. Dolžina predpone ne sme biti poljubna, temveč je na izbiro šest različnih dolžin. Običajno lahko uporabimo tudi tako imenovano znano predpono (ang. Well-Known Prefix) “64:ff9b::/96” [6], njena uporaba pa se odsvetuje, saj ne omogoča naslavljanja privatnih naslovov IPv4. Glede na dolžino predpone je struktura naslova IPv6 sledeča:

1. Če je predpona dolga 32 bitov, je naslov IPv4 kodiran med bitoma 32 in 63. Biti od 64 do 72 so enaki 0.

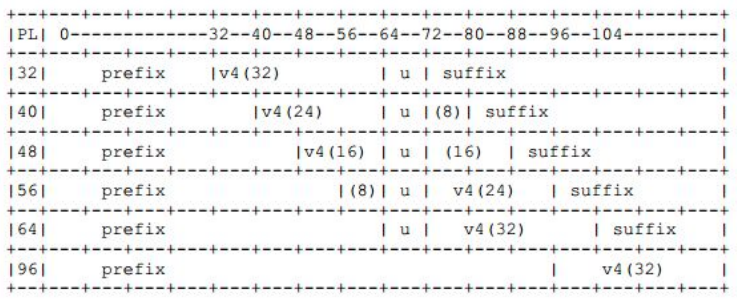
2. Če je predpona dolga 40 bitov, je 24 bitov naslova IPv4 zakodiranih med bitoma 40 in 63, ostalih 8 pa med 72 in 79.

3. Če je predpona dolga 48 bitov, je 16 bitov naslova IPv4 zakodiranih med bitoma 48 in 63, ostalih 16 pa med bitoma 72 in 87.

4. Če je predpona dolga 56 bitov, je 8 bitov naslova IPv4 zakodiranih med bitoma 56 in 63, ostalih 24 bitov pa med 72 in 95.

5. Če je predpona dolga 64 bitov, je naslov IPv4 zakodiran med bitoma 72 do 103.

6. Če je predpona dolga 96 bitov, je naslov IPv4 zakodiran med bitoma 96 in 127.



Slika 3.1: Naslovi IPv4 znotraj naslovov IPv6. (Dostopno na [6])

Za boljšo predstavo je pripeta tudi slika 3.1, kjer je zgradba naslovov IPv6 prikazana še shematsko.

Kadar dobi NAT64 brez stanj paket, ki vsebuje naslove IPv4, jih pretvori v naslove IPv6 po naslednjem algoritmu:

1. Združi predpono, naslov IPv4 in končnico (ang. suffix), da dobiš 128 biten naslov.

2. Če je dolžina predpone manj kot 96, vstavi ničle med bitoma 64 do 71.

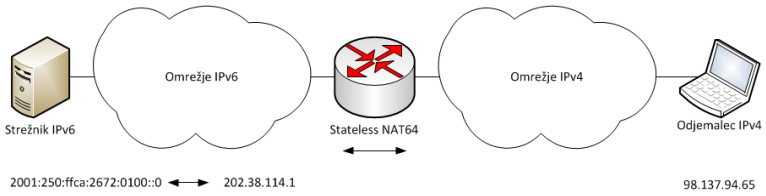
Kadar dobi NAT64 brez stanj paket, ki vsebuje naslove IPv6, jih pretvori v naslove IPv4 po naslednjem algoritmu:

1. Če je predpona dolga 96 bitov, vzemi zadnjih 32 bitov.

2. Drugače odstrani bajt "u" (biti 64 do 71), da dobiš 120 bitov dolgo sekvenco. Nato vzemi 32 bitov za predpono.

3.1.2 Prednosti in slabosti

Zaradi algoritmične pretvorbe naslovov je pričakovano, da je NAT64 brez stanj hitrejši kot s stanji. NAT64 brez stanj ohranja princip od konca do konca in omogoča vzpostavitev povezave tako s strani IPv6 proti IPv4 kot tudi obratno. Zaradi teh lastnosti lahko pričakujemo, da je NAT64 bolj



Slika 3.2: Običajen scenarij delovanja NAT64 brez stanj

primeren takrat, kadar želimo iz omrežja IPv4 vzpostaviti povezavo proti strežniku IPv6 ali pa uporabljamo aplikacijo, ki zahteva vzpostavitev povezave iz ene ali druge smeri [33]. Takšen scenarij je prikazan na sliki 3.2

Zaradi lastnosti NAT64 brez stanj obstajajo omejitve, katere naslove IPv6 imajo lahko vozlišča, zato je obvezna uporaba strežnika DHCPv6 ali ročna nastavitev naslovov [35].

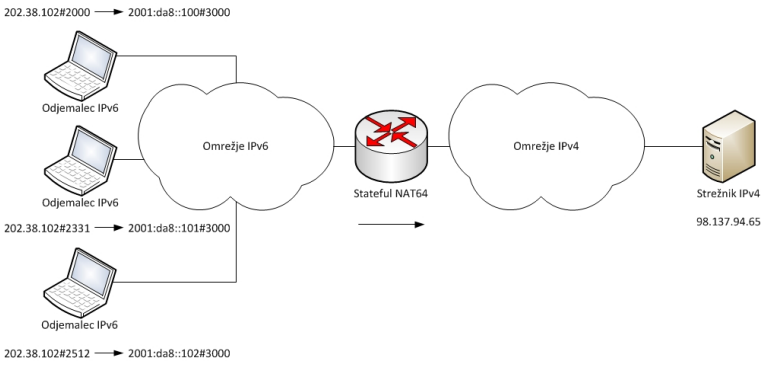
Pod največjo slabost NAT64 brez stanj lahko štejemo to, da se za vsak naslov IPv6 porabi en naslov IPv4. To postavlja pod vprašanje njegovo uporabnost, saj s tem ne rešujemo problema izčrpanosti naslovov IPv4 [35, 33]. Temu se lahko izognemo tako, da za NAT64 postavimo navaden NAT44. Vendar pa je pri tem potrebno razmisliti, ali bi bilo morda bolje postaviti kakšen drug prehodni mehanizem, ki ohranja naslove IPv4 [33].

3.2 NAT64 s stanji

3.2.1 Delovanje

NAT64 s stanji in brez stanj se razlikujeta v tem, kako preslikujeta naslove vozlišč znotraj omrežja IPv6. Za preslikovanje naslovov vozlišč iz omrežja IPv4 uporablja povsem isti algoritem kot NAT64 brez stanj in prav tako potrebuje predpono NAT64 in njeno dolžino.

Za preslikovanje naslovov iz omrežja IPv6 ne uporablja algoritma, temveč uporablja stanja. To pomeni, da si mora naprava zapomniti informacijo o preslikavi. Preslikava je ena proti mnogo, torej se več naslovov IPv6 preslika



Slika 3.3: Običajen scenarij delovanja NAT64 s stanji

v en naslov IPv4. To se izvede tako, da NAT64 za različne naslove IPv6 uporabi različna vrata. Delovanje je podobno kot pri NAT44, ki je danes uporabljen za deljenje enega naslova IPv4 [35].

3.2.2 Prednosti in slabosti

Prednost NAT64 s stanji je, da ohranja naslove IPv4. Tako lahko za več naslovov IPv6 porabimo samo en naslov IPv4 [35]. Zato je veliko bolj uporaben kot NAT64 brez stanj, kadar imamo v omrežju IPv6 veliko odjemalcev. Takšen scenarij je prikazan na sliki 3.3.

Vozlišča v omrežju IPv6 imajo lahko poljubne naslove in ne predpisane vnaprej, kot je to pri NAT64 brez stanj. Zaradi tega lahko uporabimo več vrst mehanizmov za dodeljevanje mrežnih naslovov IPv6 [35].

Slabost NAT64 s stanji je, da je neuporaben, kadar želimo postaviti strežnik, ki mora biti prav tako dostopen iz omrežja IPv4.

3.3 Postavitev hipotez

Na podlagi navedenega, da NAT64 brez stanj uporablja algoritmično prevajanje naslovov, bomo postavili prvo hipotezo:

Hipoteza 1 *Prevajanje omrežnih naslovov na način brez stanj iz IPv6 v*

Kriterij	**Stateless**	**Stateful**
Vrsta preslikave	1:1	1:n
Vzpostavitev povezave iz obeh smeri	Da	Ne
Shranjuje stanja	Ne	Da
Omejitve pri naslovih IPv6	Da	Ne
Omejitve pri dodeljevanju naslovov IPv6	Da	Ne
Ohranja naslove IPv4	Ne	Da

Tabela 3.1: Primerjava NAT64 brez stanj in s stanji (Povzeto po [35, 36])

IPv4 je hitrejše kot enako prevajanje s stanji.

Poleg tega uporablja NAT64 brez stanj princip od konca do konca in omogoča vzpostavitev povezave tudi s strani omrežja IPv4. Zaradi tega lahko postavimo drugo hipotezo:

Hipoteza 2 *NAT64 s stanji ni primeren za dostop do strežnika iz omrežja IPv4 v omrežja IPv6 in ni primeren za uporabo aplikacij, ki zahtevajo vzpostavitev seje iz obeh smeri*

Sporočila nekaterih aplikacijskih protokolov je treba pred in/ali po prevajanju še dodatno obdelati; primer takega protokola je FTP. Tretja hipoteza je torej naslednja:

Hipoteza 3 *Z dodatnim procesiranjem, ki ga izvaja aplikacijski prehod, sicer neprevedljiv protokol FTP postane prevedljiv.*

Poglavje 4

Primerjava Tayge in Ecdysisa

V tem poglavju je narejena primerjava Tayge, ki je NAT64 brez stanj, in Ecdysisa, ki je NAT64 s stanji. Namen poglavja je, da poskušamo ovreči ali potrditi prvi dve hipotezi iz poglavja 3. Narejena sta opis in pregled njunih lastnosti. Za oba programa je opisano testno omrežje, na katerem je bilo preizkušeno delovanje protokolov. Narejena je primerjava učinkovitosti posameznega prehodnega mehanizma in je opisan način, kako je bila učinkovitost izmerjena. Prikazani sta tudi razpredelnici, ki nam povesta, kateri protokoli delujejo pri določenem prehodnem mehanizmu in kateri ne. Na koncu so tudi navedeni razlogi, zakaj nekateri protokoli odpovejo in ali obstaja možnost, kako jih pripravimo do delovanja.

4.1 Tayga

4.1.1 Opis

Tayga je odprtokodna implementacija NAT64 brez stanj za operacijski sistem Linux. Je izvenjedrna, kar pomeni, da se ne naloži kot jedrni modul, temveč jo moramo zagnati kot samostojen proces. Tayga je delo avtorja Nathan Lutchanskyja in je dostopna na naslovu [38]. Za preslikovanje naslovov uporablja gonilnik TUN [39]. Gonilnik TUN je navidezna mrežna kartica, ki je v celoti implementirana programsko. To nam omogoča, da pakete naslovljene

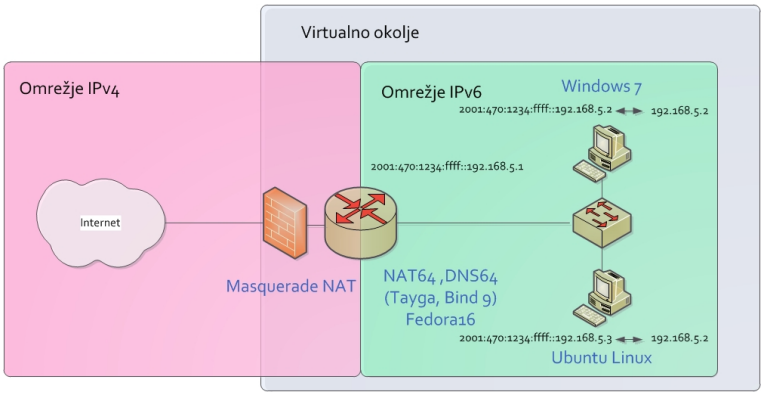
na to mrežno kartico programsko obdelamo in jih nato pošljemo spremenjene po omrežju naprej. Gonilnik TUN se recimo uporablja tudi pri programih za ustvarjanje virtualnih privatnih omrežjih in za virtualne računalnike. Taygo smo brez težav namestili in preizkusili na operacijskem sistemu Fedora 16.

Ker je Tayga NAT64 brez stanj, je tudi preslikovanje naslovov temu ustrezno. Poleg načina, ki smo ga že opisali v razdelku 3.2.1, pozna Tayga tudi dodatna dva načina za preslikovanje naslovov iz omrežja IPv6. Prvi način je statičen in ga moramo ročno nastaviti v nastavitveni datoteki Tayge. Omogoča nam, da nek naslov IPv6 preslikamo v poljuben naslov IPv4. Statičen način je priporočljiv, kadar želimo postaviti strežnik, saj točno vemo, v kateri naslov IPv6 se bo preslikal naslov IPv4. Drugi način je dinamičen in je nekakšna kombinacija NAT64 s stanji in brez stanj. Informacijo o preslikavi si zapomni v datoteko, obenem pa je preslikava še vedno ena proti ena. Deluje tako, da Tayga vozlišču IPv6 naključno dodeli naslov IPv4 iz bazena naslovov, ki jih vnaprej določi administrator v nastavitveni datoteki. Vozlišče nato uporablja naključni naslov 2 uri. Prednost tega načina je, da imajo lahko vozlišča IPv6 poljubne naslove in lahko uporabimo enake mehanizme za dodeljevanje naslovov kot pri NAT64 s stanji. Slabost pa je, da ne moremo točno vedeti, kateri naslov IPv4 se preslika v naslov IPv6, saj je ta informacija znana samo Taygi.

4.1.2 Testno okolje

Testno okolje je prikazano na sliki 4.1. Na sliki vidimo dva računalnika, ki sta preko stikala povezana na prehod Taygo, ki zagotavlja povezljivost z internetom. Računalnika in Tayga so postavljeni kot virtualni računalniki s pomočjo programske opreme VMware Workstation [40]. Testno omrežje smo poskušali narediti čim bolj podobno kot v viru [22], saj je bil cilj primerjava z mehanizmom Ecdysis.

Uporabljena je predpona “2001:470:1234:ffff::/96”. Računalniku Windows 7, ki ima naslov IPv6 “2001:470:1234:ffff::192.168.5.2/96”, se preslika v IPv4 “192.168.5.2”. Na prehodu je prav tako nameščen DNS64 strežnik



Slika 4.1: Testno okolje pri Taygi

BIND 9, ki poskrbi, da lahko iz omrežja IPv6 naslavljamo vozlišča IPv4 preko domenskih imen.

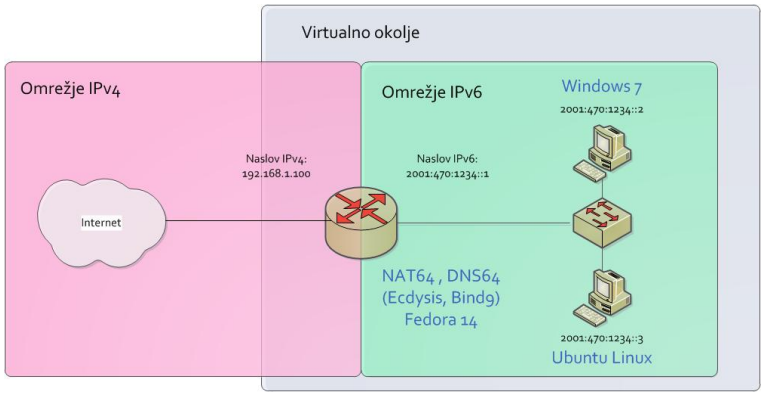
Za Taygo je nameščen masquerade NAT, ki poskrbi da se v vseh paketih, ki gredo v internet, naslovi IPv4 preslikajo v en naslov IPv4. Potreben je, ker nimamo na voljo tolikšnega bloka javnih IPv4 naslovov. Paketom IPv4, ki prihajajo v virtualno omrežje, se preslikava naslovov ne zgodi.

4.2 Ecdysis

4.2.1 Opis

Poglavitna razlika med Taygo in Ecdysisom je v tem, da je Ecdysis potrebno naložiti kot jedrni modul podobno kot gonilnike naprav. Ecdysis za prestrezanje paketov na mrežni plasti ne uporablja TUN gonilnika, temveč ogrodje Netfilter [37], katerega del je tudi požarni zid Iptables. Ta poskrbi, da gredo mrežni paketi skozi ustrezno navidezno mrežno napravo.

Ecdysis je naredilo podjetje Viagénie in je dostopno na viru [27]. V nasprotju s Taygo ga je možno namestiti samo na jedra verzije 2.6.31 do 2.6.35, kar daje občutek, da se program ne razvija več. Na njihovi spletni strani je tudi dodelana Linux distribucija Fedora 14, ki ima že nameščen



Slika 4.2: Testno okolje pri Ecdysisu

Ecdysis, ter strežnika DNS64 Unbound in Bind.

4.2.2 Testno okolje

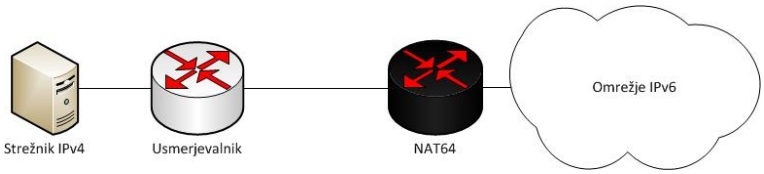
Testno okolje je podobno kot v viru [22] in je prikazano na sliki 4.2. Razlikuje se le v mrežnih naslovih. Razlog, da smo izbrali podobno testno okolje, je, da smo večino rezultatov o delovanju protokolov na Ecdysisu dobili iz tega vira.

4.3 Primerjava učinkovitosti

4.3.1 Testni scenarij

Učinkovitost Tayge in Ecdysisa je bila merjena na tri načine. Prvi način je bil merjenje prepustnosti omrežja, drugi merjenje odzivnega časa, tretji pa merjenje faznega trepetanja (ang. jitter). V vseh primerih smo se povezali z računalnikom v omrežju IPv4, ki je bil oddaljen dva skoka od NAT64, kot je prikazano na sliki 4.3. Računalnik in NAT64 so bili povezani z 1 gigabitno ethernet povezavo. Za primerjavo je dodan tudi rezultat, kadar uporabljamo samo protokol IPv4 brez NAT64.

Meritev prepustnosti je bila opravljena s programom Iperf [32]. Meritev



Slika 4.3: Testno omrežje za merjenje učinkovitosti

je bila opravljena tako, da smo pognali test za 300 sekund, sproti pa smo izpisovali, kakšna je trenutna prepustnost znotraj 10-sekundnih intervalov.

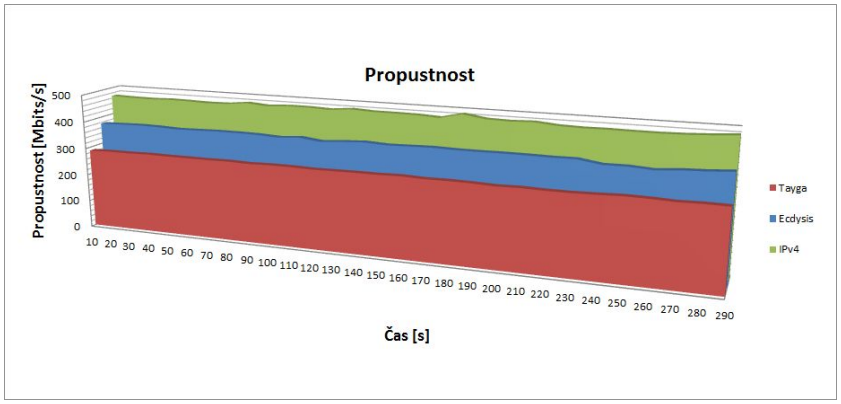
Meritev odzivnega časa in faznega trepetanja je bila opravljena s programom Ping. Meritev je bila opravljena tako, da je bilo poslanih 300 paketov s sekundnim razmakom. Na koncu je bilo očitano povprečje in standardni odklon odzivnega časa. Pred testom je bilo pričakovano, da se bo bolje odrezala Tayga, saj izvaja preslikavo omrežnih naslovov brez stanj. Rezultati meritev so prikazani v tabeli 4.1

4.3.2 Prepustnost

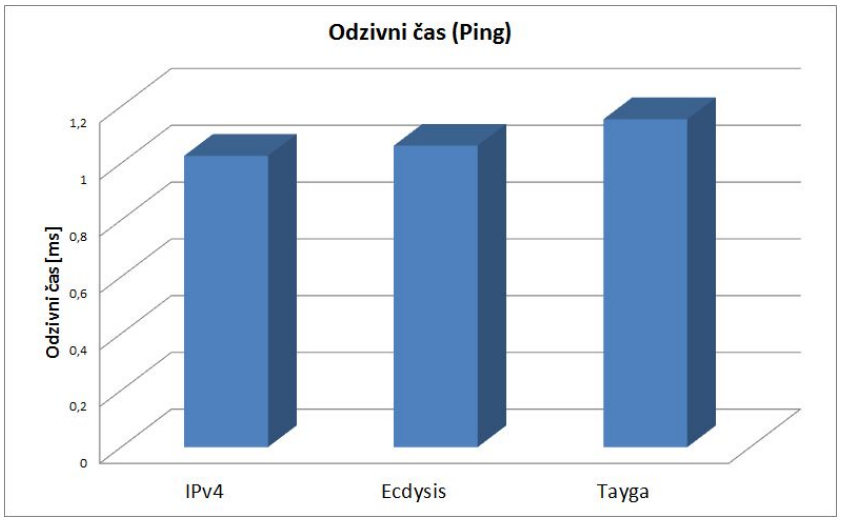
Ta metrika je pomembna, ker nam pove, kolikšna je največja možna hitrost prenašanja podatkov. Rezultati meritve prepustnosti so prikazani na grafu 4.4. Razvidno je, da je prepustnost večja pri Ecdysisu, kar je v nasprotju z našimi pričakovanji. Kot razlog za to lahko morda krivimo slabo implementacijo Tayge, saj se ne naloži direktno kot modul. V tabeli 4.1 so prikazane tudi povprečne vrednosti prepustnosti.

4.3.3 Odzivni čas

Odzivni čas nam pove, koliko časa v povprečju porabi mehanizem za prevajanje omrežnega naslova. Povprečni odzivni čas je prikazan na grafu 4.5, iz katerega je razvidno, da je povprečni odzivni čas manjši pri Ecdysisu.



Slika 4.4: Rezultati testa meritve prepustnosti



Slika 4.5: Rezultati testa meritve odzivnega časa

Metrika	IPv4	Tayga	Ecdysis
Prepustnost [Mbits/s]	396	244	316
Odzivni čas [ms]	1,026	1,155	1,062
Fazno trepetanje [ms]	0,110	0,109	0,108

Tabela 4.1: Primerjava propustnosti, odzivnega časa in faznega trepetanja

4.3.4 Fazno trepetanje

Ta metrika nam pove, kako močno se razlikujejo zamiki paketov. Merili smo jo tako, da smo odčitali standardni odklon odzivnega časa. Ta informacija je pomembna predvsem za aplikacije v realnem času, za katere je bolje, da imajo prispeli paketi čim bolj enak zamik.

Iz grafa 4.6 je razvidno, da se fazno trepetanje ne razlikuje od tistega pri navadni povezavi IPv4. Iz tega lahko sklepamo, da prehodni mehanizmi ne vplivajo na delovanje aplikacij v realnem času.

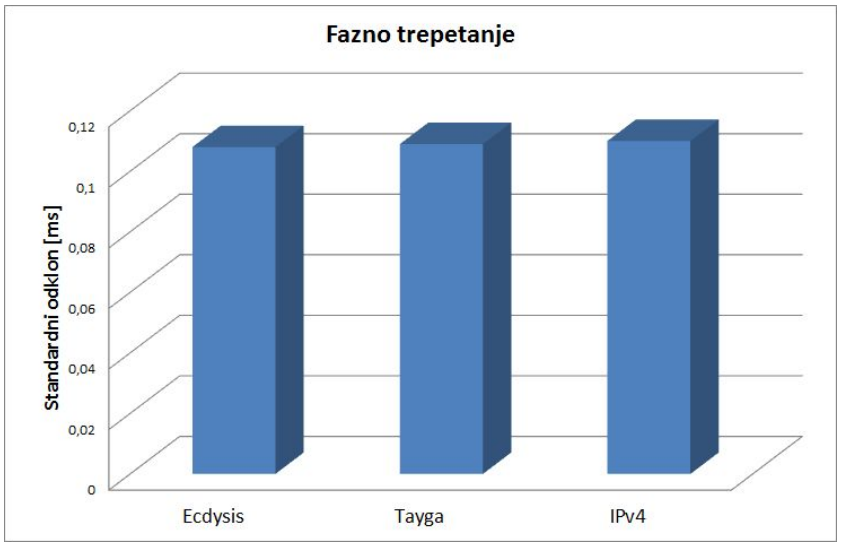
4.3.5 Potrditev ali ovržba prve hipoteze

Iz dobljenih rezultatov hitrosti delovanja vidimo, da je hitrejši Ecdysis. Razlog za to je lahko slaba implementacije Tayge, saj se ne naloži v jedro kot to stori Ecdysis.

Prve hipoteze ne moremo potrditi ali ovreči, saj mehanizma zaradi implementacije nista bila preizkušena na isti način. Preizkus bi bilo potrebno ponoviti, ko bosta obstajali podobni implementaciji NAT64 s stanji in NAT64 brez stanj.

4.4 Primerja prevajanja protokolov

V tem razdelku je opravljena primerjava, kateri protokoli delujejo bolje s Taygo in kateri z Ecdysisom. Namen primerjave je bila ugotovitev, katero vrsto NAT64 je bolje uporabiti, kadar uporabljamo določen protokol. Večino



Slika 4.6: Rezultati testa meritve odzivnega časa

podatkov o tem, kateri protokoli delujejo skupaj z Ecdysisom, smo pridobili iz vira [22]. Iz vira smo prav tako pridobili seznam protokolov, h kateremu smo dodali tudi nekaj svojih. Seznam s katerimi smo preizkušali delovanje, je prikazan v tabeli 4.2.

Za oba mehanizma smo naredili teoretično in empirično oceno. Prvo smo naredili tako, da smo predvideli, kako se bo obnesel protokol na podlagi njegovih lastnosti in načina delovanja. Drugo oceno smo dobili tako, da smo protokole dejansko preizkusili na virtualnem okolju.

Navedeni so tudi razlogi, zakaj se nekateri protokoli ne prevedejo in dodatni pogoji, ki jih moramo upoštevati pri uporabi določenega protokola. Prav tako je razloženo, zakaj nekatere aplikacije delujejo samo na Taygi, na Ecdysisu pa ne.

4.4.1 Rezultati

Rezultati preizkusa delovanja protokolov so prikazani v tabeli 4.3. Če je bil protokol dobro preveden, je označen z DP, če je bil pogojno preveden s PP in če je bil slabo preveden s SP. Dobro prevedeni protokoli so tisti, ki

Akronim	**Opis protokola**	**Aplikacija**
BitTorrent	P2P protokol za deljenje datotek	Odjemalec P2P
FTP	Protokol za prenos datotek	Odjemalec FTP
HTTP	Protokol za prenos hiperteksta	internetni brskalnik
HTTPS	Protokol za varni prenos hiperteksta	internetni brskalnik
IMAP	Protokol za dostop do internetne pošte	Poštni odjemalec
NTP	Mrežni časovni protokol	Operacijski sistem
POP3	Protokol za dostop do internetne pošte	Poštni odjemalec
RDP	Protokol oddaljenega namizja	Terminal
Skype	P2P VoIP protokol	Odjemalec instantnih sporočil, VoIP
MSNP	Mikrosoftov protokol za obvestila	Odjemalec instantnih sporočil, VoIP
SIP	Protokol za vzpostavitev seje	Odjemalec VoIP
CIFS	Protokol za internetni datotečni sistem	Operacijski sistem
SMTP	Protokol za dostop do internetne pošte	Poštni odjemalec
SSH	Odaljen dostop	Terminal
TELNET	Odaljen dostop	terminal
OpenVPN	Protokol za virtualna privatna omrežja	Odjemalec VPN
IPSec	Protokol za virtualna privatna omrežja	Odjemalec VPN
PPTP	Protokol za virtualna privatna omrežja	Odjemalec VPN
IRC	Protokol za klepetanje po internetu	Odjemalec IRC
MySQL	Protokol za podatkovno bazo MySQL	Odjemalec MySQL
HTTP server	Strežnik za protkol HTTP	Strežnik HTTP Apache
SSH server	Strežnik za protkol SSH	sshd

Tabela 4.2: Seznam aplikacij in protokolov, ki smo jih preizkusili

Akronim	Tayga		Ecdysis	
	Teoretična ocena	Empirična ocena	Teoretična ocena	Empirična ocena
BitTorrent	PP	PP	PP	PP
FTP	PP	PP	PP	PP
HTTP	DP	DP	DP	DP
HTTPS	DP	DP	DP	DP
IMAP	DP	DP	DP	DP
NTP	DP	DP	DP	DP
POP3	DP	DP	DP	DP
Skype	SP	SP	SP	SP
MSNP	SP	SP	SP	SP
SIP	DP	DP	SP	SP
CIFS	DP	DP	DP	DP
SMTP	DP	DP	DP	DP
SSH	DP	DP	DP	DP
Telnet	DP	DP	DP	DP
OpenVPN	PP	SP	PP	SP
IPSec	SP	SP	SP	SP
PPTP	SP	SP	SP	SP
IRC	DP	DP	DP	DP
MySQL	DP	DP	DP	DP
HTTP server	DP	DP	SP	SP
SSH server	DP	DP	SP	SP

Tabela 4.3: Rezultati preizkusa protokolov

se prevedejo brez problemov. Pogojno prevedeni so tisti, ki se običajno ne prevedejo, vendar jih lahko prevedemo pod pogojem, da uporabimo kakšen dodaten mehanizem, kot je recimo ALG. Slabo prevedeni protokoli pa so tisti, ki jih ni mogoče prevesti z uporabo dodatnih mehanizmov. Razloge za delovanje in nedelovanje protokolov smo povzeli po članku [22].

4.4.2 Dobro prevedeni protokoli

To so protokoli, ki so se prevedli brez težav, kar pomeni, da se je glava IPv4 paketa zamenjala z glavo IPv6 ali obratno. Kljub temu obstajajo primeri, ko ti protokoli odpovejo. To je predvsem takrat, kadar želimo komunicirati s strežniki, ki uporabljajo samo mrežni naslov brez imena. Takšni protokoli so HTTP, SSH, IRC in podobni. Zaradi tega je potrebno naslovom ročno doda-

jati predpono NAT64 ali pa uporabiti aplikacijski prehod, ki to avtomatsko naredi.

4.4.3 Pogojno prevedeni protokoli

Med pogojno prevedene protokole spadajo tisti, ki potrebujejo aplikacijski prehod ali kakšno drugo vrsto mehanizma, da se lahko prevedejo. To so predvsem tisti, ki uporabljajo omrežne naslove znotraj aplikacijske plasti. Protokola, ki potrebujeta aplikacijski prehod, sta FTP in BitTorrent.

4.4.4 Slabo prevedeni protokoli

Pod to skupino spadajo protokoli, ki v transportni plasti uporabljajo kakšen drug protokol kot TCP ali UDP. V tem primeru NAT64 odpove, saj prevaja samo tiste, ki temeljijo na TCP-ju, UDP-ju ali ICMP-ju. Slabo prevedeni so tudi tisti protokoli, ki ne podpirajo protokola IPv6.

Med takšne, ki ne podpirajo IPv6, spada Skype in MSNP, ki za komunikacijo s strežnikom uporabljata kar zakodirane naslove IPv4. Protokol OpenVPN je odpovedal, ker zaenkrat še ne podpira protokola IPv6.

Med tiste, ki uporabljajo druge protokole na transportni plasti, spadata IPSec in PPTP. IPsec uporablja ESP za enkapsulacijo transportnih protokolov, PPTP pa protokol GRE.

4.4.5 Razlike v delovanju protokolov pri Taygi in Ecdysisu

Iz tabele 4.3 je razvidno, da Ecdysis odpove, kadar vozlišče IPv6 nastopa kot strežnik in pri protokolu SIP. Razlog za to smo že opisali v poglavju 3, ko smo razlagali razlike med NAT64 s stanji in NAT64 brez stanj. V primeru uporabe NAT64 s stanji IPv4 vozlišča ne morejo ustvariti povezave proti omrežju IPv6, saj ne vedo, kakšna kombinacija naslova IPv4 in vrat se preslika v ciljni naslov IPv6.

4.4.6 Potrditev ali ovržba druge hipoteze

Iz rezultatov je razvidno, da NAT64 s stanji ne deluje, kadar hočemo komunicirati s strežnikom v omrežju IPv6 iz omrežja IPv4. Prav tako ne deluje protokol SIP, ki zahteva vzpostavitev seje iz smeri IPv6, kot tudi iz smeri IPv4. Posledično drugo hipotezo lahko potrdimo.

Poglavje 5

Izdelava aplikacijskega prehoda FTP64

V poglavju 4 je opisano, da obstajajo takšni protokoli, ki potrebujejo aplikacijski prehod, kadar hočemo, da delujejo skupaj z NAT64. Med takšne spada tudi protokol za prenos datotek ali FTP.

V tem poglavju je opisana izdelava aplikacijskega prehoda FTP64 za uporabo NAT64 brez stanj. Poglavje je razdeljeno na tri dele: načrtovanje, izvedba in namestitev aplikacijskega prehoda. V načrtovanju je predstavljena teorija, ki jo je potrebno poznati o aplikacijskem prehodu FTP64. Predstavljene so osnove delovanja protokola FTP in v katerem koraku njegovega delovanja se pojavijo težave, ko poskušamo komunicirati s strežnikom IPv4. Naštete so vse potrebne zahteve, ki jih mora aplikacijski prehod izpolnjevati, da lahko protokol FTP deluje normalno. V drugem delu, v izvedbi, je opisano, kako je bil aplikacijski prehod dejansko implementiran. V namestitvi pa je opisano, kako se aplikacijski prehod namesti na računalnik in kako se ga uporablja.

5.1 Načrtovanje aplikacijskega prehoda

5.1.1 Protokol za prenos datotek

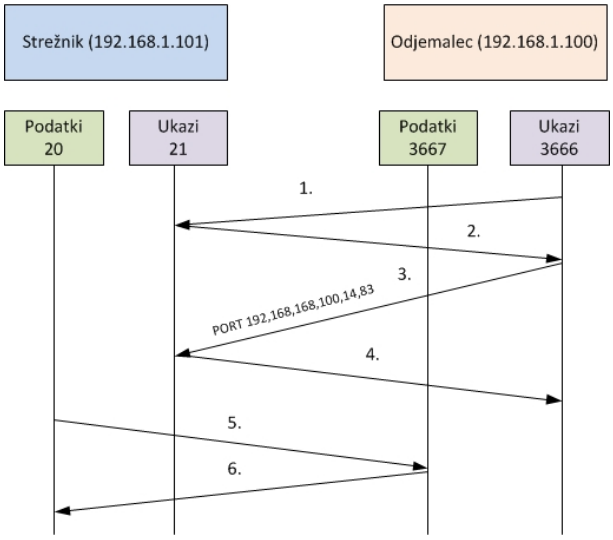
V tem razdelku so opisane osnove protokola za prenos datotek, ki jih je potrebno razumeti za implementacijo aplikacijskega prehoda. Protokol za prenos datotek se primarno, kot je že iz imena razvidno, uporablja za prenos datotek. Osnovan je na protokolu TCP in temelji na programski arhitekturi odjemalec strežnik. Informacije o protokolu FTP smo pridobili iz virov [26, 19].

Posebnost protokola FTP je, da uporablja dve povezavi ali kanala med odjemalcem in strežnikom. Prvi kanal se imenuje kontrolni kanal (ang. control channel) in se uporablja za pošiljanje ukazov, kot je recimo ukaz za pridobivanje seznama datotek. Odjemalec pošilja po kontrolnem kanalu ukaze sestavljene iz običajno štirih velikih tiskanih črk in argumenta (recimo USER anonymous). Strežnik odjemalcu odgovori s trimestno kodo in človeku berljivim opisom kode (recimo 331 Please specify the password.). Podobno kot pri protokolu HTTP ima vsaka števka v odgovoru svoj pomen. Kontrolni kanal vzpostavi odjemalec proti strežniku, ki običajno posluša na vratih 21.

Drugi kanal se imenuje podatkovni kanal (ang. data channel) in je namenjen prenosu datotek od strežnika do odjemalca ali v obratni smeri. Za razliko od kontrolnega kanala je lahko podatkovni ustvarjen tako s strani odjemalca kot tudi s strani strežnika. Kadar je podatkovni kanal ustvarjen s strani strežnika proti odjemalcu, to imenujemo aktivni način delovanja. Ko pa je ustvarjen s strani odjemalca, pa mu pravimo pasivni način. Ker je razumevanje aktivnega in pasivnega načina pomembno pri implementaciji aplikacijskega prehoda, je v nadaljevanju podroben opis posameznega načina.

Aktivni način

Kot že omenjeno se aktivni način uporablja takrat, ko strežnik vzpostavi podatkovni kanal proti odjemalcu. Vzpostavitev aktivnega načina je predstavljena z naslednjimi koraki:



Slika 5.1: Potek vzpostavitve aktivnega načina pri protokolu FTP

1. Odjemalec (recimo 192.168.1.100) ustvari kontrolni kanal iz naključnih prostih nepriviligiranih ($N > 1024$) vrat (recimo 3666) proti strežniku (recimo 192.168.1.101), ki posluša na vratih 21.

2. Strežnik odgovori odjemalcu s potrditvijo.

3. Odjemalec pošlje po kontrolnem kanalu ukaz “PORT a1,a2,a3,a4,p1,p2”, kjer predstavljajo simboli “a1-a4” mrežni naslov odjemalca IPv4 ločen z vejicami, “p1” in “p2” pa vrata namenjena za podatkovni kanal. Vrata so enaka $N + 1$, torej v našem primeru 3667. “P1” dobimo tako, da vrata delimo z 256, “p2” pa je ostanek tega deljenja. V našem primeru bi bil ukaz enak “PORT 192,168,168,100,14,83”.

4. Strežnik odgovori s potrditvijo.

5. Strežnik ustvari podatkovni kanal iz vrat 20 proti odjemalcu na vrata $N + 1$, torej 3667.

6. Odjemalec odgovori s potrditvijo.

Kadar uporabljamo IPv6, je ukaz PORT neuporaben, saj vanj ne moremo vključiti naslova IPv6. Zaradi tega uporabljajo novejši strežniki in odjemalci ukaz EPRT (Extendend PORT). Za mrežni naslov IPv6 "2001:db8:1234:ffff:c0a8:202" in enaka vrata kot pri primeru PORT bi odjemalec poslal ukaz "EPRT |2|2001:db8:1234:ffff:c0a8:202|3666|". Ukaz EPRT lahko uporabimo tudi z IPv4. Kadar ga uporabimo z IPv4, je namesto prve dvojke v ukazu ena. Namesto ukaza PORT, bi lahko naš odjemalec v primeru s slike 5.1 poslal ukaz "EPRT |1|192.168.1.100|3666|".

Pasivni način

Aktivni način se izkaže za neuporabnega, kadar je med odjemalcem in strežnikom požarni zid ali NAT. V tem primeru strežnik ne more vzpostaviti podatkovne povezave proti odjemalcu, saj so vrata bodisi zaprta zaradi požarnega zidu ali pa je naslov nedosegljiv, ker je preslikan. Zaradi tega je bila kasneje dodana možnost pasivnega načina delovanja, ki omogoča vzpostavitev podatkovnega kanala s strani odjemalca. Pasivni način je danes pogosteje uporabljen, saj je redko na voljo direktna povezava brez požarnih zidov in NAT-ov [19].

Koraki pri pasivnem načinu so sledeči:

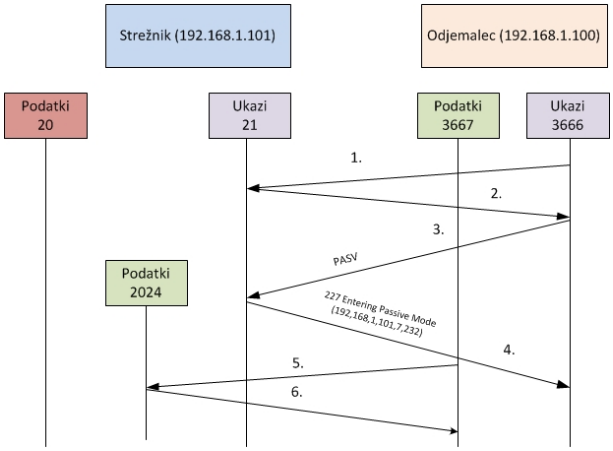
1. Odjemalec (recimo 192.168.1.100) ustvari kontrolni kanal iz naključnih prostih nepriviligiranih ($N > 1024$) vrat (recimo 3666) proti strežniku (recimo 192.168.1.101), ki posluša na vratih 21.

2. Strežnik odgovori odjemalcu s potrditvijo.

3. Odjemalec pošlje po kontrolnem kanalu ukaz PASV.

4. Če je potek procesa brezhiben, odgovori strežnik odjemalcu z odgovorom "227 Entering Passive Mode (a1,a2,a3,a4,p1,p2)". Struktura odgovora je podobna kot pri ukazu PORT. Strežnik namesto vrat 20 odpre neka druga naključna prosta vrata M.

5. Odjemalec se poveže iz vrat $N + 1$ proti strežniku na vrata M.



Slika 5.2: Potek vzpostavitve pasivnega načina pri protokolu FTP

6. Strežnik mu odgovori s potrditvijo.

Prav tako ima tudi PASV svojo alternativo za IPv6. Ukaz se imenuje EPSV (Extended PASV), ki mu ponavadi sledi odgovor "229 Entering Extended Passive Mode (|||p|)", kjer je "p" cela številka vrat. Odgovor se glede na PASV razlikuje tudi v tem, da ne vsebuje naslova strežnika. To je zaradi tega, ker se pričakuje, da se bo odjemalec s podatkovnim kanalom povezal na isti strežnik kot s kontrolnim kanalom. V nasprotnem primeru bi lahko šlo za tako imenovan napad "Man-in-the-middle", ki ga dopušča ukaz PASV. Prav tako kot EPRT se lahko tudi EPSV uporablja z vozlišči IPv4.

Razlogi za nedelovanje FTP ob uporabi NAT64

Glavna razloga za nedelovanje FTP, kadar želimo z odjemalcem v omrežju IPv6 vzpostaviti povezavo s strežnikom v omrežju IPv4, sta nepodprta ukaza EPRT in EPSV v omrežju IPv4 in preslikava mrežnih naslovov, ki jo opravi NAT64. Kljub temu da lahko ukaza EPRT in EPSV uporabljamo tudi z naslovi IPv4, je dejansko malo strežnikov, ki bi imeli to opcijo omogočeno ali implementirano. Kot je navedeno v viru [8] se zato še vedno v večini uporabljata PORT in PASV.

Kadar želi odjemalec IPv6 vzpostaviti povezavo s strežnikom IPv4, mu pošlje, v primeru aktivnega načina, ukaz EPRT. Ker strežnik IPv4 ne razume ukaza EPRT, se podatkovni kanal ne more ustvariti. Podobna situacija se zgodi pri pasivnem načinu. Zato je glavna funkcionalnost, ki jo mora aplikacijski prehod omogočati, prevajanje ukaza EPRT v PORT oziroma prevajanje ukaza EPSV v PASV. Poleg tega mora aplikacijski prehod omogočati preslikavo mrežnih naslovov, ki so vsebovani znotraj ukazov EPRT in EPSV.

5.1.2 Lastnosti aplikacijskega prehoda

V tem razdelku so opisane glavne lastnosti, ki jih mora imeti aplikacijski prehod, kot jih predlaga dokument [8]. Kjer je predlagano več možnosti, je argumentirano, zakaj smo izbrali določeno možnost.

Prevajanje kontrolnega kanala

Aplikacijski prehod mora poslušati na vratih 21 in prestrezati vse povezave proti tem vratom. Kontrolni kanal mora biti implementiran kot protokol Telnet [18], da lahko aplikacijski prehod prejema in posreduje ukaze in odgovore.

Obstajata dve možnosti implementacije kontrolnega kanala:

1. Aplikacijski prehod prestreže sejo IPv6 proti vozlišču IPv4 in ustvari novo sejo IPv4 proti vozlišču IPv4.

2. Paketi, ki so del kontrolnega kanala, so prevedeni individualno na omrežni plasti.

Druga možnost zahteva od aplikacijskega prehoda spreminjanje sekvenčne številke TCP in deljenje paketov TCP na manjše. Paketi lahko zaradi spremembe dolžine na transportni plasti razpadejo na dva dela. Izbrali smo prvo možnost, ker bo implementacija bolj učinkovita, saj bo manj programske kode.

Kadar pošlje odjemalec ukaz AUTH, pomeni, da se želi pogajati o varnostnih mehanizmih s strežnikom. To je nezdružljivo z aplikacijskim prehodom, saj sta v tem primeru kontrolni in podatkovni kanal nevidna. V tem primeru lahko aplikacijski prehod naredi sledeče:

1. Transparentno posreduje ukaze in podatke, da se lahko pogajanja o varnostnih mehanizmih normalno zaključijo.

2. Blokira pogajanja o varnostnem mehanizmu, čemur najverjetneje sledi prekinitev povezave s strani odjemalca ali strežnika.

3. Aplikacijski prehod naj se ločeno pogaja z odjemalcem in strežnikom o varnostnih mehanizmih.

V dokumentu je predlagana implementacija prve možnosti. Mi se ji bomo zaradi zahtevnosti izognili, saj nima bistvenega pomena pri delovanju aplikacijskega prehoda.

Jezikovno pogajanje

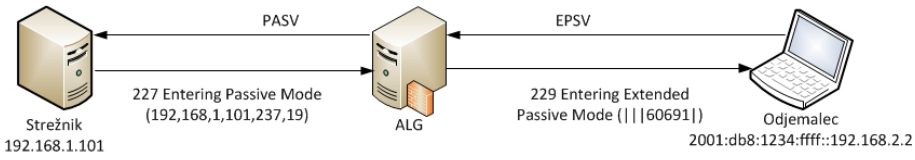
Kadar odjemalec pošlje ukaz LANG, pomeni, da se hoče pogajati o jezikovnih nastavitvah kontrolnega kanala. Težava nastane, ker ima strežnik vzpostavljeno ločeno sejo z aplikacijskim prehodom, ta pa naprej svojo sejo z odjemalcem. Velika verjetnost je, da aplikacijski prehod ne bo podpiral želenega jezika. V taki situaciji dokument predlaga naslednje rešitve:

1. Nadziraj ukaz LANG in pošiljaj besedilo v želenem jeziku, če je na voljo, sicer uporabi privzetega.

2. Ne nadziraj ukaza LANG. Besedilo je poslano v privzetem jeziku.

3. Blokiraj ukaz LANG, tako da strežniku posreduješ ukaz NOOP in njegov odgovor spremeni v odgovor s kodo 502, ki pomeni nepodprt ukaz.

Ker ukaz LANG ni ključnega pomena pri delovanju protokola FTP ga nismo implementirali.



Slika 5.3: Prevajanje EPSV v PASV

Prevajanje EPSV v PASV

Aplikacijski prehod mora ukaz EPSV prevesti v ukaz PASV in spremeniti odgovor strežnika iz kode 227 v kodo 229.

Pri prevajanju ukaza EPSV v PASV ločimo naslednje tri situacije:

- Kadar odjemalec pošlje ukaz EPSV, se ta prevede v PASV. Če je naslov strežnika na primer "192.168.1.101", bo aplikacijski prehod nato prevedel odgovor "227 Entering Passive Mode (192,0,2,31,237,19)" v odgovor "229 Entering Extended Passive Mode (|||60691|)". Aplikacijski prehod naj ignorira naslov IPv4 in naj ga ne posreduje odjemalcu. Če je naslov IPv4 drugačen kot naslov IPv4 kontrolnega kanala, mora aplikacijski prehod odjemalcu odgovoriti z "425 Can't open data connection".

- Kadar odjemalec pošlje ukaz EPSV z drugačnim številčnim argumentom kot 2, mora aplikacijski prehod odgovoriti z "522 error" in ne sme posredovati ukaza naprej proti strežniku.

- Kadar odjemalec pošlje ukaz EPSV ALL, mora aplikacijski prehod odgovoriti s "504 Command not implemented for that parameter". To prepreči situacijo, kadar želi odjemalec uporabljati samo EPSV, strežnik pa se negativno odzove na ukaz PASV.

Prevajanje EPRT v PORT

Aplikacijski prehod mora ukaz EPRT prevesti v ukaz PORT. Ker mora prevesti tudi naslov odjemalca, je implementacija EPRT v PORT zahtevnejša

kot EPSV v PASV, saj mora poznati način, kako se naredi preslikava. Aplikacijski prehod mora različno obravnavati prevode, kadar se uporablja NAT64 brez stanj in kadar se uporablja NAT64 s stanji. Ker je naša implementacija namenjena samo za NAT64 brez stanj, bomo razložili kako se ukaz prevede v tem primeru.

Tayga uporablja tri načine preslikovanja naslovov. Prvi način je statičen, kjer preslikovanje določi administrator. Pri drugem načinu si Tayga naključno izbere naslov IPv4, ki nato pripada naslovu IPv6 nekega odjemalca. V tretjem načinu pa se preslikava zgodi algoritmično, kot je opisano v razdelku 3.1.1. Ker pri prvem in drugem načinu aplikacijski prehod ne more vedeti, kakšen naslov IPv6 se preslika v naslov IPv4, smo implementirali samo preslikovanje za tretji način.

Delovanje bi lahko ilustrirali z naslednjim primerom. Če je naslov našega odjemalca "2001:db8:1234:ffff::192.168.2.2", bo odjemalec poslal ukaz "EPRT |2|2001:db8:1234:ffff::192.168.2.2|5282|", ki ga bo aplikacijski prehod prevedel v "PORT 192,168,2,2,20,162". Številka vrat mora pri NAT64 brez stanj ostati ista.

Kadar odjemalec pošlje naslov IPv6, ki ni njegov oziroma naslov IPv4, je delovanje aplikacijskega prehoda nedefinirano. Predlagane so naslednje tri rešitve:

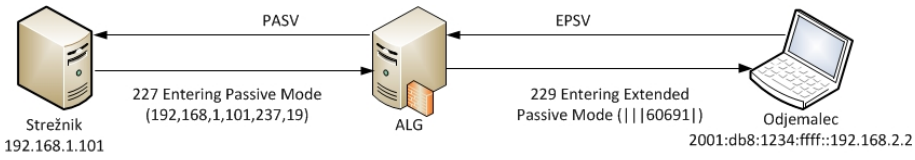
1. Aplikacijski prehod naj posreduje ukaz nespremenjen.

2. Aplikacijski prehod naj odgovori z napako.

3. Aplikacijski prehod naj poskuša narediti ustrezen prevod.

V naši implementaciji je narejeno tako, da v primeru naslova IPv4 posreduje ukaz nespremenjen. Kadar je naslov IPv6, ga bo v vsakem primeru poskušala prevesti.

Ko odjemalec pošlje ukaz EPRT, ki vsebuje naslov IPv4 ali naslov IPv6, ki je odjemalčev, ampak nima prevoda, mu aplikacijski prehod odgovori z "425 Can't open data connection".



Slika 5.4: Prevajanje EPRT v PORT

Privzeto prevajanje za vrata 20

Kadar odjemalec ne pošlje niti EPSV niti EPRT, preden se začne prenos podatkov, pomeni, da bo strežnik vzpostavil aktiven način iz vrat 20 proti istim vratom, kot jih uporablja odjemalec za kontrolni kanal. Pri NAT64 brez stanj to ne predstavlja večjih težav. Težje je to narediti pri NAT64 s stanji, saj bi to zahtevalo dodatno reprogramiranje NAT64. Ker je Tayga NAT64 brez stanj, nam tega ni bilo potrebno narediti.

Aktiven in pasiven način hkrati

Ob posredovanju ukaza PORT in kasneje PASV ali obratno je delovanje aplikacijskega prehoda nedefinirano. V dokumentu je predlagano, naj aplikacijski prehod ne zaznava takšne situacije in naj normalno prevaja EPSV v PASV in EPRT v PORT, kar smo tudi implementirali.

Običajno delovanje

Kadar pošlje odjemalec ukaz, ki ni namenjen temu, da ga aplikacijski prehod prevede, naj ga pošlje transparentno naprej proti strežniku. Primer takšnega delovanja je, kadar odjemalec pošlje ukaz PASV ali PORT.

Ukaz ALGS

Aplikacijski prehod mora imeti podprt tudi novi ukaz ALGS, ki odjemalcu omogoča nastavljanje aplikacijskega prehoda in poizvedovanje o njegovem stanju. Ukaz ALGS ne sme biti posredovan naprej proti strežniku.

Kadar odjemalec pošlje ukaz ALGS z argumentom STATUS64, so možni naslednji odgovori:

1. 216 NONE: aplikacijski prehod ne bo prevedel niti EPSV niti EPRT.
2. 216 EPSV: aplikacijski prehod bo prevedel EPSV v PASV, EPRT ni preveden v PORT.
3. 216 EPSVEPRT: aplikacijski prehod bo prevedel EPRT v PORT in EPSV v PASV.

Odjemalec lahko posreduje tudi sledeče argumete:

1. ALGS DISABLE64: odjemalec zahteva, naj aplikacijski prehod ne prevede niti EPRT niti EPSV.
2. ALGS ENABLE64: odjemalec zahteva, naj aplikacijski prehod prevede tako EPRT v PORT kot tudi EPSV v PASV.

Časovna kontrola in prevajanje v ukaz NOOP

Če je možno, naj ne preteče časovna kontrola. Zaradi tega je za kontrolni kanal priporočen čas vsaj 30 sekund. Kadarkoli se ukaz ne posreduje do strežnika, mora aplikacijski prehod strežniku poslati ukaz NOOP, da se ohrani sinhronizacija med strežnikom in odjemalcem. Ko dobi odgovor od strežnika s kodo 200, ga prilagodi, da ustreza ukazu, ki ga je poslal odjemalec in ga pošlje naprej proti odjemalcu. Če strežnik odgovori z odgovorom različnim od 200, prekine kontrolni kanal in odjemalcu sporoči napako.

5.2 Izvedba aplikacijskega prehoda

Aplikacijski prehod smo naredili tako, da smo nadgradili odprtokodni namestniški strežnik ftp.proxy [28], ki je delo avtorjev Wolfganga Zekolla in Andreasa Schoenberga. Izdan je pod licenco “ GNU General Public License” [29],

zato smo imeli vse pravice za spreminjanje njegove programske kode. Odprtokodni namestniški strežnik smo izbrali zato, ker smo že vnaprej pridobili kar nekaj funkcionalnosti, ki jih mora aplikacijski prehod imeti. Te funkcionalnosti so vzpostavitev kontrolnega in podatkovnega kanala, posredovanje ukazov, prekinitev seje ob morebitni napaki itd. Zaradi tega smo imeli precej manj dela, saj smo se lahko osredotočili le še na prevajanje ukazov in ukaz ALGS.

Namestniški strežnik ftp.proxy je napisan za uporabo z operacijskim sistemom Linux. Namestimo pa ga lahko tudi na operacijske sisteme OpenBSD in Windows. Kot edino slabost lahko navedemo, da strežnik ne podpira protokola IPv6 in to, da je v programski kodi obilo nemških komentarjev. Za namestniški strežnik smo se odločili tudi zato, ker nam omogoča spreminjanje ukazov na aplikacijskem nivoju in nam zaradi tega ni treba skrbeti za sekvenčne številke ter razbijanje na manjše segmente na transportnem nivoju.

V nadaljevanju je prikazano, kako so bile implementirane glavne funkcionalnosti, ki jih mora imeti aplikacijski prehod FTP64.

5.2.1 Podpora IPv6

Ko smo programirali aplikacijski prehod, smo imeli v mislih, da bo postavljen na odjemalcu. Tako smo se odločili, ker mora aplikacijski prehod omogočati, da lahko uporabniki s pomočjo ukaza ALGS nastavijo, katere ukaze naj prevaja. Zaradi takšne postavitve smo morali zagotoviti podporo IPv6 tako iz smeri od odjemalca do namestniškega strežnika kot tudi v obratni smeri. Odjemalec ne uporablja dvojnega sklada, torej nima podprtega protokola IPv4. Kljub temu je aplikacijski prehod implementiran tako, da ga še vedno lahko uporabimo kot navaden namestniški strežnik.

Podporo IPv6 smo naredili tako, da smo implementirali posebno funkcijo, ki glede na to, ali je argument funkcije naslov IPv4 ali naslov IPv6, vrne ustrezen odgovor. Nato smo v vseh ostalih delih programske kode, kjer so programirane vtičnice (ang. socket), glede na vrnjen odgovor določili, ali naj

vzpostavimo vtičnico IPv4 ali IPv6.

Da lahko aplikacijski prehod ve, kakšna predpona NAT64 in dolžina je v uporabi, smo morali dodati, da program ob inicializaciji prebere te informacije iz nastavitvene datoteke.

5.2.2 EPSV in prevajanje EPSV v PASV

Za implementacijo ukaza EPSV in prevajanja ukaza EPSV v PASV smo nadgradili funkcijo za vzpostavitev pasivnega načina, ki je že bila implementirana v ftp.proxy-ju. Morali smo implementirati tako navaden EPSV kot tudi prevajanje iz EPSV v PASV, saj lahko uporabnik prevajanje izklopi. Ker je prevajanje EPSV v PASV kompleksnejše kot sam ukaz EPSV, bomo glavne korake algoritma opisali samo za prvo možnost:

1. Najprej aplikacijski prehod prestreže ukaz EPSV in se na podlagi tega, ali je prevajanje vklopljeno ali ne, odloči za ustrezno funkcijo.

2. Kadar je prevajanje vklopljeno, pošlje strežniku po kontrolnem kanalu ukaz PASV in prestreže njegov odgovor.

3. Iz odgovora izlušči naslov IPv4 in številko vrat.

4. Ko to naredi, mora naslov IPv4 spremeniti v naslov IPv6 s pomočjo predpone NAT64 in algoritma, da se aplikacijski prehod sploh lahko poveže na strežnik.

5. Aplikacijski prehod nato odpre svoja vrata za podatkovni kanal. Svoj naslov in številko vrat pošlje odjemalcu v obliki odgovora EPSV.

6. Aplikacijski prehod nato čaka, da se odjemalec poveže s podatkovnim kanalom nanjo.

7. Ko se odjemalec poveže s podatkovnim kanalom, ustvari še sam podatkovni kanal proti strežniku FTP.

5.2.3 EPRT in prevajanje EPRT v PORT

Podobno kot ukaz EPSV smo EPRT oziroma prevajanje EPRT v PORT naredili tako, da smo nadgradili obstoječi ukaz PORT. Glavni koraki algoritma za prevajanje iz EPRT v PORT so sledeči:

1. Najprej prestreže odjemalčev ukaz EPRT in shrani odjemalčev naslov IPv6 (ker gre za isti računalnik, bo to ::1) ter številko vrat za kasneje.
2. Nato pretvori naslov aplikacijskega prehoda (dejanski naslov odjemalca) v IPv4. To naredi s pomočjo predpone NAT64 in algoritma.
3. Nato preoblikuje številko IPv4, da je primerna za ukaz PORT, odpre vrata in pošlje ukaz PORT proti strežniku.
4. Aplikacijski prehod nato čaka, da se strežnik poveže nanj.
5. Ko se strežnik FTP uspešno priključi na aplikacijski prehod, se sam nato še poveže na naslov in vrata, ki jih je prej pridobil od odjemalca.

5.2.4 Ukaz ALGS

Ukaz ALGS nam omogoča, da lahko odjemalec preko kontrolnega kanala nastavi prevajanje ukazov oziroma poizveduje, kakšne nastavitve so trenutno nastavljene. Za nastavitev smo uporabili logični zastavici, ki se lahko bereta oziroma nastavita na želeno vrednost. Algoritem je sledeč:

1. Če je argument enak STATUS64, preverimo, kateri zastavici sta postavljeni:
   (a) Če sta obe zastavici postavljeni, odgovorimo z EPSVEPRT.
   (b) Če je postavljena samo zastavica EPSV, odgovorimo z EPSV.
   (c) Če ni postavljena nobena zastavica, odgovorimo z NONE.
2. Če je argument enak ENABLE64, se obe zastavici postavita na 1.
3. Če je argument enak DISABLE64, se obe zastavici postavita na 0.

5.3 Namestitev na računalnik in uporaba

V prihodnjem razdelku je opisana namestitev na operacijski sistem Linux in uporaba aplikacijskega prehoda. Zaradi lažje razumljivosti so pripete tudi zaslonske slike.

5.3.1 Namestitev

Aplikacijski prehod prenesemo na svoj računalnik in ga razširimo v poljuben imenik. Ko to storimo, naredimo standardna dva koraka pri nameščanju programov na operacijski sistem Linux:

```
make
make install
```

Ko to naredimo, se nam bo običajno v imenik "/usr/local/sbin/" namestil program ftp.proxy . Ftp.proxy se ne zažene kot običajen daemon, ampak ga moramo zagnati s pomočjo super strežnika. Super strežnik je strežnik, ki ga običajno uporabljamo za zaganjanje ostalih strežnikov oziroma daemonov, kot je recimo sshd, ftpd, httpd itd. Če uporabljamo super strežnik Xinetd, moramo v nastavitveno datoteko xinetd.conf dodati sledeče:

```
service ftp
        {
         socket_type = stream
         wait        = no
         flags       = IPv6
         user        = nobody
         server      = /usr/local/sbin/ftp.proxy
         server_args = –m -e -f /etc/ftp.proxy.conf
        }
```

Predvsem sta pomembni predzadnja in zadnja argumenta. S predzadnjim povemo, da lahko odjemalec sam določi, s katerim strežnikom se bo povezal in nam ni treba tega nastavljati v nastavitveni datoteki. Zadnji nam pove,

iz katere datoteke bomo brali nastavitve. Pomembna je predvsem zato, ker mora aplikacijski prehod obvezno dobiti predpono NAT64. Za ostale zastavice priporočamo, da si pogledate priročnik za ftp.proxy.

Ko nastavimo xinetd.conf, moramo nastaviti še nastavitveno datoteko za ftp.proxy, saj mora aplikacijski prehod obvezno prebrati predpono NAT64. Primer nastavitve te datoteke bi bil sledeč:

```
[::1]
  selectserver    yes
  monitormode     yes
  nat64prefix     2001:db8:1234:ffff::/96
```

Pri tem je pomembno navesti, da so te nastavitve namenjene za localhost IPv6 ([::1]) in da ne pozabimo predpone NAT64 (nat64prefix). Ko opravimo vse te korake, lahko začnemo z uporabo.

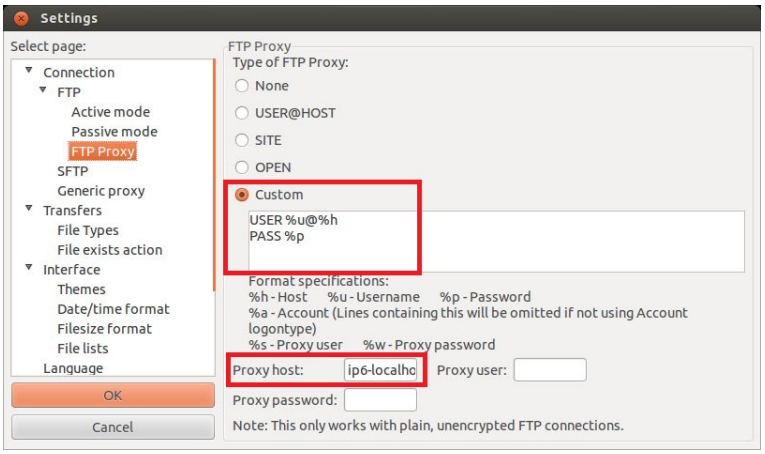
5.3.2 Uporaba

Prikazani so primeri uporabe za odjemalec FTP Filezilla. Za ostale odjemalce bi morale biti nastavitve podobne. Prvi korak, ki ga moramo storiti, je nastavitev namestniškega strežnika, kot je prikazano na sliki 5.5. Pomembno je, da izberimo poljuben tip namestniškega strežnika ter ustrezne nastavitve za uporabniško ime in geslo:

```
USER %u@%h
PASS %p
```

To je pomembno predvsem zato, ker lahko na ta način ftp.proxy izlušči strežnik iz uporabniškega imena. Pomembno je tudi, da za naslov namestniškega strežnika nastavimo ip6-localhost ali "::1" namesto localhost. Odjemalec bo tako uporabil IPv6 namesto IPv4.

Privzeti način delovanja pri Filezilli je pasivni način. Ciljni strežnik FTP vpišemo v polje gostitelj. Strežnik je lahko predstavljen kot številka IPv6 ločena z oglatimi oklepaji ali pa z imenom. Če je strežnik anonimen, kliknemo na poveži in povezava bo vzpostavljena. V nasprotnem primeru moramo



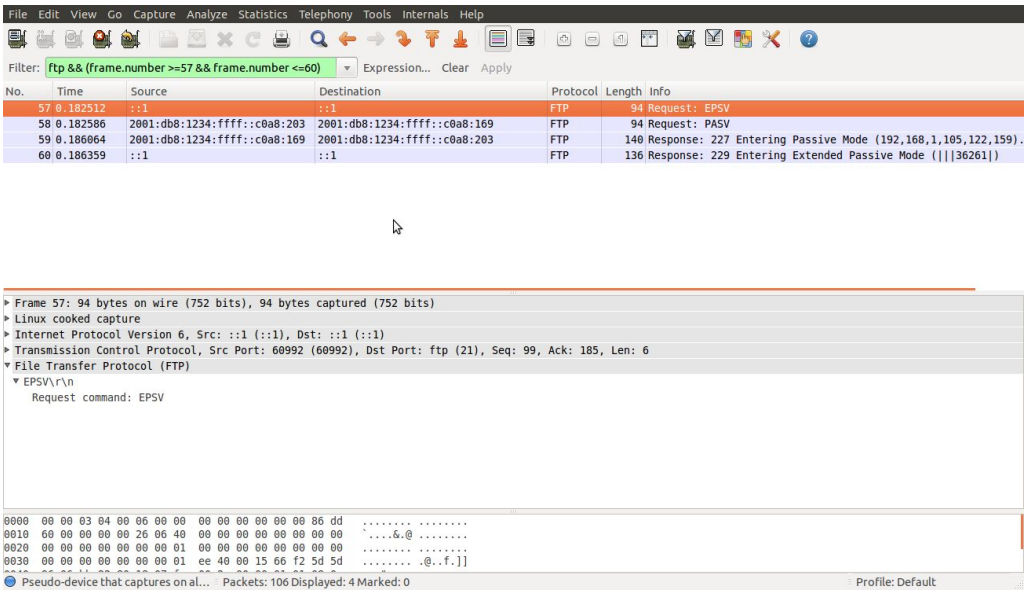
Slika 5.5: Nastavitev namestiškega strežnika v Filezilli

prej vpisati še uporabniško ime in geslo. Primer pasivnega načina oziroma prevajanja iz EPSV v PASV je prikazan na sliki 5.6.

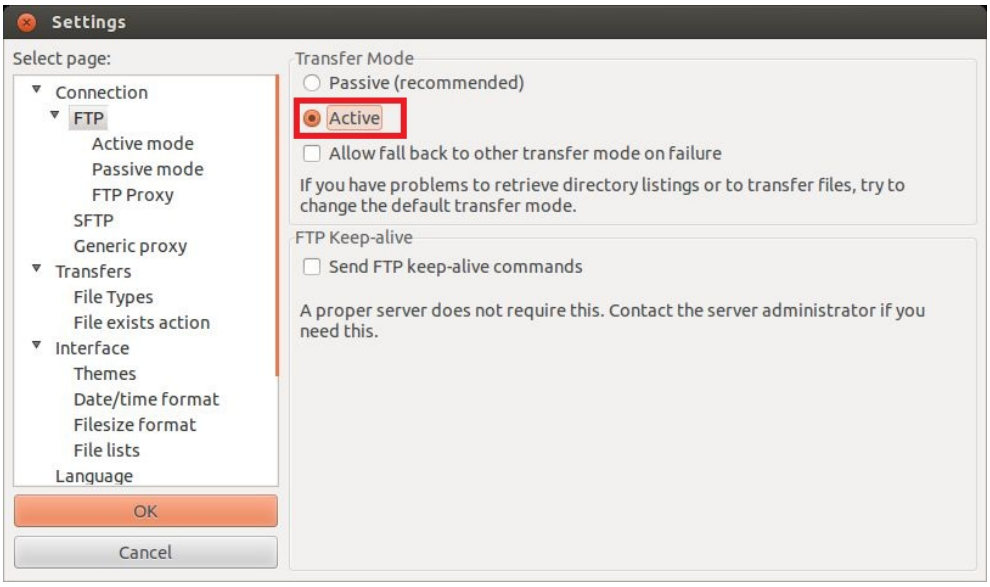
Če hočemo uporabljati aktivni način, moramo narediti nastavitev na odjemalcu. Primer nastavitve v Filezilli je prikazan na sliki 5.7. Velja omeniti, da moramo imeti za aktivni način ustrezno omrežje, torej brez požarnih zidov in ostalih NAT-ov. Dokaz, da prevajanje deluje, je prikazan na sliki 5.8. V našem primeru smo se povezali na strežnik v lokalnem omrežju. Za gostitelja smo vnesli kar številko naslova IPv6, ki smo jo dobili tako, da smo ročno dodali naslovu IPv4 predpono NAT64.

5.3.3 Potrditev ali ovržba tretje hipoteze

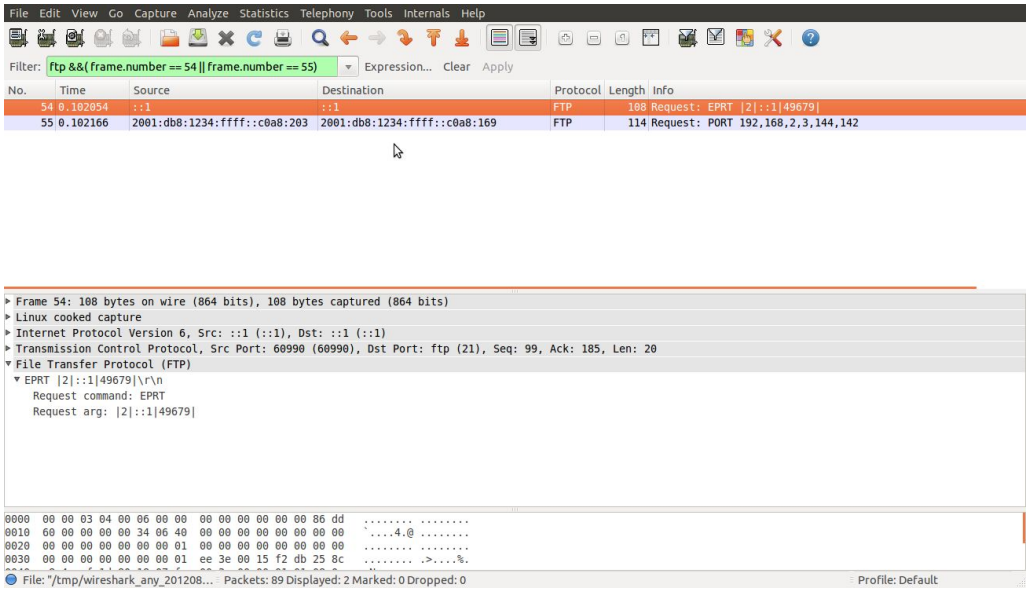
Aplikacijski prehod prevaja naslove in ukaze, kar lahko vidimo v slikovnem prikazu 5.6 in 5.8 . Zato lahko potrdimo tudi tretjo hipotezo iz poglavja 3.



Slika 5.6: Dokaz, da se EPSV prevede v PASV



Slika 5.7: Nastavitev aktivnega načina v Filezilli



Slika 5.8: Dokaz, da se ukaz EPRT prevede v PORT

Poglavje 6

Sklepne ugotovitve

V diplomski nalogi je bila narejena primerjava NAT64 s stanji in NAT64 brez stanj oziroma primerjava implementacij Ecdysisa in Tayge. Narejen je bil tudi aplikacijski prehod za protokol FTP, kadar uporabljamo NAT64 brez stanj.

Primerjava prehodnih mehanizmov NAT64 je pokazala, da je NAT64 s stanji bolj uporaben, kadar imamo več odjemalcev v omrežju IPv6 in želijo uporabljati storitev strežnika, ki je v omrežju IPv4. NAT64 brez stanj pa je bolj uporaben, kadar želimo imeti strežnik dostopen iz omrežja IPv4 ali pa uporabljamo aplikacije, ki potrebujejo vzpostavitev seje tako iz smeri IPv4 proti IPv6 kot obratno. S temi ugotovitvami si lahko pomagajo zlasti skrbniki omrežij, ki bi želeli postaviti NAT64, ne vedo pa, katera vrsta je bolj primerna za njihovo omrežje.

Izdelava aplikacijskega prehoda je prispevala prvi aplikacijski prehod takšne vrste, saj ga v času nastajanja diplome še ni bilo. Pri izdelavi je bilo prikazano, kako se na enostaven način s pomočjo namestniškega strežnika naredi pretvorba ukazov na aplikacijski plasti, kar nam olajša tudi razvoj aplikacijskih prehodov za druge protokole, kot je recimo BitTorrent.

Literatura

[1] R. AlJa'afreh, J. Mellor, I. Awan, "Implementation of IPv4/IPv6 BDMS Translation Mechanism", v zborniku: *UKSIM European Symposium on Computer Modeling and Simulation*, Liverpool, sept. 2008, str. 512–517.

[2] R. AlJa'afreh, J. Mellor, I. Awan, "Evaluating BDMS and DSTMTransition Mechanisms", v zborniku: *UKSIM European Symposium on Computer Modeling and Simulation*, Liverpool, sept. 2008, str. 488–493.

[3] R. AlJa'afreh, J. Mellor, I. Awan, "A Comparison Between the Tunneling Process and Mapping Schemes for IPv4/IPv6 Transition", v zborniku: *International Conference on Advanced Information Networking and Applications*, Bradford, maj 2009, str. 601–606.

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

[5] M. Bagnulo, A. Sullivan, P. Matthews, I. van Beijnum. "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, apr. 2011.

[6] C. Bao, C. Huitema, M. Bagnulo, M. Boucadair, X. Li, "IPv6 Addressing of IPv4/IPv6 Translators", RFC 6052, okt. 2010.

[7] C. Bao, X. Li, Y. Zhai, W. Shang, "dIVI: Dual-Stateless IPv4/IPv6 Translation", draft-xli-behave-divi-04 (IETF Internet Draft, expired), maj 2012.

[8] I. van Beijnum, " An FTP Application Layer Gateway (ALG) for IPv6-to-IPv4 Translation", RFC 6384, okt. 2011.

[9] R. Bush, "The Address plus Port (A+P) Approach to the IPv4 Address Shortage", RFC 6346, avg. 2011.

[10] X. Che, D. Lewis, "IPv6: Current Deployment and Migration Status", v reviji: *International Journal of Research and Reviews in Computer Science*, št. 2, zv. 1, str. 22–29, 2010.

[11] Y. Cui, Q. Sun, M. Boucadair, T. Tsou, Y. Lee, I. Farrer, "Lightweight 4over6: An Extension to the DS-Lite Architecture", draft-cui-softwire-b4-translated-ds-lite-07 (IETF Internet Draft), jan. 2013.

[12] R. Despres, S. Matsushima, T. Murakami, O. Troan, "IPv4 Residual Deployment across IPv6-Service networks (4rd) ISP-NAT's made optional", draft-despres-intarea-4rd-01 (IETF Internet Draft, expired), sep. 2011.

[13] A. Durand, R. Droms, J. Woodyatt, Y. Lee, "Dual-Stack Lite Broadband Deployments Following IPv4 Exhaustion ", RFC 6333, avg. 2011.

[14] X. Li, C. Bao, F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, apr. 2011.

[15] X. Li, C. Bao, M. Chen, H. Zhang, J. Wu, "The China Education and Research Network (CERNET) IVI Translation Design and Deployment for the IPv4/IPv6 Coexistence and Transition", RFC 6219, maj 2011.

[16] O. Martin, "Where is the Internet heading to?", v reviji: *Journal of Physics: Conference Series* , zv. 219, del 6, 2010.

[17] T. Nishitani, S. Miyakawa, A. Nakagawa, H. Ashida, "Common Functions of Large Scale NAT (LSN)", draft-nishitani-cgn-01(IETF Internet Draft, expired), maj 2009.

[18] J. Postel, J. Reynolds, "Telnet Protocol Specification", STD 8, RFC 854, maj 1983.

[19] J. Postel, J. Reynolds. " FILE TRANSFER PROTOCOL (FTP)", STD 9, RFC 959, okt. 1985.

[20] Y. Shirasaki, S. Miyakawa, A. Nakagawa, J. Yamaguchi, H. Ashida, "NAT444 with ISP Shared Address", draft-shirasaki-nat444-isp-shared-addr-02 (IETF Internet Draft, expired), mar. 2010.

[21] P. Srisuresh, H. Holdrege, "IP Network Address Translator (NAT) Terminology and Considerations", RFC 2663, avg. 1999.

[22] N. Škoberne, M. Ciglarič, "Practical Evaluation of Stateful NAT64/DNS64 Translation", v reviji: *Advances in Electrical and Computer Engineering*, št. 3, zv. 11, str. 49-54, 2011.

[23] D. G. Waddington, F. Chang, "Realizing the Transition to IPv6", v reviji: *IEEE Communications Magazine*, zv. 40, str. 138–148, jun. 2002.

[24] D. Wing, "Network Address Translation: Extending the Internet Address Space", v reviji: *IEEE Internet Computing* , zv. 14, str. 66–70, 2010.

[25] I. Yamagata, S. Miyakawa, A. Nakagawa, J. Yamaguchi, H. Ashida. "ISP Shared Address", draft-shirasaki-isp-shared-addr-07 (IETF Internet Draft, expired), 2012.

[26] (2012) Active FTP vs. Passive FTP, a Definitive Explanation. Dostopno na:
http://slacksite.com/other/ftp.html

[27] (2012) Ecdysis: Open-Source Implementation of a NAT64 Gateway Dostopno na:
http://ecdysis.viagenie.ca/

[28] (2012)Ftp.proxy - FTP Proxy Server. Dostopno na:
http://www.ftpproxy.org/

[29] (2012) GNU General Public License. Dostopno na:
http://www.gnu.org/copyleft/gpl.html

[30] (2012)Internet Usage Statistics. Dostopno na:
http://www.internetworldstats.com/stats.htm

[31] (2012)IPv4 Address Report. Dostopno na:
http://www.potaroo.net/tools/ipv4/index.html

[32] (2012)Iperf. Dostopno na:
http://sourceforge.net/projects/iperf/

[33] (2012) Ivan Pepelnjak: Stateless NAT64 is useless. Dostopno na:
http://blog.ioshints.info/2011/05/stateless-nat64-is-useless.html

[34] (2012)NAT64 and DNS64: IPv4/IPv6 Co-existence via Server Load Balancer. Dostopno na:
http://www.a10networks.com/products/axseries-NAT64_DNS64.php

[35] (2012) NAT64—Stateless versus Stateful. Dostopno na:
http://www.cisco.com/en/US/prod/collateral/iosswrel/
ps6537/ps6553/white_paper_c11-676277.html

[36] (2012)NAT64 Technology: Connecting IPv6 and IPv4 Networks. Dostopno na:
http://www.cisco.com/en/US/prod/collateral/iosswrel/
ps6537/ps6553/white_paper_c11-676278.html

[37] (2012)Netfilter: firewaling, nat, and packet managing for Linux. Dostopno na:
http://www.netfilter.org/

[38] (2012)Tayga: Stateless NAT64. Dostopno na:
http://www.litech.org/tayga

[39] (2012)Universal TUN/TAP driver: Virtual Point-to-Point(TUN) and Ethernet(TAP) devices. Dostopno na:
http://vtun.sourceforge.net/tun/

[40] (2012)VmWare Workstation. Dostopno na:
http://www.vmware.com/products/workstation/overview.html