SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 706

Prilagodba OpenWRT distribucije Linux operacijskog sustava za upotrebu u Terastream arhitekturi

Petar Alilović

Zagreb, lipanj 2014.

Zahvaljujem se ekipi iz T-Hta na pruženoj pomoći Hrvoju Habjaniću i Branimiru Rajtaru i profesorima Leonardu Jelenkoviću i Stjepanu Grošu na pruženoj prilici.

Sažetak

Mreže pružatelja Internet usluga su evoluirale u vrlo složene sustave koji kombiniraju niz različitih tehnologija kako bi u konačnici pružile usluge efikasnog prijenosa podataka korisnicima. Problem takvog organskoga razvoja je u tome da ih je vrlo teško održavati, a pogotovo je teško razvijati i nuditi nove usluge u okruženju koje je vrlo konkurentno. U tom smislu određen broj pružatelja Internet usluga i proizvođača mrežne opreme započeo je Terastream projekt čiji prvenstveni cilj je pojednostavljivanje arhitekture kako bi pružatelji usluge bili agilniji na tržištu. Terastream mreža bazirana je na korištenju IPv6 verzije IP protokola. Svi ostali protokoli uključujući i IPv4 verziju IP protokola smatraju se servisom. U sklopu diplomskog rada ostvaren je jedan od mehanizama predviđen u Terastream mreži koji omogućava dobivanje IPv4 konfiguracije potrebne za IPv4 komunikaciju.

Ključne riječi:

OpenWrt, Linux, DHCP, IP, DHCPv4oDHCPv6, Terastream, odhcp, udhcp, netifd

Abstract

Internet service providers networks have evolved into highly complex systems that combine a number of different technologies in order to ultimately provide efficient data transmission services to users. The problem of such organic development is that it is very difficult to maintain, and it is particularly difficult to develop and offer new services in an environment that is highly competitive. In this regard, a number of Internet service providers and network equipment manufacturers began Terastream project whose primary goal is to simplify the architecture so service providers can be more agile in the market. Terastream network is based on IPv6 version of the IP protocol. All other protocols including IPv4 version of the IP protocol are considered service. As part of the thesis one of the mechanism in Terastream network has been accomplished. Mechanism allows to obtain the IPv4 configuration required for IPv4 communication in IPv6 network only.

Keywords:

OpenWrt, Linux, DHCP, IP, DHCPv4oDHCPv6, Terastream, odhcp, udhcp, netifd

Sadržaj

1. Uvod

Ovaj dokument predstavlja diplomski rad pisan na Fakultetu elektrotehnike i računarstva u Zagrebu. Diplomski rad pisan je kao popratna dokumentacija te opis projekta na kojem su sudjelovali Fakultet elektrotehnike i računarstva i Hrvatski telekom. Suradnja na projektu odvijala se u sklopu Terastream projekta kojeg je inicirao Deutsche telekom. Zadatak projekta je reorganizacija njihove interne mreže s ciljem smanjenja troškova i modernizacije. Projekt koji je odrađen u sklopu Terastream projekta ostvarenje je jednog od mehanizma predviđenog u novoj, modernijoj mreži. Terastream mreža podržava jedino promet *Internet Protocol Version 6* (skr. IPv6) paketa. Kako se *Internet Protocol Version 4* (skr. IPv4) verzija protokola još intenzivno koristi, potrebno je bilo osigurati mogućnost njegovog korištenja. Za korištenje IPv4 protokola potrebno je provesti konfiguraciju mrežnog sučelja uz pomoć *Dynamic Host Configuration Protocol Version 4* (skr. DHCPv4) protokola. No kako je DHCPv4 protokol temeljen na IPv4 protokolu, takav paket nije moguće usmjeravati u internoj mreži. U sklopu diplomskog rada ostvaren je mehanizam čijom je upotrebom moguće dobiti DHCPv4 konfiguraciju u IPv6 mreži. Rješenje problema bazirano je na posebnom protokolu koji je smišljen upravo za potrebe Terastream mreže. Implementacija rješenja zahtijevala je modifikaciju klijenta kojeg pokreće OpenWrt Linux distribucija i poslužiteljske aplikacije koja pruža mogućnosti DHCPv4 komunikacije.

Projekt je obavljen sa tročlanim timom čiji sam bio član. U radu su detaljnije opisani dijelovi na kojima sam radio, a to su: *odhcp6c*, *netifd* i *OpenWrt SDK* alat.

Rad je podijeljen na četiri poglavlja sa jednim dodatkom. U drugom poglavlju obrađene su teme vezane za OpenWrt Linux distribuciju uključujući njegovu povijest, mehanizam prevođenja distribucije i njezinih paketa te su opisani odabrani interni mehanizmi i pojedine korisničke aplikacije koje su korištene u svrhu rješenja problema projekta. Treće poglavlja opisuje zamisli oko ostvarenja Terastream mreže te se detaljnije obrađuje mehanizam kojeg je bilo potrebno ostvariti. Četvrto poglavlje vezano je za implementaciju ranije navedenoga mehanizma. Rad završava zaključkom u petom poglavlju te popisom korištene literature u šestom poglavlju.

2. OpenWrt Linux distribucija

Zahvaljujući napretku poluvodičke tehnologije, procesori kao osnovne komponente upravljanja ugrađenih sustava, postaju sve brži [1] što omogućava izvršavanje sve kompliciranijih poslova u manjem vremenskom periodu. Kako napreduje sklopovlje tako i programska podrška koja se izvršava na njemu obavlja kompliciranije zadatke. OpenWrt [2] je Linux distribucija operacijskog sustava namijenjena za ugrađena računala. Najveći dio ugrađenih računala koji koriste OpenWrt distribuciju prvenstveno su mrežni uređaji čija je zadaća usmjeravanje mrežnog prometa. Ono sto čini OpenWrt distribuciju specifičnu u odnosu na ostale Linux distribucije je njegova minimalna veličina, brzina izvođenja mrežnih zadataka te veliki naglasak na štednju dostupnih računalnih resursa.

2.1. Povijest

OpenWrt, u obliku kakav je danas, plod je velikog truda uloženog od skupine Linux entuzijasta. Njegovi počeci sežu od početka 2004. godine [3] kada je tvrtka Linksys predstavila uređaj pod nazivom WRT54G [4]. Navedeni uređaj služio je kao centralna jedinica za dijeljenje konekcije između određenog broja računala. Nedugo nakon što je uređaj postao dostupan korisnicima, Andrew Miklas primijetio je da navedeni uređaj koristi modificiranu Linux jezgru kao operacijski sustav koji ga pokreće [5]. Po General Public Licence (skr. *GPL*) licenci po kojoj je Linux jezgra dostupna u obliku slobodnog programskog koda, Linux jezgra ne smije se koristiti u komercijalne svrhe bez dijeljenja tog programskog koda potrošaču proizvoda na kojem se koristi. Nakon što je Andrew Miklas poslao apel Linksys kompaniji da se korišteni kod objavi te kako nije bilo nikakvih reakcija sa njihove strane, Andrew Miklas je incident prijavio na Linux grupu elektroničke pošte [6] (engl. mailing list). Linksys tvrtka je uslijed pritiska Linux zajednice popustila te je u javnost izašla Linux jezgra korištena na uređaju [7]. U tom trenutku započinju različiti projekti koji svoj programski kod baziraju na onom koji je objavila Linksys tvrtka. Jedni od popularnijih projekata su bili OpenWrt, DD-wrt, HyperWRT i projekt Sveasoft firme. Programski kod koji je izašao u javnost bila je uvelike modificirana Linux jezgra kao što se i pretpostavljalo, no nekoliko ljudi brzo je uvidjelo bespotrebnost većine programskog koda u njemu. Nakon čišćenja nepotrebnih dijelova koda te ostavljanja samo dijelova koji su najnužniji, veličina sustava pala je sa 30MB na oko 3MB te je ubrzo nakon toga, početkom 2004., napravljena i prva verzija sustava sa oznakom *build 1*. (skr. B1) koja nije bila javno dostupna. Sa daljnjim razvojem OpenWrt distribuciji dodana je mogućnost pisanja po disku koja u to vrijeme nije bila dostupna na konkurentskim projektima. OpenWrt projekt je pod GPL licencom te je kao takav slobodni programski kod. Razvoj OpenWrt Linux distribucije potpomaže velika zajednica širom svijeta. Tablicu OpenWrt verzija sa pripadnim datumom izlaska moguće je vidjeti u tablici 2.1 [8].

Naziv OpenWrt verzije	OpenWrt verzija	Datum izlaska	Verzija Linux jezgre
Prije Buildroot-ng	0.X	%	%
White Russian	0.9	2006-01	2.4.30
Kamikaze	7.06	2007-06	2.6.19
Kamikaze	7.07	2007-07	2.6.21
Kamikaze	7.09	2007-09	2.6.21
Kamikaze	8.09	2008-09	2.6.26
Kamikaze	8.09.1	2009-06	2.6.26
Kamikaze	8.09.2	2010-01	2.6.26
Backfire	10.03	2010-04	2.6.32
Backfire	10.03.1	2011-12	2.6.32
Attitude Adjustment	12.09	2013-04	3.3
Barrier Breaker	Razvojna verzija	Kontinuirani izlasci	3.10.36

Tablica 2.1: OpenWrt verzije

2.2. Prevođenje u OpenWrt Linux distribuciji

OpenWrt je uvelike definiran željom za što manjom ukupnom veličinom programskog koda kojeg se instalira na mrežni uređaj. Sljedeći navedenu želju, OpenWrt izostavlja sve funkcionalnosti koje nisu krucijalno potrebne za rad sustava te ih ostavlja kao opcionalne. Također, kako mrežni uređaji imaju širok raspon korištenih procesorskih arhitektura, potrebno je bilo osmisliti mehanizam koji će olakšati razvoj OpenWrt Linux distribucije za takav širok spektar arhitektura. OpenWrt u tom smislu pripada kategoriji meta-distribucija. Cilj navedenih distribucija je da služe kao polazna točka pri izgradnji sustava, da nude samo osnovne mogućnosti u svom primarnom obliku no da ih je moguće mijenjati po svojim potrebama na niz različitih načina. Želja za što manjom veličinom i što lakšim razvojem za širok spektar procesorskih arhitektura dijeli sva programska podrška namijenjena ugrađenim računalima pa tako i OpenWrt Linux distribucija. U svrhu pružanja bolje podrške pri razvoju sustava za ugrađena računala napravljeni su mehanizmi koji odvajaju proces izrade programske podrške od procesa njegovog prevođenja. Takvi mehanizmi pokušavaju minimalizirati veličinu prevedenog sustava te maksimizirati podršku za različite arhitekture procesora. Navedene mehanizme pruža nekoliko sustava kao što su: Buildroot [9], Yocto-project [10] i Openembedded [11]. OpenWrt koristi modificiranu verziju Buildroot mehanizma.

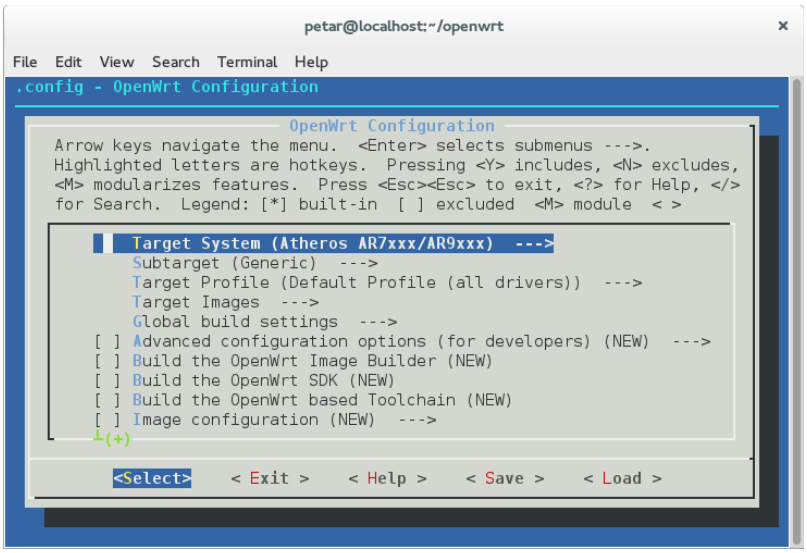
2.2.1. OpenWrt Buildroot

U računarstvu, *toolchain* sustav definiran je kao skup alata koji se koristi za stvaranje ciljne aplikacije ili sustava. Ime *toolchain* sustava ujedno opisuje i njegov rad, komponente skupa alata djeluju u neprekidnom lancu, tako kada jedna komponenta sustava završi svoju zadaću rezultat se prenosi kao ulazni parametar alata koji je sljedeći u nizu. *Toolchain*

sustav nužno se sastoji od barem prevoditelja (engl. *compiler*) i poveznika (engl. *linker*). Kada se aplikacija prevodi za izvršavanje na računalu na kojem se nalazi, koristi se prevoditelj, a s time i *toolchain* sustav koji je namijenjen za tu arhitekturu procesora. Najpoznatiji takav skup alata za Linux distribucije operacijskih sustava poznat je pod imenom *GNU toolchain*. Svaki takav *toolchain* potpun je skup alata sa kojima je moguće dobiti izvršnu datoteku ili paket namijenjen sustavu za koji se prevodi. Prevođenje aplikacije ili sustava za arhitekturu koja je različita od arhitekture računala na kojem se prevodi zahtijeva poseban prevoditelj, a s time i poseban skup *toolchain* alata koji su poznati pod nazivom *cross-compilation toolchain*.

OpenWrt koristi modificirani Buildroot mehanizam pri svome prevođenju te prevođenju paketa namijenjenih njemu. OpenWrt Buildroot mehanizam rješava probleme izgradnje minimalnoga sustava za široki spektar procesorskih arhitektura. Za potrebe prevođenja izvornog koda namijenjenoga arhitekturi koja nije jednaka arhitekturi računala na kojem se prevodi, OpenWrt Buildroot stvara potreban *cross-compilation toolchain* skup alata. Upotrebom stvorenoga *toolchain* skupa alata OpenWrt Buildroot prevodi sve izvorne kodove za ciljanu arhitekturu. Izvorni kod same OpenWrt Linux distribucije i njezinih paketa raspodijeljen je na mnoge druge repozitorije. Pri prevođenju cijelog sustava, sa svih potrebnih ili odabranih repozitorija po potrebi se skidaju izvorni kodovi, nad njima se upotrebljavaju posebne zakrpe [12] (engl. *patch*) te se takvi tada prevode i spremaju u zapis pogodan instaliranju na OpenWrt sustav zajedno sa potrebnim OpenWrt skriptama. Izvorni kod korisničkih aplikacija u svom originalnom obliku ponekad nije u potpunosti pogodan direktnom prevođenju i instaliranju na OpenWrt distribuciju, promjene koje je potrebno upotrijebiti na izvorni kod sadržane su u datotekama zakrpama. Prije prevođenja takvih aplikacija, OpenWrt Buildroot mehanizam prvo primjenjuje promjene opisane u datotekama zakrpama na izvorni kod te se takav izvorni kod tada prevodi. Navedeni postupak upotrebljava se i na Linux jezgru prije njezinoga prevođenja za OpenWrt distribuciju.

Skidanjem OpenWrt izvornog koda iz službenog repozitorija zapravo se skida OpenWrt Buildroot mehanizam. OpenWrt Buildroot mehanizam, osim izgradnje ugrađenog programskog koda koji se instalira na ugrađenog računalo, podržava i izradu raznih alata koji olakšavaju prevođenje paketa ili ciljnog programskog koda. Mogućnosti OpenWrt Buildroot mehanizma te odabir željene akcije moguće je inicirati pokretanjem `make menuconfig` naredbe iz korijenskoga OpenWrt Buildroot direktorija. Pokretanjem naredbe za konfiguraciju otvara se prozor prikazan na slici 2.1. Odabir konfiguracije prvi je korak prevođenja OpenWrt alata ili OpenWrt Linux distribucije. Nakon odabira konfiguracije i njezinoga spremanja u konfiguracijsku datoteku moguće je pokrenuti postupak prevođenja pomoću `make` naredbe.



Slika 2.1: Prozor za konfiguraciju prevođenja

OpenWrt Buildroot mehanizam definirana je struktura direktorija i posebnih `Makefile` datoteka. `Makefile` datoteke definiraju preuzimanje izvornog programskog koda, primjenjuju zakrpe i prevode preuzeti izvorni kod. Jedna od velikih prednosti Buildroot mehanizma je to što podržava široki spektar procesorskih arhitektura za koje je moguće prevoditi. `Makefile` datoteke sadrže sve potrebne informacije za stvaranje kompletnog sustava. Glavna `Makefile` datoteka pri svome pokretanju redom obavlja sljedeće zadatke

- Stvara izlazne direktorije
- Stvara *toolchain* alate ili povezuje vanjske *toolchain* alate u svoju okolinu
- Stvara sve odabrane aplikacije zajedno sa obaveznim dijelovima sustava

Kako je već navedeno, OpenWrt Buildroot je unaprijed definirana struktura direktorija i datoteka. Datotečna struktura kod OpenWrt Buildroot mehanizma sastoji se od tri glavna direktorija:

- `toolchain`
- `target`
- `package`

`Toolchain` direktorij sadrži alate koji su potrebni za izradu *toolchain* mehanizma. Direktorij se uglavnom sastoji od C-biblioteka, C-prevoditelja te još niza alata koji služe kao pomoć pri prevođenju i izgradnji ciljnoga sustava kao što su *gdb* i *binutils*. U `target` direktoriju nalaze se datoteke potrebne za izgradnju Linux jezgre, sustava za izgradnju ciljnog OpenWrt sustava te datoteke za izgradnju *OpenWrt SDK* alata, *OpenWrt Image Builder* alata i *toolchain* alata. Svaki od navedenih ciljeva izgradnje moguće je izgraditi nezavisno od ostalih ciljeva te ga zasebno i koristiti. U `package` direktoriju nalaze se datoteke koje služe za prevođenje dostupnih korisničkih aplikacija. U navedenom

direktoriju svaka korisnička aplikacija nalazi se u zasebnom direktoriju sa vlastitom `Makefile` datotekom, skupom zakrpa i skupom skripti koje potpomažu njezino izvođenje. Ciljni zapis prevedene korisničke aplikacije je *ipk* paket. U procesu izgradnje OpenWrt sustava redom se prevode direktoriji kako je prikazano na slici 2.2. Nakon uspješnoga završetka prevođenja ciljeva, odabranih tijekom konfiguracije, rezultate prevođenja moguće je pronaći u *bin* direktoriju.

1) tools
2) toolchain/binutils
3) toolchain/gcc (prevoditelji)
4) target/linux (moduli Linux jezgre)
5) package
6) target/linux (slika Linux jezgre)
7) target/linux/image

Slika 2.2: Redoslijed prevođenja OpenWrt sustava

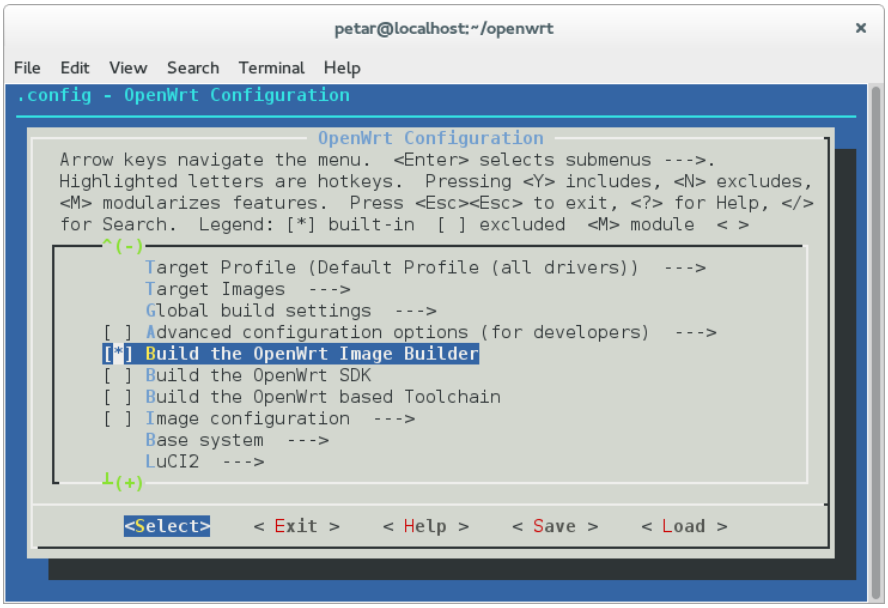
2.2.2. OpenWrt Image Builder

Za fino ugađanje ciljnog ugrađenog programskog koda koji se instalira na ugrađeno računalo moguće je koristiti alat naziva OpenWrt Image Builder. Navedeni alat služi za izradu ciljnog binarnog koda bez prolaska kroz cijeli postupak prevođenja OpenWrt Linux distribucije. Koristeći takav proces moguće je smanjiti vrijeme potrebno za pronalazak takve OpenWrt konfiguracije koja zadovoljava ograničenja ugrađenog računala te sadrži dovoljan skup paketa za ispravan i namijenjeni rad uređaja.

OpenWrt Image Builder alat za određenu procesorsku arhitekturu moguće je skinuti sa službene OpenWrt stranice ili ga je moguće prevesti uz pomoć OpenWrt Buildroot mehanizma. Za prevođenje OpenWrt Image Builder alata potrebno je skinuti OpenWrt Buildroot [13]. Prije pokretanja prevođenja, pokretanjem konfiguracije te odabirom opcije *Build the OpenWrt Image Builder* osigurava se izrada navedenoga alata kao što je prikazano na slici 2.3. Izrada prilagođenog ciljnog programskog koda upravlja se pomoću tri parametara `make` naredbe:

- PROFILE
- PACKAGES
- FILES

Na samom početku izrade OpenWrt Image Builder alata odabrana je ciljna procesorska arhitektura. Kako odabranu procesorsku arhitekturu koristi niz različitih uređaja, sa argumentima PROFILE parametra moguće je odabrati profil za dostupni uređaj. Profili su već unaprijed složene konfiguracije za određene uređaje. Konfiguracije uključuje minimalni skup paketa koji će se koristiti na tako napravljenoj OpenWrt Linux distribuciji. Sa parametrima PACKAGES argumenta dodatno se definira skup paketa čija se instalacija



Slika 2.3: Opcija za izradu OpenWrt Image Builder alata

želi uključiti ili isključiti u ciljnoj OpenWrt Linux distribuciji. Parametri FILES argumenta određuju skup direktorija ili datoteka koje će biti prenesene na ciljnu OpenWrt Linux distribuciju. Vrijeme potrebno za izradu novog ciljnog OpenWrt sustava drastično je smanjeno korištenjem već prevedenih komponenti koje je tada samo potrebno složiti u ciljni sustav. Primjer generičke naredbe za izradu ciljnog OpenWrt sustava pomoću OpenWrt Image Builder alata prikazana je na slici 2.4.

```
$ make image PROFILE=XXX PACKAGES="pkg1 pkg2 pkg3 -pkg4" FILES=files/
```

Slika 2.4: Primjer naredbe za pokretanje izrade slike

2.2.3. OpenWrt SDK alat

Završni korak pri izradi novih aplikacija za korisničku okolinu je prevođenje izvornog koda u izvršnu datoteku. U većini slučajeva kod izrade većih ili složenijih aplikacija, jednostavna izvršna datoteka nije dovoljna te se tada prelazi na izradu paketa za određenog upravitelja paketima. Kod OpenWrt Linux distribucije koristi se *opkg* upravitelj paketima, a paketi koji se koriste zapravo su inačica *zip* arhive sa ekstenzijom *.ipk*. Osim prikladnog zapisa u obliku paketa, također je potrebno isti prilagoditi za arhitekturu stroja na kojem će se korisnička aplikacija izvoditi.

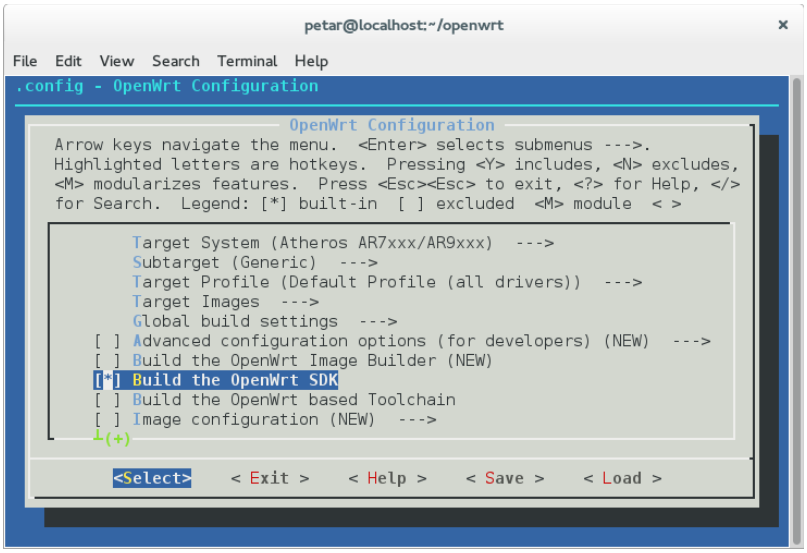
Izradi paketa za OpenWrt distribuciju moguće je pristupiti na dva različita načina. Prvi način zahtijeva prevođenje paketa u sklopu prevođenja cijele OpenWrt distribucije što katkad zna trajati i po nekoliko sati. Drugi način izrade paketa je upotrebom *OpenWrt SDK* alata. Navedeni alat namijenjen je prevođenju jednog ili manjeg broja paketa za OpenWrt Linux distribuciju bez potrebe za prevođenjem cijelog OpenWrt sustava. Korištenjem

OpenWrt SDK alata uvelike se štedi vrijeme izrade paketa te je takav način i korišten pri izradi programskog dijela diplomskog zadatka.

U ovom potpoglavlju biti će objašnjen postupak pribavljanja *OpenWrt SDK* alata, pripremanja korisničke aplikacije prije prevođenja te prevođenje aplikacije u OpenWrt paket.

Pribavljanja OpenWrt SDK alata

OpenWrt SDK moguće je pribaviti na dva različita načina: skidanjem već prevedenog *OpenWrt SKD* alata ili prevođenje vlastitog. Skidanje već prevedenog *OpenWrt SDK* alata moguće je obaviti sa OpenWrt stranice koja pruža izbor OpenWrt distribucije, a s tim i verzije *OpenWrt SDK* alata [14]. Drugi način koji će u ovom poglavlju biti detaljnije objašnjen je postupak izrade vlastitog *OpenWrt SDK* alata. Kako je već prije navedeno, OpenWrt Buildroot mehanizam kao jedan od svojih ciljeva pri izgradnji ima mogućnost izgradnje *OpenWrt SDK* alata te *toolchain* mehanizma koji će se koristiti u zavisnosti od arhitekture računala na kojem se prevodi i arhitekture računala za kojeg se prevodi. Za prevođenje vlastitog *OpenWrt SDK* alata na takav način, potrebno je odabrati opciju izgradnje prilikom pokretanja konfiguracije OpenWrt Buildroot mehanizma kako je prikazano na slici 2.5.



Slika 2.5: Odabir prevođenja OpenWrt SDK alata

Nakon odabira ostalih opcija te spremanja istih, izrada *OpenWrt SDK* alata pokreće se naredbom `make`. Nakon uspješnog završetka postupka prevođenja, *OpenWrt SDK* moguće je pronaći u *bin* direktoriju.

Prevođenje paketa uz pomoć OpenWrt SDK alata

Kao primjer aplikacije korisničke okoline koja će se prevoditi uzeta je *odhcp6c* aplikacija, no isti postupak moguće je upotrijebiti za bilo koju aplikaciju namijenjenu OpenWrt distribuciji. U općenitom slučaju aplikacija se sastoji od izvornog koda te skupa datoteka koje služe za konfiguriranje aplikacije ili su u obliku skripti koje potpomažu izvođenje

aplikacije. Izvorni kod aplikacije koja se želi mijenjati pribavlja se preko službenog repozitorija aplikacije. Skup datoteka potrebnih aplikaciji za izvođenje nalaze se u OpenWrt repozitoriju izvornog koda. U istom repozitoriju moguće je vidjeti zahtijevanu strukturu direktorija i datoteka koja je potrebna pri prevođenju sa OpenWrt SDK-om. Tako je za *odhcp6c* aplikaciju, koja se nalazi na putanji `/package/network/ipv6/odhcp6c` [15], moguće vidjeti da se sastoji od direktorija `files` u kojem su smještene skripte te datoteke `Makefile`. `Makefile` datoteka ključni je resurs pri prevođenju aplikacije. Više o `Makefile` datoteci i njezinoj modifikaciji moguće je naći u dodatku Dodatak A. Nakon što se navedena datoteka prilagodi korisničkim potrebama, sve je spremno za početak stvaranja paketa. Direktorij `odhcp6c`, u kojem je `Makefile` datoteka i direktorij `files` sa potrebnim skupom datoteka, potrebno je smjestiti u `package` direktorij *OpenWrt SDK* alata. Nakon smještanja direktorija na navedeno mjesto, prevođenje je moguće započeti izvršavanjem naredbe `make`. Za detaljniji ispis procesa prevođenja moguće je koristiti zastavicu `V` i brojčanu konstantu koja označuje razinu detalja u ispisu (npr. `make V=99`). Pri uspješnom završetku prevođenja stvara se paket aplikacije koji je moguće naći u *bin* direktoriju.

2.3. OpenWrt interna struktura

U ovom poglavlju biti će objašnjenje glavne komponente i mehanizmi pokrenutog OpenWrt sustava. Kako se smanjenje cijene ugrađenih računalnih sustava pokušava ostvariti smanjivanjem potrebnog spremničkog prostora, programi koje uređaj obavlja optimiraju se obzirom na njihovu veličinu [16]. OpenWrt Linux distribucija slijedi takav primjer optimizacije te postoji niz aplikacija koje su prilagođene ili nanovo napravljene izrazito u svrhu OpenWrt projekta. Jedan od problema kojeg susrećemo ugrađena računala čiji je programski kod napisan u C programskom jeziku je problem veličine GNU C biblioteke *glibc*. Veličina C programa koji koriste navedenu *glibc* biblioteku te sama biblioteka nije optimizirana za korištenje na sustavima namijenjenim ugrađenim računalima. Zbog navedenoga problema OpenWrt sustav pruža mogućnost korištenja dvije različite C biblioteke koje su optimizirane za korištenje na sustavima za ugrađena računala, a pružaju veliki skup mogućnosti *glibc* biblioteke. Popularnija od njih je uClibc biblioteka, a druga, čije je održavanje prekinuto početkom 2014. godine [17] je Embedded glibc (skr. eglibc) biblioteka. U tablici 2.2 moguće je vidjeti neke od razlika između OpenWrt Linux distribucije i tipične Linux distribucije namijenjene za stolna računala [18]. U nastavku, detaljnije će biti objašnjenje samo odabrane komponente.

	Stolno računalo	**OpenWrt**
Veličina radne memorije	128MiB – 16GiB	32MiB - 512MiB
Podržani skup instrukcija	Skoro svi skupovi	Skoro svi skupovi
Standardna C biblioteka	glibc	uClibc
IPC	D-Bus	ubus
Mrežna konfiguracija	Network Manager	netifd
Upravitelj paketima	yum, apt, pacman...	Opkg

Tablica 2.2: OpenWrt karakteristike

2.3.1. ubus

OpenWrt Micro Bus Architecture [19] (skr. ubus) implementacija je mehanizma koji omogućava komunikaciju između različitih procesa istog sustava poznat pod imenom Interprocess communication (skr. *IPC*). Projekt je pokrenut specifično za potrebe OpenWrt Linux distribucije. Projekt se sastoji od ubus daemona, ubus biblioteke *libubus* te dodatnih pomoćnih alata. Biblioteka *libubus* razvijena je kao pomoć za razvoj programerima čije aplikacije koristite ubus mehanizam. Jezgra projekta je ubus daemon koji je zadužen za međuprocesnu komunikaciju. Svaki proces koji namjerava komunicirati s drugim procesima obvezan je definirati sučelja preko kojih mu se može pristupiti. Takva sučelja grupirana su u hijerarhijske strukture definirane nazivom. Registracija sučelja obavlja se definiranim pozivom ubus daemona. Pregled najčešćih registriranih aplikacija i odabranih njihovih dostupnih procedura moguće je vidjeti u tablici 2.3.

Ime sučelja	**Procedura**	**Argumenti procedure**
network	restart	{ }
network.*device*	status	{ "name": "ifname" }
network.interface.*name*	up	{ }
network.interface.*name*	down	{ }
network.interface.*name*	status	{ }
log	read	{ "lines": "Integer" }
log	write	{ "event": "String" }

Tablica 2.3: Standardna ubus sučelja i procedure

Kao jedan od pomoćnih alata sa kojim je moguć prikaz stanja ubus daemona i obavljanje različitih akcija nad registriranim sučeljima koristi se *ubus* korisnička aplikacija. Pregled akcija navedene aplikacije moguće je vidjeti u tablici 2.4. *Ubus* korisničku aplikaciju uglavnom koriste OpenWrt skripte za pozivanje procedure, praćenje događaja na sučelju ili slanje događaja na definirana sučelja.

Akcija	**Objašnjenje akcije**
ubus list	Ispis svih trenutno registriranih naziva sučelja
ubus list <sučelje> -v	Ispis dostupnih procedura sučelja
ubus call <sučelje> <procedura> [<argumenti>]	Poziv procedura sa potrebnim parametrima
ubus listen	Slušanje događaja na definiranom sučelju

Tablica 2.4: Neke od mogućnosti ubus korisničke aplikacije

2.3.2. UCI

Unified Configuration Interface [20] (skr. *UCI*) dio je OpenWrt sustava koji služi kako bi centralizirao konfiguraciju OpenWrt Linux distribucije i njezinih komponenata. Navedeni alat prvenstveno služi kako bi spremao krucijalne podatke koji su neophodni za rad sustava no moguće ga je koristiti i za spremanje različitih drugih postavki. Komponente sustava kao što su mrežna konfiguracija, bežična konfiguracija, sustav za logiranje i sustav za udaljeni pristup koriste UCI za spremanje svojih postavki. Postoje dva načina na koja je moguće mijenjati UCI postavke. Prvi način su UCI konfiguracijske datoteke koje služe za definiranje inicijalne konfiguracije aplikacije pri njenom pokretanju. Navedene konfiguracijske datoteke nalaze se u `/etc/config` putanji. Drugi način mijenjanja UCI konfiguracije je upotrebom standardnog sučelja namijenjenoga upravo za modifikacije tijekom izvršavanja aplikacije. Sučelju je moguće pristupiti u nizu programskih jezika kao što su LUA, C i jezik komandne linije ili sa alatom komandne linije pod nazivom *uci*. Navedeni alat najčešće se koristi u OpenWrt skriptama za čitanje stanja pojedinih komponenti te za njihovu modifikaciju. Osim čitanja i mijenjanja trenutnog stanja komponenti, *uci* alat omogućava i mijenjanje konfiguracijskih datoteka. U tablici 2.5 prikazane su neke od dostupnih akcija *uci* alata.

Akcija	Objašnjenje akcije
uci show	Prikazuje popis svih opcija u obliku ključ - vrijednost
uci get <konf.>.<sekcija>[.<opcija>]	Prikazuje vrijednost opcije za zadanu konfiguraciju
uci set <konf.>.<sekcija>[.<opcija>]=<vrijednost>	Postavlja vrijednost opcije

Tablica 2.5: Neke od mogućnosti uci korisničke aplikacije

2.3.3. netifd

Network interface configuration deamon [21] (skr. netifd) centralna je komponenta OpenWrt mrežnog podsustava. Njegovo područje djelovanja obuhvaća administraciju mrežnih uređaja i mrežnih sučelja. Pri izradi netifd komponente programeri su veliku važnost davali njegovoj veličini, ovisnosti o drugim paketima te štednji radne memorije pri njegovom izvršavanju. Netifd mrežni upravitelj koristi UCI i ubus mehanizme u svojem radu. Glavna konfiguracijska datoteka koju mrežni upravitelj koristi, nalazi se u `/etc/config/network` putanji. Navedena datoteka zapravo predstavlja UCI konfiguracijsku datoteku. U konfiguracijskoj datoteci definirana su mrežna sučelja i postavke preklopnika. Kod definiranje mrežnih sučelja minimalna konfiguracija mora sadržavati sljedeće:

- jedinstveno ime sučelja
- protokol koji se koristi na sučelju
- fizičko sučelje koje će biti povezano sa definiranom sekcijom

Primjer tako definiranog sučelja moguće je vidjeti na slici 2.6.

```
config interface 'wan'
        option proto 'dhcp'
        option ifname 'eth0'
```

Slika 2.6: Primjer definiranog sučelja

Nakon završetka obrade konfiguracijske datoteke, definirane postavke moguće je vidjeti upotrebom *uci* korisničke aplikacije sa naredbom `uci show network`. Kako je ranije navedeno, netifd mrežni upravitelj koristi ubus mehanizam u svrhu pružanja usluga drugim aplikacijama. Tim uslugama moguće je pristupiti korištenjem sučelja. Uobičajena sučelja prikazana su na slici 2.7. Osim standardnih sučelja, svaki definirani preklopnik ili mrežno sučelje sadržavati će svoje vlastito sučelje.

```
root@OpenWrt:/# ubus list
...
network
network.device
network.interface
network.wireless
...
```

Slika 2.7: Primjer ispisa ubus list naredbe

Tako bi za ranije definirano mrežno sučelje `wan` postojalo sučelje `network.interface.wan` koje bi sadržavalo procedure za upravljanje tim sučeljem. Osim procedura za upravljanje sučeljem, postoji i `status` procedura za ispis trenutnog stanja sučelja. Primjer pozivanja status procedure za ranije definirano mrežno sučelje prikazano je na slici 2.8.

```
root@OpenWrt:/# ubus call network.interface.wan status
```

Slika 2.8: Primjer ubus status procedure za wan sučelje

2.3.4. BusyBox

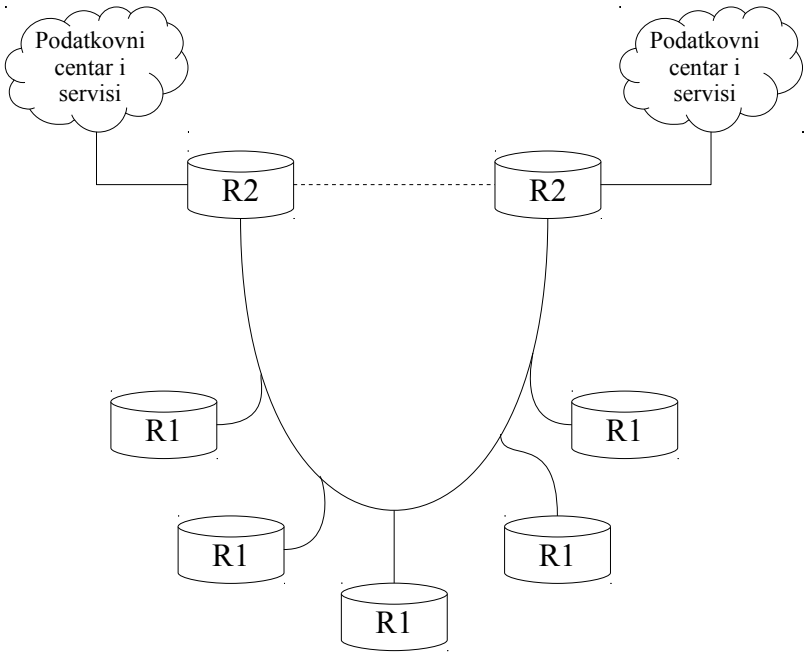
Busybox [22] je korisnička aplikacija koja pruža funkcionalnosti velikog broja standardnih Linux alata. Listu alata koje pruža BusyBox te njihove funkcionalnosti moguće je pogledati na službenoj stranici aplikacije [23]. Navedeni alati implementirani su uz pomoć programskog koda standardnih alata te su optimizirani s obzirom na veličinu sa svrhom uštede spremničkog prostora. Alat koji pruža Busybox te standardni alat ne podudaraju se uvijek u skupu pruženih mogućnosti. Alati koje pruža BusyBox sadrže samo najčešće korištene mogućnosti tog alata. Busybox je korisnička aplikacija koja kao parametar prima naziv naredbe koja se želi izvršiti. Razlog zašto su svi alati povezani u jednu izvršivu datoteku je optimizacija veličine i ušteda spremničkog prostora. Ako bi svaki alat bio zasebna izvršiva datoteka skup takvih izvršivih datoteka imao bi veću veličinu nego u navedenom načinu pakiranja. Razlog tome je što format izvršive datoteke (engl. Executable and Linkable Format, skr. ELF) sadrži dodatni trošak spremničkog prostora [24].

3. Terastream projekt

Projekt pod nazivom Terastream [25] projekt je Deutsche Telekoma čija je zadaća smanjiti trošak kod prijenosa velikog broja IP paketa. Motivacija za projekt su procjene povećanja IP prometa u nadolazećim godinama koja će nastati upotrebom modernijih protokola i povećanjem propusnosti veza koje spajaju krajnjeg korisnika sa pružateljem usluga. U sklopu Terastream projekta predložen je drugačiji model arhitekture koji bi bio kompetentan nositi se sa takvom promjenom. Kao glavna misao vodilja bila je potreba za pojednostavljenjem mreže te boljim usklađivanjem arhitekture sa TCP/IP modelom [26]. Glavne smjernice koje bi riješile navedene probleme su:

- smanjenje broja korištenih tehnologija
- korištenje samo IPv6 tehnologije za jezgrene funkcije i servise
- pojednostavljenje optičke mreže te čvršće povezivanje optičke mreže sa IP mrežom
- korištenje jedne mreže za sve servise

Na slici 3.1 prikazana je osnovna topologija mreže. Arhitektura sadrži dva tipa korištenih usmjerivača R1 i R2.



Slika 3.1: Arhitektura Terastream mreže

Usmjerivači naziva R1 bliži su krajnjim korisnicima te kao takvi predstavljaju ulaznu i izlaznu točku između krajnjih korisnika i interne mreže. Navedeni usmjerivači spojeni su takozvanom "konjskom potkovom" tj. polukružnim optičkim vlaknom na čijim se krajevima nalaze usmjerivači naziva R2. Usmjerivači R2 predstavljaju ulaznu i izlaznu

točku između Interneta i interne mreže. Također, svaki R2 usmjerivač ima pristup vlastitom podatkovnom središtu gdje su smješteni servisi i usluge. Svaki R2 usmjerivač spojen je sa svakim drugim R2 usmjerivačem te takvim povezivanjem stvara se potpuno povezana topologija R2 usmjerivača. Kao adresni mehanizam u internoj mreži koristi se IPv6 adresni prostor. Korištenjem ekskluzivno IPv6 adresa onemogućava se bilo kakav protok IPv4 paketa kroz mrežu što stvara određene probleme ako se želi koristiti IPv4 protokol. Terastream projekt, IPv4 tehnologiju vidi kao tehnologiju koja će se u nadolazećim godinama sve manje koristiti. U tom pogledu IPv4 protokol biti će dostupan kao servis. Takvo rješenje ima i svoje prednosti. U trenutku kada se IPv4 protokol prestane koristiti mreža će i dalje raditi pošto njezin ispravan rad nije ovisan o IPv4 protokolu. Svaki IPv4 paket na ulazu u internu mrežu enkapsulirati će se u IPv6 paket koji će tada biti usmjeren na ispravno odredište. Pri izlasku takvog IPv6 paketa događa se dekapsuliranje tj. "odmotavanje" IPv4 paketa koji se tada IPv4 protokolom usmjerava do odredišta. Ovakva arhitektura ostvaruje fleksibilnost u pogledu na korištenje IPv4 protokola.

3.1. DHCPv4oDHCPv6

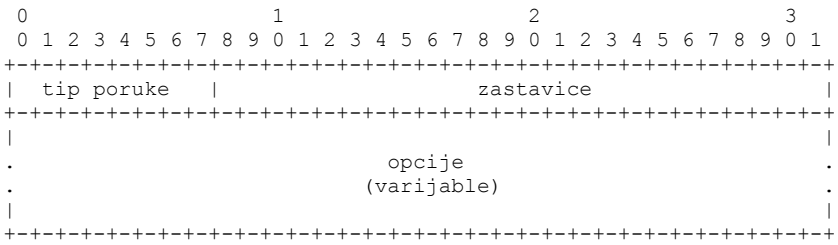
Terastream interna mreža koristi samo IPv6 protokol. Pošto IPv6 protokol još nije u potpunosti prihvaćen od većine korisnika te neće ni biti u dogledno vrijeme, nužno je krajnjem korisniku omogućiti korištenje IPv4 protokola. Za omogućavanje IPv4 veze potrebno je sučelje sa IPv4 adresom. U svrhu dobivanja IPv4 konfiguracije za spajanje koristi se DHCPv4 protokol. Kako se DHCP poslužitelji nalaze u podatkovnom centru interne mreže koja koristi IPv6 protokol, nemoguće je preuzeti IPv4 konfiguraciju slanjem DHCPv4 paketa u internu mrežu. Rješenje navedenog problema predlaže nekoliko IETF dokumenata. Kao prvi pristup koji predlaže rješenje opisan je u *Internet Engineering Task Force* (skr. IETF) *draft* [27] *DHCPv4 over IPv6 Transport* [28] dokumentu. U tom mehanizmu, IPv4 zaglavlje DHCPv4 protokola zamjenjuje se sa IPv6 zaglavljem te se takav paket šalje kroz mrežu. Zbog nekoliko problema kao što su nužnost postojanja DHCPv4 poslužitelja te njegovo održavanje, takav je pristup odbačen od strane IETF zajednice te se ne predlaže njegovo korištenje. Drugi mehanizam, koji je bilo potrebno ostvariti u sklopu diplomskog rada, opisan je u dokumentu pod nazivom *DHCPv4 over DHCPv6 Transport* [29]. Dokument predlaže komunikaciju u kojoj je DHCPv4 paket enkapsuliran u DHCPv6 paket. Arhitektura koja se predlaže u dokumentu pretpostavlja slučaj u kojem klijent i pružatelj usluga podržavaju samo IPv6 protokol. U tom slučaju, klijent može koristiti samo IPv6 mrežu za komunikaciju sa IPv4 uslugama te je takve usluge potrebno podesiti koristeći IPv6 infrastrukturu.

Za potrebe takve komunikacije u slučaju dobivanja IPv4 konfiguracije pomoću DHCP protokola, potrebno je enkapsulirati DHCPv4 paket u DHCPv6 paket te ga takvog poslati u IPv6 mrežu gdje se nalazi DHCPv6 poslužitelj. Takav poslužitelj, osim odgovaranja na DHCPv6 poruke, mora znati davati pravilne DHCPv4 odgovore enkapsulirane u DHCPv6 pakete.

Za potrebe opisane komunikacije DHCPv6 protokol definira dva nova tipa poruka: *DHCPv4-QUERY* koji označava poruku poslanu od strane klijenta te *DHCPv4-RESPONSE* koja sadrži odgovor od poslužitelja. Također, definirana je i nova opcija pod nazivom *DHCPv4 Message* u koju se postavlja DHCPv4 poruka koja se prenosi u oba slučaja.

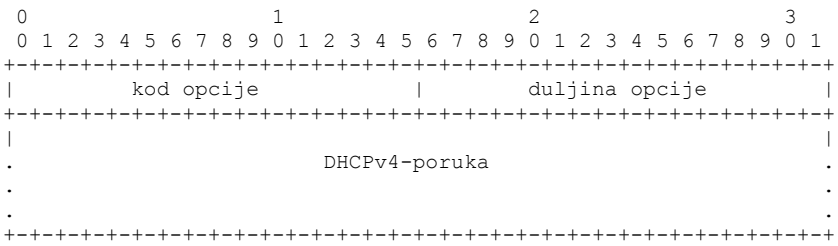
Prije početka procesa dobivanja DHCPv4 konfiguracije klijent mora imati valjanu IPv6 adresu koju može dobiti koristeći DHCPv6 protokol. U toj početnoj komunikaciji, ako korisnik želi koristiti *DHCPv4 over DHCPv6* (dalje u tekstu samo DHCPv4oDHCPv6) protokol nužan je poslati zahtjev za dobivanjem adresa DHCPv4oDHCPv6 poslužitelja. Zahtjev za opcijom postavlja se u bloku sa ostalim zahtjevima za opcije. Ako poslužitelj odgovori sa dostupnom traženom opcijom klijentu se obznanjuje da je u mogućnosti koristiti DHCPv4oDHCPv6 protokol za dobivanje IPv4 konfiguracije. Ako je poslana opcija prazna, tj. ne sadrži ni jedan dostupan DHCPv4oDHCPv6 poslužitelj, klijent koristi unaprijed poznatu *All_DHCP_Relay_Agents_and_Servers* višeodredišnu adresu za komunikaciju sa DHCPv4oDHCPv6 poslužiteljima.

Kako je navedeno, za potrebe DHCPv4oDHCPv6 mehanizma DHCPv6 definira dva nova tipa poruka: *DHCPV4-QUERY* i *DHCPV4-RESPONSE*. Prva poruka koristi se za komunikaciju od DHCPv4oDHCPv6 klijenta prema DHCPv4oDHCPv6 poslužitelju te je u njoj sadržana originalna DHCPv4 poruka u *DHCPv4 Message* opciji. Druga poruka koristi se za komunikaciju DHCPv4oDHCPv6 poslužitelja prema DHCPv4oDHCPv6 klijentu te ona sadrži DHCP4 odgovor koji se nalazi u *DHCPv4 Message* opciji. Oba tipa poruka sadrže isti format prikazan na slici 3.2.



Slika 3.2: Format DHCPv4-QUERY i DHCPv4-RESPONSE poruka

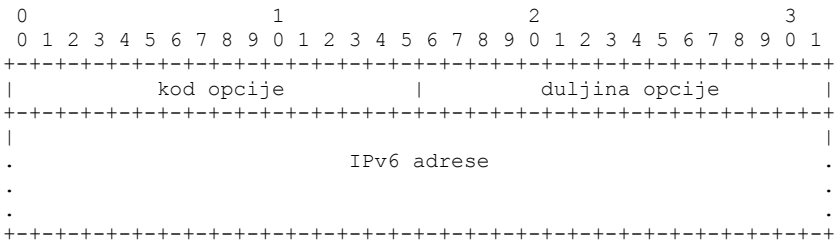
U DHCPv4oDHCPv6 mehanizmu koristi se i nova opcija pod nazivom *DHCPv4 Message* opcija u kojoj je sadržana DHCPv4 poruka kako je prikazano na slici 3.3.



Slika 3.3: Format DHCPv4 Message opcije

Kako je već prije navedeno, za potrebe korištenja DHCPv4oDHCPv6 mehanizma potrebno je znati IPv6 adrese DHCPv4oDHCPv6 poslužitelja. Te adrese dobivaju se pri IPv6 konfiguraciji sučelja koristeći DHCPv6 protokol. U tom procesu dobivanja konfiguracije

definirana je nova opcija pod nazivom *OPTION_DHCPV4_O_DHCPV6_SERVER*. Format navedene opcije prikazan je na slici 3.4.

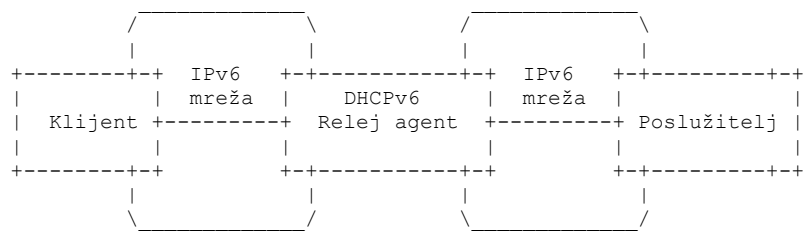


Slika 3.4: Format OPTION_DHCPV4_O_DHCPV6_SERVER opcije

Klijent koji bi želio koristiti DHCPv4oDHCPv6 mehanizam obvezan je zatražiti tu opciju od DHCPv6 poslužitelja navodeći je u posebnom bloku u kojem se nalazi popis traženih opcija. Ako se navedena opcija nalazi u odgovoru DHCPv6 poslužitelja klijent je u mogućnosti koristiti DHCPv4oDHCPv6 mehanizam za dobivanje DHCPv4 konfiguracije. Ako opcija nije uključena u DHCPv6 odgovor, klijent ne smije koristiti DHCPv4oDHCPv6 mehanizam.

3.1.1. Slučaj sa relejom

DHCPv4oDHCPv6 mehanizam mora raditi i u slučaju da se između klijenta i poslužitelja nalazi relej kako je prikazano na slici 3.5.



Slika 3.5: Primjer arhitekture sa relej agentom

U tom slučaju potrebno je koristiti *Flags suboption* koji je definiran IETF standardom RFC5010 [30]. Navedeni zahtjev potreban je zbog slučaja u kojem se stanje klijenta iščitava iz tipa komunikacije. Tip komunikacije može biti slanje na jednu adresu ili slanje svima (engl. broadcast). Na primjer, ako je klijent u RENEWING stanju on koristi jednoodredišnu adresu za dobivanje nove konfiguracije. Klijent u REBINDING stanju koristi broadcast adresu. U slučaju da je između klijenta i poslužitelja relej agent te je klijent u RENEWING stanju i pošalje DHCPREQUEST poruku, poslužitelj ne može odrediti u kojem je stanju klijent. Koristeći *Flags suboption* relej agent eksplicitno definira tip poslane poruke.

Također, pošto se koristi DHCPv6 protokol za enkapsuliranje DHCPv4 poruke te kako ta dva entiteta nemaju nikakvu korelaciju, nemoguće je odrediti da li je DHCPv4 poruka trebala biti poslana na jednu adresu ili svima gledajući IPv6 zaglavlje DHCPv6 poruke.

Nadalje kako bi se obznanio način na koji bi se DHCPv4 poruka trebala poslati koristi se *Unicast* zastavica u *DHCPv4-QUERY* poruci. Tu zastavicu potrebno je postaviti na 1 ako je DHCPv4 paket trebao biti poslan koristeći jednoodredišnu adresu ili na 0 ako je DHCPv4 paket trebao biti poslan svima. Izbor načina na koji je IPv4 poruka trebala biti poslana vrši se prema IETF RFC2131 dokumentu [31] i njegovim proširenjima.

3.1.2. DHCPv4oDHCPv6 tok rada

U početnom stanju na klijentu, DHCPv4oDHCPv6 mehanizam mora biti isključen. Prije korištenja DHCPv4oDHCPv6 mehanizma klijent mora dobiti potrebnu IPv6 konfiguraciju. Klijent koji želi koristiti DHCPv4oDHCPv6 mehanizam obvezan je zatražiti *DHCP4o6_SERVER_ADDRESS* opciju navodeći je u *Option Request Option* bloku u svakom stanju. Poslužitelj može odgovoriti sa navedenom opcijom te u tom slučaju klijent je obvezan uključiti DHCPv4oDHCPv6 mehanizam. U slučaju da poslužitelj u svom odgovoru ne uključi opciju, klijent mora isključiti DHCPv4oDHCPv6 mehanizam. Ako postoji tražena opcija ali je prazna, klijent šalje višeodredišnu poruku na adresu *ALL-DHCP_Relaty_agents_and_servers*, no ako postoji lista poslužitelja, klijent bi trebao poslati zahtjev na svaku adresu koja se nalazi u listi.

Kada poslužitelj dobije *DHCPv4-QUERY* poruku od klijenta, on traži *DHCPv4 Message* opciju te u slučaju da opcija nije nađena odbacuje poslanu poruku. Ako tražena poruka postoji, poslužitelj je dekapsulira.

3.1.3. Sigurnost DHCPv4o6 mehanizma

Klijent je obavezan pregledati listu dobivenih DHCPv4oDHCPv6 poslužitelja te ukloniti višestruka pojavljivanja iste IPv6 adrese DHCPv4oDHCPv6 poslužitelja. U slučaju zanemarivanja tog koraka, moguć je napad amplificiranjem [32].

4. Implementacija rješenja

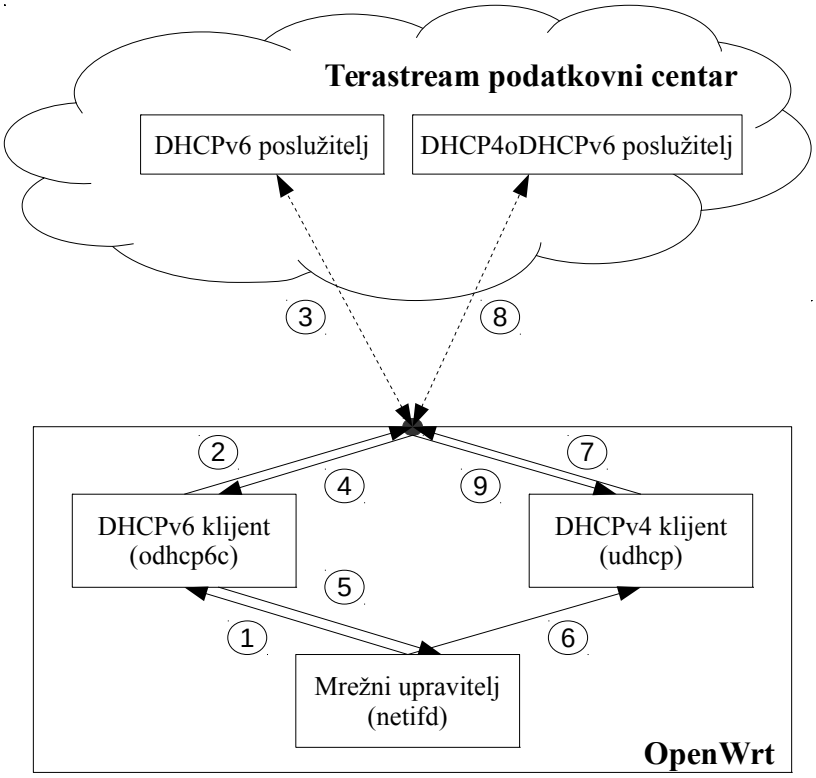
Za implementaciju rješenja potrebno je bilo modificirati tri OpenWrt korisničke aplikacije: odhcp6c, udhcp, netifd i skripte koje koriste te ISC DHCPv6 poslužitelj. Da bi DHCPv4oDHCPv6 mehanizam funkcionirao potrebno je bilo modificirati DHCPv6 klijenta, DHCPv4 klijenta te mrežnog upravitelja koji je zadužen za njihovo pokretanje. Na OpenWrt Linux distribuciji odhcp6c korisnička aplikacija pruža mogućnosti DHCPv6 klijenta, busybox aplikacija udhcp pruža mogućnosti DHCPv4 klijenta, a netifd aplikacija zadužena je za njihovo pokretanje. Implementirano rješenje moguće je razložiti na dvije glavne komponente: DHCPv4oDHCPv6 klijenta i DHCPv4oDHCPv6 poslužitelja. Klijent na kojem je instalirana OpenWrt Linux distribucija uređaj je pod nazivom Homegateway [33]. Navedeni uređaj pozicioniran je između klijentske lokalne mreže i R1 usmjerivača te predstavlja entitet preko kojega je krajnjem korisniku omogućeno korištenje usluga mreže.

Prije početka DHCPv4oDHCPv6 mehanizma potrebno je dobiti IPv6 konfiguraciju i u tom postupku popis adresa DHCPv4oDHCPv6 poslužitelja. Ako DHCPv4oDHCPv6 popis adresa poslužitelja postoji ili je prazan potrebno je pokrenuti DHCPv4oDHCPv6 mehanizam. Za potrebe komunikacije u ovom rješenju koristi se DHCPv4 klijent koji svoje poruke enkapsulira u DHCPv6 poruke te ih šalje na specificiranu adresu. Nakon što se od DHCPv6 poslužitelja dobije zatražena opcija, ona se mora zapamtiti te je potrebno zapamtiti i dobiveni popis adresa DHCPv4oDHCPv6 poslužitelja. Za pamćenja dobivene liste poslužitelja potrebno je bilo staviti tu listu na raspolaganje netifd mrežnom upravitelju koji u procesu podizanja sučelja pamti njegovu konfiguraciju i različite varijabilne podatke. Nakon što je DHCPv6 konfiguracija dobivena te sa njom i lista adresa DHCPv4oDHCPv6 poslužitelja te je sučelje konfigurirano, moguće je započeti dobivanje IPv4 konfiguracije korištenjem DHCPv4oDHCPv6 mehanizma. U UCI konfiguracijskoj datoteci za mrežne postavke, navedeno je da se DHCPv4 klijent pokreće tek nakon dobivanja DHCPv6 konfiguracije. Pri podizanju DHCPv4 klijenta potrebno mu je proslijediti parametre kao što su adresa DHCPv4oDHCPv6 poslužitelja, jedinstveni identifikator klijenta i IPv6 adresa. Nakon pokretanja DHCPv4 klijenta započinje dobivanje IPv4 konfiguracije DHCPv4oDHCPv6 mehanizmom kako je opisano u poglavlju DHCPv4oDHCPv6. DHCPv4 klijent dalje koristi standardni DHCPv4 protokol te svaku njegovu poruku enkapsulira te je šalje na definiranu DHCPv4oDHCPv6 adresu poslužitelja. Poslužitelj je također morao biti modificiran. Modifikacija poslužitelja bila je potrebna stoga što DHCPv4oDHCPv6 poslužitelj mora moći primati DHCPv4 poruke enkapsulirane u DHCPv6 pakete. Osim primanja, DHCPv4oDHCPv6 poslužitelj mora imati sve mogućnosti DHCPv4 poslužitelja te na dekapsulirane DHCPv4 poruke treba ispravno odgovarati. Svoje DHCPv4 odgovore treba enkapsulirati u DHCPv6 pakete te ih takve poslati kroz IPv6 mrežu. Modifikacija DHCPv4oDHCPv6 poslužitelja ne razmatra se u sklopu ovog poglavlja.

Na slici 4.1 prikazana je pojednostavljena komunikacija između komponenti koje sudjeluju u DHCPv4oDHCPv6 procesu koja se odvija sljedećim redom:

1. mrežni upravitelj pokreće aplikaciju koja je zadužena za dobivanje IPv6 konfiguracije korištenjem DHCPv6 protokola
2. aplikacija započinje komunikaciju dobivanja IPv6 konfiguracije i liste DHCPv4oDHCPv6 poslužitelja

3. komunikacija za dobivanje IPv6 konfiguracije odvija se po DHCPv6 standardu [34]
4. aplikacija pamti dobivenu konfiguraciju i listu DHCPv4oDHCPv6 poslužitelja
5. listu DHCPv4oDHCPv6 poslužitelja aplikacija prenosi mrežnom upravitelju koji ju tada interno pohranjuje
6. prije pokretanja aplikacije koja omogućuje DHCPv4oDHCPv6 mehanizam mrežni upravitelj prenosi listu adresa DHCPv4oDHCPv6 poslužitelja skripti za pokretanje aplikacije koja omogućuje DHCPv4oDHCPv6 mehanizam
7. aplikacija korištenjem DHCPv4oDHCPv6 mehanizma započinje dobivanje DHCPv4 konfiguracije
8. komunikacija za dobivanje DHCPv4 konfiguracije odvija se prema DHCPv4 standardu [35] uz korištenje DHCPv4oDHCPv6 mehanizma
9. nakon uspješnog završetka komunikacije klijent dobiva potrebne parametre te konfigurira sučelje



Slika 4.1: DHCPv4oDHCPv6 proces dobivanja IPv4 adrese

4.1. DHCPv4oDHCPv6 klijent

U implementaciji DHCPv4oDHCPv6 klijenta sudjeluju tri korisničke aplikacije: odhcp6c, netifd i udhcp. U sljedećim potpoglavljima navedene su promjene koje su bile potrebne za postizanje DHCPv4oDHCPv6 mehanizma na klijentu.

4.1.1. odhcp6c

Odhcp6c je aplikacija koja nudi mogućnosti DHCPv6 klijenta. Za ostvarivanje DHCPv4oDHCPv6 mehanizma odhcp6c aplikaciju je bilo potrebno modificirati. Prema DHCPv4oDHCPv6 specifikaciji, prije pokretanja DHCPv4oDHCPv6 mehanizma nužno je dobiti DHCPv6 konfiguraciju. U DHCPv6 komunikaciji, osim same konfiguracije moguće je dobiti popis DHCPv4oDHCPv6 poslužitelja na temelju kojeg se dalje odlučuje da li će se koristiti DHCPv4oDHCPv6 mehanizam. Odhcp6c aplikaciju bilo je potrebno modificirati tako da u svakoj DHCPv6 poruci zatraži listu adresa DHCPv4oDHCPv6 poslužitelja te da zapamti listu ako je dobije. Prema DHCPv4oDHCPv6 specifikaciji, za traženje liste DHCPv4oDHCPv6 poslužitelja potrebno je zatražiti tu opciju standardnim putem traženja opcija od DHCP poslužitelja, a to je navođenjem brojčane konstante opcije u *Option Reuquest Option* (skr. *ORO*) bloku [36]. Za tu promjenu u odhcp6c aplikaciji koristila se konstanta *DHCPV6_OPT_DHCP4_O_DHCP6_SERVER* [37] koja je sadržavala brojčanu konstantu 99 (150 u lokalnoj testnoj okolini). U odhcp6c aplikaciji ORO se stvara te se pamti kao interno stanje aplikacije [38]. U svakom zahtjevu odhcp6c će iščitati stanje konstruiranog ORO bloka te će ga takvog upotrijebiti u poruci. Dalje, kada odhcp6c aplikacija dobije odgovor od DHCPv6 poslužitelja ona kreće u parsiranje dobivenih opcija. U funkcijama `dhcpv6_handle_reply` i `dhcpv6_handle_advert` potrebno je bilo dodati još jedan uvjet koji ispituje da li je trenutna opcija u procesu parsiranja jednaka *DHCPV6_OPT_DHCP4_O_DHCP6_SERVER* konstanti [39][40]. U slučaju da je prije navedeno istina, potrebno je bilo zapamtiti te podatke. Za pamćenje dobivenih podataka koristila se odhcp6c funkcija `odhcp6c_add_state` sa prikladnom konstantom *STATE_DHCP4O6_SERVER*S koja označuje da je primljena *DHCPV6_OPT_DHCP4_O_DHCP6_SERVER* opcija te da u podacima može postojati lista od nula ili više IPv6 adresa DHCPv4oDHCPv6 poslužitelja. Elementi dobivene liste ne upotrebljavaju se direktno u odhcp6c aplikaciji nego tek kod pokretanja i rada DHCPv4 klijenta. Tu listu bilo je potrebno staviti na raspolaganje netifd mrežnom upravitelju te jednom kada se lista zapamti, moguće ju je upotrijebiti u druge svrhe. Pošto je odhcp6c aplikacija pokrenuta od strane netifd mrežnog upravitelja te se ona i izvršava u sprezi sa mrežnim upraviteljem potrebno je bilo izmijeniti skriptu [41] koja reagira na promjenu internog stanja odhcp6c aplikacije. Prije tog poziva, odhcp6c aplikacija od svih važnih zapamćenih parametara stvara varijable okoline koje stavlja na raspolaganje skripti nakon njenog poziva [42]. U skripti je dodatno trebalo od dobivene liste DHCPv4oDHCPv6 poslužitelja napraviti prikladan zapis te takav uz pomoć funkcije `proto_add_dhcp4o6_server` [43] staviti na raspolaganje netifd mrežnom upravitelju.

4.1.2. netifd

Lista DHCPv4oDHCPv6 poslužitelja stavlja se na raspolaganje netifd mrežnom upravitelju pozivom funkcije `proto_send_update` iz odhcp6c skripte [44]. Zadaća netifd

mrežnog upravitelja je zapamtiti dobivenu listu adresa DHCPv4oDHCPv6 poslužitelja te na prikladan način omogućiti njeno dohvaćanje. Nakon poziva funkcije `proto_send_update` poslana se lista preuzima u `proto_shell_update_link` funkciji [45] te se stvara VList struktura [46] upotrebom `interface_add_dhcp4o6_server_list` [47] i `interface_add_dhcp4o6_server` [48] funkcije. Poslije potonjih akcija lista adresa DHCPv4oDHCPv6 poslužitelja zapamćena je unutar netifd mrežnog upravitelja. Za dohvaćanje liste adresa DHCPv4oDHCPv6 poslužitelja iskorištena je već dostupna netifd ubus procedura *status*. Navedena procedura dostupna je na svakom mrežnom sučelju te služi za dohvaćanje liste trenutnih parametara tog sučelja. Funkcija za ispis parametara modificirana je za potrebe ispisa liste adresa DHCPv4oDHCPv6 poslužitelja. Funkcija koja je registrirana da reagira na *status* proceduri nalazi se u `ubus.c` datoteci. Funkcija se naziva `netifd_dump_status` [49] te kao argument dobiva pokazivač na mrežno sučelje kojemu se *status* treba ispisati. Uz pomoć funkcije `interface_ip_dump_dhcp4o6_server_list` [50] stvara se tekstualni zapis IPv6 adresa DHCPv4oDHCPv6 poslužitelja te se u ispisu pokazuje kao JavaScript Object Notation [51] (skr. JSON) polje pod ključem naziva *dhcp4o6-servers*. Nakon modifikacije implementacije ubus *status* procedure te pozivom te procedure moguće je dobiti listu adresa DHCPv4oDHCPv6 poslužitelja ako ona postoji.

4.1.3. udhcp

Zadaća udhcp aplikacije je provođenje DHCPv4 protokola. Modificirana udhcp aplikacija osim DHCPv4 protokola pruža mogućnost upotrebe DHCPv4oDHCPv6 mehanizma. DHCPv4oDHCPv6 komunikacija smije početi tek kada se napravi IPv6 konfiguracija koju obavlja odhcp6c aplikacija. Navedena restrikcija ostvarena je preko UCI mrežne konfiguracijske datoteke navođenjem da se DHCPv4 sučelje podiže tek nakon DHCPv6 sučelja. Netifd mrežni upravitelja zadužen je za administraciju svih mrežnih uređaja koji su navedeni u konfiguracijskoj datoteci. Nakon što je odhcp6c aplikacija obavila Ipv6 konfiguraciju i dohvaćanje liste DHCPv4oDHCPv6 poslužitelja, nakon što je netifd mrežni upravitelj dohvatio i zapamtio listu DHCPv4oDHCPv6 poslužitelja, pokreće se udhcp aplikacija. Ako se udhcp aplikacija želi koristiti u DHCPv4oDHCPv6 načinu rada potrebno je eksplicitno aktivirati takav način rada te pružiti i dodatne parametre koji su potrebni: DHCPv4oDHCPv6 adresa poslužitelja, IPv6 adresa i jedinstveni identifikator klijenta (engl. Client ID). Također, ako se želi koristiti DHCPv4oDHCPv6 mehanizam u mrežnoj UCI konfiguracijskoj datoteci potrebno je navesti ime sučelja koje je zaduženo za DHCPv6 komunikaciju korištenjem *dhcp4o6if* opcije. Na slici 4.2 prikazana je takva jedna jednostavna konfiguracija sučelja.

```
config interface 'wan6'
        option ifname 'eth1'
        option proto 'dhcpv6'
config interface 'wan'
        option ifname '@wan6'
        option proto 'dhcp'
        option dhcp4o6if 'wan6'
```

Slika 4.2: Primjer korištenja dhcp4o6if opcije

U slučaju da se opcija ne koristi udhcp aplikacija pokreće se u normalnom načinu rada. U slučaju da se opcija koristi potrebno je dohvatiti listu DHCPv4oDHCPv6 poslužitelja, aktivnu IPv6 adresu te izračunati jedinstveni klijentski identifikator iz IPv6 adrese. Navedeni podatci upotrebljavaju se kao parametri pri pokretanju udhcp aplikacija u DHCPv4oDHCPv6 načinu rada. Skripta [52] koja pokreće udhcp klijenta dio je netifd mrežnog upravitelja. Skripta je modificirana kako bi omogućila upotrebu DHCPv4oDHCPv6 mehanizma ako se on koristi. Pozivom ubus procedure status netifd ispisuje parametre sučelja koji se prate. U netifd poglavlju opisana je modifikacija funkcije koja je asocirana za *status* proceduru na način da dodatno ispisuje i listu adresa DHCPv4oDHCPv6 poslužitelja. Pozivom ubus status procedure [53] učitava se lista parametra dobivena u JSON formatu. Udhcp aplikaciji u DHCPv4oDHCPv6 načinu rada potrebna je adresa jednog DHCPv4oDHCPv6 poslužitelja sa kojim će aplikacija započeti komunikaciju. Iz učitane liste adresa DHCPv4oDHCPv6 poslužitelja uzima se zadnja u nizu te se ona koristiti pri komunikaciji. Sljedećim pozivom ubus status procedure [54] preuzima se IPv6 adresa. Kao zadnji parametar koji se predaje udhcp aplikaciji pri DHCPv4oDHCPv6 načinu rada je jedinstveni identifikator klijenta. Računanje identifikatora ostvareno je također u skripti koja pokreće udhcp aplikaciju.

Nakon izračunavanja parametara te uspješnog pokretanja udhcp aplikacije započinje komunikacija sa DHCPv4oDHCPv6 poslužiteljem čija je adresa predana kao jedan od parametara.

4.2. Osvrt na moguća poboljšanja

Implementacija klijentskog DHCPv4oDHCPv6 mehanizma je testirana te radi, no neki programski dijelovi nisu u skladu sa DHCPv4oDHCPv6 specifikacijom ili ih je moguće znatno poboljšati. U ovom poglavlju navedena su moguća poboljšanja implementacije klijentskog DHCPv4oDHCPv6 mehanizma.

Jedno od mogućih poboljšanja odnosi se na dohvaćanje IPv6 adrese za potrebe udhcp aplikacije. Iako je u mrežnoj UCI konfiguracijskoj datoteci postavljeno da se DHCPv4 sučelje pokreće tek nakon DHCPv6 sučelja, u trenutku dohvaćanja IPv6 adrese za potrebe pokretanja udhcp aplikacije IPv6 adresa još nije postavljena. Navedeni problem riješen je čekanjem definiranog broja sekundi prije nastavljanja procesa pokretanja udhcp aplikacije [55]. Navedeno rješenje je vrlo loše te unosi nesigurnost u radu sustava i nepotrebno čekanje. Točno i pravilno rješenje problema bilo bi pamćenje IPv6 adrese koje je odhcp6c aplikacija dobila preko Router Advertisement (skr. RA) poruka. U tom slučaju može se uzeti u obzir da će IPv6 adresa biti pročitana i zapamćena prije pokretanja udhcp aplikacije.

Slijedeće poboljšanje odnosi se na dohvaćenu listu DHCPv4oDHCPv6 poslužitelja i udhcp aplikaciju. Prema DHCPv4oDHCPv6 specifikaciji se dobije neprazna lista adresa DHCPv4oDHCPv6 poslužitelja na svaku jedinstvenu adresu iz liste šalje se DHCPv4 zahtjev. Trenutna implementacija udhcp aplikacije predviđa samo jednog DHCPv4oDHCPv6 poslužitelja sa kojim se komunicira. Trenutna implementacija iz liste dobivenih adresa DHCPv4oDHCPv6 poslužitelja uzima zadnju adresu [56] u listi te nju predaje kao parametar pri pokretanju udhcp aplikacije.

Još jedno poboljšanje odnosi se na listu adresa DHCPv4oDHCPv6 poslužitelja i udhcp aplikacije. U slučaju da je dobivena lista adresa DHCPv4oDHCPv6 poslužitelja prazna, prema specifikaciji potrebno je komunikaciju započeti sa

All_DHCP_Relay_Agents_and_Servers višeodredišnom adresom. Trenutna implementacija modifikacije udhcp aplikacije ne razmatra taj slučaj. Također, slučajevi u kojima DHCPv4oDHCPv6 poslužitelj pošalje praznu listu adresa DHCPv4oDHCPv6 poslužitelja i slučaj u kojem DHCPv4oDHCPv6 klijent uopće ne dobije listu DHCPv4oDHCPv6 adresa nisu jednaki te je potrebno drugačije reagirati na svaki. Trenutna implementacija DHCPv4oDHCPv6 mehanizma na klijentu ne razmatra slučaj u kojem DHCPv4oDHCPv6 klijent ne dobije listu adresa DHCPv4oDHCPv6 poslužitelja.

5. Zaključak

Implementacija DHCPv4oDHCPv6 mehanizma uspješno je obavljena i primijenjena u testnoj okolini no njena upotreba u produkcijskoj okolini još nije testirana. Iako je projekt završen, daljnja poboljšanja implementacije sigurno će napredovati kako će napredovati i Terastream arhitektura mreže.

Također, kako se implementacija DHCPv4oDHCPv6 klijenta temelji na nekoliko korisničkih aplikacija potrebno će biti pratiti izmjene u tim aplikacijama zbog postojanja mogućnosti prestanka rada. Prestanak rada implementacije mogao bi se dogoditi zbog toga što programski kod rješenja nije uvršten u glavne repozitorije pripadnih aplikacija. Najbolja opcija bila bi ulazak napisanog koda u glavni repozitorij aplikacije kojeg bi tada održavala zajednica te ljudi zaduženi za razvoj aplikacije. Navedeni postupak iniciran je za udhcp aplikaciju te se očekuje uvrštavanje napisanog programskog koda u glavni repozitorij aplikacije.

6. Literatura

1. Leonardo Jelenković, Skripta iz predmeta Operacijski sustavi za ugrađena računala
2. OpenWrt, URL: https://openwrt.org/ (20/03/2014)
3. OpenWrt Version History – OpenWrt Wiki,
   URL: http://wiki.openwrt.org/about/history (20/03/2014)
4. Linksys WRT54G series – Wikipedia , the free encyclopedia,
   URL: http://en.wikipedia.org/wiki/Linksys_WRT54G_series (20/03/2014)
5. History of OpenWrt (Page 1) – WhiteRussian – OpenWrt,
   URL: https://forum.openwrt.org/viewtopic.php?id=10221 (15/04/2014)
6. LKML: Andrew Miklas: Linksys WRT54G and the GPL,
   URL: https://lkml.org/lkml/2003/6/7/164 (15/04/2014)
7. Talk: OpenWRT | Dspace.org.nz,
   URL: http://dspace.org.nz/2011/08/08/openwrt-talk-notes/ (15/04/2014)
8. OpenWrt - Wikipedia , the free encyclopedia,
   URL: http://en.wikipedia.org/wiki/OpenWrt (20/03/2014)
9. Buildroot: making embedded Linux easy,
   URL: http://buildroot.uclibc.org/ (20/03/2014)
10. Yocto Project | Open Source embedded Linux build system, package metadata and SDK generator, URL: https://www.yoctoproject.org/ (20/03/2014)
11. Openembedded.org,
    URL: http://www.openembedded.org/wiki/Main_Page (20/03/2014)
12. OpenWrt/Patch – Gateworks,
    URL: http://trac.gateworks.com/wiki/OpenWrt/Patch (15/04/2014)
13. trunk – OpenWrt, URL: https://dev.openwrt.org/browser/trunk (20/03/2014)
14. Index of /, URL: http://downloads.openwrt.org/ (20/03/2014)
15. trunk – OpenWrt, URL: https://dev.openwrt.org/browser/trunk?rev=41030#package/network/ipv6/odhcp6c (20/03/2014)
16. Leonardo Jelenković, Skripta iz predmeta Operacijski sustavi za ugrađena računala
17. Embedded GLIBC - Wikipedia , the free encyclopedia,
    URL: http://en.wikipedia.org/wiki/Embedded_GLIBC (05/05/2014)
18. OpenWrt - operating system architecture – OpenWrt Wiki,
    URL: http://wiki.openwrt.org/doc/techref/architecture (20/03/2014)
19. ubus (OpenWrt micro bus architecture) – OpenWrt Wiki,
    URL: http://wiki.openwrt.org/doc/techref/ubus (15/03/2014)
20. The UCI System – OpenWrt Wiki, http://wiki.openwrt.org/doc/uci (15/03/2014)
21. netifd (Network Interface Daemon) – Technical Reference – OpenWrt Wiki,
    URL: http://wiki.openwrt.org/doc/techref/netifd (15/03/2014)

22. BusyBox, URL: http://www.busybox.net/ (15/03/2014)

23. BusyBox, URL: http://www.busybox.net/downloads/BusyBox.html (15/03/2014)

24. BusyBox, URL: http://www.busybox.net/FAQ.html

25. Terastream, URL: https://ripe67.ripe.net/presentations/131-ripe2-2.pdf (15/03/2014)

26. TCP/IP Protocol Architecture,
URL: http://technet.microsoft.com/en-us/library/cc958821.aspx (15/03/2014)

27. Internet Draft - Wikipedia, the free encyclopedia,
URL: http://en.wikipedia.org/wiki/Internet_Draft (15/03/2014)

28. draft-ietf-dhc-dhcpv4-over-ipv6-09 - DHCPv4 over IPv6 Transport,
URL: http://tools.ietf.org/html/draft-ietf-dhc-dhcpv4-over-ipv6-09 (15/03/2014)

29. draft-ietf-dhc-dhcpv4-over-dhcpv6-09 - DHCPv4 over DHCPv6 Transport,
URL: http://tools.ietf.org/html/draft-ietf-dhc-dhcpv4-over-dhcpv6-09 (15/03/2014)

30. RFC5010 - The Dynamic Host Configuration Protocol Version 4 (DHCPv4) Relay Agent Flags Suboption, URL: http://tools.ietf.org/html/rfc5010 (15/03/2014)

31. RFC 2131, URL: http://www.ietf.org/rfc/rfc2131.txt (15/03/2014)

32. What is DNS amplification attack ? - Definition from WhatIs.com,
URL: http://whatis.techtarget.com/definition/DNS-amplification-attack (15/03/2014)

33. Residential gateway - Wikipedia, the free encyclopedia,
URL: http://en.wikipedia.org/wiki/Residential_gateway (15/03/2014)

34. RFC 3315 - Dynamic Host Configuration Protocol for IPv6 (DHCPv6),
URL: http://tools.ietf.org/html/rfc3315

35. RFC 2131 - Dynamic Host Configuration Protocol,
URL: http://tools.ietf.org/html/rfc2131 (15/03/2014)

36. terastream/odhcp6c, URL:
https://github.com/terastream/odhcp6c/blob/77afb422922ae27804b9fdd59af6ed57d3d38aa9/src/dhcpv6.c#L166 (25/05/2014)

37. terastream/odhcp6c, URL:
https://github.com/terastream/odhcp6c/blob/77afb422922ae27804b9fdd59af6ed57d3d38aa9/src/odhcp6c.h#L66 (25/05/2014)

38. terastream/odhcp6c, URL:
https://github.com/terastream/odhcp6c/blob/77afb422922ae27804b9fdd59af6ed57d3d38aa9/src/dhcpv6.c#L189 (25/05/2014)

39. terastream/odhcp6c, URL:
https://github.com/terastream/odhcp6c/blob/77afb422922ae27804b9fdd59af6ed57d3d38aa9/src/dhcpv6.c#L981 (25/05/2014)

40. terastream/odhcp6c, URL:
https://github.com/terastream/odhcp6c/blob/77afb422922ae27804b9fdd59af6ed57d3d38aa9/src/dhcpv6.c#L816 (25/05/2014)

41. terastream/OpenWRT, URL: https://github.com/terastream/OpenWRT/blob/596c883c510d401d1a1839963d109972dec4c217/package/network/ipv6/odhcp6c/files/dhcpv6.script (25/05/2014)

42. terastream/odhcp6c, URL: https://github.com/terastream/odhcp6c/blob/77afb422922ae27804b9fdd59af6ed57d3d38aa9/src/script.c#L376 (25/05/2014)

43. terastream/netifd, URL: https://github.com/terastream/netifd/blob/4082f04f0198d92946401a170978ae49a76eac6f/scripts/netifd-proto.sh#L100 (25/05/2014)

44. terastream/OpenWRT, URL: https://github.com/terastream/OpenWRT/blob/596c883c510d401d1a1839963d109972dec4c217/package/network/ipv6/odhcp6c/files/dhcpv6.script#L83 (25/05/2014)

45. terastream/netifd, URL: https://github.com/terastream/netifd/blob/4082f04f0198d92946401a170978ae49a76eac6f/proto-shell.c#L421 (25/05/2014)

46. VList - Wikipedia , the free encyclopedia, URL: http://en.wikipedia.org/wiki/VList

47. terastream/netifd, URL: https://github.com/terastream/netifd/blob/4082f04f0198d92946401a170978ae49a76eac6f/interface-ip.c#L1060 (25/05/2014)

48. terastream/netifd, URL: https://github.com/terastream/netifd/blob/4082f04f0198d92946401a170978ae49a76eac6f/interface-ip.c#L1034 (25/05/2014)

49. terastream/netifd, URL: https://github.com/terastream/netifd/blob/4082f04f0198d92946401a170978ae49a76eac6f/ubus.c#L660 (25/05/2014)

50. terastream/netifd, URL: https://github.com/terastream/netifd/blob/4082f04f0198d92946401a170978ae49a76eac6f/ubus.c#L642 (25/05/2014)

51. JSON, URL: http://json.org/ (25/05/2014)

52. terastream/OpenWRT, URL: https://github.com/terastream/OpenWRT/blob/596c883c510d401d1a1839963d109972dec4c217/package/network/config/netifd/files/lib/netifd/proto/dhcp.sh (25/05/2014)

53. terastream/OpenWRT, URL: https://github.com/terastream/OpenWRT/blob/596c883c510d401d1a1839963d109972dec4c217/package/network/config/netifd/files/lib/netifd/proto/dhcp.sh#L41 (25/05/2014)

54. terastream/OpenWRT, URL: https://github.com/terastream/OpenWRT/blob/596c883c510d401d1a1839963d109972dec4c217/package/network/config/netifd/files/lib/netifd/proto/dhcp.sh#L52 (25/05/2014)

55. terastream/OpenWRT, URL: https://github.com/terastream/OpenWRT/blob/596c883c510d401d1a1839963d109972d

ec4c217/package/network/config/netifd/files/lib/netifd/proto/dhcp.sh#L40 (25/05/2014)

56. terastream/OpenWRT, URL: https://github.com/terastream/OpenWRT/blob/596c883c510d401d1a1839963d109972dec4c217/package/network/config/netifd/files/lib/netifd/proto/dhcp.sh#L46 (25/05/2014)

Popis skraćenica

IPv6	Internet Protocol version 6
IPv4	Internet Protocol version 4
DHCPv4	Dynamic Host Configuration Protocol 4
GPL	General Public License
eglibc	Embedded glibc
ubus	Micro Bus Architecture
IPC	Inter-Process Communication
UCI	Unified Configuration Interface
netifd	Network Interface Configuration Deamon
ELF	Executable and Linkable Format
IETF	Internet Engineering Task Force
ORO	Option Request Option
JSON	JavaScript Object Notation
RA	Router Advertisement

7. Dodatak A

7.1. Objašnjenje OpenWrt Buildroot `Makefile` datoteke

Na slici 7.2 prikazan je primjer Buildroot Makefile datoteke za odhcp6c korisničku aplikaciju. Linije 10-19 određuju osnovne informacije kao što je poveznica na repozitorij sa izvornim kodom aplikacije, naziv, verzija i autor aplikacije. Ostatak datoteke sastoji se od direktiva koje započinju sa ključnom riječju *define*. Tako direktiva *Package/odhcp6c* služi za definiranje ovisnosti između paketa te smještaj paketa u određenu kategoriju. Direktiva *install* služi za smještanje pojedinih datoteka pri instalaciji paketa. Ako se izrađuje nova korisnička aplikacija ili čisto zbog jednostavnosti, moguće je umjesto poveznice na repozitorij izvornog koda, koristiti putanja do izvornog koda na lokalnom stroju. Navedenu izmjenu potrebno je definirati u *Build/Prepare* direktivi na način kako je prikazano na slici 7.2.

```
define Build/Prepare:
	mkdir -p $(PKG_BUILD_DIR) $(CP) <path_to_local_repository>/* $(PKG_BUILD_DIR)/
endef
```

Slika 7.1: Primjer Build/Prepare direktive namijenjene za lokalni razvoj

Gdje `<path_to_local_repository>` predstavlja putanju do direktorija sa izvornim kodom.

```
...
10 PKG_NAME:=odhcp6c
11 PKG_VERSION:=2014-03-13
12 PKG_RELEASE=$(PKG_SOURCE_VERSION)
13
14 PKG_SOURCE:=$(PKG_NAME)-$(PKG_VERSION).tar.bz2
15 PKG_SOURCE_SUBDIR:=$(PKG_NAME)-$(PKG_VERSION)
16 PKG_SOURCE_URL:=git://github.com/sbyx/odhcp6c.git
17 PKG_SOURCE_PROTO:=git
18 PKG_SOURCE_VERSION:=9c7c654cb2d5ac6ac536f603cd5a9372416e91da
19 PKG_MAINTAINER:=Steven Barth <steven@midlink.org>
20
21 include $(INCLUDE_DIR)/package.mk
22 include $(INCLUDE_DIR)/cmake.mk
23
24 ifneq ($(CONFIG_PACKAGE_odhcp6c_ext_prefix_class),0)
25   CMAKE_OPTIONS += -DEXT_PREFIX_CLASS=$(CONFIG_PACKAGE_odhcp6c_ext_prefix_class)
26 endif
27
28 define Package/odhcp6c
29   SECTION:=net
30   CATEGORY:=Network
31   TITLE:=Embedded DHCPv6-client for OpenWrt
32   DEPENDS:=+kmod-ipv6
33 endef
34
35 define Package/odhcp6c/config
36   config PACKAGE_odhcp6c_ext_prefix_class
37     int "Prefix Class Extension ID (0 = disabled)"
38     depends on PACKAGE_odhcp6c
39     default 0
40 endef
41
42 define Package/odhcp6c/install
43     $(INSTALL_DIR) $(1)/usr/sbin/
44     $(INSTALL_BIN) $(PKG_BUILD_DIR)/odhcp6c $(1)/usr/sbin/
45     $(INSTALL_DIR) $(1)/lib/netifd/proto
46     $(INSTALL_BIN) ./files/dhcpv6.sh $(1)/lib/netifd/proto/dhcpv6.sh
47     $(INSTALL_BIN) ./files/dhcpv6.script $(1)/lib/netifd/
48 endef
49
50 $(eval $(call BuildPackage,odhcp6c))
```

Slika 7.2: Primjer Buildroot Makefile datoteke