Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matevž Podnar

Nadgradnja Linux stikala s podporo multicast preklapljanju, ki temelji na vohljanju IGMP paketov

DIPLOMSKA NALOGA
NA UNIVERZITETNEM ŠTUDIJU

doc. dr. Iztok Lebar Bajec
MENTOR

doc. ddr. Iztok Humar
SOMENTOR

Ljubljana, 2010

Univerza
v Ljubljani

Fakulteta za računalništvo
in informatiko

Tržaška 25
1000 Ljubljana, Slovenija
telefon: 01 476 84 11
faks: 01 426 46 47
www.fri.uni-lj.si
e-mail: dekanat@fri.uni-lj.si

Št. naloge: 01668/2010
Datum: 05.04.2010

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

Kandidat: **MATEVŽ PODNAR**

Naslov: **NADGRADNJA LINUX STIKALA S PODPORO MULTICAST PREKLAPLJANJU, KI TEMELJI NA VOHLJANJU IGMP PAKETOV**

IMPLEMENTATION OF IGMP SNOOPING FOR THE LINUX ETHERNET BRIDGE

Vrsta naloge: Diplomsko delo univerzitetnega študija

Tematika naloge:

Nadgradite odprtokodno Ethernet stikalo (npr. temelječe na operacijskem sistemu Linux), katerega osnovna funkcionalnost promet multicast poplavlja na vsa vrata, in sicer tako, da bo to v nadgrajeni različici promet pošiljalo le tistim odjemalcem, ki so prijavljeni v skupino uporabnikov zainteresiranih za obravnavani tok podatkov. Osredotočite se na implementacijo na podlagi vohljanja IGMP paketov. Razvijte orodja za krmiljenje in konfiguracijo sistema ter verificirajte delovanje predstavljene rešitve. Izvedite analizo zmogljivosti nadgrajenega stikala in jo primerjajte z zmogljivostjo izhodiščne verzije. Preverite tudi robustnost stikala v primeru napadov DoS s paketi IGMP. Predlagajte možnosti za nadaljnje izboljšave.

Mentor:

doc. dr. Iztok Lebar Bajec

Somentor:

doc. ddr. Iztok Humar

Dekan:

prof. dr. Franc Solina

Fakulteta za računalništvo in informatiko
2
UNIVERZA V LJUBLJANI

IZJAVA O AVTORSTVU

DIPLOMSKEGA DELA

Spodaj podpisani Matevž Podnar,

z vpisno številko 24950407,

sem avtor diplomskega dela z naslovom:

Nadgradnja Linux stikala s podporo multicast preklapljanju, ki temelji na vohljanju IGMP paketov

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom

  doc. dr. Iztoka Lebarja Bajca

  in somentorstvom

  doc. ddr. Iztoka Humarja

- 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, dne 07.05.2010 Podpis avtorja:

Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Matevž Podnar

Nadgradnja Linux stikala s podporo multicast preklapljanju, ki temelji na vohljanju IGMP paketov

POVZETEK

V diplomskem delu predstavljam izvedbo funkcionalnosti vohljanja IGMP na stikalu Ethernet, zasnovanem na operacijskem sistemu Linux. Izvedba nove funkcionalnosti je zahtevala spremembe in dopolnitve delov operacijskega sistema, in sicer tako jedra, kot tudi uporabniške programske opreme, s katero stikalo upravljamo. Implementirani so bili algoritmi in podatkovne strukture, ki se uporabljajo pri gradnji tabele multicast, poleg tega pa je bilo dodelano tudi preklapljanje, da poteka v skladu z omenjeno tabelo.

Poleg izvedbe sem v okviru diplomskega dela opravil tudi preizkus delovanja in analizo zmogljivosti nadgrajenega stikala. Prvi del preizkusov in meritev je pomenil postavitev ustreznega okolja, ki je bilo v vseh primerih sestavljeno iz stikala z nadgrajeno programsko opremo, različno pa je bilo število računalnikov, ki so promet generirali, oziroma sprejemali. Analiza je bila opravljena za dve skupini primerov: v prvi skupini se v omrežju pojavlja običajen promet multicast, v drugi pa gre za napad z velikim številom paketov IGMP.

Dobljeni rezultati so pokazali, da je izvedba vohljanja IGMP pri programskih stikalih smotrna. Poleg osnovne naloge vohljanja IGMP, to je zmanjšanja obremenitve omrežja, namreč v večini primerov pride tudi do zmanjšanja števila preklapljanj, s tem pa do manjše obremenitve in večje prepustnosti stikala. Preizkusi so dokazali ustreznost zasnove in izvedbe, saj se je izkazalo, da je nadgrajeno stikalo v primerjavi z izvirnim robustnejše tudi v primeru napada s paketi IGMP.

Ključne besede: multicast, unicast, broadcast, IGMP, vohljanje, jedro, Linux

University of Ljubljana
Faculty of Computer and Information Science

Matevž Podnar

Implementation of IGMP Snooping for the Linux Ethernet Bridge

ABSTRACT

The objective of this thesis was to implement IGMP Snooping for the Ethernet switch, based on Linux operating system. Implementation of the new functionality required modifications and improvements of different parts of the operating system: user space management software and parts of kernel. Algorithms and data structures, used for building multicast table, have been implemented in order to improve switching logic, which is now based on entries in the multicast table.

Functionality tests and performance analyses of the upgraded switch represent the second part of the thesis. First part of tests and measurements was to set up a proper testing environment, which in all cases consisted of a switch with upgraded software and a different number of computers with a role to either generate or receive multicast traffic flow. Analyses has been made for two suits of cases: in the first suit only normal multicast traffic occured in the network. The second suit was a simulation of an attack where network was flooded with a large amount of different IGMP packets.

Results showed that implementation of IGMP snooping in most cases brings positive effects on overall performance of software switches. Besides its main task, that is to lower network load, IGMP snooping in most cases also provides a method to reduce the number of switchings, which leads to lower load and higher through-put of a switch. Tests also proved suitability of the design and implementation: measurements showed that upgraded switch is more robust comparing to the original one even in the case of an attack with IGMP packets.

Key words: multicast, unicast, broadcast, IGMP, snooping, kernel, Linux

ZAHVALA

Za pomoč in koristne nasvete pri izdelavi tega diplomskega dela se zahvaljujem mentorju doc. dr. Iztoku Lebarju Bajcu, za strokovno pomoč in prijateljsko podporo pa tudi somentorju doc. ddr. Iztoku Humarju. Prav tako se zahvaljujem Mitju Marondiniju za posojeno strojno opremo ter Fakulteti za elektrotehniko za možnost izvajanja meritev na njihovi opremi in v njihovih prostorih.

Posebna zahvala tudi Živi in družini za podporo pri študiju, pomoč, spodbudo in potrpežljivost.

Brez vas tega dela ne bi bilo.

— Matevž Podnar, Ljubljana, maj 2010.

KAZALO

SPISEK KRATIC

API (angl. *Application Programming Interface*) - programski vmesnik

ASM (angl. *Any Source Multicast*) - multicast iz katerega koli vira

CAM (angl. *Content-Addressable Memory*) - pomnilnik z naslovljivo vsebino ali asociativni pomnilnik

CGMP (angl. *Cisco Group Management Protocol*) - Ciscov protokol za upravljanje skupin

DoS (angl. *Denial of Service*) - ohromitev storitve

FDDI (angl. *Fiber Distributed Data Interface*) - standard prenosov prek optičnih kablov

IEEE (angl. *Institute of Electrical and Electronics Engineers*)

IETF (angl. *Internet Engineering Task Force*) - delovna skupina za internetno tehniko

IGMP (angl. *Internet Group Management Protocol*) - internetni protokol za upravljanje skupin

IP (angl. *Internet Protocol*) - internetni protokol

LAN (angl. *Local Area Network*) - lokalno omrežje

MAC (angl. *Media Access Control*) - krmiljenje dostopa do medija

MLD (angl. *Multicast Listener Discovery*) - odkrivanje poslušalcev multicast

OSI (angl. *Open Systems Interconnection*) - odprti sistem povezovanja

OSPF (angl. *Open Shortest Path First*) - najprej odprta prosta najkrajša pot

OUI (angl. *Organization Unique Identifier*) - enolični identifikator organizacije

PGM (angl. *Pragmatic General Multicast*) - pragmatičen splošni multicast

PPP (angl. *Point-to-Point Protocol*) - protokol od točke do točke

RFC (angl. *Request For Comment*) - zahtevek za spremembo

RSTP (angl. *Rapid Spanning Tree Protocol*) - hitri protokol vpetega drevesa

SNMP (angl. *Simple Network Management Protocol*) - preprost protokol za upravljanje omrežij

SSM (angl. *Specific Source Multicast*) - multicast iz določenega vira

STP (angl. *Spanning Tree Protocol*) - protokol vpetega drevesa

TCP (angl. *Transmission Control Protocol*) - protokol za krmiljenje prenosa

UDP (angl. *User Datagram Protocol*) - uporabniški datagramski protokol

VLAN (angl. *Virtual Local Area Network*) - navidezno lokalno omrežje

1 Uvod

Vse pogostejša prisotnost širokopasovnih omrežnih povezav je omogočila uporabo medmrežja in medmrežnih protokolov tudi na področjih, kjer so se do nedavnega uporabljale druge prenosne poti. Tak primer je prenos televizije, videa in zvoka prek interneta. V okviru prenosa prek interneta poznamo več načinov distribuiranja podatkov (prometa), ki se med seboj ločijo po številu prejemnikov in virov prometa. Pri „klasičnih“ primerih uporabe interneta, kot so na primer elektronska pošta, svetovni splet, prenos datotek in podobno, je tipično v uporabi način prenosa, kjer obstajata samo en izvor in en ponor. Za prenašanje takšnih vsebin je zato najprimernejši način pošiljanje enemu samemu konkretnemu uporabniku ali pošiljanje vrste *unicast*. Nekatere druge vrste internetnih storitev pa so namenjene pošiljanju več uporabnikom. Pri tem lahko v glavnem naletimo na dve možnosti - pošiljanje samo določenim uporabnikom, oziroma *multicast*, ali pošiljanje vsem uporabnikom, oziroma *broadcast*. Televizija in radio sta primera, kjer želimo enako vsebino poslati več različnim porabnikom.

Prenos videa ali zvoka je bil do nedavnega za prenosno infrastrukturo prevelik zalogaj, zato se je uporabljal le v redkih in posebej dobro opremljenih okoljih. Danes razširjenost

tehnologij, kot so različne oblike omrežij DSL in vedno širša dostopnost optičnega omrežja omogoča prenos zahtevanih količin podatkov. Ob tem morda ni odveč omeniti, da vse vrste prenosa videa in zvoka niso najbolj primerne za prenos v obliki prenosa več prejemnikom. Na primer prenos videa na zahtevo zaradi točno določenega časa, ko nek posamezni uporabnik želi prejemati določeno vsebino, še vedno zahteva unicast način prenosa podatkov.

Teoretično bi lahko seveda tudi pri prenašanju televizijskega ali radijskega programa prek interneta uporabili prenos unicast, vendar bi za to potrebovali izjemno zmogljive računalniške sisteme, ki bi bili sposobni obdelati veliko količino zahtev po povezavah ter infrastrukturo, ki bi bila sposobna vse podatke prenesti. Prenos prometa tipa unicast namreč običajno deluje na naslednji način: računalnik, ki se v tem primeru imenuje *strežnik*, na točno določenem naslovu čaka na zahtevo po prenosu podatkov. Vsak uporabnik, ki se v tem primeru imenuje *odjemalec* in ki želi dostopati do podatkov na strežniku, strežniku posreduje zahtevo po podatkih. Na zahtevo se strežnik odzove tako, da prične pošiljati podatke. Predstavljamo si lahko obremenitev strežnika v primeru, ko na tisoče ljudi želi spremljati, denimo, nogometno tekmo.

Očitno potrebujemo drugačno rešitev, in sicer takšno, ki omogoča dostavo podatkov velikemu številu uporabnikov ob sorazmerno nizki obremenitvi strežnika, ki vsebine ponuja. Načini prenosa podatkov, ki to omogočajo, so lahko:

- prenos do skupine odjemalcev ali multicast,
- prenos do vseh odjemalcev ali broadcast ter
- prenos vrste *anycast*, ki označuje prenos, ki je prav tako vrste „od enega do več“ in kjer sicer obstaja množica uporabniških končnih točk, vendar v vsakem trenutku samo en izmed teh končnih odjemalcev prejema informacije od izbranega vira.

V diplomskem delu se s prometom vrste anycast ne bom posebej ukvarjal, posvetil pa se bom prometu multicast [1] in tudi broadcast. Z njima je mogoče ob sorazmerno nizki obremenitvi strežnika podatke prenesti večjemu številu odjemalcev. To je mogoče zato, ker za distribucijo podatkov ne skrbi strežnik, temveč drugi omrežni elementi - stikala in usmerjevalniki. Poleg očitne prednosti glede obremenitve strežnikov pri zagotavljanju dostopa do nekaterih vrst vsebin, pa vse tehnologije prenosa od ene točke v več točk prinašajo tudi nekaj pasti. Eno izmed njih predstavlja nevarnost izpostavljenosti uporabnikov pred napadi tipa *ohromitve storitve* (angl. *Denial of Service* ali skrajšano *DoS*),

kjer napadalec povzroči odpoved delovanja omrežne opreme s povzročanjem izredno velike količine prometa, ki ga omrežna oprema ne more več ustrezno obdelati.

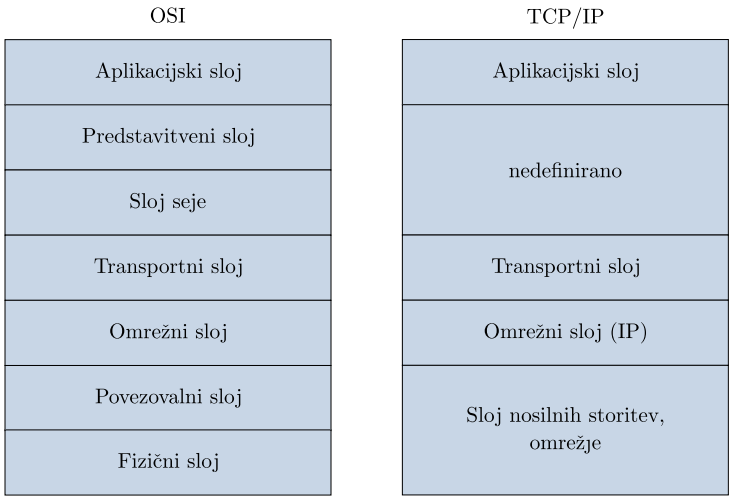
Kot smo že ugotovili, prenos televizijskega ali radijskega programa prek interneta ni primeren za unicast način prenosa podatkov. Za nekatere primere uporabe prav tako ni primeren niti broadcast, saj takšen prenos pomeni „poplavljanje“ celotnega omrežja in vseh odjemalcev s podatki. Slednje v primeru, ko vsi odjemalci niso zainteresirani za poslane vsebine, predstavlja nepotrebno obremenitev tako omrežja, omrežne opreme, kot tudi računalnikov odjemalcev. Vsak podatek, ki ga v omrežje pošlje strežnik, se na vsakem stikalu in usmerjevalniku „razmnoži“ in pošlje po vseh prenosnih poteh, ki so priključene na omrežni element, na koncu svoje poti pa povzroči, da se strojna oprema končnih uporabnikov ukvarja s prometom, ki je v resnici ne zanima. Odvečen promet predstavlja težavo predvsem s stališča izkoriščenosti omrežja - strojna oprema zaradi zavračanja odvečnega prometa ni posebej dodatno obremenjena in ji to ne predstavlja težav - vsekakor pa omrežja v času, ko je zasedeno z odvečnim prometom, ne moremo uporabljati za prenos koristnih podatkov.

Promet je potrebno dostaviti samo zainteresirani skupini odjemalcev, torej pride v poštev le multicast. Zaradi določenih razlogov, ki so podrobneje opisani v nadaljevanju, pa do nezaželenega podvojevanja podatkov prihaja tudi pri prenosu multicast. S to zagato so se proizvajalci omrežne opreme spopadli na različne načine, nekateri z uvajanjem novih protokolov, kot je na primer to storil znani proizvajalec omrežne opreme Cisco Systems Inc., drugi pa z eno od implementacij tako imenovanega *vohljanja* (angl. *snooping*), ki odkriva in razlikuje promet, ter poskuša zagotoviti kar najmanjšo obremenitev omrežja in omrežne opreme.

1.1 Omrežni sloji in multicast

Podobno kot pri vseh ostalih vrstah komunikacije, tudi komunikacije v računalniških omrežjih potekajo po vnaprej definiranih predpisih ali protokolih. Glede na infrastrukturo in glede na potrebe konkretnih posameznih uporab (aplikacij) pa pridejo v poštev kombinacije različnih protokolov, ki komuniciranje opredeljujejo na različnih ravneh oziroma slojih. Zato je za podrobnejšo predstavitev dogajanja pri komuniciranju prek računalniških omrežij in boljše razumevanje namena in ciljev tega diplomskega dela potrebno uvesti ter razložiti nekaj pojmov s tega področja.

Ko govorimo o prenosu podatkov prek interneta, se ves čas srečujemo z izrazi, kot so: protokoli, protokolni skladi, podatkovni okvirji in paketi. V primeru interneta največkrat govorimo o modelu[1] ali skladu TCP/IP, ki je sestavljen iz štirih slojev. Vsak sloj opravlja neko storitev in pri tem komunicira samo s slojema, ki sta eno stopnjo nižje oziroma višje v hirearhiji slojev. Najnižji ali prvi sloj v primeru modela TCP/IP je *sloj nosilnih storitev*, sledijo *omrežni sloj, transportni sloj* in *aplikacijski sloj*. Splošnejši model, o katerem se pri telekomunikacijah pogosto govori, je sklad OSI, ki določa sedem slojev, zaradi česar so posamezni protokoli, ki se v primeru modela TCP/IP uvrščajo v isti sloj, lahko v primeru sklada OSI uvrščajo v različne sloje (slika 1.1). Nivoji modela OSI so: *fizični sloj, povezovalni sloj, omrežni sloj, transportni sloj, sloj seje, predstavitveni sloj, aplikacijski sloj*. Podatki po omrežju potujejo združeni v povezane enote, pri čemer je konkretna oblika podatkov seveda odvisna od sloja, ki ga opazujemo. Na prvem sloju tako lahko vidimo dejanske električne, optične ali drugačne impulze, ki predstavljajo zaporedje ničel in enic, na višjih slojih pa govorimo o logičnih enotah, ki jim na drugem sloju modela OSI rečemo okvirji, na tretjem pa paketi. Ker je model OSI vsaj v teoriji telekomunikacij



Slika 1.1 Primerjava protokolnih skladov OSI in TCP/IP.

pogosteje uporabljen, se bom tudi pri navajanju slojev v tem delu odslej vedno skliceval nanj. Za opis prometa, ki potuje do skupine odjemalcev, oziroma k vsem, ki se nahajajo na nekem segmentu omrežja, si je potrebno nekoliko podrobneje ogledati drugi, tretji

[1] Zbirki pravil, ki so namenjena vzpostavitvi komunikacije ter komunikaciji med dvema ali več točkami v omrežju.

in četri sloj. Višji sloji (peti, šesti in sedmi) so namreč odvisni od konkretnih aplikacij, ki uporabljajo omrežne povezave, zato niso predmet tega diplomskega dela. Prav tako podrobnejši opis prvega sloja, ki se ukvarja z električnimi in fizičnimi specifikacijami naprav presega obseg tega dela. V okviru diplomskega dela zadošča vedenje, da prvi sloj določa povezave med napravami in fizičnim medijem (določa torej omrežne vmesnike, specifikacije kablov, napetosti itd.).

Lahko rečemo, da se prvi sloj od drugega loči v tem, da če prvi sloj določa predvsem interakcijo ene naprave z nekim komunikacijskim medijem, je drugi sloj namenjen določitvi interakcije več naprav, ki so s takšnim komunikacijskim medijem povezane. Drugi, oziroma povezovalni sloj, tako zagotavlja:

- določanje enote sporočil (znake, bloke, pakete),
- način ugotavljanja napak med dvema sosednjima vozliščema,
- odpravo napak,
- omrežno topologijo,
- mehanizme dostopa do prenosnega medija,
- kontrolo pretoka.

Primeri protokolov, ki delujejo na drugem sloju, so: Ethernet, FDDI, PPP itd. Na tem mestu naj omenim, da se bom v nadaljevanju ob omembi drugega sloja konkretno ukvarjal s skupino protokolov Ethernet (IEEE 802.3). Čeprav na tem sloju poznamo tudi druge rešitve, se danes za lokalna omrežja najpogosteje uporablja prav Ethernet.

Ugotovitev, do katere pridemo po pregledu obstoječe omrežne opreme in ki je zelo pomembna za problematiko tega diplomskega dela je: *večina omrežne opreme, ki deluje na drugem sloju ne razlikuje med prometom vrste broadcast in multicast.* Tipične naprave, ki delujejo na tem sloju so stikala, ki imajo množico vrat, v katere so priključene ostale omrežne naprave. Ker običajna stikala ne ločujejo med prometom multicast in broadcast, oboje obravnavajo enako in promet vrste multicast enako kot broadcast preprosto razpošljejo vsem napravam in podomrežjem, ki so na stikalo priključena.

Nivo nad povezovalnim slojem se nahaja omrežni sloj, ki izvaja usmerjanje paketkov skozi vozlišča, od ponora do izvora. Usmerjanje je možno na osnovi izvornega in ciljnega naslova, ki se nahajata v glavi paketa. Usmerjanje poteka po različnih kanalih, odvisno

od razmer v omrežju. Za usmerjanje so odgovorni usmerjevalni algoritmi, ki jih izvajajo *usmerjevalniki*.

Omrežni sloj je tudi prvi, kjer se na običajni omrežni opremi pojavi razlika med prometom broadcast in multicast. To prinaša oviro pri zagotavljanju varnosti: uporabniki so na nekem lokalnem omrežju (ki je povezano prek stikal in ne usmerjevalnikov) potencialne tarče napada, ki bi pod krinko distribuiranja, denimo televizijske vsebine, povzročil preveliko obremenitev in posledično odpoved delovanja omrežne opreme. Včasih se takšni dogodki zgodijo tudi nehote. Znani so na primer primeri težav, ki se pojavljajo na (lokalnih) omrežjih, kjer se mešajo različni tipi storitev, velik delež celotnega prometa pa predstavlja promet tipa multicast. Slednji tako dobesedno poplavi obstoječe kapacitete in povzroči popolno odpoved delovanja vseh vrst storitev, vezanih na delovenje omrežja.

Tako z vidika varnosti kot učinkovitosti je torej vohljanje vsekakor privlačna rešitev, vendar zahteva pregledovanje vsebine (vseh) okvirjev, ki se znajdejo v stikalih, saj se le tako lahko že dovolj zgodaj (na dovolj nizkem sloju) ugotovi, kam je okvir namenjen v resnici. Na ta način uporabnike sicer resnično lahko obvarujemo pred nezaželenim prometom, vendar to prinese dodatno delo omrežnemu elementu, ki skrbi za preklapljanje okvirjev v omrežju. Vprašanje, ki se postavi, je, kolikšno dodatno obremenitev bi to prineslo?

Pred zaključkom podpoglavja o slojih je potrebno omeniti še četrti sloj, ki predstavlja transportni sloj, kjer se večina podatkov prenese s pomočjo dveh protokolov: TCP in UDP. Protokol TCP temelji na paketnem prenosu podatkov, kjer je vsak paket v grobem sestavljen iz glave, ki vsebuje podatke o izvoru in ponoru podatkov ter podaja informacije o samih podatkih in podatkovnega dela, ki vsebuje dejansko informacijo. Poleg tega je sestavni del glave TCP paketa tudi zaporedna številka, ki pove, kateri v nizu poslanih paketov je posamezni paket. Nekateri paketi, ki se pošiljajo v okviru protokola TCP, so namenjeni potrjevanju prejema - prejemnik pošilja potrditvene pakete, s katerimi sporoča, da je uspešno prejel vse pakete, ki so imeli nižjo zaporedno številko od tiste, ki jo potrjuje. Ta mehanizem omogoča tako imenovano *zanesljivo* pošiljanje podatkov, pri čemer se beseda zanesljivo ne nanaša na kakršnokoli metodo prikrivanja vsebine podatkov, temveč na to, da že sam protokol zagotavlja dostavo in pravilen vrstni red dostave podatkov višjim slojem. Prav zato se ta protokol zelo pogosto uporablja v primeru komunikacije med dvema znanima uporabnikoma. Promet tipa multicast je v

osnovi zasnovan na uporabi protokola UDP, ki je preprostejši in ne zagotavlja dostave paketov. Paketi so lahko dostavljeni v različnem vrtnem redu, vmes pa lahko kakšen tudi manjka (se je iz enega ali drugega razloga izgubil). Takšno obnašanje pa večinoma ne predstavlja težav, saj si lahko predstavljamo, da v tipični aplikaciji, ki uporablja multicast, denimo pri internetni televiziji, delci informacij, ki so se nekje na poti izgubili, ne predstavljajo več aktualne informacije in je zato nima smisla pridobivati za nazaj (če program teče naprej, nas slika ali delček slike izpred nekaj sekund ne zanima več).

Kot rečeno, promet vrste multicast za svojo osnovo uporablja protokol UDP. Za veliko večino aplikacij, ki uporabljajo multicast, to zadostuje, obstajajo pa seveda tudi primeri, ko je potrebno poskrbeti za večjo zanesljivost dostave. Možnosti zagotavljanja zanesljivejše dostave je ta hip več, omrežij, ki jih uporabljajo, pa je v primerjavi z vsemi uporabniki prometa multicast izjemno malo. Kot prvo je rešitev ponudilo že omenjeno podjetje Cisco Systems Inc., ki je predlagalo protokol PGM, s katerim se približuje lastnostim protokola TCP - paketi so vedno dostavljeni (s pomočjo mehanizma potrjevanja sprejema), so v pravilnem vrstnem redu in niso podvojeni.

Bistvo prometa vrste multicast je dostavljanje podatkov skupini odjemalcev, zato za upravljanje skupin obstaja poseben protokol, ki se imenuje IGMP (internetni protokol za upravljanje skupin). Namen protokola je prejemnikom (odjemalcem) omogočiti prijavo in odjavo v oziroma iz skupine. Z uporabo tega protokola se odjemalci tudi prijavijo na sprejem določenega prometa, ali pa se od njega odjavijo. V preteklih 20 letih so bile sprejete tri verzije tega protokola, za vse tri pa je v sklopu razprave o slojih značilno, da v okviru modela OSI delujejo na tretjem sloju in praktično ne vplivajo na delovanje omrežnih elementov, ki delujejo na drugem sloju. Ti v primeru standardnega delovanja kljub obstoju protokola ne razlikujejo med prometom vrste broadcast in multicast.

1.2 Cilji diplomskega dela

Cilj tega diplomskega dela je torej pregled in implementacija funkcionalnosti vohljanja IGMP, kar pomeni pregledovanje prispelih okvirjev, odkrivanje paketov IGMP na drugem sloju ter preklapljanje prometa multicast na način, ki bo ustrezal prej pregledanim paketom IGMP. Poleg tega je cilj preizkus delovanja in analiza zmogljivosti stikala, ki mu je dodana omenjena funkcionalnost. Ker je vohljanje IGMP sestavljeno iz dveh korakov: odkrivanja okvirjev IGMP in kasnejšega preklapljanja običajnega prometa vrste mul-

ticast, je potrebno stikalo preizkusiti v dveh skrajnih primerih. V prvem primeru je potrebno ugotoviti, kako se nadgrajeno stikalo odziva na napad z veliko količino paketov IGMP, v drugem primeru pa raziskati, kakšen je vpliv pregledovanja običajnih okvirjev multicast na obremenitev procesorja stikala in kakšne so zmogljvosti nadgrajenega stikala v primerjavi z izvirnim.

V okviru tega diplomskega dela se bom omejil na daleč najbolj razširjeno obliko prometa multicast, ki temelji na kombinaciji protokolov: na povezovalnem sloju je to Ethernet, omrežni sloj predstavlja IPv4, transportnega pa UDP.

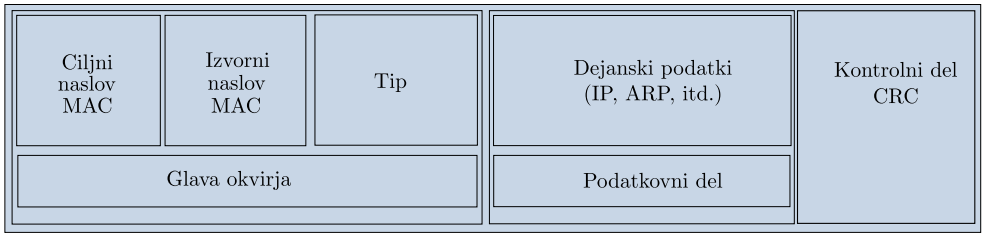
2 Delovanje stikal, multicast in protokol IGMP

Stikalo Ethernet (angl. *Ethernet switch*) je omrežni element, ki je namenjen povezovanju različnih lokalnih fizičnih omrežij. V okviru modela OSI deluje na drugem sloju. Stikalo predstavlja množica vrat, med katerimi stikalo usmerja oziroma preklaplja promet[1]. Danes so stikala široko uporabljen omrežni element, ki je zaradi zanemarljive razlike v ceni skoraj popolnoma izpodrinil nekdaj razširjeni *hub*. Slednjega podobno kot stikalo določa množica vrat, med katerimi se distribuira promet, deluje pa na prvem, torej fizičnem sloju. Njegova velika slabost je, da ves dohodni promet preprosto razširi na vse priključene naprave. Tako vse naprave, priključene na hub, prejmejo ves promet, ki je namenjen v ta segment omrežja. Nekateri aktivni hubi sicer lahko opravljajo tudi funkcijo *repetitorja*, kar pomeni, da pred pošiljanjem signala priključenim uporabnikom signal prej aktivno električno ojačajo, še bolj izpopolnjene verzije pa tudi odpravljajo težave, ki jih prinaša hkratno oddajanje prometa različnih uporabnikov, torej kolizijo signalov. Stikala opravljajo vse funkcije hubov, poleg tega pa (načeloma) odpravljajo nepotrebno pošiljanje prometa vsem uporabnikom. Hubi so tako ostali v svoji funkciji

[1] Pogosteje se na drugem sloju uporablja izraz preklapljanje prometa, saj je izraz „usmerjanje“ primernejši za usmerjevalnike, ki delujejo na tretjem sloju.

le še v primerih, kjer je razpošiljanje vsega prometa vsem uporabnikom zaželeno - na primer pri analiziranju prometa v omrežju, vendar tudi tu lahko enake učinke dosežemo s primerno konfiguracijo stikal.

Stikala torej bolj ali manj inteligentno preklapljajo promet na drugem sloju omrežja. Podatki, na katerih temelji odločanje, kako naj se stikalo pri danem okvirju obnaša, oziroma na katera vrata naj stikalo okvir preklopi, so podatki v glavi okvirja. Sestavo tipičnega okvirja prikazuje slika 2.1.



Slika 2.1 Ethernet II okvir- najpogostejši tip Ethernet okvirja danes.

Ethernet okvir tipa II je danes najbolj razširjen. Okvir je sestavljen iz treh delov:

- glave okvirja, ki je sestavljena iz treh delov:
  - ciljnega naslova MAC[2], ki predstavlja identifikacijo ponorne naprave;
  - izvornega naslova MAC, ki predstavlja identifikacijo izvorne naprave;
  - polja Tip, ki določa protokol, ki se prenaša v podatkovnem delu okvirja;
- podatkovnega dela, kjer se nahajajo dejanski podatki;
- kontrolnega dela, ki je namenjen zagotavljanju integritete okvirjev.

Stikalo vzdržuje posebno tabelo naslovov MAC, ki služi kot osnova za preklapljanje okvirjev. Ko okvir pride v stikalo, stikalo v to tabelo shrani izvorni naslov in številko vrat, kjer je stikalo zaznalo okvir. Ime, ki se za to tabelo pogosteje uporablja, je tabela CAM, ime pa izhaja iz pomnilnika z naslovljivo vsebimo (angl. *Content-Addressable Memory*), katerega lastnost je, da vsebina določa pomnilniški naslov, zato je s pomočjo takšne vrste pomnilnika zelo hitro mogoče ugotoviti katera vrata so povezana s posameznim izvornim naslovom MAC. Takšnemu pomnilniku v slovenščini pogosteje pravimo asociativni pomnilnik, več o njem in o ostalih vrstah pomnilnika si lahko preberemo v [2]. Polnjenje tabele

[2]Naslovi MAC so fizični naslovi omrežnih elementov, ki jih znotraj določenega omrežnega segmenta enolično identificirajo.

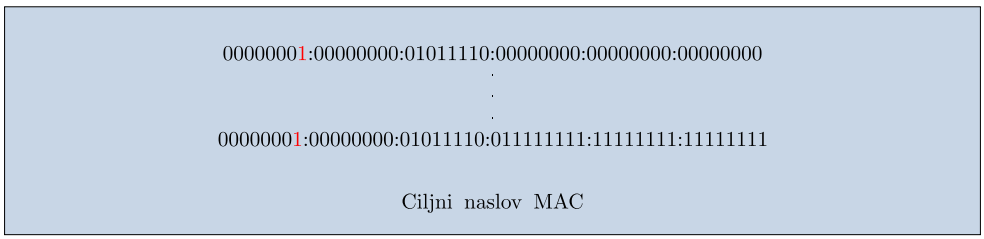
CAM, kar se pogosto označuje kot učenje, se torej vrši s pomočjo izvornih naslovov. Pri preklapljanju stikalo s pomočjo te tabele in ciljnega naslova okvirja ugotovi, na katera vrata naj preklopi dohodni promet. Če ciljnega naslova ni v tabeli CAM (je torej za stikalo neznan), ali gre za naslov broadcast oziroma multicast, se takšen okvir prenese na vsa vrata, razen izvornih. Če so ciljna vrata enaka izvornim, se okvir zavrže.

Poleg te osnovne funkcionalnosti stikala prinašajo še nekaj prednosti, vendar za problem, predstavljen v tem diplomskem delu, te niso tako pomembne, zato jih bom samo na kratko omenil. Stikala lahko izvajajo preverjanje okvirjev glede napak, podpirajo pa lahko tudi protokol (R)STP, ki je namenjen preprečevanju nastanka zank v omrežju. Gre za pomembni lastnosti stikal, vendar je na tej točki za nas pomembno le preklapljanje prometa.

2.1 Preklapljanje v primeru prometa vrste multicast

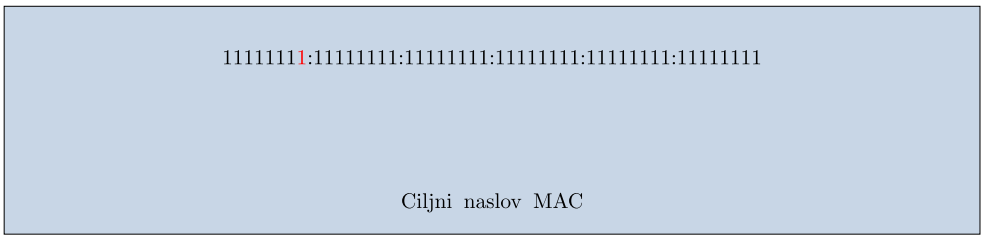
Naslovi MAC predstavljajo enolično identifikacijo omrežne naprave na drugem sloju, kar načeloma velja tako za izvorni kot ciljni naslov. Izjemo predstavlja primer, ko je promet namenjen več kot enemu odjemalcu. Namesto naslova naprave se v primeru prometa multicast prometa in broadcast kot ciljni naslov uporabljajo vnaprej določene skupine naslovov, oziroma posebni naslovi MAC.

Konkretno je prometu tipa broadcast namenjen naslov ff:ff:ff:ff:ff:ff, medtem ko je za multicast določen razpon naslovov med 01:00:5e:00:00:00 in 01:00:5e:7f:ff:ff. Razpon naslovov multicast, prikazan v dvojiškem zapisu, prikazuje slika 2.2, razpon naslovov broadcast pa slika 2.3.



Slika 2.2 Razpon naslovov multicast v dvojiškem zapisu.

Kot je z rdečo barvo poudarjeno na slikah, imajo vsi naslovi, ki so namenjeni naslavljanju več kot enega prejemnika, vrednost najmanj pomembnega bita najpomembnejšega okteta enako 1, to pa je tudi edini kriterij, po katerem Ethernet stikala, ki ne upo-

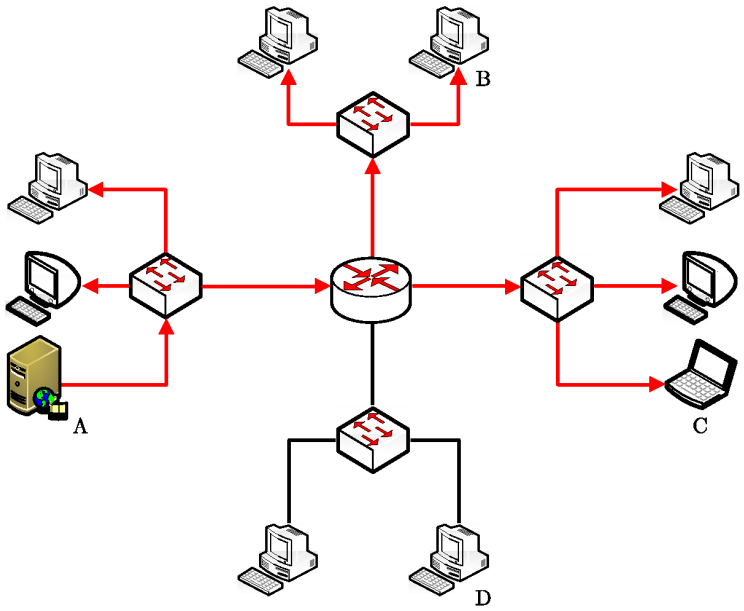


Slika 2.3 Naslov broadcast v dvojiškem zapisu.

rabljajo vohljanja IGMP, tovrstne okvirje prepoznavajo kot „skupinske“ in jih kot take razpošljejo - preklopijo na vse izhode stikala. V primeru prometa vrste multicast torej običajno stikalo deluje podobno kot na začetku opisani hub, saj promet, ki je namenjen le določenim uporabnikom, razpošlje vsem.

Na drugem sloju se okvirji, ki so namenjeni skupini, in okvirji, ki so namenjeni vsem uporabnikom, med seboj sicer razlikujejo že na prvi pogled, vendar standardno stikalo med njimi ne razlikuje. Stikala, ki podpirajo funkcionalnost vohljanja IGMP, pa zanima celoten naslov MAC. Poleg samega prepoznavanja okvirjev vrste multicast je za takšna stikala pomembno še nekaj: poleg najpomembnejših treh oktetov, ki povedo, da gre za okvir multicast (01:00:5e), jih zanima tudi preostalih 23 bitov - ti namreč določajo *skupino*, v katero spada obravnavani okvir. Stikalo, ki podpira funkcionalnost vohljanja IGMP, se prav na podlagi informacije o skupini odloča, na katera vrata je okvir potrebno distribuirati in na katera ne.

Dogajanje v tipičnem omrežju, ki ne uporabja vohljanja IGMP, prikazuje slika 2.4. Na tej sliki je računalnik A ponudnik vsebine, ki se prenaša v obliki multicast prometa (denimo televizijski program), računalnika B in C sta naročnika tega programa, oziroma odjemalca. Zaradi omenjene lastnosti tipičnega Ethernet stikala, da o prometu tipa multicast ne ve nič ter da preverja samo najmanj pomemben bit najpomembnejšega okteta, se promet vedno distribuira vsem odjemalcem, priključenim na stikalo. Zato okvirje, ki so namenjeni računalnikoma A in C prejmeta tudi sosednja računalnika, ki sta priključena na isto stikalo. Kot vidimo, pa računalnika, ki sta v segmentu, kjer se nahaja računalnik D, prometa ne prejmeta. Za to je poskrbel usmerjevalnik, ki pozna razliko med prometom broadcast in multicast.



Slika 2.4 Običajno omrežje z multicast prometom, vohljanje IGMP na stikalih ni vključeno.

2.2 Naslavljanje multicast

Iz slike 2.4 vidimo tudi, da omrežje na sloju višje (na omrežnem sloju), že pozna razliko med prometom, namenjenim le enemu uporabniku, skupini ali pa vsem. Razlog tiči v dejstvu, da so različnim vrstam prometa namenjeni tudi različni naslovi IP. Tako ima promet, ki je namenjen vsem uporabnikom (broadcast) rezervirano posebno skupino naslovov IP, ki se izračunajo tako, da se izvede logična operacija disjunkcije med bitnim komplementom podomrežne maske (angl. *subnet mask*) in naslovom IP naprave[3].

IP naslovi multicast obstajajo v razponu med 224.0.0.0 do 239.255.255.255, kar se je včasih imenovalo „razred D“ IP naslovov. V splošnem različni naslovi IP predstavljajo različne *skupine multicast*, odjemalcem se dodeljujejo dinamično, nekateri naslovi pa so rezervirani. Rezervirani naslovi so lahko lokalni (224.0.0.1 za naslavljanje vseh naprav multicast, 224.0.0.2 za naslavljanje vseh usmerjevalnikov multicast prometa, 224.0.0.5 za naslavljanje usmerjevalnikov OSPF) ali globalni (med 224.0.1.0 in 224.0.1.255 - namenjeni so različnim protokolom, ki temeljijo na prometu tipa multicast). Preostali naslovi se delijo na globalne (med 224.0.2.0 in 238.255.255.255) in lokalne (med 239.0.0.0 in

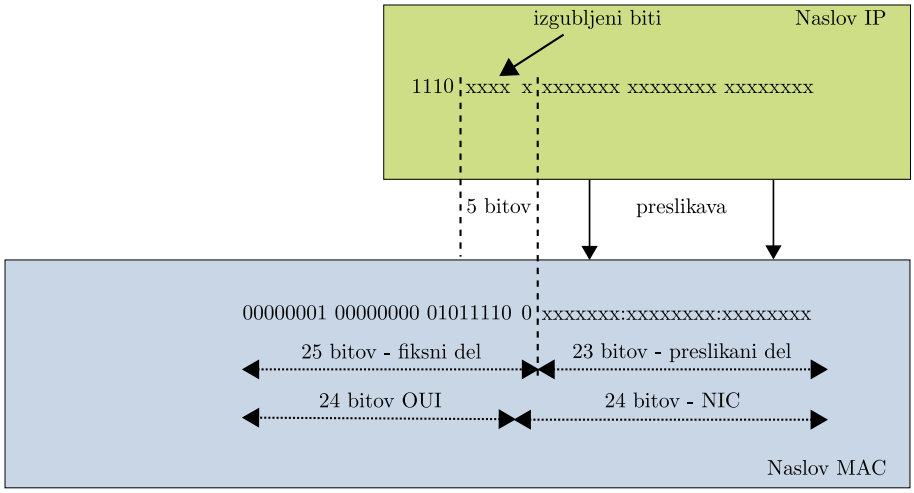
[3]Naslov broadcast je v primeru naslova 172.16.0.0 in podomrežne maske 255.240.0.0 tako enak 172.16.255.255.

239.255.255.255).

Ko govorimo o skupinah multicast, je morda zavoljo jasnosti dobro podati primer: tipičen predstavnik takšne skupine je denimo televizijski ali radijski kanal - vsak kanal pripada svoji skupini.

2.2.1 Preslikava med naslovi IP in MAC

Skupine sem omenjal že pri opisu delovanja stikal, ki se zanašajo na naslove MAC. Med naslovi IP in MAC torej v primeru multicast prometa obstaja povezava. Prikazuje jo slika 2.5. Gre za preslikavo, oziroma mapiranje, kjer je gornjih 25 bitov naslova MAC fiksno določenih. Ostalih 23 bitov se preslika neposredno iz dela naslova IP, ki je vsebovan v paketu, ki ga nosi dani okvir.



Slika 2.5 Prikaz mapiranja naslova IP v MAC.

Iz slike lahko vidimo tudi, da so naslovi MAC v splošnem sestavljeni iz dveh delov. Naslovi MAC so sicer lahko različnih dolžin, 48-bitni naslovi, kakršne uporabljamo v današnjih omrežjih Ethernet, pa so razdeljeni na:

- zgornji del v dolžini 24 bitov, ki določa proizvajalca omrežnega elementa z obravnavanim naslovom MAC ter
- spodnji del enake dolžine, ki pri unicast naslovih določa obravnavani omrežni element, pri multicast naslovih pa spodnjih 23 bitov določa skupino, najpomembnejši bit pa je vedno 0.

Proizvajalci omrežne opreme morajo svojo identifikacijsko število, ki predstavlja zgornjih 24 bitov, registrirati pri mednarodnem konzorciju IEEE, ta koda pa se imenuje OUI. Za potrebe multicasta je, kot vidimo, rezervirana ena sama številka OUI, kar je razlog, da ne moremo uporabiti celotnega naslova IP, ki je dolg 32 bitov. Glede na razpon naslovov IP, ki so rezervirani za multicast, bi bilo namreč potrebno rezervirati 16 številk OUI.

Dodaten fiksno določeni bit (najpomembnejši bit drugega okteta v naslovu IP oziroma 25. bit v naslovu MAC) pa je cena odločitve, da bodo polovico drugega dela naslova MAC rezervirali za prihodnjo rabo. Razlogi za današnje dokaj nenavadno mapiranje naslovov IP v MAC so torej zgodovinski.

Slika 2.5 tudi kaže, da mapiranje naslova IP v naslov MAC ni bijektivno, pač pa surjektivno: ker se 5 bitov, ki določajo naslov IP, pri mapiranju izgubi, se v enak naslov MAC preslika 32 naslovov IP. V enak MAC se tako preslikajo vsi naslovi IP, ki se razlikujejo samo v prvem oktetu, poleg tega pa se zaradi samo sedmih uporabljenih bitov drugega okteta naslova IP, naslova, kot sta 224.129.1.2 in 224.1.1.2 prav tako preslikata v enak naslov MAC. To pomeni, da se na drugem sloju samo s pregledovanjem naslovov MAC ne moremo izogniti temu, da bi nekaj neželenega prometa zašlo tudi k sosednjim uporabnikom. Kljub temu pa lahko preprečimo, da bi ves promet tipa multicast, ki se znajde na nekem fizičnem segmentu omrežja, romal k vsem.

2.3 Protokol IGMP

Protokol IGMP je komunikacijski protokol, namenjen prijavi in odjavi odjemalca v skupino odjemalcev, ki želijo prejemati okvirje, ki pripadajo določeni multicast skupini (denimo, naročiti se želijo na sprejem določenega televizijskega programa). IGMP je sestavni del specifikacije *IPv4 multicast*, ki določa vse podrobnosti v zvezi s pošiljanjem podatkov od vira (ali virov) do skupine odjemalcev. Ker je protokol namenjen nadzoru omrežja, lahko rečemo, da je IGMP v primeru multicast analogen protokolu ICMP[4], ki se uporablja v primeru unicast prenosa podatkov. IGMP ima analogen protokol tudi v domeni protokola IPv6, in sicer gre za protokol MLD, ki ga bom v nadaljevanju opisal le na kratko. Protokol IGMP je definiran v okviru tretjega sloja, zato se znotraj okvirjev Ethernet nahaja v podatkovnem delu okvirja (slika 2.1).

Trenutno poznamo tri različice protokola, definirane pa so v naslednjih dokumentih: prva verzija je definirana v RFC 1112 [3], druga v RFC 2236 [4], tretja pa v RFC 3376 [5].

Ko govorimo o protokolu IGMP, ločimo:

- odjemalce (prejemnik podatkovnega toka, običajno osebni računalnik, podatkovni komunikator za IP televizijo...) in
- povpraševalce (robni usmerjevalnik multicast).

Odjemalec, ki želi prejemati določen multicast promet, pošlje poseben paket (na tem sloju že govorimo o paketih), ki označuje vključitev v skupino, oziroma ohranitev članstva v skupini, v katero je že priključen.

Povpraševalci, ki delujejo na tretjem sloju (usmerjevalniki), takšne pakete prepoznajo in si ob njihovem prihodu v posebno tabelo shranijo identifikacijsko številko skupine ter številko vrat. Nato paket pošljejo naprej. S pomočjo te tabele nato usmerjevalniki usmerjajo multicast promet, ki prihaja iz nasprotne smeri (ponudnika vsebin) do odjemalcev, ki so se prijavili v določeno skupino. Ti paketi so lahko poslani samoiniciativno s strani odjemalca, lahko pa nastopijo kot odgovor na povpraševanje s strani povpraševalca. Iz skupine je mogoče tudi izstopiti. Podrobnosti se v tem primeru med različnimi verzijami protokola nekoliko razlikujejo.

[4]ICMP se v osnovi ne uporablja za prenašanje podatkov, pač pa za prenos sporočil o napakah in stanju omrežja. Neposredno ga uporablja le malo aplikacij, izjemi sta znani storitvi *ping* in *traceroute*.

2.3.1 Protokol IGMP, verziji 1 in 2

Najstarejša verzija protokola IGMP, verzija 1, je bila predstavljena že leta 1989 [3]. Leta 1997 ji je sledila verzija 2 [4], ki pa se od prve po svoji zasnovi ne razlikuje bistveno. Obe verziji protokola predvidevata dve vrsti sporočil, in sicer:

- *IGMP Report* - sporočilo odjemalca, da želi biti vključen v skupino in
- *IGMP Query* - sporočilo povpraševalca, na katerega pričakuje odziv odjemalcev.

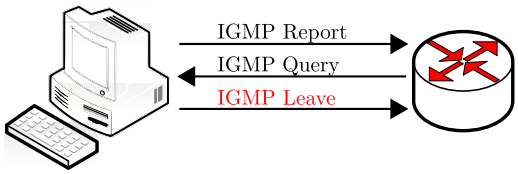
Poleg teh dveh pa druga različica protokola IGMP pozna še tretjo vrsto sporočila, in sicer:

- *IGMP Leave* - sporočilo odjemalca, da želi izstopiti iz skupine.

Pravilna imena sporočil, ki jih najdemo v standardih [3] in [4], so sicer nekoliko drugačna: IGMP Report je *Host Membership Report*, IGMP Query je *Host Membership Query*, IGMP Leave pa *Leave Group*. Zaradi novih imen sporočil z enakim ali sorodnim namenom, ki jih je uvedel standard IGMPv3, pa teh v nadaljevanju diplomskega dela ne bom uporabljal in se bom držal imen IGMP Report, IGMP Query ter IGMP Leave, razen, kadar je posebej pomembno, za katero verzijo protokola IGMP gre.

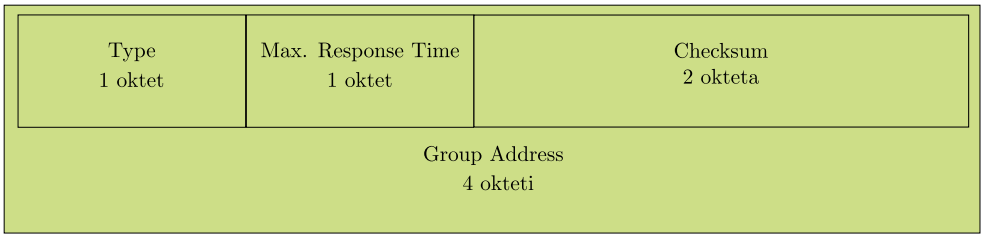
Prvi dve vrsti sporočil sta namenjeni prijavi odjemalcev v skupino. Povpraševalci periodično pošiljajo sporočila (zahteve) IGMP Query, na katera pričakujejo odgovor IGMP Report, ki ga odjemalci lahko pošljejo tudi brez predhodne zahteve povpraševalca. Na ta način se odjemalci prijavijo v skupino, obenem pa tudi sporočajo, da želijo v skupini ostati. Če želijo skupino zapustiti, preprosto prenehajo pošiljati sporočila IGMP Report. To je tudi edini način odjave od skupine v primeru IGMPv1, saj po poteku določenega časa povpraševalec sam izbriše neaktivne odjemalce. V tem primeru govorimo o pasivni odjavi iz skupine. IGMPv2 pa predvideva še eksplicitno, oziroma aktivno odjavo iz skupine, pri čemer odjemalec pošlje sporočilo IGMP Leave. Komunikacijo med povpraševalcem in odjemalcem prikazuje slika 2.6.

Naj omenim še eno podrobnost, ki loči doslej obravnavani verziji protokola, in sicer v primeru IGMPv2 obstajata dve obliki sporočila IGMP Query. Povpraševalec lahko preverja stanje vseh skupin na določenem segmentu IP ali pa preverja stanje specifične skupine. Tako lahko povprašuje splošno (angl. *General Membership Query*) ali specifično za skupino (angl. *Group-Specific Membership Query*), vendar kot bomo videli v nadaljevanju, to za nas ni posebej pomembno. Vsi trije tipi sporočil imajo enotno obliko, ki jo



Slika 2.6 Komunikacija med povpraševalcem in odjemalcem v primerih IGMPv1 in IGMPv2.

vidimo na sliki 2.7.



Slika 2.7 Splošna oblika sporočil IGMP za verziji 1 in 2.

Polja v sporočilu so:

- Type - določa tip sporočila;
- Max Response Time - velja le za povpraševanja, privzeta nastavitev je 10 sekund, pomeni pa čas, v katerem morajo odjemalci odgovoriti na sporočilo;
- Checksum - namenjen je zagotavljanju integritete sporočil;
- Group Address - to polje nosi informacijo o naslovu skupine multicast. Če gre za sporočilo tipa IGMP Report ali IGMP Leave, je to skupina, v katero se odjemalec prijavlja, oziroma iz katere se odjavlja. Če pa gre za povpraševanje, je to polje v primeru splošnega povpraševanja nastavljeno na nič, v primeru povpraševanja, specifičnega za skupino pa je tu naslov skupine, za katero povpraševanje velja.

2.3.2 Protokol IGMP, verzija 3

Tretja različica protokola IGMP [5] uvaja nekaj bistvenih novosti. V osnovi lahko pri klasičnem prometu tipa multicast govorimo o modelu „multicast iz katerega koli vira“ (označuje ga kratica ASM), kar pomeni, da lahko katera koli naprava v omrežju oddaja promet v katero koli skupino. Takšen način je primeren za aplikacije tipa „iz več točk v več točk“ (angl. *multipoint-to-multipoint*), kjer za isto multicast skupino obstaja več

izvorov. Nasprotje temu konceptu predstavlja „iz ene točke v več točk“(angl. *point-to-multipoint*), kar je zahteva ravno pri najpogostejši uporabi multicasta danes (internetna radio in televizija). Zagotavljanju takšnega tipa multicasta, ki mu pravimo tudi multicast iz določenega vira (označuje ga kratica SSM), je namenjen IGMPv3.

V primeru SSM morajo odjemalci ob prijavi v skupino navesti tudi unicast naslov strežnika, od katerega želijo prejemati podatkovni tok. Tako lahko govorimo o kanalih, ki imajo en izvor in več ponorov podatkov, zato lahko rečemo, da je ključna novost tretje verzije možnost neposredne odjemalčeve izbire izvora prometa.

Odjemalci imajo dve možnosti vključevanja v skupine: vključitveni in izključitveni način. Prvi način omogoča, da odjemalec eksplicitno določi strežnike, od katerih želi prejemati podatkovni tok, medtem ko drugi način omogoča, da odjemalec določi oddajnike, od katerih ne želi sprejemati prometa. Podobno kakor v primeru prvih dveh verzij tudi tu standard določa dva tipa sporočil:

- *Membership Query* - sporočila, ki jih pošiljajo usmerjevalniki,
- *Membership Report* - sporočila, ki jih pošiljajo odjemalci.

IGMP tretje verzije v bistvu razširja sporočila, ki jih že najdemo v prvih dveh verzijah, saj je namesto podajanja skupine sedaj potrebno podati par: *izvor, skupina*. Zato je v sporočilih, kjer se je v verzijah 1 in 2 navajalo samo skupino, sedaj potrebno dodati še spisek enega ali več izvorov, od katerega želimo, oziroma nočemo prejemati podatkovnega toka.

Sporočila tretje verzije protokola IGMP se tako močno razlikujejo od sporočil prejšnjih dveh različic. Na kratko lahko rečemo, da so vsa sporočila tretje verzije opisana z naslednjo peterico: *povpraševalec, vmesnik, naslov multicast, filtrski način, spisek naslovov IP*. Elementi te peterice imajo naslednji pomen:

- Povpraševalec - parameter, ki ga uporabljamo za razlikovanje med različnimi povpraševalci;
- Vmesnik - identifikator vmesnika, na katerem bomo sprejemali multicast promet;
- Naslov multicast - naslov, oziroma skupina, na katero se zahteva nanaša, za različne naslove torej pošljemo različne zahteve;
- Filtrski način - vključitveni ali izključitveni način;

- Spisek naslovov - naslovi unicast, s katerih želimo v vključitvenem filtrskem načinu sprejemati podatkovni tok, oziroma naslovi, od katerih v izključitvenem načinu ne želimo sprejemati podatkovnega toka.

Zaradi različnih filtrskih načinov je ekvivalent IGMP Report sporočila IGMPv2 za določen naslov multicast naslednja peterica podatkov: {*povpraševalec, vmesnik, naslov, EXCLUDE, ()*}. Kombinacija izključevalnega načina filtriranja in praznega spiska naslovov namreč pomeni, da odjemalec ne želi izključiti nikogar. To je torej vsebinsko enako zahtevi po priključitvi skupini (oziroma naslovu multicast), kakršne lahko pošiljamo v okviru IGMPv2. Podobno lahko opazimo, da je ekvivalent sporočilu IGMP Leave, ki ga poznamo iz druge verzije IGMP, naslednje sporočilo Membership Report tretje verzije: {*povpraševalec, vmesnik, naslov multicast, INCLUDE, ()*}. Odjemalec s tem sporočilom sporoča, da ne želi prejemati prometa od nobenega vira. Lastnosti vseh treh verzij protokola IGMP so strnjeno podane v tabeli 2.8.

2.4 Protokol MLD

Pri omrežjih, ki na tretjem sloju uporabljajo protokol IPv6, se za odkrivanje poslušalcev pri oddajanju več prejemnikom ne uporablja IGMP, temveč MLD (angl. *Multicast Listener Discovery*). Obstajata dve verziji tega protokola: MLDv1 omogoča funkcije, ki so ekvivalentne IGMPv2, druga različica MLDv2 pa je ekvivalentna IGMPv3. Princip delovanja je sicer identičen protokolu IGMP, opisuje pa ga dokument RFC 3810 [6].

2.5 Vohljanje IGMP

Vohljanje IGMP sem že predstavil kot eno izmed rešitev za zmanjšanje odvečnega prometa, ki nastaja zaradi lastnosti običajnih stikal, da ne ločujejo med prometom vrste broadcast in multicast.

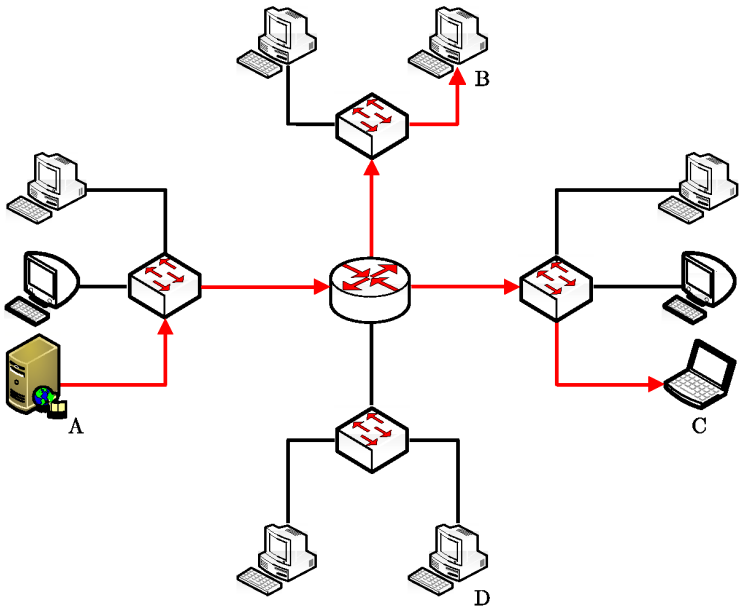
V osnovi ta funkcionalnost predstavlja prenos funkcionalnosti IGMP z izključno tretjega še na drugi sloj, torej v praksi večinoma na sloj delovanja Ethernet stikal, to pa za seboj prinaša nekaj posledic. Vohljanje lahko označuje tako pasivno kot aktivno pregledovanje okvirjev. V prvem primeru okvirje, ki se znajdejo na stikalu, v resnici samo pregledujemo, ne izvajamo pa filtriranja ali kakor koli drugače vplivamo na potek podatkovnega toka na stikalu. Nas v tem delu bolj zanima aktivno vohljanje, kjer aktivno posegamo v delo stikala, in sicer tako, da že na drugi sloj prenesemo funkcionalnost, ki

Protokol	Lastnosti
IGMPv1	- omogoča le prijavo v skupino, posebne zahteve za odjavo ne pozna, - omogoča le preverjanje stanja vseh skupin, ki se nahajajo na določenem segmentu LAN - splošno povpraševanje, - odjavljanje odjemalcev temelji na konceptu periodičnega preverjanja stanja odjemalcev s strani izvora podatkov, - če usmerjevalnik ne dobi odgovora, po preteku časovne kontrole (tipično okoli tri minute) preneha pošiljati multicast podatkovni tok.
IGMPv2	- omogoča prijavo v skupino, - odjemalcem omogoča aktivno odjavo iz skupine: - krajši čas za odjavo iz skupine, - manj neželjenega prometa. - omogoča eksplicitno preverjanje naročnikov na specifično skupino.
IGMPv3	- možnost neposredne izbire izvora multicasta znotraj izbrane skupine, - vključitveni način - odjemalec eksplicitno določi oddajnike, od katerih bo sprejemal podatkovni tok vrste multicast, - izključitveni način - odjemalec določi oddajnike, od katerih ne bo sprejemal podatkovnega toka vrste multicast.

Slika 2.8 Lastnosti različnih verzij protokola IGMP

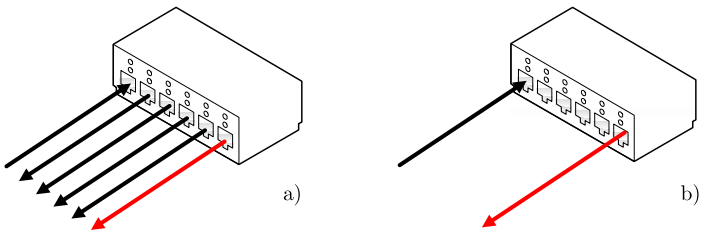
jo sicer zagotavlja protokol IGMP šele v tretjem sloju.

Priporočila v zvezi z vohljanjem IGMP so opisana v [7]. Namen je zmanjševanje obremenitve omrežja v vseh smereh: tako v smeri pretoka podatkov od odjemalcev k usmerjevalniku in obratno, kot tudi zmanjševanje prometa znotraj lokalnega omrežja. Podatkovni tok znotraj lokalnega omrežja v primeru uporabe vohljanja IGMP prikazuje slika 2.9. Kot v prejšnjem primeru na sliki 2.4, je računalnik A v tem primeru ponudnik vsebine, ki se prenaša v obliki multicasta, računalnika B in C pa sta naročnika. Zaradi uporabe vohljanja IGMP sedaj vidimo, da promet prejemajo samo odjemalci, ki ta promet želijo prejemati. Oglejmo si še dogajanje ob multicastu na stikalih. Prikazana sta primera, ko je vohljanje IGMP podprto in ko ni. Slika 2.10a prikazuje stikalo, ki funkcionalnosti vohljanja IGMP ne pozna. V tem primeru je samo eno izmed štirih vrat dejanski „naročnik“ prometa (označeno z rdečo barvo), vseeno pa so tudi ostala vrata



Slika 2.9 Promet v primeru uporabe vohljanja IGMP.

prejemniki prometa. Posledično naprave, ki so priključene na ostala vrata, ne morejo izvajati nobene druge komunikacije na teh povezavah, saj so le-te že zasedene. Kot prikazuje slika 2.10b, pa je v primeru vključene funkcionalnosti vohljanja IGMP sedaj prejemnik prometa le ena naprava, s tem pa je obremenjenost omrežja in na omrežje priključenih naprav močno zmanjšana.



Slika 2.10 Preklapljanje prometa stikala v primeru izključene (a) in vključene (b) funkcionalnosti vohljanja IGMP.

2.5.1 Delovanje stikala v primeru vohljanja IGMP

Stikalo mora v primeru funkcionalnosti vohljanja IGMP neprestano pregledovati prihajajoči promet. Promet, ki pride v stikalo, slednjemu služi za beleženje dejstev o tem, na katerih vratih se dejansko nahajajo odjemalci, ki želijo prejemati določen multicast promet in na katerih ne. Na ta način lahko stikalo tudi v resnici omeji distribuiranje

prometa. V praksi beleženje pomeni, da stikalo vzdržuje podatkovno strukturo, kjer so v grobem shranjeni elementi, ki posamezna vrata povezujejo s skupino ali skupinami. To podatkovno strukuro bom v nadaljevanju imenoval *tabela multicast*. Seveda so poleg teh dveh podatkov v praksi potrebni še nekateri drugi, denimo podatek o tem, kdaj je na določena vrata nazadnje prišlo sporočilo, da nek odjemalec želi ostati v določeni skupini (torej podatek o tem, kdaj je stikalo na določenih vratih nazadnje zaznalo sporočilo IGMP Report za določeno skupino). Ker standarda, ki bi točno določal podatkovne strukture in način vohljanja ni, so te podrobnsti prepuščene implementaciji.

Poleg ugotavljanja, katera vrata želijo ostati povezana z določeno skupino, je za podporo drugi različici protokola in tudi za zagotavljanje večje učinkovitosti implementacije, potrebno okvirje pregledovati za sporočili, ki nakazujejo odjavo od skupine. Ker je na določena vrata stikala lahko posredno povezanih več odjemalcev, mora stikalo pregledovati tudi izvorne naslove okvirjev, saj le tako lahko ugotovi kdaj lahko določena vrata v resnici zapre za določeno skupino multicast. Če tega ne bi bilo, bi v primeru, ko je na določena vrata povezano stikalo, prek katerega se v omrežje povezuje več končnih odjemalcev prvo stikalo ta vrata za promet zaprlo že, ko bi se samo en izmed končnih odjemalcev odjavil od skupine.

Teoretično bi bilo v primeru protokolov različice 1 in 2 mogoče vohljanje IGMP izvajati na dveh različnih nivojih: na nivoju končnega naslova IP ali na nivoju končnega naslova MAC. To bi bilo možno zaradi preslikovanja naslovov IP v naslove MAC, kakor to opisuje že poglavje 2.2.1. Vendar pri vohljanju na nivoju naslovov MAC hitro naletimo na težavo, ki sem jo že omenil že, in sicer, da preslikava ni bijektivna, zato se v en naslov MAC preslika več različnih naslovov IP. Vseeno ta težava ni posebej velika, saj sicer dobimo nekaj odvečnega prometa, oziroma se promet pojavlja tudi na vratih, ki niso nujno prijavljena v skupino, a je prometa še vedno veliko manj, kot če stikalo sploh ne omejuje prometa. Večji izziv predstavlja IGMP verzije 3.

IGMP verzije 3 deluje tako, da odjemalci sporočila IGMP Report, s katerimi se prijavljajo v določeno skupino, v skladu s standardom [5] vedno pošiljajo na ciljni naslov 224.0.0.22. Zaradi tega je tudi ciljni naslov MAC vseh sporočil z vseh odjemalcev vedno enak, in sicer 01:00:5e:00:00:16 - zadnji trije okteti torej predstavljajo tudi zadnje tri oktete naslova IP, število 16 pa je šestnajstiška predstavitev števila 22. Dejanski ciljni naslov ene ali več skupin, na katero oziroma katere se odjemalec prijavlja je zapisan šele v podatkih, ki jih vsebuje paket IP.

Zaradi naštetih razlogov mora torej stikalo okvirje pregledati na mestu, kjer so zapisani ciljni naslovi IP in si ob vsakem ustreznem okvirju ustrezno posodobiti svojo tabelo multicast.

Opazimo lahko, da proces pregledovanja okvirjev ločuje dve situaciji:

1. pregledovanje okvirjev, ki nosijo sporočila IGMP in
2. pregledovanje okvirjev, ki nosijo vsebine, ki se prenašajo prek multicasta.

Glede slednjih naj omenim, da tudi ti okvirji, enako kot okvirji s sporočili IGMP, uporabljajo mapiranje med naslovi IP in MAC. Zato je pri „vsebinskem“ paketu možno in dovoljeno vohljanje izvajati po ciljnih naslovih MAC. Preverjanje teh naslovov je hitrejše, saj zadostuje preverba ciljnega naslova MAC, ki je zapisan v sami glavi okvirja, ne pa tudi ciljnega naslova IP, ki je skrit v notranjosti okvirja. Posledično bodo seveda vrata, ki bi morala prejemati samo promet s ciljnim naslovom, denimo 224.1.2.3, prejemala tudi promet s ciljnim naslovom 239.129.2.3, vendar je osnovni namen funkcionalnosti vohljanja IGMP še vedno v pretežni meri dosežen.

Poleg doslej opisanega mora stikalo poskrbeti tudi za ustrezen časovnik, ki iz tabele periodično briše vnose, ki jim je v skladu s protokolom IGMPv1 že potekel čas. Uporaba časovnika je seveda nujna tudi v okoljih, kjer vsi odjemalci uporabljajo protokol verzij 2 oziroma 3, saj ta protokola zahtevata kompatibilnost s prejšnjimi verzijami, poleg tega pa lahko odjemalci iz omrežja izpadejo tudi brez predhodne odjave iz skupin, v katere so bili prijavljeni.

Christensen, et al. [7] opisujejo številne zahteve, ki morajo biti izpolnjene pri implementaciji funkcionalnosti vohljanja IGMP, nekatere pa samo priporočajo. Pred izdajo omenjenega dokumenta so različni proizvajalci stikal funkcionalnost vohljanja IGMP na svojih stikalih že podpirali, in sicer vsak na svoj način, zato je na trgu precej naprav, ki temu dokumentu sledijo le deloma. Še posebej veliko težav, oziroma odstopanj od priporočil, je pri protokolu IGMPv3, saj ta v primerjavi s predhodnjima različicama uvaja nekaj konceptualnih novosti, denimo izključevalni in vključevalni način delovanja ter možnost prijavljanja in odjavljanja v in iz množice skupin.

Vohljanje IGMP na delovanje stikal vpliva še v nekaj točkah:

- Stikalo mora razločevati med navadnimi in usmerjevalniškimi vrati. Slednji označujejo vrata, na katere so priključeni usmerjevalniki, ki podpirajo multicast. To je potrebno iz dveh razlogov. Prvi je ta, da je sporočila IGMP Report verzij 1 in 2 dovoljeno poslati samo na usmerjevalniška vrata. IGMP omenjenih verzij namreč določa, da odjemalci, ki prejmejo tovrstni paket, prenehajo s pošiljanjem svojih sporočil IGMP Report. To sledi iz obnašanja standardnih stikal, ki ves promet vrste multicast tako ali tako vedno pošiljajo vsem napravam, priključenim na stikalo. Zato lahko odjemalec, ki prejme takšno sporočilo, upravičeno sklepa, da na tem segmentu že obstaja nek drug odjemalec, ki prav tako želi prejemati enak podatkovni tok. Posledično takšen odjemalec preneha s pošiljanjem sporočil IGMP report. V primeru vohljanja IGMP pa pride do tega, da odjemalci, ki ne pošiljajo tovrstnega sporočila, ne bodo prejemali prometa. Drugi razlog je v prometu, ki ima ciljni naslov drugačen od 224.0.0.X, saj gre za promet, ki ni namenjen lokalnemu omrežju. Pakete s tem ciljnim naslovom naj bi stikalo poslalo tako na vrata, registririrana za to skupino, kot tudi na usmerjevalniška vrata [7]. Razlikovanje med navadnimi in usmerjevalniškimi vrati je mogoče doseči na tri načine:

  1. Spisek usmerjevalniških vrat lahko stikalo gradi s pomočjo sporočil *Multicast Router Solicitation*, kot je to opisano v IGMP Multicast Router Discovery [8]. Prav tako lahko stikalo s pomočjo pregledovanja okvirjev odkriva sporočila *Multicast Router Advertisement*, ki jih pošiljajo usmerjevalniki in so namenjena ostalim vozliščem omrežja.

  2. Stikalo lahko usmerjevalniška vrata odkrije tako, da posluša za sporočili IGMP Query, ki jih pošiljajo usmerjevalniki multicast, kjer izvorni naslov ni 0.0.0.0. Ta naslov namreč nakazuje, da sporočilo IGMP Query ni prišlo z usmerjevalnika multicast. V omrežju lahko obstajajo omrežni elementi, ki delujejo kot „proxy“, ki prestrezajo različna sporočila IGMP Query, naprej pa posredujejo samo eno takšno sporočilo. Ti omrežni elementi kot izvorni naslov navedejo 0.0.0.0.

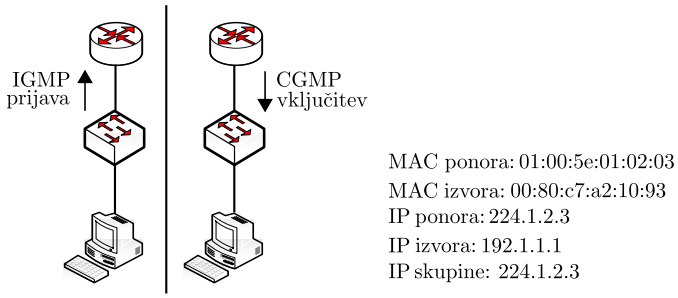
  3. Administrator s pomočjo posebnega programa določi, katera vrata so usmerjevalniška in katera ne.

- stikalo ne sme zavračati sporočil IGMP Report, ki imajo izvorni naslov enak 0.0.0.0. Razlog, podobno kot prej, tiči v tako imenovanem „proxy-reportingu“, kar pomeni, da morda v omrežju obstajajo elementi, ki prestrezajo sporočila IGMP, ki jih pošiljajo odjemalci. V primeru, ko ti elementi ugotovijo, da se več odjemalcev želi priključiti k isti skupini, pošljejo samo eno sporočilo IGMP Report.

2.6 Protokol CGMP

Ustrezen dokument z navodili in priporočili [7], ki opisuje delovanje vohljanja IGMP, je bil objavljen relativno pozno, in sicer leta 2006. Nekateri proizvajalci so omenjeno funkcionalnost poskušali implementirati še pred nastankom dokumenta.

Poleg doslej obravnavanega pregledovanja okvirjev je podjetje Cisco Systems Inc., v želji po omejevanju (nepotrebnega) prometa, pokazalo še en pristop, ki se od vohljanja IGMP nekoliko razlikuje, saj uvaja sodelovanje med usmerjevalniki in stikali. Razvili so lasten protokol, imenovan CGMP [9], ki opravlja enako nalogo kot vohljanje IGMP, njegova slaba lastnost pa je, da morata protokol podpirati tako stikalo kot usmerjevalnik. Protokol deluje v povezavi s protokolom IGMP, namenjen pa je omejevanju razpošiljanja



Slika 2.11 Delovanje protokola CGMP.

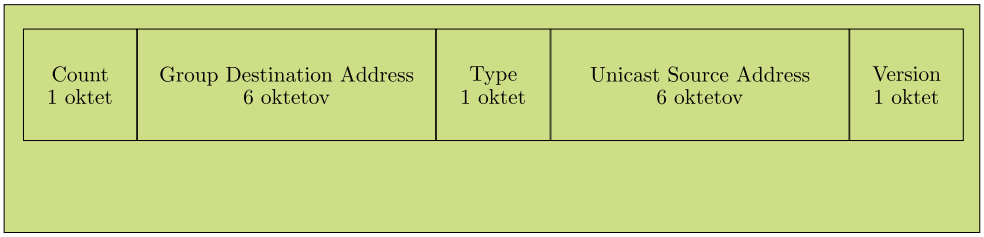
IP multicast paketov samo na tista vrata stikal, ki so povezana z odjemalci za multicast promet. Ti odjemalci se na multicast promet prijavljajo in odjavljajo z že opisanimi sporočili, ki so del protokola IGMP. Usmerjevalniki, ki prepoznajo ustrezen paket IGMP, pošljejo novo sporočilo vrste CGMP, ki ga prepoznajo stikala in ta v skladu s tem v nadaljevanju posredujejo promet. CGMP predvideva do 64 različnih skupin multicast. Podobno kot vohljanje IGMP tudi CGMP ne zahteva spreminjanja strojne ali programske opreme končnih odjemalcev.

Usmerjevalniki, ki podpirajo in imajo omogočen protokol CGMP ter so zaznali paket

IGMP, s katerim se je nek odjemalec želel priključiti k multicast skupini, periodično oddajajo ustrezne CGMP pakete. Takšno delovanje je v skladu z IGMP verzije 1, ko pa se je pojavila verzija 2, je Cisco nadgradil tudi svoj protokol, ki sedaj omogoča tudi odjavo. Cisco tej funkcionalnosti pravi *hitri odhod* (angl. *Fast-Leave*), saj za odjavo odjemalca iz skupine ni potrebno čakati, da preteče vnaprej definiran čas. Tudi v primeru protokola CGMP mora stikalo podatke o prijavah odjemalcev v določeno skupino in vratih, na katera so ti odjemalci povezani, hraniti v posebni tabeli (tabela CAM). Funkcionalnosti protokola IGMP verzije 3 Ciscov CGMP zaenkrat ne podpira.

Strukturo paketov CGMP opisuje slika 2.12, pri čemer posamezni elementi pomenijo naslednje:

- Count: nepredznačeno 8 bitno celo število,
- Group Destination Address: naslov MAC ciljne naprave,
- Type: tip sporočila,
- Unicast Source Address: naslov MAC izvorne naprave (ki je tipa unicast),
- Version: verzija CGMP.



Slika 2.12 Struktura paketa CGMP.

Protokol CGMP breme pregledovanja paketov multicast jemlje s stikal in jih prenaša na usmerjevalnike, ki pa morajo prav tako pregledovati celoten multicast promet. Stikala morajo pregledovati le kontrolna sporočila CGMP. Slaba stran takšnega pristopa je prav gotovo dejstvo, da morajo tako usmerjevalniki kot stikala poznati dodaten protokol.

2.7 Nadgradnja obstoječih stikal

Pri obstoječem omrežju, kjer se pojavlja tudi multicast promet, je zamenjava obstoječe opreme precej drag poseg. Večina stikal, ki so na trgu in predstavljajo nižji do srednji

razred ponudbe, ne podpira nobene izmed opisanih tehnik omejevanja prometa. Kot smo videli, stikala, ki bi uporabljala protokol CGMP prav tako ne morejo delovati brez ustrezne opreme na sloju usmerjevalnikov. Ugodno alternativo menjavi strojne opreme predstavljajo sistemi, kjer je z nadgradnjo programske opreme mogoče dodajati nove funkcionalnosti. Zadnje čase se vedno bolj uveljavljajo sistemi, ki temeljijo na operacijskem sistemu Linux. Večinoma gre za posebej izdelane sisteme, ki so namenjeni preklapljanju ali usmerjanju prometa ali kombinaciji obojega. Na trgu so v obliki majhnih in cenenih hišnih usmerjevalnikov, ki obenem običajno nudijo tudi dostopno točko za brezžično dostopanje do omrežja. Dobra stran teh usmerjevalnikov je njihova možnost menjave programske opreme. Prav tako pa lahko funkcijo stikala prevzame kar klasični namizni računalnik ali drug sistem, na katerem teče ustrezna programska oprema. Pri tem se zaradi možnosti dostopanja do izvorne kode programske opreme in možnosti spreminjanja le-te kot dobra izbira ponuja ravno operacijski sistemu Linux.

3 Izvedba vohljanja IGMP na stikalu Ethernet v operacijskem sistemu Linux

Pri praktični izvedbi funkcionalnosti vohljanja IGMP sem se osredotočil na operacijski sistem Linux. Ta se na področju sistemov, ki tako ali drugače sodelujejo v računalniških omrežjih, široko uporablja. Lahko predstavlja platformo, na kateri temeljijo praktično vsi omrežni elementi: stikala, usmerjevalniki, strežniki, končni odjemalci itd. Glavne prednosti so: preizkušenost, robustnost, nadgradljivost in zanesljivost. Pomembno prednost v primeru, ki ga obravnavam v tem delu, predstavlja tudi dejstvo, da gre za tako imenovani odprtokodni sistem, kjer je mogoč prost, neomejen in brezplačen dostop do izvorne kode praktično vseh delov sistema. To dejstvo omogoča popravljanje, spreminjanje in nadgradnjo obstoječih funkcionalnosti.

V času pisanja diplomskega dela uradna verzija jedra operacijskega sistema, kjer se nahaja in izvaja tudi večina funkcionalnosti v zvezi z omrežnimi dejavnostmi, nima podpore za vohljanje IGMP, podpira pa funkcionalnosti običajnih stikal.

Razvoj operacijskega sistema Linux je zelo hiter in sproti se pojavljajo nove različice tako samega jedra sistema, kot tudi podpornih programov, ki delujejo v uporabniškem naslovnem prostoru. Zato sem se odločil za eno izmed zadnjih verzij, in sicer verzijo

2.6.31. Med razvojem moje rešitve za podporo multicast in pisanjem diplomskega dela so bile sicer že objavljene tudi novejše različice, ki pa se od izbrane v delu, ki pokriva preklapljanje prometa na povezovalnem sloju, ne razlikujejo bistveno in še vedno ne podpirajo funkcionalnosti vohljanja IGMP.

3.1 Računalniški sistem z operacijskim sistemom Linux kot Ethernet stikalo

V okviru operacijskega sistema Linux je funkcionalnost stikala implementirana v modulu `bridge` [10]. Kljub imenu, ki v prevodu pomeni „most“, kar se največkrat uporablja kot sinonim za preprostejšo verzijo stikala s samo dvema vhodoma, ta modul nudi polno funkcionalnost stikala z več vhodi.

Modul `bridge` v Linuxu ustvari *navidezni omrežni vmesnik* (angl. *virtual network interface*), na katerega je mogoče povezati poljubno število drugih (navideznih) omrežnih vmesnikov, ki jih most vidi kot vhode oziroma vrata. Omrežni vmesniki so logične entitete, ki sprejemajo in oddajajo okvirje Ethernet. Lahko predstavljajo fizične omrežne naprave, ki se nahajajo v sistemu, kakršne so na primer omrežne kartice, lahko pa navidezne omrežne vmesnike, ki ne predstavljajo nobene fizične naprave. Primer omrežnih vmesnikov, ki predstavljajo fizične naprave so vmesniki z običajnimi imeni `eth0, eth1, eth2...`, ki predstavljajo omrežne vmesnike (denimo omrežne kartice) v sistemu. Primer navideznega omrežnega vmesnika, ki ne predstavlja nobene fizične naprave, pa je kar most sam. To pomeni, da lahko povezujemo tudi mostove med seboj, kar pomeni, da lahko nek most predstavlja enega izmed vhodov drugega mostu.

Most v Linuxu deluje podobno kot strojne izvedenke stikal - vsak vstopni okvir most, najprej pregleda in v primeru napake zavrže. Nato ugotovi, ali je okvir namenjen lokalnemu cilju, kar pomeni, da ga modul odda v obdelavo višjim slojem omrežnega sklada ali pa ga preklopi na ena oziroma več vrat. Za preklapljanje most uporablja tabelo naslovov MAC, ki sem jo opisal v uvodu v 2. poglavje. Na osnovi ciljnega naslova okvirjev most v tabeli najde ustrezna ciljna vrata in na takšen način usmerja promet. Poseben primer predstavljajo okvirji, ki imajo najmanj pomemben bit najpomembnejšega okteta ciljnega naslova MAC vrednost 1; gre torej za okvirje, namenjene več uporabnikom. V obravnavani različici operacijski sistem Linux deluje kot standardno stikalo in promet vrste multicast razpošlje na vsa razpoložljiva vrata.

Linuxov most v splošnem deluje popolnoma transparentno - ko so omrežni vmesniki enkrat povezani, most med njimi na drugem sloju usmerja promet kakor da bi bili ti vmesniki v omrežnem segmentu. Izjemo od tega načela predstavlja možnost, da tudi mostu samemu dodelimo naslov IP - na ta način se lahko na most povežemo tudi na daljavo in na njem izvajamo upravljanje. Dostop do dodeljenega naslova IP je mogoč prek katerega koli omrežnega vmesnika, ki je povezan v most, dostopati je možno prek vseh standardnih storitev, kot so na primer telnet, ssh, rsh, SNMP in tako naprej. V osnovi je Linuxov most deloval zgolj kot programska verzija stikala, kasneje pa so dodali tudi možnost dodajanja funkcionalnosti požarnega zidu, most pa vsebuje tudi podporo protokolu STP [10].

3.1.1 Most in podpora VLAN

Zanimivo vprašanje predstavlja podpora tako imenovanim *navideznim lokalnim omrežjem* ali VLAN. Podpora VLAN je v Linuxu izvedena ločeno od izvedbe mostu, saj je ta realizirana v obliki novega omrežnega vmesnika. Omrežni vmesnik s podporo za VLAN se torej tako kot vsi ostali omrežni vmesniki „pripne“ mostu. V primeru operacijskega sistema Linux je naloga tega vmesnika preprosta - gre za odznačevanje vhodnega in označevanje izhodnega prometa. Takšno delovanje je tudi neodvisno od drugih omrežnih vmesnikov. Most sam se za morebitne oznake za VLAN, ki jih okvir vsebuje, preprosto ne meni. Pravilna uporaba in izkoriščanje funkcionalnosti VLAN je tako od mostu ločena in zato ni stvar tega diplomskega dela.

3.1.2 Upravljanje standardnega mostu v okolju Linux

Za upravljanje mostu se v operacijskem sistemu Linux uporablja program `brctl`. Pomnilniški prostor v računalniškem sistemu, na katerem teče operacijski sistem, je med drugim razdeljen na dva dela, kar predstavlja tudi razdelitev vse v nadaljevanju opisane, programske opreme. Tako govorimo o:

- *uporabniškem* naslovnem prostoru in
- *jedrnem* naslovnem prostoru.

Večina osnovne funkcionalnosti mostu se izvaja v jedrnem naslovnem prostoru, kar pomeni, da je tudi za vse spremembe in dopolnitve te funkcionalnosti potrebno spreminjati in dopolnjevati jedro operacijskega sistema. To pa ne velja za upravljalni del.

Ta se izvaja v uporabniškem prostoru in deluje tako, da ukaze uporabnika iz ukazne vrstice prek posebnih sistemskih klicev posreduje jedrnemu delu. Program `brctl` počne natanko to. Sintakso in spisek ukazov, ki jih `brctl` sprejme, dobimo s klicem programa brez parametrov:

```
brtcl commands:
  addbr         <br>                add bridge
  addif         <br> <device>       add interface to bridge
  delbr         <br>                delete bridge
  delif         <br> <device>       delete interface from bridge
  show                              show a list of bridges
  showbr        <br>                show bridge info
  showmacs      <br>                show a list of mac addrs
  setageing     <br> <time>         set ageing time
  setbridgeprio <br> <prio>         set bridge priority
  setfd         <br> <time>         set bridge forward delay
  setgcint      <br> <time>         set garbage collection int.
  sethello      <br> <time>         set hello time
  setmaxage     <br> <time>         set max message age
  setpathcost   <br> <port> <cost>  set path cost
  setportprio   <br> <port> <prio>  set port priority
  stp           <br> <state>        {dis,en}able stp
```

Za nas ta hip najpomembnejša ukaza sta preprosto `addbr` in `addif`, s katerima naredimo nov most in mostu dodamo omrežni vmesnik. Primer:

```
brctl addbr br0
brctl addif br0 eth0
brctl addif br0 eth1
```

Poleg programa `brctl` se je za delo z mostom potrebno seznaniti še s programom `ifconfig`. To je orodje, s katerim lahko dodajamo, upravljamo in brišemo omrežne vmesnike.

Program je mogoče uporabljati v treh oblikah:

```
ifconfig
ifconfig <interface>
ifconfig <interface> [aftype] <options> | <address> ...
```

Prva oblika, torej klic programa brez imena omrežnega vmesnika izpiše vse omrežne vmesnike in njihov status. Druga oblika izpiše samo status konkretno navedene naprave. S tretjo obliko spreminjamo nastavitve za posamezni omrežni vmesnik. Nastavljamo lahko naslove IP, MAC, omrežne maske in podobno. Podrobna predstavitev vseh ukazov presega obseh tega dela, predstavil bom le dodelitev naslova IP, prek katerega je kasneje mogoče dostopati do mostu. To naredimo z ukazom:

```
ifconfig br0 naslov_IP
```

Za nas sta na tej točki pomembna samo še ukaza `ifconfig <interface> up` in `ifconfig <interface> down`, s katerima aktiviramo in deaktiviramo omrežni vmesnik.
Z zaporedjem teh ukazov dobimo nov primerek mostu `br0` z naslovom `naslov_IP`. Na most smo pripeli dva omrežna vmesnika, ki vsak zase predstavlja svojo omrežno kartico (kartici morata biti seveda prisotni v računalniku). Most `br0` je potrebno še aktivirati z `ifconfig br0 up`. Po tem dejanju z že omenjenim orodjem `ifconfig` lahko vidimo nov omrežni vmesnik `br0`:

```
br0     Link encap:Ethernet  HWaddr b2:a3:87:8d:ec:68
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:0 errors:0 dropped:0 overruns:0 frame:0
        TX packets:5 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:0
        RX bytes:0 (0.0 B)  TX bytes:398 (398.0 B)

eth0    Link encap:Ethernet  HWaddr b2:a3:87:8d:ec:68
        inet6 addr: feb2::aa3:87ff:fe8d:8dec/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:1574 errors:0 dropped:0 overruns:0 frame:0
        TX packets:1173 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:132061 (132.0 KB)  TX bytes:1560666 (1.5 MB)
```

```
eth1    Link encap:Ethernet  HWaddr 08:00:27:a7:6e:21
        inet6 addr: fe80::a00:27ff:fea7:6e21/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:1173 errors:0 dropped:0 overruns:0 frame:0
        TX packets:1574 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:1560666 (1.5 MB)  TX bytes:132061 (132.0 KB)
```

To je okleščen izpis, ki ga vidimo po zgoraj opisanem postopku vzpostavljnja mostu `br0` z dvema omrežnima vmesnikoma `eth0` in `eth1`.

Komunikacija programa `brctl` z jedrom operacijskega sistema, kjer se, kot rečeno, nahaja dejanska implementacija preklapljanja, poteka deloma prek klicev `ioctl`, deloma pa prek navideznega datotečnega sistema `sysfs`, ki se ga nadzoruje prek psevdo-datotek v imeniku `/sys`. V splošnem je oblika klica `ioctl` naslednja:

```
int ioctl(int d, int request, ...);
```

Pri čemer je `int d` enolični datotečni deskriptor, ki ga dobimo, ko odpremo povezavo z omrežnim vmesnikom, naslednji argument predstavlja zahtevo in tudi določa zadnji argument, ki je lahko različnih oblik, odvisno od argumenta `request`.

Izsek iz izvorne kode programa `brctl`, ki vzpostavi nov most (ukaz `brctl addbr <ime_mostu>`):

```
ret = ioctl(fd, SIOCBRADDBR, ime_mostu);
```

Argument `fd` je tu datotečni deskriptor, ki ga dobimo, ko odpremo povezavo z omrežnim vmesnikom `br0`, naslednji argument `SIOCBRADDBR` predstavlja zahtevo po novi instanci mostu. Ime te nove instance je določeno z zadnjim argumentom.

Način konfiguracije prek navideznega datotečnega sistema `sysfs` predstavlja novejši pristop h konfiguriranju nekaterih naprav. Pri tem pristopu se v imeniku `/sys` ob nalaganju gonilnikov pojavi množica psevdo-datotek, ki vsebujejo podatke o lastnostih naprav. Gonilnik `bridge`, ki nadzoruje delovanje mostu tako ustvari imenik `/sys/devices/virtual/net/br0/`. V tem imeniku se nahajajo naslednje psevdo-datoteke:

```
address    carrier  features  ifindex    mtu        type
addr_len   dev_id   flags     iflink     operstate  uevent
brforward  broadcast  dormant  ifalias   link_mode  tx_queue_len
```

Kot primer si oglejmo vsebino ene izmed njih, na primer datoteka `address` vsebuje naslov MAC našega mostu: `b2:a3:87:8d:ec:68`. Še več psevdo-datotek s podatki o mostu najdemo v podimeniku `/sys/devices/virtual/net/br0/bridge`:

```
ageing_time  forward_delay  hello_time   priority    root_port
topology_change  bridge_id  gc_timer     hello_timer root_id
stp_state    topology_change_detected    flush       group_addr
max_age      root_path_cost tcn_timer    topology_change_timer
```

Ta podimenik je tudi pravo mesto za konfiguracijo dopolnitev mostu s funkcionalnostjo vohljanja IGMP. Poleg tega podimenika so zanimivi tudi podimeniki `/sys/devices/virtual/net/br0/ethX`, pri čemer X pomeni zaporedno številko vmesnika, priključnega k mostu. V teh imenikih najdemo psevdo-datoteke s parametri, pomembnimi za posamezen vmesnik.

Tako uporaba `ioct` kot navideznega datotečnega sistema `sysfs` predstavljata samo način komunikacije uporabniške programske opreme z jedrom; uporabniški program mora poleg tega parametre iz ukazne vrstice še prebrati, jedro pa shraniti in jih uporabiti. Ta dva dela izvorne kode sta enaka ne glede na način komunikacije.

3.2 Popravki in dopolnila v zvezi z upravljanjem

Nova funkcionalnost stikala poleg sprememb v samem delovanju mostu prinaša spremembe in dopolnila tudi na področju upravljanja. Glavnina dela v zvezi z upravljanjem teče v uporabniškem pomnilniškem prostoru, zaradi razlogov, ki smo jih spoznali v podpoglavju 3.1.2, pa so nekatere spremembe potrebne tudi v samem jedru.

3.2.1 Uporabniški pomnilniški prostor

Del upravljanja, ki poteka v uporabniškem prostoru, dejansko pomeni dodajanje nove funkcionalnosti v obliki razširitev pri klicih funkcije `ioctl` in pri dostopanju do sistema `sysfs`. Možna sta dva pristopa - razvoj novega orodja (programa) ali preprosto dodajanje funkcionalnosti obstoječemu orodju `brctl`. Ker je za `brctl` na voljo vsa izvorna koda, sem se odločil za slednje. Ta rešitev se je zdela elegantnejša, saj tako ohranjamo možnost celotne konfiguracije mostu z enega mesta.

Po pregledu nove funkcionalnosti in dokumentacije [7] sem se odločil za naslednje spremembe:

- orodje mora omogočiti označevanje usmerjevalniških vrat na stikalu,
- omogočeno mora biti spreminjanje privzetega časa, ko posamezno sporočilo IGMP Report izgubi svoj pomen,
- potrebno je zagotoviti pregled tabele multicast, iz katere je razvidno stanje posameznih vrat, oziroma na katera vrata naj se usmerja promet iz posameznih skupin,
- omogočeno mora biti ročno priključevanje in izključevanje posameznih vrat k posameznim skupinam.

V ta namen sem orodje `brctl` razširil s petimi dodatnimi ukazi:

```
showmulti    <br>                    prikaz skupin multicast po vratih
settimeout   <br> <cas>              sprememba privzetega časa
setrouter    <br> <vrata> {on|off}   nastavi usmerjevalniška vrata
addsnoop     <br> <vrata> <skupina>  dodaj skupino vratom
removesnoop  <br> <vrata> <skupina>  odstrani skupino vratom
```

Odločitev, da konfiguriranje mostu poteka prek razširjenega programa `brctl`, je deloma botrovala tudi programerskemu slogu. Tako se dodatne funkcionalnosti kličejo na način, ki predvideva tako uporabo funkcije `ioctl`, kot tudi imenika `/sys`.

Osrednji del programa `brctl` predstavlja funkcija `br_set`, prek katere se nastavlja večina parametrov, ki vplivajo na delovanje mostu. Funkcija je zasnovana dovolj robustno, da najprej ugotovi, ali je na sistemu krmiljenje prek imenika `/sys` sploh podprto in nato izvede ustrezen način konfiguracije.
Dodane so bile naslednje funkcije, ki po vrsti izvedejo prej opisane ukaze:

```
int br_show_multicast_table(const char *br);
int br_change_mcast_timeout(const char *br, int new_time);
int br_add_router_port(const char *br, int port);
int br_del_router_port(const char *br, int port);
int br_add_group(const char *br, int port, const char *group);
int br_del_group(const char *br, int port, const char *group);
```

Vse dodane funkcije so narejene na enak način, in sicer vsebujejo klic `br_set(const char *bridge, const char *name, unsigned long value, unsigned long oldcode)`, kot to kaže primer:

```
int br_show_multicast_table(const char *br) {
     return br_set(br, "show_mc_table", 0,
                    BRCTL_MC_SHOW_TABLE);
}
```

3.2.2 Jedro

Razprava v prejšnjem podpoglavju je nakazala, da bodo dopolnitve jedrnega dela upravljanja obsegale dodajanje podpore obema vrstama komunikacije z uporabniškim delom in razširitev struktur, kjer se nastavitve shranijo. Slednje je povezano s strukturama `net_bridge` in `net_bridge_port`, ki sta definirani v datoteki `br_private.h`, razširjeni pa sta bili s podatkoma o trajanju veljavnosti sporočila IGMP Report in nastavitvijo usmerjevalniških vrat:

```
struct net_bridge
{
  .
  .
  unsigned long long snoop_timeout; // group timeout
};


struct net_bridge_port
{
  .
  .
  int router_port;                  // router port flag
};
```

V strukturi `net_bridge` je tudi definicija tabele multicast, ki jo bomo natančneje spoznali v nadaljevanju. Komunikacijo s programom `brctl` predstavljajo spremembe v datotekah `br_ioctl.c`, `br_sysfs_br.c` in `br_sysfs_if.c`. V datoteki `br_ioctl.c` se nahajajo funkcije, ki se pokličejo ob klicu `ioctl`. Zaradi večje jasnosti sem napisal svojo

funkcijo, ki pregleda vhodno zahtevo in pokliče ustrezno funkcijo. Funkcija tako v bistvu vsebuje samo en odločitveni stavek `switch`, vse skupaj pa izgleda takole:

```
case BRCTL_MC_SHOW_TABLE:
{
  show_igmp_snoop_table(br);
  return 0;
}
```

Enak vzorec se ponovi še za `BRCTL_MC_CH_TO, BRCTL_MC_ADDPORT, BRCTL_MC_DELPORT`, ki po vrsti izvedejo ukaze za prikaz skupin multicast po vratih, spremembo privzetega časa, označitev danih vrat za usmerjevalniška, izbris oznake usmerjevalniška za določena vrata.

Zaradi možnosti upravljanja mostu prek datotečnega sistema `sysfs` je bilo na podoben način potrebno napisati še spremembe v datotekah `br_sysfs_br.c` in `br_sysfs_if.c`. Popravki v prvi datoteki služijo nastavljanju časa, v katerem poteče veljavnost poslanega sporočila IGMP Report, kot sem to opisal že v poglavju 2.5.1. Kot vsi ostali parametri, ki se nastavljajo v tej datoteki, tudi ta parameter velja za celoten most. Popravki v datoteki `br_sysfs_if.c` pa so namenjeni parametrom, ki veljajo za posamezna vrata. V to datoteko sem dodal nastavljanje statusa vrat glede zastavice: usmerjevalniška ali običajna.

Posledica posegov v `br_sysfs_br.c` in `br_sysfs_if.c` sta dve novi psevdo-datoteki, in sicer psevdo-datoteka `router` v ustreznem imeniku `/sys/devices/virtual/net/-br0/ethX` ter `snoop_time` v imeniku `/sys/devices/virtual/net/br0/bridge`.

3.3 Preklapljanje prometa - algoritem in podatkovne strukture

Tudi pri implementaciji osnovne funkcionalnosti vohljanja IGMP, torej pregledovanja in preklapljanja okvirjev, sem se držal načela najmanjšega možnega poseganja v izvorno kodo mostu. To pomeni, da je osnovna funkcionalnost mostu, ko ne gre za multicast, ostala nespremenjena. Zato sem se odločil napisati dodatne funkcije, ki se izvedejo samo v primeru, ko most zazna okvir vrste multicast.

Kot je bilo pričakovati, most v Linuxu že opravlja osnovni pregled okvirjev, kjer ugotovi, ali gre za okvir, namenjen enemu ali več uporabnikom. Na tej točki sem zato izvorno kodo popravil tako, da v primeru, ko most ugotovi, da gre za skupinski ciljni

naslov, pregleda okvir še natančneje. Gre za funkcijo `int br_handle_frame_finish (struct sk_buff *skb)`, v datoteki `br_input.c`. Če funkcija ugotovi, da je okvir vrste broadcast, se koda izvaja v nespremenjeni obliki, v nasprotnem primeru pa program začne izvajati novo funkcionalnost.

To funkcionalnost predstavlja nekaj podpornih funkcij, ki skrbijo za pregled tabele multicast, ugotavljajo, ali so določena vrata že v tej tabeli, vrata v tabelo dodajajo in jih iz nje tudi brišejo.

3.3.1 Podatkovne strukture

Glavno podatkovno strukturo, ki se pri tem ves čas uporablja, predstavlja slika 3.1. Kot je vidno na sliki, je struktura sestavljena iz dveh delov:

1. razpršene tabele skupin multicast, v kateri ključ predstavlja skupina, vsebino pa kazalec na enosmerno povezan seznam in

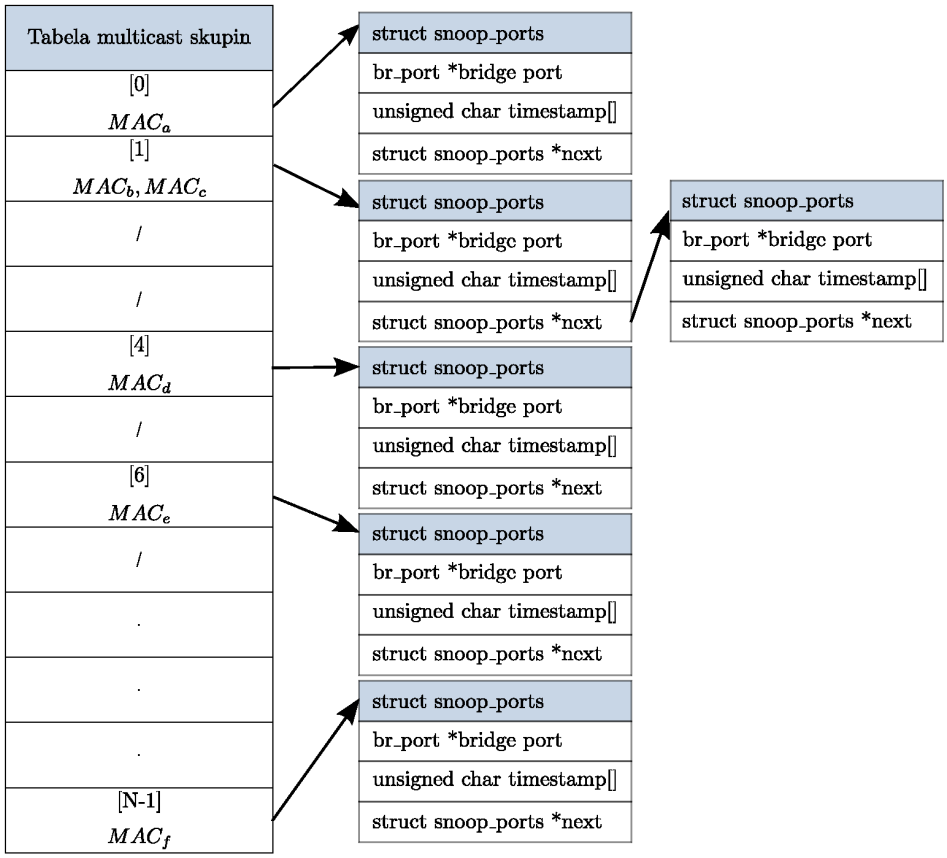
2. elementov seznama, ki vsebuje strukturo s podatki o vratih, na katerih je most zaznal sporočilo IGMP Report za to skupino, in čas prejetja okvirja.

Tabela multicast

Tabela multicast predstavlja povezave skupin multicast z vrati stikala. Skupin je seveda veliko, saj vsak naslov IP iz skupine naslovov multicast predstavlja svojo skupino. To pomeni, da imamo v razponu med 224.0.0.0 in 239.255.255.255 več kot 268 milijonov različnih skupin. Če se omejimo na vohljanje po naslovih MAC, je skupin sicer 32-krat manj, vendar še vedno zelo veliko. V pomnilniku torej ne moremo hraniti podatkov o vsaki skupini posebej, saj bi to zahtevalo zaseganje prevelike količine pomnilniškega prostora (najmanj 256MB, če bi vsako skupino lahko predstavil samo z enim oktetom, kar pa ni mogoče).

Zaradi hitrosti sem se odločil, da na tem mestu ne uporabim katere izmed podatkovnih struktur z dinamičnim zaseganjem pomnilnika. Tabela multicast je tako implementirana kot *razpršena* (angl. *hash*) tabela, torej tabela, ki jo določajo ključ, vrednost in posebna razpršitvena funkcija, ki vrednost v tabeli poišče glede na podani ključ [11].

Za učinkovito delovanje razprešene tabele je izbira razpršitvene funkcije zelo pomembna. Prva možnost, ki se ponuja, je, da bi razpršitveno funkcijo predstavljala kar številka skupine. Zaradi prostorske potratnosti je to nesprejemljivo. Ostale možnosti pa prinašajo



Slika 3.1 Tabela multicast.

tako imenovane kolizije, kar pomeni, da se bo več skupin preslikalo v isti ključ. Običajna in najpreprostejša rešitev tega problema je *veriženje* (angl. *chaining*), kar pomeni, da je v okviru razpršene tabele implementirana še ena struktura, običajno kar *povezani seznam* (angl. *linked list*). Ta omogoča, da se za enim ključem nahaja več elementov, pravilnega pa nato poiščemo z iskanjem po seznamu. Učinkovitost tabele se zaradi uporabe seznama (iskanje po seznamu ima linearno časovno zahtevnost) seveda zmanjša, vendar so ob ustrezni izbiri razprišitvene funkcije ti seznami kratki. Namesto enosmernega povezanega seznama bi bilo mogoče uporabiti tudi učinkovitejše strukture, ki za iskanje elementa nimajo linearne časovne zahtevnosti, a se za to nisem odločil, saj je največja dolžina seznamov omejena s številom vrat, povezanih v most, teh pa je običajno občutno manj kot 100 (še večkrat pa samo nekaj). Poleg tega je operacija iskanja elementa v seznamu potrebna samo v primeru sporočil IGMP. Drugače je v primeru dospetja okvirja, ki nosi podatke za določeno skupino. V tem primeru je potrebno pregledati vsa vrata,

ki so prijavljena v skupino, za kar pa pri uporabi kakšne druge podatkovne strukture (denimo rdeče-črnega drevesa [11]) potrebujemo najmanj enako časa kot pri običajnem seznamu. Ker je podatkovnih okvirjev v primerjavi s številom okvirjev nekajkrat več, je iskanje elementa v seznamu sorazmerno redko, izbira enosmernega linearnega seznama pa zadovoljiva rešitev.

Elementi seznamov vrat s podatki o odjemalcih

Vsako polje v osnovni tabeli torej kaže na seznam vrat, ki so prijavljeni v določeno skupino. Elemente teh povezanih seznamov predstavlja struktura `struct snoop_ports`. Ta struktura pa ne nosi samo podatka o vratih, temveč tudi podatke o odjemalcih, prijavljenih na ta vrata. Struktura vsebuje:

- `net_bridge_port *port` - kazalec na strukturo, ki definira vrata mostu,
- `unsigned char timestamp[]` - čas dospetja zadnjega sporočila IGMP Report, ki ga je za to skupino in na ta vrata poslal posamezni odjemalec, določen z naslovom MAC,
- `struct snoop_ports *next` - kazalec na naslednji element seznama.

Tabela odjemalcev za določena vrata je potrebna za aktivno spremljanje prijav v in odjav iz skupine. To tabelo predstavlja element `unsigned char timestamp[]`, označuje pa čas zadnjih prihodov sporočil IGMP Report na določena vrata z nekega naslova MAC. Tabelo pregleduje poseben časovnik, ki pretečena sporočila izbriše iz tabele oziroma izvaja pasivno odjavo vrat iz skupine. Časovnik bom natančneje opisal v nadaljevanju. Poleg pasivne odjave, oziroma izbrisa zaradi poteka časa, tabela omogoča tudi aktivno odjavo iz skupine, torej odjavo s pomočjo sporočila IGMP Leave.

Operacije na podatkovnih strukturah

Operacije na podatkovnih strukturah predstavljajo podporne funkcije, ki sem jih omenjal prej. V bistvu gre za običajne operacije nad seznami: dodajanje v seznam, brisanje iz seznama in pregled seznama. Te operacije se izvajajo ob upoštevanju, da imajo vsi seznami svoj začetek v tabeli skupin, kot to kaže slika 3.1.

Konkretna implementacija vsebuje naslednje funkcije:

- `int add_to_hash(int group, struct net_bridge_port *pt, unsigned char mac_in_gr[])`. Namenjena je dodajanju elementov v tabelo. Funkcija poišče seznam glede na parameter `group`. Nato s pomočjo parametra `struct net_bridge_port *pt` najde ustrezni element v seznamu. Če elementa v seznamu še ni (gre za nova vrata v skupini), ga v seznam doda. Parameter `mac_in_gr` določa, kateremu odjemalcu v tabeli `timestamp[]` ustreznega elementa je potrebno posodobiti čas vključitve v skupino.
- `void show_hash(int group, struct net_bridge *br)`. Ta funkcija prikaže vse elemente, ki obstajajo v seznamu za dano skupino.
- `void del_from_group(int group, struct net_bridge_port *pt, unsigned short mac_in_gr)` odjemalcu, ki je poslal sporočilo IGMP Leave čas zadnjega pošiljanja sporočila IGMP Report v tabeli `timestamp[]` ponastavi na 0.
- `void del_from_hash(int group, long max_time, struct net_bridge *br)`. Funkcija ugotovi, kateri odjemalci v okviru skupine `group` imajo čas zadnjega prispelega sporočila IGMP Report v tabeli `timestamp[]` starejši od `max_time`. Takšnim odjemalcem čas v tabeli `timestamp[]` ponastavi na 0. Če za posamezna vrata v okviru obravnavane skupine noben izmed elementov tabele `timestamp[]` ni različen od 0, se takšna vrata izbrišejo iz seznama vrat, ki pripadajo skupini. Izbris pomeni tudi dealokacijo pomnilnika.

Časovnik

Kot sem v tem poglavju že omenil, se s tabelo `timestamp[]` v strukturi `snoop_ports` ukvarja tudi poseben časovnik. Časovnik je implementiran v obliki posebne niti, ki neprestano teče v ozadju in periodično vsaki dve sekundi preverja, če je kakšnemu zapisu v tabeli že potekel čas. Čas dveh sekundi sem izbral empirično, in sicer kot kompromis med zahtevnostjo preverjanj in potrebo po dovolj ažurni tabeli multicast. Zapisom, ki je potekel čas, ali pa so bili aktivno odjavljeni s sporočilom IGMP Leave, se čas v tabeli ponastavi na 0. Če so vsi vnosi v tabeli `timestamp[]` enaki 0, se element izbriše iz seznama, s tem pa se vrata pasivno odjavijo od skupine. Tudi za to poskrbi časovnik. Omejitev izbrane izvedbe je, da lahko vrata promet za skupino, od katere so se že odjavili vsi odjemalci (ali jim je potekel čas), sprejemajo še največ dve sekundi. Časovnik

se požene ob inicializaciji modula `bridge`, torej ob nalaganju modula v pomnilnik in aktivaciji le-tega z ukazom `ifconfig <interface> up`. Zagon niti izvede ukaz:

```
struct task_struct *tthread=kthread_run(timer_thread, br, "kthread");
```

Ta ukaz vrne kazalec na novo jedrno nit `tthread`. Prvi parameter je ime funkcije, ki se bo izvajala v niti, vhodni parameter tej funkciji je `br`, ki predstavlja kazalec na instanco mostu, zadnji parameter pa je opisno ime niti. Ukaz se pokliče iz funkcije `void br_dev_setup(struct net_device *dev)`, ki jo najdemo v datoteki `br_dev.c`, izvede pa se ob aktivaciji modula.

Nit se ob ustavljanju mostu, ko se izvede funkcija `static int br_dev_stop(struct net_device *dev)`, ustavi sama, in sicer z ukazom:

```
kthread_stop(tthread);
```

Dejanska implementacija časovnika je preprosta - gre za neskončno zanko, ki periodično vsaki dve sekundi za vsako skupino izvaja že opisano funkcijo `del_from_hash`:

```
int timer_thread(void *data) {
  int i;
  struct net_bridge *br = (struct net_bridge *) data;

  while (1) {

    for (i=0;i<GROUP_NUM;i++)
      del_from_hash(i, br->snoop_time, br);

    msleep(2000);

    if (kthread_should_stop()) break;
  }
}
```

3.3.2 Algoritem

Že v uvodu v to poglavje sem omenil težnjo po čim manjšem poseganju v obstoječi most, kar je prineslo možnost, da izvorno kodo dopolnim samo od točke, ko je most že ugotovil, da je okvir, ki ga obravnava, namenjen več uporabnikom. Naslednji korak je razločevanje med okvirji multicast in broadcast. Če je okvir vrste multicast, temu sledi algoritem, prikazan na sliki 3.2.

1. Algoritem najprej ugotovi, ali okvir vsebuje paket IP, ki vsebuje podatke po protokolu IGMP. Preverjanje je mogoče z ugotavljanjem tipa protokola znotraj paketa IP, ki je vsebovan v pregledovanem okvirju: če paket vsebuje podatke po protokolu IGMP, je vrednost polja Tip enaka 0x0002 (slika 2.1). Algoritem se v tem primeru nadaljuje z naslednjim korakom, sicer pa s korakom 4.

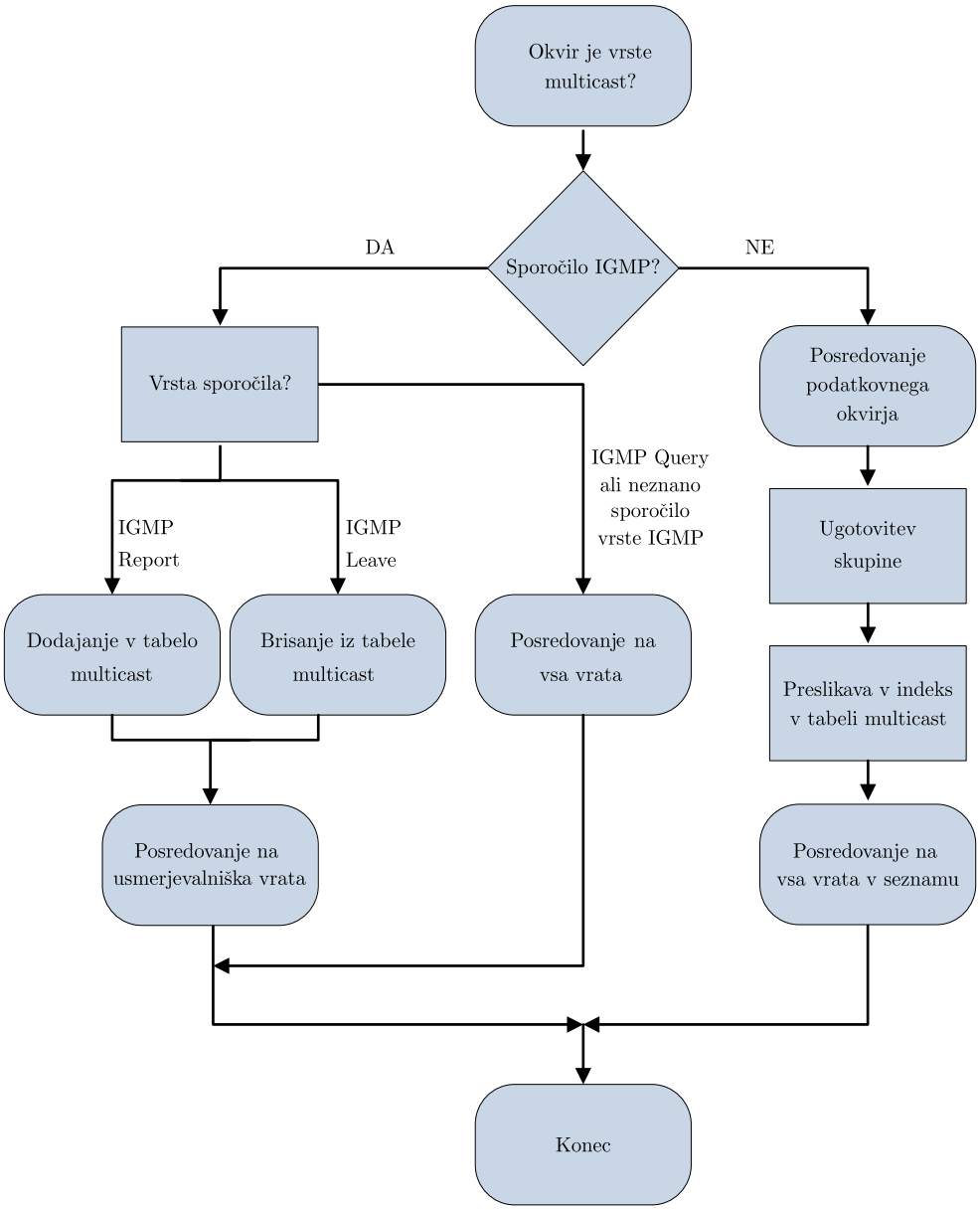
2. Glede na vsebino po protokolu IGMP lahko pričakujemo:

   - dodajanje elementov v tabelo (sporočilo IGMP Report, elementa še ni v tabeli),
   - osveževanje elementov (sporočilo IGMP Report, element je že v tabeli),
   - izbris elementov iz tabele (sporočilo IGMP Leave).

   Dejanska implementacija v primerih IGMPv1 in IGMPv2 neposredno uporablja podporne funkcije, opisane v podpoglavju 3.3.1, več težav pa je seveda z verzijo 3.

3. Okvir se posreduje naprej. Posredovanje je odvisno od vrste sporočila. Sporočila IGMP Report ter IGMP Leave se posredujejo samo aktivnim usmerjevalniškim vratom. Sporočila IGMP Query se posredujejo vsem vratom, razen izvornim. Enako se vsem vratom, razen izvornim, posreduje vsa sporočila IGMP, ki jih most ne prepozna.

4. Če se algoritem znajde v tej točki, obravnavani okvir nosi običajne podatke in ne informacij po protokolu IGMP. Takšne okvirje posreduje brez spreminjanja tabele multicast. Ciljna skupina se v implementaciji ugotavlja na podlagi ciljnega naslova MAC, kar je hitreje kot ugotavljanje skupine na podlagi naslova IP, obenem pa še vedno v skladu s priporočili [7]. Vrata, na katera most posreduje okvirje dobimo s sprehodom po seznamu, ki ga v tabeli multicast določa skupina.



Slika 3.2 Algoritem - posredovanje okvirjev.

Algoritem je s tem končan, na tem mestu, pa je potrebno dodati še komentar k podpori IGMPv3. Ker je v tej izvedbi IGMPv3 podprt le delno, je drugi korak algoritma poenostavljen, in sicer tako, da program ne pregleduje spiska naslovov unicast, s katerih odjemalec želi, oziroma ne želi prejemati prometa, ampak upošteva samo skupino multicast, ki ji je okvir namenjen. Takšna implementacija seveda ne zadošča zahtevam tretje različice protokola IGMP, je pa v velikem številu praktičnih primerov zadostna - tipičen odjemelec za spremljanje videa prek IP, ki se želi priključiti določeni skupini, v resnici ne navaja od koga želi prejemati promet (spisek virov je prazen) in kljub uporabi IGMPv3 deluje kot bi uporabljal IGMPv2. Razlog za takšno stanje, ki vlada v trenutku pisanja tega diplomskega dela, je tudi v tem, da večina odjemalcev protokola IGMP ne implementira neposredno, temveč uporablja API operacijskega sistema in tako prek funkcij operacijskega sistema pošilja sporočila IGMP. Današnji operacijski sistemi (Linux, MS Windows XP in kasnejši) sicer podpirajo IGMPv3 in na zahtevo odjemalskega programa reagirajo tako, da pošljejo okvirje v obliki sporočil IGMPv3, vendar programi niso prilagojeni dodatnim funkcionalnostim, ki jih ta verzija omogoča. Tako v trenutnih različicah najbolj priljubljenih programov za spremljanje videa ni mogoče navesti vira, oziroma virov, kar posledično pomeni, da so sporočila poslana v obliki IGMPv3, ki je ekvivalentna IGMP Report oziroma IGMP Leave IGMPv2. Podrobnosti so opisane v poglavju 2.3.2.

4 Preizkus delovanja in analiza zmogljivosti nadgrajenega stikala

Namen tega poglavja je v praksi preizkusiti delovanje nadgrajenega stikala ter analizirati in oceniti njegovo delovanje, oziroma učinkovitost, v primerjavi s stikalom z osnovno funkcionalnostjo. S tem želim raziskati, kako se stikalo odziva na različne količine dohodnega prometa in kakšne so zmogljivosti stikala pri obremenitvi z običajnimi paketi multicast in kakšne pri simuliranem napadu s paketi IGMP.

Že uvodno razmišljanje, opis delovanja stikal, predvsem pa opis praktične izvedbe so nakazali, da nova funkcionalnost v delovanje stikala prinaša dve spremembi z nasprotujočim si vplivom na učinkovitost. Prva sprememba je, da mora stikalo z dodano funkcionalnostjo vohljanja IGMP pregledati vse vstopne okvirje in občasno poseči v tabelo multicast. Ta sprememba pomeni dodatno obremenitev in s tem tudi možnost poslabšanja učinkovitosti delovanja stikala. Druga sprememba pa ima lahko na učinkovitost ugodnejši učinek: ker stikalo okvirje preklaplja „pametneje“, je preklapljanj v primerih, ko vsa vrata niso ciljna vrata za določeno skupino prometa, manj. Bistveno za analizo je torej vprašanje, katera od obeh sprememb je za delovanje stikala pomembnejša.

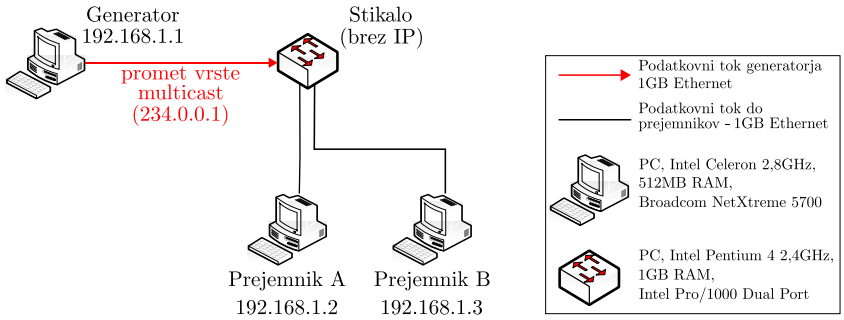
Preizkus delovanja ter analizo sestavljajo trije sklopi:

1. preizkus pravilnega delovanja programske opreme z uporabo praktične aplikacije,
2. analiza zmogljivosti stikala v primeru običajnega multicasta in
3. analiza delovanja stikala v primeru napada DoS z množico sporočil IGMP Report ter IGMP Leave.

Primerjavo nadgrajenega stikala z izvirnim stikalom vsebujeta zadnja dva sklopa. Ta primerjava je pomembna tudi zato, ker z njo dobimo odgovor na vprašanje o smotrnosti implementacije vohljanja IGMP na programskih stikalih.

4.1 Preizkus pravilnosti delovanja

S preizkusi pravilnosti delovanja sem se želel prepričati v pravilnost delovanja stikala in skladnost z ustreznimi standardi. Cilj preizkusov je ugotoviti ali pravilno delujejo in med seboj tudi sodelujejo vsi glavni sklopi programske opreme. Uvod v preizkuse predstavlja postavitev testnega okolja.



Slika 4.1 Testno okolje v primeru preizkusa pravilnosti delovanja

4.1.1 Testno okolje

Testno okolje za ta sklop je prikazano na sliki 4.1, predstavlja pa omrežje, v katerega so vključeni štirje osebni računalniki z operacijskim sistemom Linux. Osrednji računalnik deluje kot stikalo in na njem teče opisani modul `bridge`. Prek tega stikala so povezani še trije računalniki: generator prometa in dva prejemnika. Programsko opremo, ki generira in sprejema promet, bom opisal v nadaljevanju, saj se od primera do primera nekoliko razlikuje, na stikalu pa je v vseh primerih iz te skupine nameščen modul `bridge`.

Modul je po potrebi izviren ali pa nadgrajen, delovati pa začne po vnosu naslednjega zaporednja ukazov:

```
modprobe bridge
brctl addbr br0
brctl addif br0 eth0
brctl addif br0 eth1
brctl addif br0 eth2
ifconfig br0 up
```

Prvi ukaz naloži modul v pomnilnik, nadaljnji ukazi pa vzpostavijo novo instanco mostu (`br0`) in nanj pripnejo omrežne vmesnike `eth0`, `eth1` ter `eth2`, ki predstavljajo omrežne kartice. Modul je mogoče z ukazom `rmmod bridge` iz pomnilnika tudi odstraniti, kar je potrebno narediti ob menjavi izvirnega modula s nadgrajenim in obratno. Po takšni menjavi je potrebno zgoraj zapisano zaporedje ukazov `modprobe`, `brctl` in `ifconfig` seveda ponoviti, saj se nastavitve ob odstranitvi modula iz pomnilnika izgubijo.

4.1.2 Preizkusi delovanja

Del preizkusov iz tega sklopa temelji na uporabi odprtokodnega programa VLC. Program skupina prostovoljcev razvija v okviru projekta VideoLAN (`http://www.videolan.org`), gre pa za predvajalnik in tudi oddajnik večpredstavnih vsebin. VLC lahko večpredstavne vsebine predvaja iz lokalne datoteke ali prek omrežja. V času pisanja diplomskega dela je bila na voljo različica 1.0.5. Ker ta program večpredstavne vsebine v omrežje po želji tudi pošilja, lahko z njim vzpostavimo vir podatkovnega toka multicast. Za potrebe testiranja sem uporabljal predvajanje videa, ki sem ga v omrežje pošiljal na naslov 234.0.0.1. Program VLC kot izvor prometa teče na generatorju s slike 4.1, kot odjemalec pa tudi na strani obeh prejemnikov.

Poleg programa VLC sem za generiranje prometa uporabil še program `nemesis`, ki je dostopen na naslovu `http://nemesis.sourceforge.net`, z njim pa je v omrežje mogoče pošiljati različne vrste okvirjev. Konkretne primere uporabe bom opisal v okviru posameznega preizkusa.

Za spremljanje prometa je bil na vseh štirih računalnikih nameščen tudi program Wireshark (`http://www.wireshark.org`), s katerim lahko za vsak omrežni vmesnik zajemamo in spremljamo promet. Zajem prometa omogoča zelo natančno analizo doga-

janja, saj nam omogoča dostop do vsakega okvirja, ki pride v računalniški sistem; vsak okvir lahko razčlenimo in pogledamo njegovo vsebino. Preizkus je obsegal štiri sklope:

- preklapljanje prometa na ustrezna vrata (osnovna funkcionalnost),
- pregled ločevanja navadnih in usmerjevalniških vrat,
- pregled poseganja v tabelo multicast z nadgrajenim programom `brctl`,
- pregled delovanja časovnika.

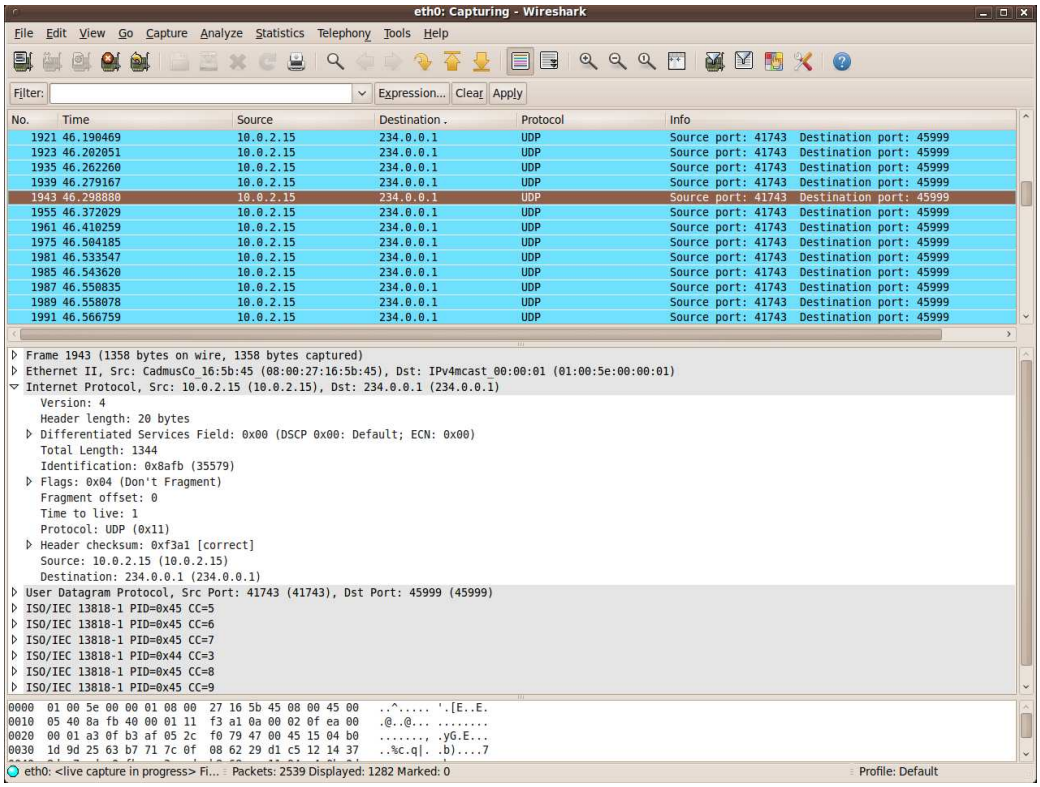
Uspešno izvedeni testi teh štirih sklopov bi pokazali pravilno delovanje glavnih elementov sistema, in sicer tako delovanje programa `brctl`, kot tudi delovanje jedra operacijskega sistema.

Osnovna funkcionalnost

Program VLC na generatorju nastavimo tako, da prične oddajati izbrano video vsebino. Nastavitev dosežemo z uporabo menija `Odpri -> Napredno odpiranje datotek`. Izberemo gumb `Dodaj` in si izberemo video vsebino, ki jo želimo predvajati. S pritiskom kombinacije tipk `Alt+S` se pomaknemo na naslednji korak. Izberemo gumb `Naslednji`. V skupini `Cilj` izberemo `UDP`, pritisnemo gumb `Dodaj`. V polje `Naslov` vtipkamo `234.0.0.1`. V polje `vrata` vpišemo neko številko, denimo 45999. S pritiskom na gumb `Pretok` v istem pogovornem oknu se začne predvajanje vsebine. Program Wireshark, ki teče na stikalu, pokaže, da se je na omrežju pojavil promet vrste multicast (slika 4.2).

Na obeh odjemalcih, ki sta na sliki 4.1 označena kot Prejemnika A in B, opravimo podoben postopek. Izberemo meni `Odpri -> Odpri omrežni pretok`. V spisku `Protokol` izberemo `UDP`, v sosednje polje `Naslov` pa vpišemo 234.0.0.1. V polje `vrata` vpišemo neko številko, ki pa mora biti enaka oznaki vrat na generatorju. Na računalniku Prejemnik A pritisnemo gumb `Predvajaj`, na drugem prejemniku pa ne.

Na tej točki se pojavi razlika med primeroma, ko na stikalu teče nadgrajeni ali izvirni modul `bridge`. V obeh primerih se na računalniku, kjer smo pritisnili gumb `Predvajaj`, začne predvajati video vsebina. Razlika je vidna s programom Wireshark - ta pokaže, da v primeru izvirnega modula `bridge` promet prejema tudi Prejemnik B, v primeru uporabe nadgrajenega modula `bridge` pa ne.



Slika 4.2 Programa Wireshark med sprejemanjem multicast prometa.

Ločevanje usmerjevalniških in navadnih vrat

Usmerjevalniška vrata se od navadnih ločijo po tem, da je nanje povezan usmerjevalnik. Razloge, zakaj je potrebno razlikovanje med navadnimi in usmerjevalniškimi vrati sem podrobneje opisal v 2.5.1, kjer sem omenil tudi, da standard [7] dovoljuje, da usmerjevalniška vrata označimo administrativno z uporabo posebnega programa. V ta namen sem program `brctl` razširil še s to funkcionalnostjo (poglavje 3.2.1). Kot usmerjevalniška sem določil vrata, na katera je priključen Prejemnik B; označena so kot `eth1`:

```
brctl setrouter br0 eth1 on
```

Kot generator okvirjev sem pri tem testu uporabil program `nemesis`. Za potrebe tega preizkusa je zanimiv zato, ker med drugim omogoča pošiljanje okvirjev z vsebino, ki ustreza protokolu IGMP.

V ta namen je na voljo nekaj opcij, med njimi so najbolj zanimive:

```
-p <tip sporočila IGMP v decimalni obliki>
-i <naslov IP skupine IGMP>
-D <ciljni naslov IP>
```

Program sem pognal z naslednjimi parametri:

```
nemesis igmp -p 22 -i 234.0.0.1 -D 234.0.0.1
```

Generator je na stikalo povezan prek vrat `eth0`. Zaradi parametra `-p 22` je poslani okvir predstavljal sporočilo IGMP Report. Stikalo je okvir preklopilo na vrata `eth1`, na vrata `eth2` pa ne. Poskus ponovim še v primeru, ko vrata `eth1` niso več označena kot usmerjevalniška:

```
brctl setrouter br0 eth1 off
```

V tem primeru stikalo prispelega okvirja ne preklopi nikamor, kar je prav, saj nobena izmed vrat niso označena kot usmerjevalniška. Opisani postopek sem ponovil še v primeru pošiljanja sporočil IGMP Query:

```
nemesis igmp -p 17 -i 234.0.0.1 -D 234.0.0.1
```

V tem primeru je stikalo okvir preklopilo na vsa vrata (razen izvornih), kar je v skladu z zahtevami standarda. Prav tako stikalo na vsa vrata preklopi neznane okvirje IGMP.

Časovnik

Časovnik predstavlja eno izmed zahtev standarda [7], ki ohranjajo združljivost vohljanja IGMP med različnimi verzijami. Izvedbo časovnika sem opisal v podpoglavju 3.3.1. Za potrebe tega testa sem periodo časovnika nastavil na 20 sekund. To omogoča ukaz

```
brctl settimeout br0 20
```

Ker je časovnik sedaj nastavljen na 20 sekund, bi moral vsak odjemalec sporočila IGMP Report pošiljati s periodo, krajšo od 20 sekund. Ker tega VLC s privzetimi nastavitvami ne počne, pričakujem, da se bo predvajanje videa po 20 sekundah ustavilo.

Test sem pognal z nastavitvami, ki sem jih opisal v podpoglavju o testiranju osnovne funkcionalnosti, rezultati pa so bili v skladu s predvidevanji - predvajanje videa na odjemalcu Prejemnik A se je po približno 20 sekundah ustavilo, program Wireshark pa kaže,

da generator v omrežje še vedno pošilja podatke, ki pa jih stikalo blokira. Nov test z nastavitvijo

```
brctl settimeout br0 80
```

spet pokaže pričakovano funkcionalnost - video se predvaja 80 sekund, nato se predvajanje prekine.

Poseganja v tabelo multicast

Programu `brctl` sem dodal tudi nekaj ukazov, ki omogočajo manipulacijo tabele multicast. Gre za ukaza `addsnoop` ter `removesnoop`, ki sem ju opisal v poglavju 3.2.1. Ker ukaza interno le ponovita dogajanje, ki se izvede ob prihodu sporočil IGMP Report ali IGMP Leave, sem za testiranje znova uporabil program `nemesis`, le da sem namesto ukazov (na generatorju):

```
nemesis igmp -p 22 -i 234.0.0.1 -D 234.0.0.1
```

in

```
nemesis igmp -p 17 -i 234.0.0.1 -D 234.0.0.1
```

uporabil ukaza:

```
brctl addsnoop br0 eth1 01:00:5e:00:00:01
```

in

```
brctl removesnoop br0 eth1 01:00:5e:00:00:01.
```

Oba ukaza sem pognal na stikalu, z njima pa sem vrata `eth1` uvrstil med, oziroma izločil iz seznama prejemnikov prometa v skupini, ki jo določa naslov MAC 01:00:5e:00:00:01. Poleg testa s programom `nemesis` sem si pomagal še z ukazom `brctl showmac br0`, ki je ustrezno prikazal vključitev in izključitev ustreznih skupin v tabelo multicast za instanco mostu `br0`.

4.2 Analiza delovanja ob običajnem multicast prometu

Namen tega sklopa je ugotoviti učinek vohljanja IGMP na zmogljivost, oziroma učinkovitost stikala ob običajnem multicastu - torej ne ob napadu s paketi IGMP. Kot sem omenil že v uvodu, se tu prepletata dva dejavnika: po eni strani lahko pričakujemo, da se

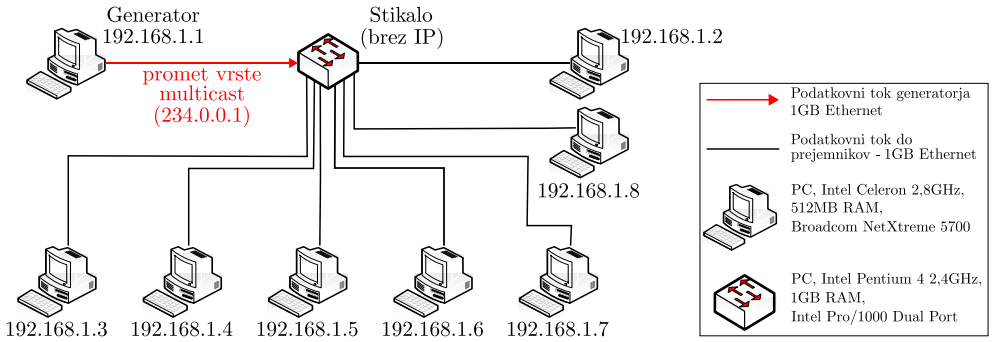
bo zmogljivost stikala zaradi obremenitve procesorja z dodatnim pregledovanjem paketov nekoliko znižala, po drugi strani pa lahko pričakujemo morebitno izboljšanje delovanja stikala v primerih, ko vsa vrata niso prejemniki prometa.

Stikalo ob prihodu okvirjev multicast izvaja preklapljanje prometa na ciljna vrata. Ker je v primeru stikal s podporo vohljanju IGMP število teh vrat lahko različno, pričakujem, da bo tudi zmogljivost stikala v vseh teh primerih različna. Zmogljivost je zato potrebno meriti v odvisnosti od dveh neodvisnih spremenljivk - količine prometa in števila ciljnih vrat. Ker želim v tem poglavju izvesti tudi primerjavo med izvirnim in spremenjenim stikalom, bo potrebno meritve ponoviti tudi na izvirnem stikalu - seveda pa bo teh meritev manj, saj je število ciljnih vrat zaradi edine možnosti uporabe poplavljanja v tem primeru vedno enako številu vseh vrat.

Za merjenje zmogljivosti sem uporabil posredno meritev, in sicer meritev obremenjenosti procesorja. Razlog za to je v tem, da želim meriti tudi obremenjenost stikala, ko le-to prometa ne preklaplja na nobena vrata. Ta primer je v praksi povsem mogoč - do njega pride, ko je v omrežju nek vir podatkov multicast (denimo ponudnik vsebin internetne televizije), obenem pa nihče od uporabnikov ne želi prejemati tovrstnih podatkov. To predstavlja tudi mejni primer, v katerem pride do največje razlike med stikali, ki vohljanje IGMP podpirajo, in tistimi, ki te podpore nimajo. Merjenje obremenjenosti CPE kaže, v kateri točki pride do „točke zloma“ sistema, ko sistem ne zmore več procesirati dohodnega prometa in zato začne izgubljati dohodne okvirje.

4.2.1 Postavitev okolja

Preizkusi v sklopu analize zmogljivosti stikala so potekali v testnem okolju, podobno tistemu s slike 4.1, le da je bilo razširjeno, kot to prikazuje slika 4.3. Na stikalo je bilo



Slika 4.3 Testno okolje v primeru analize zmogljivosti in preizkusa delovanja v primeru običajnega multicast prometa.

priključenih osem računalnikov - en izmed njih je bil namenjen generiranju prometa, ostalih sedem je bilo potencialnih prejemnikov. Stikalo je predstavljal osebni računalnik s štirimi prostimi režami PCI. Podrobnejša konfiguracija tega računalnika je naslednja:

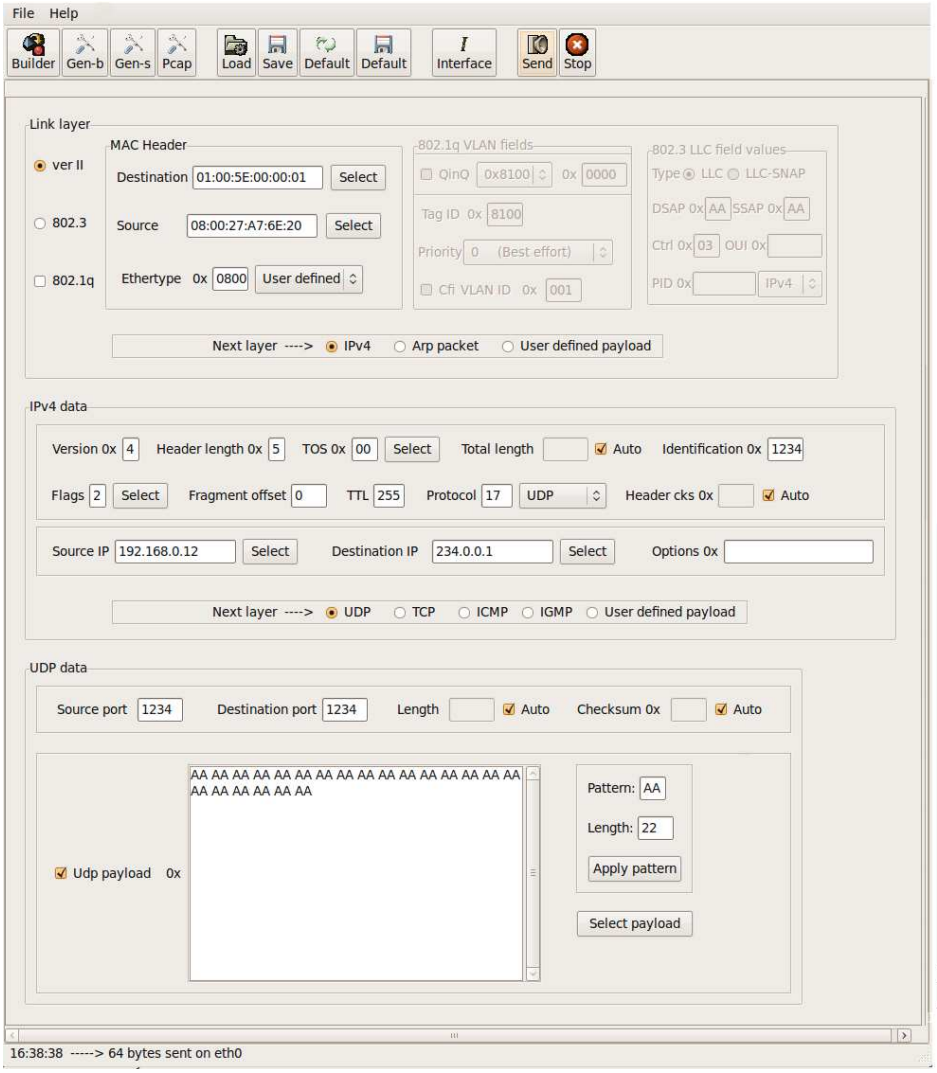
- CPE: Intel Pentium 4, 2,5GHz,
- RAM: 1GB,
- Disk: SM-Fujitsu MHT2040AT, 40GB.
- Operacijski sistem: okleščena verzija Ubuntu Linux 9.04 brez grafičnega vmesnika.

V računalniku, ki je imel vlogo stikala, so bile nameščene štiri omrežne kartice Intel Pro/1000 MT Dual Port Server Adapter, vsaka z dvemi vrati. Vsa vrata - tudi tista na isti kartici - so med seboj neodvisna tako s stališča strojne kot programske opreme. V operacijskem sistemu so bila vrata predstavljena kot omrežni vmesniki z ustreznimi imeni `eth0...eth7`. Vsi omrežni vmesniki so bili po že opisanem postopku povezani v en most `br0`.

Tudi prejemniki prometa so bili osebni računalniki, ki pa so imeli vgrajene naslednje komponente:

- CPE: Intel Celeron 2,8GHz,
- RAM: 512MB,
- Disk: Maxtor 6L080M0, 66GB.
- Omrežna kartica: Broadcom NetXtreme Gigabit Controller serije 5700,
- Operacijski sistem: Ubuntu Linux 9.04.

Promet v tem sklopu je generiral program `packeth` (`http://packeth.sourceforge.net`). Program odlikuje grafični uporabniški vmesnik, prek katerega je mogoče nadzorovati pošiljanje okvirjev multicast v omrežje. Dolžina okvirjev je bila 64 oktetov. Nastavitve, ki določajo obliko okvirjev, prikazuje slika 4.4.



Slika 4.4 Nastavitve programa `packeth` za pošiljanje običajnega multicast prometa

4.2.2 Meritve

Kot sem omenil že v uvodu v to podpoglavje, je merjenje zmogljivosti potekalo posredno prek merjenja obremenjenosti sistema, oziroma natančneje, procesorja. Za operacijski sistem Linux obstaja več orodij, ki prikazujejo obremenjenost sistema, široko se na primer uporablja program `top`, ki v tabelarični obliki kontinuirano prikazuje trenutno obremenjenost sistema. Opazovati je mogoče tako skupno obremenjenost sistema, kot tudi obremenitev po posameznih procesih in podsistemih (poleg procesorja prikazuje še zasedenost

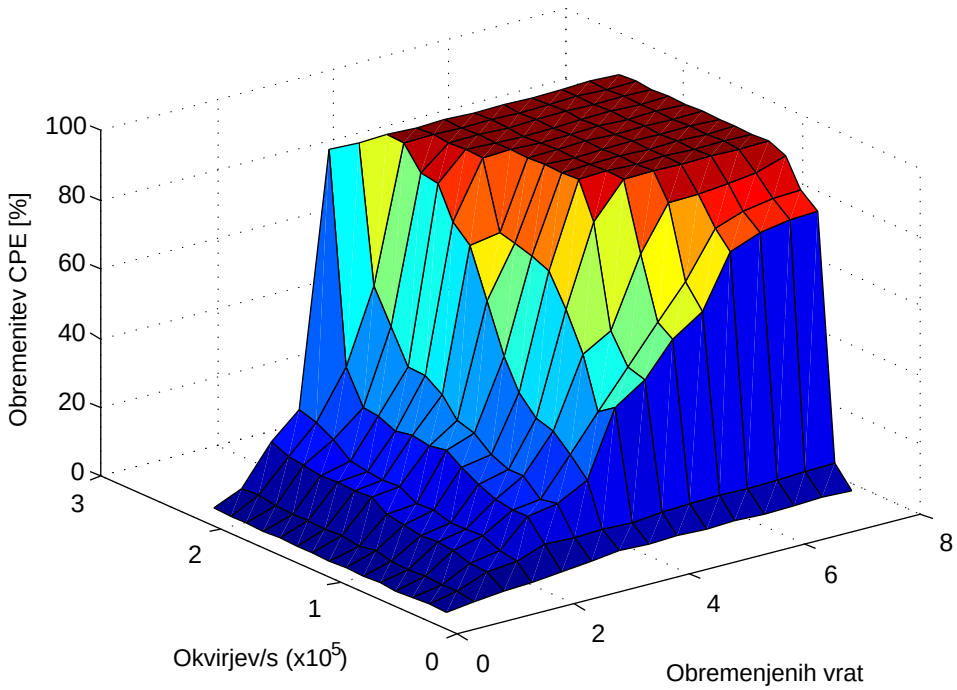
pomnilnika, izmenljivega pomnilnika itd.). Drug takšen program je `mpstat`, ki prikazuje enake podatke, vendar ne v obliki stalne tabele, lahko pa vrednosti izpisuje periodično, pri čemer periodo podamo kot parameter pri zagonu programa.

Težava pri merjenju obremenitve procesorja je v tem, da je obremenitev ves čas stohastičen proces in zato nekoliko niha - tudi takrat, ko meritev ne izvajamo. Kljub temu pa je z uporabo statističnega povprečenja prek obdobja več deset sekund mogoče opaziti, da obremenitev procesorja zgovorno kaže na odvisnost od prej omenjenih neodvisnih spremenljivk - količine prometa na stikalu in števila ciljnih vrat. Zato sem si pri tem testu pomagal z beleženjem; ker so podatki o obremenitvi na voljo ves čas, jih je mogoče zapisovati v datoteko, kar omogoča naknadno obdelavo podatkov s povprečenjem in analizo. Z empiričnimi meritvami sem ugotovil, da zadostujejo sekvence vzorčenja v trajanju približno 60 sekund.
Podatki o obremenitvi procesorja so se v datoteko zapisovali v enosekundnem intervalu. To dosežemo z ukazom: `mpstat 1 >> /home/user/meritev.txt`, kar pomeni dodajanje zapisov s podatki v datoteko `/home/user/meritev.txt`. Izsek treh naključno izbranih zapisov, kjer je bila obremenitev procesorja 42,57%, 6% in 11,88%, prikazujejo naslednje vrstice:

```
15:03:04 CPU %usr  %nice %sys %iowait %irq  %soft  %steal %guest %idle
15:03:05 all 0,00  0,00  0,99  0,00   11,88  29,70  0,00   0,00  57,43
15:03:06 all 0,00  0,00  0,00  0,00    2,00   4,00  0,00   0,00  94,00
15:03:07 all 0,99  0,00  0,00  0,00    1,98   8,91  0,00   0,00  88,12
```

4.2.3 Rezultati in analiza

Rezultate meritev prikazuj graf 4.5. Ta prikazuje dve neodvisni spremenljivki - zato je ta graf prikazan v treh dimenzijah - prva neodvisna spremenljivka je število generiranih okvirjev, druga pa število vrat, na katera mora stikalo preklopiti okvirje. Posamezna vrata se v skupino vključijo s pomočjo sporočila IGMP Report, lahko pa tudi administrativno s pomočjo ukaza `brctl`. Odvisna spremenljivka je obremenitev procesorja. Graf 4.5 kaže na hitro padanje učinkovitosti delovanja stikala ob povečevanju števila vrat, na katera stikalo preklaplja promet. Ravnina, ki kaže obremenitev CPE, se pri samo enih ciljnih vratih dviga počasi in na prikazanem grafu, kjer je največja obremenitev okoli 220000 okvirjev na sekundo, ne preseže 20%. Ob povečevanju števila ciljnih vrat pa se graf vzpenja hitreje. Najhitreje do polne obremenitve pride stikalo, ki preklaplja na vseh

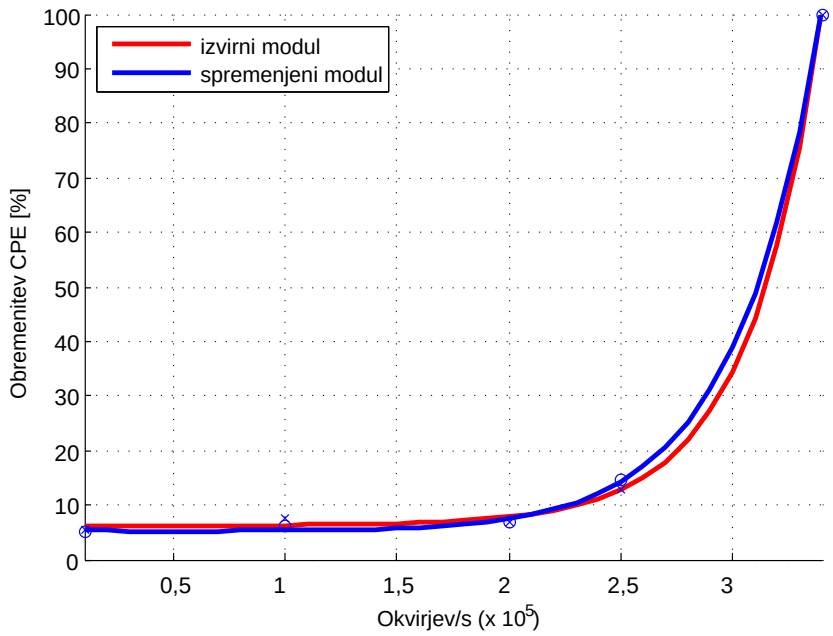


Slika 4.5 Obremenitev CPE v odvisnoti od števila dohodnih okvirjev in števila ciljnih vrat.

sedem vrat, in sicer pri okoli 33000 okvirjih na sekundo.

Da je število ciljnih vrat res glavni razlog padanja učinkovitosti stikala, kažejo tudi meritve delovanja izvirnega stikala. Ker izvirno stikalo ves dohodni multicast promet vedno preklopi na vsa vrata, meritve takšnega stikala niso odvisne od števila vrat, ki želijo prejemati promet. Graf 4.6 je zato dvodimenzionalen, na njem pa je z rdečo barvo označena zasedenost procesorja izvirnega stikala ob različni količini dohodnega prometa, z modro pa stikala s podporo vohljanju IGMP, kjer so prejemniki prometa vsa vrata. Graf 4.6 kaže veliko ujemanje med delovanjem izvirnega stikala, ki dohodni promet preklopi na vsa vrata in stikalom s podporo vohljanju IGMP v primeru, ko so vsa vrata prejemnik prometa. V tem primeru je delovanje obeh stikal namreč podobno - ves promet se preusmeri na vsa vrata, stikalo s podporo vohljanju IGMP pa obenem tudi pregleduje vhodne okvirje. Ugotovimo lahko, da učinkovitost stikala s podporo vohljanju IGMP v prikazanem primeru ne zaostaja za učinkovitostjo izvirnega stikala. To pomeni, da je za-

htevnost vohljanja v primeru običajnega multicast prometa v primerjavi z zahtevnostjo preklapljanja zanemarljiva. Iz te ugotovitve sklepam, da je v primeru programske implementacije stikal izvedba vohljanja na stikalu učinkovit ukrep, ki v normalnih razmerah, kakršne je simuliral ta test, poveča uporabnost stikala. Opravljeni preizkusi kažejo na



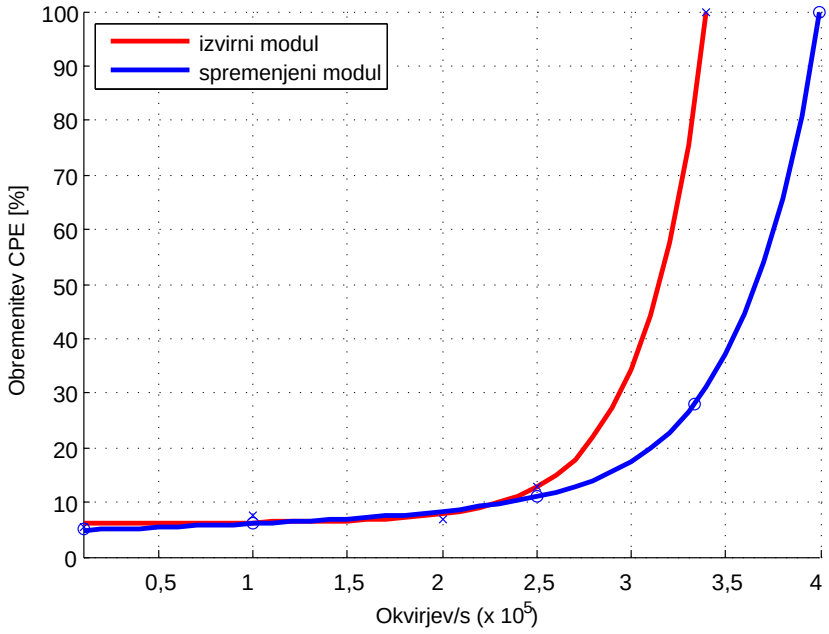
Slika 4.6 Obremenitev CPE stikala v odvisnosti od količine prometa pri izvirnem in spremenjenem modulu, kjer so kot ciljna določena vsa vrata.

koristnost vohljanja IGMP.

Da bi bolje razumeli pridobitev zmogljivosti stikala z implementirano funkcionalnostjo vohljanja IGMP na račun manjšega števila preklapljanj, si poglejmo še obremenitev v primeru pošiljanja na vsa vrata, razen enih (torej na šest vrat). Izkaže se (slika 4.7), da se točka zloma v primeru stikala, ki preklaplja na vsa vrata, razen enih, iz okoli 33000 okvirjev na sekundo pomakne na 40000 okvirjev na sekundo. Ena ciljna vrata manj so torej že pomaknila točko polne zasedenosti procesorja za skoraj 20%.

4.3 Analiza delovanja v primeru napada z okvirji IGMP

Poleg doslej opravljenih sklopov preizkusov ostaja odprto še vprašanje: kako stikalo reagira na napad DoS z množico sporočil IGMP Report in IGMP Leave. Pričakujemo namreč lahko, da bo učinkovitost stikala v tem primeru drugačna kot pri običajnem



Slika 4.7 Obremenitev CPE stikala v odvisnosti od količine prometa pri izvirnem in spremenjenem modulu, kjer je bilo kot ciljnih določenih šest vrat

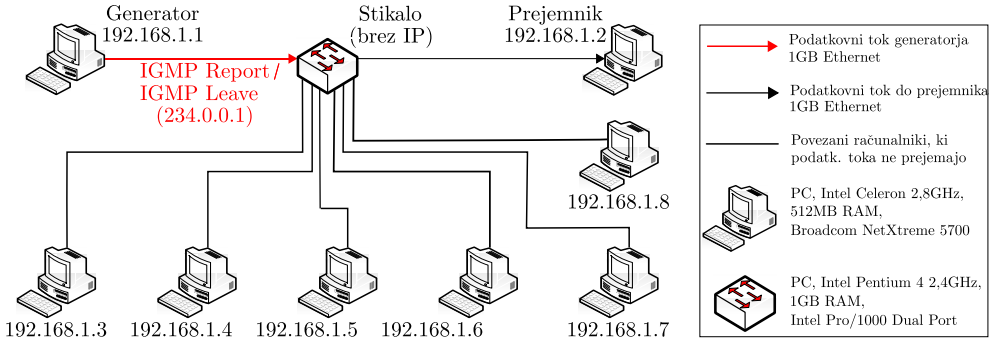
multicastu, saj tudi stikalo v teh dveh primerih deluje povsem drugače. Oba načina delovanja sem podrobneje opisal v poglavju 3.2.2, na kratko pa naj ponovim, da sporočila IGMP zahtevajo različne posege v tabelo multicast, v primeru običajnega prometa pa stikalo tabelo zgolj pregleduje.

Posegi v tabelo multicast so časovno potratni, saj zahtevajo zaseganje in sproščanje pomnilnika ob vsakem prispelem paru sporočil IGMP Report in IGMP Leave za vsaka posamezna vrata. Po drugi strani pa v primeru teh sporočil lahko ugotovimo, da stikalu prispelih okvirjev ni več potrebno razpošiljati na vsa vrata, temveč samo na usmerjevalniška (torej v tipičnem primeru na ena sama). Zato odpade zamudno preklapljanje, za katerega so meritve že pokazale, da je glavni razlog za počasno delovanje programskih stikal.

4.3.1 Testno okolje

Testno okolje je podobno okolju iz analize zmogljivosti ob običajnem prometu vrste multicast (slika 4.3). Da bi čim bolje simulirali realne razmere, v katerih običajno delujejo stikala in v katerih bi lahko prišlo do tovrstnega napada, so bila ena vrata stikala označena

kot usmerjevalniška. Stikalo je na ta vrata preklapljalo dohodna sporočila IGMP. Ker so ves promet tvorila ravno takšna sporočila, je stikalo na ta vrata preklapljalo vse dohodne okvirje. Okolje prikazuje slika 4.8. Glavna razlika med tem okoljem in okoljem



Slika 4.8 Testno okolje v primeru analize zmogljivosti in preizkusa delovanja v primeru napada DoS.

iz prejšnjega podpoglavja je v okvirjih, ki jih na stikalo pošilja generator. Ne gre več za običajne pakete UDP s ciljnim naslovom iz območja multicast (torej običajen promet vrste multicast), pač pa za zaporedje sporočil IGMP Report in IGMP Leave za določeno (in vedno enako) skupino. Takšen promet ob dani implementaciji povzroči neprestano zaseganje in sproščanje pomnilnika, pa tudi neprestane sprehode po linearnem seznamu vrat, ki pripadajo določeni skupini.

4.3.2 Meritve

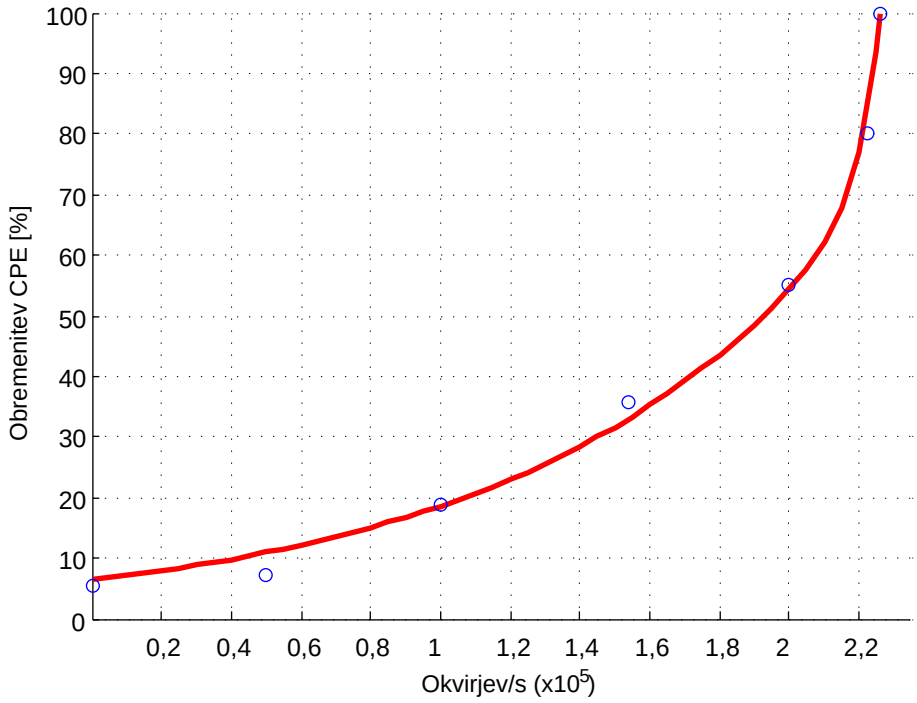
Simulacijo napada DoS z okvirji IGMP dobimo s hkratnim zagonom dveh instanc programa `packeth`, pri čemer ena instanca pošilja pakete IGMP Report, druga pa IGMP Leave. Paziti je potrebno, da so vsi paketi usmerjeni v isto skupino multicast. Ker obe instanci programa tečeta na istem generatorju, prihod okvirjev s teh dveh instanc pomeni nenehno prijavljanje in odjavljanje istih vrat v in iz neke skupine. Seveda hkratno delovanje dveh instanc na istem računalniku ne zagotavlja zaporednega prijavljanja ter odjavljanja vrat v in iz skupine, kar bi bilo idealno, se pa temu lahko približamo - zopet s podajšanim trajanjem poizkusa in povprečenjem rezultatov prek 60 sekundne časovne periode.

Obremenjenost stikala tudi v tem primeru merimo posredno - z zapisovanjem obremenjenosti CPE v datoteko in kasnejšo obdelavo, ki je enaka tisti iz prejšnjega podpoglavja.

4.3.3 Rezultati in analiza

Rezultate meritev prikazuje slika 4.9. Kot vidimo, je skupno število okvirjev, ko je sistem dosegel 100% zasedenost, približno 226000, kar je precej več (slika 4.6), kot v primeru sistema, ki ni uporabljal modula s podporo vohljanju IGMP. To pomeni, da gre razmerje med zahtevnostjo preklapljanja in poseganja v tabelo multicast - podobno kot v primeru običajnega prometa - znova v prid uporabi vohljanja IGMP.

Simulacija napada, predstavljena v tem podpoglavju, pokaže, da napad s pošiljanjem velike količine sporočil IGMP ni nevarnejši od napada na stikalo z običajnim prometom. Prav tako se je izkazalo, da je novi modul s podporo vohljanju IGMP tudi v primeru napada s prometom IGMP celo robustnejši od izvirnika, saj dodana funkcionalnost pakete IGMP blokira, medtem ko ima izvirni modul težave z intenzivnim preklapljanjem okvirjev.



Slika 4.9 Obremenitev CPE stikala v primeru napada z različnimi števili sporočil IGMP

Ta preizkus je bil obenem preizkus pravilnosti upravljanja s pomnilnikom, saj daljše trajanje preizkusa ni pokazalo zmanjševanje količine prostega pomnilnika. To kaže, da se vsak zaseženi segment pomnilnika na neki točki tudi sprosti. Za konec naj omenim, da zaradi omejenega števila vrat na stikalih ne more priti do neskončnega podaljševanja linearnega seznama v tabeli multicast, zato tudi ob neprestanem pošiljanju sporočil IGMP Report na vsa razpoložljiva vrata ne more priti do sesutja sistema zaradi pomanjkanja razpoložljivega pomnilnika.

5 Sklepne ugotovitve

Diplomsko delo predstavlja nadgradnjo stikala, ki temelji na operacijskem sistemu Linux, z izvedbo vohljanja IGMP in analizo delovanja tako nadgrajenega stikala. V uvodnih poglavjih sem opravil pregled relevantnih omrežnih slojev in pregled načinov pošiljanja enemu (unicast), vsem (broadcast) ter skupini uporabnikov (multicast). V zvezi s tem sem predstavil protokole, ki so povezani s pošiljanjem podatkov v omrežjih Ethernet. Poleg tega sem opisal delovanje stikal, ki predstavljajo osrednji omrežni element v takšnih omrežjih. Značilnost stikal je, da prometa, ki je namenjen skupini uporabnikov (multicast) ne ločujejo od prometa, ki je namenjen vsem uporabnikom (broadcast). Tako sem opisal:

- protokolna sklada OSI in TCP/IP,
- delovanje stikal in ozadje, zakaj stikala ne ločujejo med prometom broadcast in multicast,
- posebnosti prometa multicast in

- protokol za upravljanje skupin IGMP, in sicer vse tri različice, ki so trenutno standardizirane, na kratko pa tudi sorodna protokola CGMP in MLD.

V nadaljevanju sem opisal zamisel o prenosu dela funkcionalnosti IGMP, ki sicer deluje na tretjem omrežnem sloju, na drugi omrežni sloj. Zamisel temelji na predpostavki, da bi prenos koncepta skupin na stikala zmanjšal količino (nepotrebnega) prometa v omrežju, saj bi slednje stikalom omogočilo, da s prometom, ki je namenjen le skupini uporabnikov, ne poplavljajo ostalih delov omrežja. To je mogoče doseči tako, da stikalu dodamo funkcionalnost pregledovanja vhodnega prometa, kar odpira možnost pametnejšega preklapljanja prometa. Takšno pregledovanje okvirjev imenujemo vohljanje IGMP.

V grobem vohljanje IGMP pomeni pregledovanje vstopnih okvirjev in hranjenje podatkov o stanju omrežja v posebni tabeli multicast. Stikalo zato ob vsakem prispelem okvirju ugotavlja, ali gre za:

- okvir multicast, ki nosi informacije v skladu s protokolom IGMP ali
- običajen okvir vrste unicast, broadcast oziroma multicast.

V skladu z ugotovitvami o pregledanem okvirju nato sistem:

- posodobi tabelo multicast, ali pa
- okvir preklopi na ustrezna vrata oziroma skupino vrat.

Takšno pregledovanje okvirjev poleg koristi za omrežje pomeni tudi nasprotujoča si učinka na delovanje stikala: pregledovanje za stikalo pomeni dodatno delo in niža učinkovitost, zmanjšano število preklapljanj pa stikalo razbremenjuje, kar pomeni višjo učinkovitost.

Osrednji del diplomskega dela tako predstavljata poglavji o praktični izvedbi in analizi stikala. V okviru poglavja o praktični izvedbi sem opisal algoritem vohljanja, obenem pa na kratko tudi opisal izvorno kodo programske opreme. Izvorna koda obsega dva večja sklopa - prvi sklop predstavlja spremembe in dopolnitve jedra operacijskega sistema Linux, drugi sklop pa dopolnitve programa `brctl`, ki se uporablja kot osnovno orodje za upravljanje stikala. Iz opisa izvorne kode je očitno, da je rešitev, kakršno predstavljam v tem diplomskem delu, predvsem prilagojena protokoloma IGMP verzije 1 in 2, deloma pa tudi verziji 3.

V poglavju o preizkusu delovanja in analizi stikala so razvrščeni najprej osnovni preizkusi delovanja vseh delov programske opreme, nato pa še analiza učinkovitosti delovanja v primeru običajnega prometa multicast ter prometa, ki ga sestavljajo izmenjujoči

si okvirji IGMP Report in IGMP Leave. Slednje predstavlja simulacijo morebitnega napada vrste DoS na stikalo, temelji pa na razmisleku, da stikalo ob prispetju takšnih okvirjev neprestano spreminja tabelo multicast, kar pomeni veliko število zasegov in sprostitev pomnilnika.

V okviru analize je bilo potrebno najprej postaviti okolje, ki bi omogočalo dovolj dobro simulacijo sistemov v realnih razmerah. Tako sem ob pomoči mentorjev postavil okolje, ki bi lahko predstavljalo manjše lokalno omrežje. Obsegalo je stikalo s programsko opremo, opisano v prejšnjih poglavih, nanj pa je bilo priključenih še osem računalnikov. En izmed teh računalnikov je predstavljal generator prometa, ostalih sedem pa je bilo (potencialnih) prejemnikov.

Predpostavka, ki sem jo podal v uvodnih poglavjih, je bila, da vohljanje največ prispeva k zmanjšanju nepotrebnega prometa v omrežjih, ki jih povezujejo stikala Ethernet. Hkrati s to pridobitvijo za omrežja pa se pojavlja vprašanje, kaj vohljanje pomeni za sama stikala. Stikala, ki izvajajo vohljanje, morajo namreč opravljati dodatno delo, kar bi lahko pomenilo degradacijo delovanja stikal. Obenem pa v primeru, ko vsa vrata na stikalu niso naročnik prometa, namenjenega določeni skupini, odpade nepotrebno preklapljanje prometa na takšna vrata. Pri iskanju odgovora o upravičenosti implementacije vohljanja na stikalu gre torej za vprašanje razmerja med zahtevnostjo preklapljanja in zahtevnostjo vohljanja.

Rezultati analize so pokazali, da je vohljanje upravičeno v trenutku, ko vsaj eno izmed vrat ni prejemnik prometa, namenjenega vsaj eni skupini, ki se pojavlja na stikalu. Zahtevnost preklapljanja je pri programskih stikalih namreč v primerjavi z zahtevnostjo vohljanja bistveno večja. Vohljanje IGMP torej ne pomeni samo zmanjšanje obremenitve omrežja - v večini primerov pomeni tudi izboljšanje učinkovitosti delovanja samega stikala.

Podobno se je uporaba vohljanja kot pozitivna izkazala v primeru napada z okvirji IGMP. Ker je število vrat na vsakem stikalu omejeno, ne more priti do neskončnega podaljševanja linearnega seznama, ki je del tabele multicast, kar pomeni, da ne more priti do sesutja sistema z neprestanim prijavljanjem vrat v različne skupine. Tudi tu se izkaže prednost selektivnega obravnavanja prispelih okvirjev - okvirji, ki nosijo sporočila IGMP, so namreč prav tako okvirji vrste multicast, ki jih običajno stikalo razpošlje vsem. To pomeni, da znova pridemo do tehtanja, vendar gre tu za tehtanje med zahtevnostjo preklapljanja na vsa vrata in zahtevnostjo poseganja v tabelo multicast. Izkaže se, da

je zahtevnost preklapljanja znova opazno večja. Tako stikalo, ki podpira funkcionalost vohljanja IGMP točko zloma doseže pri več kot šestkrat večji obremenitvi kot običajno stikalo.

5.1 Predlogi za nadaljnje delo

Možnosti nadaljnjih izboljšav je kar nekaj. Najočitnejša je boljša, oziroma popolna podpora standardu IGMPv3, ki ga večina praktičnih aplikacij v tem trenutku sicer še ne podpira v celoti, ga pa podpirajo vsi sodobni operacijski sistemi, zato bo v prihodnje po vsej verjetnosti širšo podporo dobil tudi na aplikativni ravni.

Prav tako bi bilo mogoče izboljšati orodje `brctl` z možnostjo boljšega pregledovanja stanja stikala ali drugimi dodatki. Trenutno stikalo ponuja izpis stanja po vseh skupinah, kar ni najbolj pregledno. Konkretne potrebe bi verjetno še najbolje pokazala praktična uporaba opisane izvedbe. Prav tako standard [7] predvideva možnost samodejne zaznave usmerjevalniških vrat, kar je v tej izvedbi rešeno z uporabo posebnega ukaza `brctl setrouter`. Dodajanje in odvzemanje usmerjevalniških vrat sedaj potrebuje aktivno poseganje administratorja, čemur se je mogoče izogniti.

Izboljšave bi bile mogoče tudi na področju meritev in analiz. Dodelati bi bilo potrebno način merjenja obremenitve CPE, poleg tega pa morda vpeljati merjenje dejanske prepustnosti sistema. Takšna meritev bi bila bolj neposredna kot opisane meritve obremenitve CPE in zato za praktične primere bolj uporabna. Posredno merjenje bi tako ohranili le v primeru, ko stikalo prometa ne prepušča nikamor.

Nadgraditi bi bilo mogoče tudi testno okolje: namesto generatorja v obliki običajnega osebnega računalnika in programa `packeth`, bi bila boljša uporaba posebnih strojnih generatorjev prometa, na primer generatorjev Ixia. Z njimi bi premostili težavo neenakomernega generiranja prometa. Žal so tovrstni generatorji zelo dragi, po drugi strani pa tudi testna okolja, ki sem jih opisal v poglavjih o meritvah, lahko generirajo zadovoljivo enakomeren promet, paziti je le potrebno, da generatorjev ne obremenimo do skrajnih meja njihovih zmožnosti.

Na področju meritev kakovosti programske opreme bi bilo koristno raziskati možnost uporabe posebnih orodij za profiliranje, ki omogočajo natančno merjenje porabe, oziroma porazdelitve porabe časa po posameznih segmentih programske opreme.

Opravljena izvedba in meritve so po mojem mnenju sicer pokazale, da je vohljanje

IGMP lahko dobra možnost povečanja prepustnosti omrežij Ethernet in tudi način za povečanje učinkovitosti programsko izvedenih stikal, vseeno pa je ostalo odprtih še precej možnosti za nadaljnje delo tako na področju izvedbe kot analize.

LITERATURA

[1] S. Deering, Multicast routing in a datagram internetwork, 1991.

[2] D. Kodek, Arhitektura računalniških sistemov, Bi-tim, 2000.

[3] S. Deering, Host Extensions for IP Multicasting, 1989.
url: http://tools.ietf.org/html/rfc1112

[4] B. Fenner, Internet Group Management Protocol, Version 2, 1997.
url: http://tools.ietf.org/html/rfc2236

[5] B. Cain, S. Deering, Internet Group Management Protocol, Version 3, 2002.
url: http://tools.ietf.org/html/rfc3376

[6] R. Vida, L. Costa, Multicast Listener Discovery Version 2 (MLDv2) for IPv6, 2004.
url: http://tools.ietf.org/html/rfc3810

[7] M. Christensen, K. Kimball, F. Solensky, IGMP and MLD Snooping Switches Considerations, 2004.
url: http://tools.ietf.org/html/rfc4541

[8] B. Haberman, J. Martin, Multicast Router Discovery, RFC4286, 2005.
url: http://tools.ietf.org/html/rfc4286

[9] T. N. Cisco, IGMP and MLD Snooping Switches Considerations, 2008.
url: http://www.cisco.com/application/pdf/paws/10559/22.pdf

[10] C. Benvenuti, Understanding Linux Network Internals, O'Reilly Media Inc., 2006.

[11] I. Kononenko, Načrtovanje podatkovnih struktur in algoritmov, Založba FE in FRI, 1996.