

TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond
Automaatika instituut

ISS40LT
Siim Pöder 020616

TÖÖRIIST VÕRGULIIKLUSE KUJUNDAMISE
HINDAMISEKS
Bakalaureusetöö

Juhendaja: Rein Paluoja
Dotsent

Tallinn 2007

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

(kuupäev)

(allkiri)

Lõputöö ülesanne

Lõputöö teema eesti keeles: Võrguliikluse kujundamise hindamise tööriist

Lõputöö teema inglise keeles: Tool for testing traffic shaping solutions

Teema päritolu: Autori huvi

Lõputöö eesmärgid: Disainida ning luua tööriist, mis lihtsustaks võrguliikluse kujundamise lahenduste hindamiseks vajalike katsete läbiviimist ning võimaldaks katsetulemuste kesksel salvestamist.

Oodatavad tulemused: Tööriist on loodud ning abistab võrguliikluse kujundamise lahenduste hindamist.

Lähtetingimused: Võrguliikluse kujundamise lahenduste hindamiseks efektiivselt kasutatavat tööriista ei eksisteeri.

Lahendatavad küsimused: Võrguliikluse kujundamise lahenduste hindamisel tekkivate peamiste probleemide leidmine, tööriista disain lähtuvalt leitud probleemidest.

Täiendavad nõuded:

Kuupäev:

Üliõpilane: Siim Pöder

Allkiri:

Juhendaja: Rein Paluoja

Allkiri:

Kinnitaja:

Allkiri:

TÖÖRIIST VÕRGULIIKLUSE KUJUNDAMISE HINDAMISEKS

Annotatsioon

Käesolev töö kirjeldab võrguliikluse kujundamise lahenduste hindamise tööriista disaini ja kasutamist. Töö annab ülevaate Interneti ja tema protokollide olemusest ning tema kõige üldisematest tööpõhimõtetest, kirjeldab mõned võrguliikluse kujundamise lahenduste hindamise probleemid ning kirjeldab tööriista, mis võimaldab need probleemid ületada. Lõpuks on toodud näide kirjeldatud tööriista kasutamisest, mida on analüüsitud.

TOOL FOR TESTING TRAFFIC SHAPING SOLUTIONS

Abstract

This paper describes design and use of an application for testing traffic shaping solutions. It gives an overview of the Internet, its protocols and the most general principles it works by; describes a few problems with testing traffic shaping solutions and an application that allows one to overcome these problems. The paper ends with an example use case of the designed application and analysis of that use case.

Kasutatud terminid

Eestikeelne termin	Inglisekeelne vaste, selgitus
Võrguliikluse kujundamine	Traffic Shaping. Pakettide ümberjärjestamine võrguliidese puhvris protokollide päiste ning väliste reeglite põhjal
Viide	Round Trip Time. Töös on kasutatud iseloomustamiseks aega, mis kulub saadetud paketi vastuseks saadetud paketi saabumiseni
Ping	Ping. Tööriist viite mõõtmiseks ICMP (Internet Control Message Protocol) „Echo“ ja „Echo Reply“ sõnumite abil. Teinekord kasutatakse ka tähenduses „viide“.
Andmete edastamise kiirus	Data Throughput. Töös on kasutatud iseloomustamiseks ajaühikus edastatud andmete hulka
Teenuse tüüp	Type of Service. Soovitud pakettide erikohtlemise viis teenuse kvaliteedi tõstmiseks
Portimine	Porting. Rakenduse arendus, tema tööle saamiseks varem mitte toetatud operatsioonisüsteemides
Skoop	Scope. Tarkvara nõuete osa, nimetab tingimused, mille pihul funktsionaalsed nõuded täidetud olema peavad
Parsimine	Parsing. Andmete sisselugemine eriliselt, tavaliselt inimloetavalt tekstikujult
Marsruuter	Router. Korraga mitmes arvutivõrgus asuv seade, mis seob võrgud kokku sõnumeid vahendades
Võrgulüüs	Gateway. Võrgu vaikimisi marsruuter, mille kaudu käib suhtlus kõigi ülejäänud võrkudega

Sisukord

Annotatsioon.....	1
Abstract.....	2
Kasutatud terminid.....	3
Sissejuhatus.....	6
Võrguliikluse kujundamine.....	7
Interneti olemus.....	7
Internetiühendus.....	8
Konkurss võrguressurssidele.....	9
Interneti protokollid.....	10
IP protokoll.....	10
Transporditaseme protokollid.....	11
Teenuse tüüp.....	12
Tööriist.....	15
Võrguliikluse kujundamise lahenduse hindamine.....	15
Tööriista eesmärk.....	16
Tööriista skoop.....	16
Riist- ja tarkvaraplatvorm.....	17
Disain.....	17
Komponentide ülevaade.....	18
SeireServer.....	18
Paketigeneraator.....	19
Klient.....	20
Komponentidevaheline suhtlusprotokoll.....	21
Protokolli sõnumivahetuse näide.....	21
Seadistused.....	22
SeireServer.....	23
Süntaks.....	23
Semantika.....	23
Näide.....	24
PaketiGeneraator.....	25
Süntaks.....	25
Semantika.....	25
Näide.....	26
Katse ja analüüs.....	28
Katse kirjeldus.....	28
Võrgu seadistus.....	28
Tööriista seadistus.....	29
Tööriista kasutamine.....	30
Lahenduste katsetamine.....	30
Katse analüüs.....	33
Kokkuvõte.....	35
Resume.....	36
Kasutatud kirjandus.....	37
LISA 1 – katse seaded ning skriptid.....	38

Jooniste nimekiri

Joonis 1: Lihtsustatult kujutatud sõnumi teekond Internetis.....	8
Joonis 2: IP protokollis päis.....	10
Joonis 3: Sõnum IP protokollis.....	11
Joonis 4: TCP protokollis päis.....	12
Joonis 5: Sõnum TCP/IP protokollis.....	12
Joonis 6: Tööriista komponendid võrgutopoloogias.....	18
Joonis 7: Komponentidevahelise suhtlusprotokollis sõnum.....	21
Joonis 8: Katsetulemuste vorm (C kood).....	21
Joonis 9: SeireServer-i seadistuse definitsioon (XSD).....	23
Joonis 10: SeireServer-i seadistuse näide.....	24
Joonis 11: PaketiGeneraatori seadistuse definitsioon (XSD).....	25
Joonis 12: PaketiGeneraatori seadistuse näide.....	27
Joonis 13: Katse tulemused ilma võrguliikluse kujundamiseta.....	30
Joonis 14: Katse tulemused piirates Internetiühendust 200KB/s.....	31
Joonis 15: Katse tulemused prioritseerimisega.....	32
Joonis 16: Katse tulemused prioritseerimise ja Internetiühenduse piiramisega.....	33
Joonis 17: Katse SeireServer-i seadistus.....	38
Joonis 18: Katse esimese PaketiGeneraatori seadistus.....	39
Joonis 19: Katse teise PaketiGeneraatori seadistus.....	40
Joonis 20: Katse käivitamise skript.....	40
Joonis 21: Katse tulemuste salvestamise skript.....	41
Joonis 22: Katse tulemuste visualiseerimise skript.....	41

Tabelite nimekiri

Tabel 1: IP protokollis teenuse tüübid.....	13
Tabel 2: Komponentidevahelise suhtlusprotokollis sõnumi tüüp.....	21
Tabel 3: Komponentidevahelise suhtluse näide.....	22

Sissejuhatus

Käesoleva töö eesmärgiks on efektiivse rakenduse loomine hindamaks võrguliikluse kujundamise lahenduste toimivust. Töö jaotub kolme ossa:

Esimene osa annab lühikese ülevaate Interneti toimimise põhimõtetest ning Internetis kasutatavatest protokollidest. Ta tutvustab võrguliikluse kujundamise üldist ideed ning probleeme, mida võrguliikluse kujundamisega leevendada püütakse.

Teises osas on kirjeldatud võrguliikluse kujundamise lahenduste hindamise mõned probleemid, millega autor on kokku puutunud ning on loodud tööriist, mis võimaldab neid probleeme vältides efektiivsemat võrguliikluse kujundamise lahenduste hindamist.

Kolmandas osas on toodud näited loodud tööriista kasutamisest ning analüüsitud tööriista omadusi.

Loodud tööriist ei ole uus oma funktsionaalsuse poolest – töö eesmärgiks on pigem leevendada raskusi, mida autor on kogunud võrguliikluse kujundamise lahenduste hindamisel kasutades üldotstarbelisi tööriistu. Tööriist on oma rakenduselt väga spetsiifiline ning talle ei ole ette näha suurt kasutuspopulaarsust.

Võrguliikluse kujundamine

Et mõista võrguliikluse kujundamist, tuleks enne mõista üht-teist Interneti olemuse ning tööpõhimõtete kohta. See peatükk kirjeldab lühidalt Interneti füüsilist ehitust, ühe keskmise „tavalise“ Internetiühenduse omadusi, selgitab kuidas võrguliikluse kujundamine kasulik olla võib ning annab arusaama, kuidas võrguliikluse kujundamine toimub.

Interneti olemus

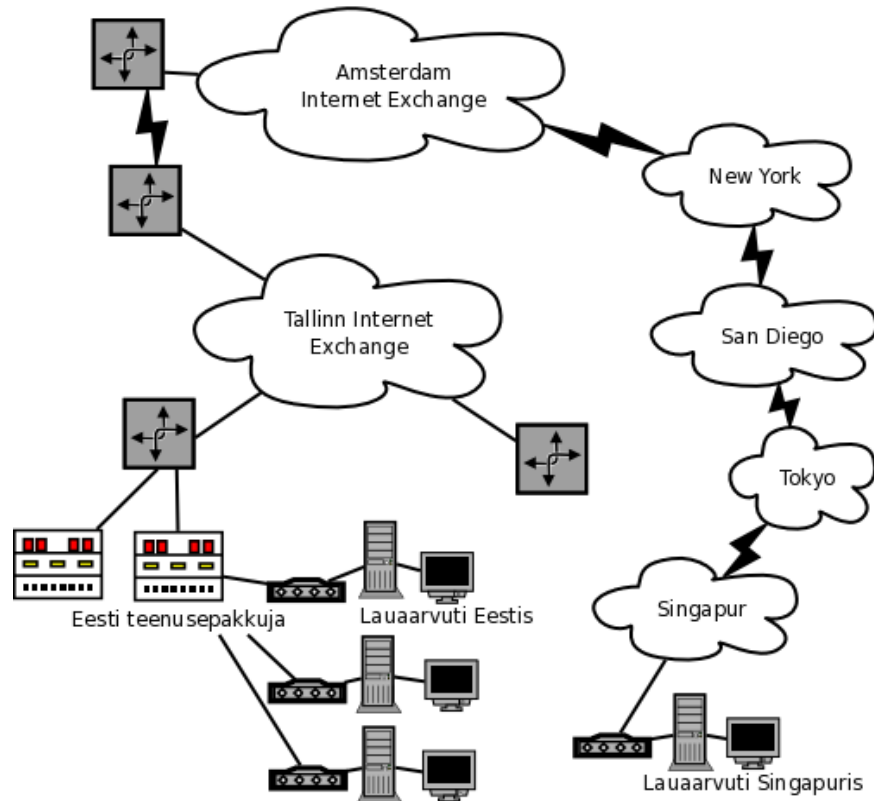
Internet on ülemaailmne arvutivõrkude võrk. Kui internet väikese tähega tähistab igasugust arvutivõrkude-vahelist võrku, siis Internet suure tähega on üks konkreetne võrk, mis ulatub praktiliselt igasse maailma otsa. Andmaks ülevaadet Interneti ehitusest, võib jaotada tema ühendused tinglikult kolmeks tasemeks.

Kõige kõrgemal tasemel koosneb Internet mandrite-vahelistest ühendustest. Näiteks ühendavad merekaablid Euroopat, Ameerikat ning Aasiat. Need ühendused kuuluvad suurtele rahvusvahelistele telekommunikatsiooni-ettevõtetele[1].

Järgmise taseme moodustavad riikide- ja linnadevahelised sideliinid, mille ehitamine ning haldamine on jõukohane peale rahvusvaheliste ettevõtete ka kohalikele Interneti-teenusepakkujatele.

Linnasisesed võrgud võivad olla ise ehitatud igal suuremal kohalikul teenusepakkujal. Omavahel suhtlevad nad nn Internet Exchange'des. Näiteks Tallinnas asub TIX (Tallinn Internet Exchange), kus vahetavad omavahel sõnumeid muuhulgas Elion, Starman ning Elisa[2].

Kuigi marsruudid Internetis on pidevas muutumises vastavalt teenusepakkujate-vahelistele kokkulepetele ning võrkudes ilmnevatele riketele või ülekoormustele, võib ette kujutada, kuidas Internetis üks sõnum liikuda võiks.



Joonis 1: Lihtsustatult kujutatud sõnumi teekond Internetis

Kui näiteks Elisa Internetiühendust kasutav inimene soovib saata IM (instant message) sõnumit tuttavale Singapuris, rändab tema sõnum esiteks läbi mõningatest Elisa võrgusisesest marsruuteritest kuni jõuab TIX-i, kus Elisa piirdemarsruuter annab sõnumi edasi Linxtelecomi piirdemarsruuterile. Sealt liigub sõnum edasi AMS-IX-i (Amsterdam Internet Exchange), kus sõnum taaskord edasi antakse. Niiviisi teenusepakkujalt teenusepakkujale ning võrgust võrku liikudes jõuab sõnum sammhaaval Dublinisse, ookeani põhjapidi New Yorki, läbi terve Põhja-Ameerika mandri San Diego'sse, sealt taaskord ookeani põhjapidi võib-olla näiteks Tokyosse, kust peamiselt maapealseid sideliine pidi Singapurini ja sealse tuttava arvutisse.

Internetiühendus

Kui suured magistraalvõrgud suudavad tihti edasi kanda kümneid või sadu gigabritte infot sekundis on lõppkasutajale kättesaadav andmete edastamise kiirus tavaliselt sadu või tuhandeid kordi väiksem.

Ühtlasi on pikem info liikumiseks kuluv aeg. Kui Prantsusmaalt USA-sse saadetud pakett läbib Atlandi Ookeani ligikaudu 50ms, siis autori lauarvutist esimese teenusepakkuja marsruuterini (ilmselt mõne kilomeetri kaugusele) liigub pakett 5ms.

Seega on üks tavaline Internetiühendus mitmeid suurusjärke aeglasem Interneti sisemuses toimivatest vaheühendustest. Just selle nn „viimase sammu“ ühenduse parameetrid piiravad Internetiühenduse toimivust kõige märgatavamalt.

Konkurss võrguressurssidele

Kui üht Internetiühendust kasutab korraga vaid üks rakendus on teenuse kvaliteet just nii hea, kui käesoleva ühendusega võimalik on. Probleem võib tekkida, kui ühe Internetiühenduse kaudu suhtlevad maailmaga mitu rakendust.

Näiteks on üks Eesti korrusmajades pakutav Internetiühendus juba tootetingimuste järgi jagatud mitmete elanike vahel[3]. See tähendab, et mitmest arvutist pärit sõnumeid on tarvis saata läbi ühe piiratud läbilaskevõimega ühenduse. Kui kaks arvutit püüavad korraga võimalikult kiiresti informatsiooni Internetti saata, kulub sama infohulga edastamiseks kaks korda rohkem aega, kui kuluks kummalgi saatjal üksinda ühenduse taga olles.

Leidub rakendusi, mis püüavad „agressiivselt“ hõivata suurt osa ühenduse läbilaskevõimest: heaks näiteks on failiedastustarkvara BitTorrent, mis loob mitmeid võrdlemisi aeglaseid ühendusi paljude masinatega kõikjal Internetis ning püüab nende arvuga saavutada suurt edastuskiirust[4]. Kui korraga töötavad BitTorrent ja veebibrauser, on brauserit käsitsedes selgesti tajutav suur viide lehtede avanemisel.

Kui BitTorrent'i andmeedastuse mõnesekundiline viibimine ei põhjusta mingeid probleeme, siis veebi sirvimise „aeglus“ mõjub kasutajale ilmselgelt häirivalt. Kuna veebi sirvimisel toimub lehtede laadimine hooti, on mõeldav, et uue lehe laadimise ajaks võiks failiedastusele vähem võrguressurssi eraldada. Failiedastusele eraldatud ressursse piirates saaks veebileht laetud võimalikult kiiresti. Selle jaoks ongi kasutusel võrguliikluse kujundamine. Eristades sobivate parameetrite põhjal erinevaid andmevoogusid ning neid erinevalt koheldes püüatakse parendada teenuse kvaliteeti.

Interneti protokollid

Võrguliikluse kujundamise realiseerimiseks on oluline ära tunda erinevat kohtlemist vajavad andmevood. Äratundmise mehhanismi mõistmiseks on oluline tunda Interneti protokolle.

IP protokoll

Arvutite omavahelise suhtlemise vaikumisi standardiks Internetis on IP protokoll. Tema kohaselt tuleb iga Internetis liikuva sõnumi ette lisada IP päis, milles sisalduva info abil leitakse tee läbi võrgu sihtkohta ning sealt tagasi.

Kõige olulisem osa IP protokollist on IP aadress, mis on 32-bitine number ning mida tavaliselt märgitakse üles 4 punktidega eraldatud arvuna 0-255, näiteks 127.0.0.1 (see konkreetne aadress tähistab IP protokollis kohaselt „kohalikku masinat“, st aadressile 127.0.0.1 sõnumi adresseerimine on kahtlemata sümptomaatiline mingisugusele kliinilisele psühhoosile).

Igas IP päises on kaks IP aadressi: saatja aadress ning vastuvõtja aadress. Vastuvõtja aadressi järgi oskavad marsruuterid sõnumit õiges suunas edasi saata, saatja aadressi saab vastuvõtja kasutada sõnumile vastamiseks.

IP päises on peale aadresside veel terve hulk kohustuslikku informatsiooni. Tervikliku IP päise struktuur on järgmine:

Nihe	Bitid 0-3	3-7	8-15	16-18	19-31
0 bitti	IP versioon	Päise pikkus	Teenuse tüüp	Sõnumi pikkus	
32	Identifikaator			Lipud	Fragmendi nihe
64	Eluiga		Protokoll	Päise kontrollsumma	
96	Saatja aadress				
128	Vastuvõtja aadress				
160	Sätted (valikulised)				

Joonis 2: IP protokollis päis

Päise täpne kirjeldus ning väljade täpne tähendus on ära toodud IETF rfc0791-s[5].

Seega, et Interneti kaudu saata sõnum „Tere hommikust“ tuleks ta vormistada järgnevalt:

IP päis sobiva saatja, vastuvõtja ning muude väljadega	Tekst „Tere hommikust“
--	------------------------

Joonis 3: Sõnum IP protokollis

Transporditaseme protokollid

Nagu selgus Interneti topoloogia kirjeldusest liigub iga pakett (nii nimetatakse üldiselt IP protokollis sõnumit) teel oma sihtkohta läbi paljude marsruuterite ning sideühenduste. Seetõttu sõltub paketi kohalejõudmine paljude marsruuterite tehnilisest seisundist, koormusest ning seadistusest. Pole ebatavaline, et pakettide järjekord teekonna jooksul muutub, pakettides ilmnevad vead või paketid kaduma lähevad.

Nimetatud (ja ka teiste) probleemide tõttu kasutatakse Internetis sõnumite saatmiseks lisaks IP protokollile ka nn transporditaseme protokolle. Selle asemel, et kohe IP protokollis päise järel sõnum saata, lisatakse IP päisele tavaliselt veel TCP protokollis päis.

TCP protokollis päises on kõige silmapaistvamateks väljadeks (mitte tehnilisest, vaid kasutaja aspektist) saatja ning vastuvõtja pordid. Kui luua metafoor IP aadressist kui tavalisest tänava-aadressist, siis võiks TCP porti vaadata kui elaniku nime – seda kasutatakse, et Interneti kaudu võiks ühes arvutis sõnumeid vahetada korraga mitu rakendust.

Mõningatel TCP portidel on ka välja kujunenud ka nn standardsed rakendused. Näiteks veebilehti jagavad HTTP serverid kasutavad suhtlemiseks TCP porti 80, faile jagavad FTP serverid TCP porti 21 ning e-kirju edastavad SMTP serverid TCP porti 25.

Tervikliku TCP päise struktuur on järgmine:

TALLINNA TEHNIKAÜLIKOOL

Nihe	Bitid 0-3	4-7	8-15	16-31
0	Saatja port			Vastuvõtja port
32	Järjekorranumber			
64	Kinnitusnumber			
96	Sõnumi nihe	Reserve eritud	Lipud	Aken
128	Kontrollsumma			Pakilisuse viit
160	Sätted (valikulised)			

Joonis 4: TCP protokollis päis

Päise täpne kirjeldus ning väljade täpne tähendus on ära toodud IETF rfc0793-s[6].

TCP asemel kasutatakse mõnikord ka UDP protokollis, millel on TCP-ga sarnased võimalused. UDP päis on lühem ning seda protokollis kasutatakse valdkondades, kus garanteeritud andmeedastus pole oluline.

Ehkki ka eelmises peatükis esitletud sõnumi struktuur on põhimõtteliselt õige, tuleks Interneti kaudu sõnumi „Tere hommikust“ saatmiseks ta tegelikult vormistada järgnevalt:

IP päis sobiva saatja, vastuvõtja ning muude väljadega	TCP päis sobivate portide ning muude väljadega	Tekst „Tere hommikust“
---	---	------------------------

Joonis 5: Sõnum TCP/IP protokollis

Teenuse tüüp

Võrguliikluse kujundamise eelduseks on võimekus tuvastada, millist teenust üks või teine sõnum vajab. See võimekus on teatud mõttes Interneti sisse ehitatud. Nimelt võib sõnumi saatja määrata soovitud teenuse tüübi IP protokollis vastava välja abil:

TALLINNA TEHNIKAÜLIKOOL

Tabel 1: IP protokollide teenuse tüübid

Teenuse tüübi identifikaator	Teenuse tüübi tähendus
1000	minimeerida viide
0100	maksimeerida läbilase
0010	maksimeerida kohalejõudmine
0001	minimeerida rahaline kulu
0000	tavaline teenus

Rakendus võib otsustada, millised kriteeriumid on tema jaoks tähtsad ning marsruuterid peaksid vastavalt sellele valima paketi jaoks sobivaima marsruudi. Kuid see on nii vaid teoorias – praktikas Interneti marsruuterid teenuse tüüpi marsruudi valikusse ei kaasa. Seetõttu kasutab enamik rakendusi vaikimisi teenuse tüüpi „tavaline teenus“ ning seetõttu on väli enamiku Interneti jaoks kasutu (kuigi eriotstarbelistes võrkudes võib ta kriitilist tähtsust omada).

Kuna teenuse tüübi väljast abi ei ole, otsustavad võrguliikluse kujundamise lahendused teenuse tüübi üle teiste protokollide päiste väljade alusel. Tüüpilised lahendused määravad teenuse tüübi sõnumite lähteadressi ja/või lähte- ning sihtportide järgi.

Enamik rakendusi kasutab võrguga suhtlemisel standardseid porte, mistõttu on võimalik paketti lähte- ja sihtportide järgi aru saada, mis rakendusega tegemist on[7]. Teenuse tüüp tuletatakse sel juhul juba rakenduse eeldatavast iseloomust – liikluskujundamise eeldab näiteks, et FTP protokollil puhul on oluline ainult läbilase. Rakendus ise sel juhul oma vajadusi määrata ei saa. Samuti pole võimalik teenuse tüüpi määrata tundmatutel portidel toimuva võrguliikluse jaoks ning on võimalus määrata vale teenuse tüüp ühendusele, mis lihtsalt „juhtub kasutama“ mõnd standardporti.

Lähteadressi saab kasutada erineva teenuse kvaliteedi määramiseks erinevatele Internetiühendust kasutavatele masinatele. Kui IP aadressid on alamvõrgus püsivalt määratud, on võimalik näiteks jagatud ühenduse puhul eelistada ühe masina pakette teisele. Kasutusjuhiks võib olla näiteks olukord, kus asutuse serverid ning tööjaamad asuvad ühe Internetiühenduse taga ning süsteemiadministraator soovib pigem, et asutuse

TALLINNA TEHNIKAÜLIKOOL

veebileht oleks partneritele kättesaadav, kui et sekretäridel oleks võimalus Internetist naljakaid videoid vaadata.

Tööriist

Võrguliikluse kujundamise lahenduse hindamine

Võrguliikluse kujundamise lahenduse hindamisel on autori kogemuse põhjal vaja teostada kolme liiki operatsioone:

1. Andmevoogude tekitamine – käivitada kujundatavast võrgust erinevaid rakendusi, et tekitada olukordi, kus lahendus saaks erinevatesse klassidesse kuuluvat võrguliiklust kujundada. Selle jaoks on peamiselt tarvis tõmmata käima HTTP allalaadimisi-üleslaadimisi, FTP-sessioone, audio- ja videostriime, telnet- või SSH-sessioone. Sõltuvalt võrguliikluse kujundamise lahendusest võib olla vajalik nende rakenduste käivitamine erinevatelt IP-aadressidelt. Mida mitmekülgsemat katsetamist on vaja, seda keerulisem on rakenduste orkestreerimine.
2. Võrguliikluse jälgimine – jälgida sobiva võrgulüli peal parasjagu huvi pakkuvat tüüpi võrguliikluse klasside mahtu, et hinnata erinevatele klasside liikluse mahtude määrasid. Üldotstarbelised võrguliikluse jälgimise tööriistad selle jaoks ei sobi, kuna nad töötavad korraga üheainsa filtriga ning erinevate liikluse klasside üheaegset jälgimist ei võimalda. Ühtlasi on raskusi üldotstaebeliste tööriistade väljundi tõlgendamise ja töötlemisega.
3. Viidete mõõtmine – käivitada *ping* või selle derivaat, et hinnata viidet huvi pakkuvat tüüpi pakettide ja nende vastuspakettide vahel. Kui jälgida on vaja viidet mitme erineva liikluse klassi jaoks eraldi, on tarvilik käivitada viidete mõõtmiseks mitu *ping*-i, misjuhul tuleb nende väljundeid ka korraga jälgida ning tõlgendada.

Nende operatsioonide käigus kogutud andmete põhjal on võimalik teha tähelepanekuid võrguressursside kasutamise efektiivsusest: Hinnata eraldatud läbilaskeribasid ning sõnumite viiteid erinevat tüüpi teenuste korraga töötamisel.

Tööriista eesmärk

Ilma ühtse süsteemita katsetuste tegemisel on autori hinnangul järgnevad puudused:

1. Ajastamine – käsitsi erinevatel masinatel erinevaid andmevoogusid plaani kohaselt käivitada ja peatada on tülikas ning ebatäpne.
2. Keerukus – käsitsi on keeruline, kui mitte võimatu, käivitada ja peatada kümneid andmevoogusid mitmel masinal ette planeeritud järjekorras.
3. Korratavus – käsitsi on raske üht katset mitu korda sarnasel viisil sooritada, mis on aga vigade taastekitamiseks või nende parandamise kinnitamiseks oluline.
4. Info vorm – tööriistade väljundeid saab küll automaatselt töötlemiseks ühtsustada, kuid see tegevus on tülikas.
5. Info talletamine – järjekordsete katsete tulemuste võrdlemiseks eelmiste katsetega tuleks kõik tulemused salvestada. Kui katsetamiseks on käigus mitu masinat, on tulemuste kokkukogumine tülikas.

Loodud tööriist peab ühendama võrguliikluse jälgimise, andmevoogude tekitamise ning viidete mõõtmise ühtsesse süsteemi ning adresseerimaks loetletud puudusi:

1. Võimaldama tsentraalset katsete juhtimist.
2. Võimaldama kirjeldada täpselt katse käik ning selle kirjelduse alusel katset korduvalt käivitada.
3. Koguma katse tulemused (viited ning mahud ajahetkede teljel) ning nad ühtsustatud vormis salvestama.

Tööriista skoop

Loodud tööriista kasutusskoobiks on väikse kuni keskmise suurusega (kuni mõnikümneid korruga aktiivset masinat) võrkude liikluse kujundamise erilahenduste hindamine. Tööriist ei taotle võimekust hinnata võrguliikluse organiseerimise efektiivsust ajakriitilistes süsteemides (reaalajasüsteemides). Ta on funktsionaalne hindamise abivahend.

Riist- ja tarkvaraplatvorm

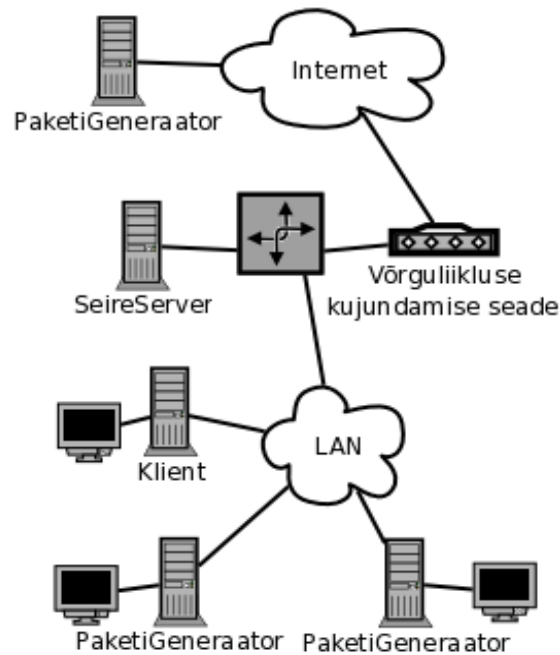
Tööriista sihtplatvormiks on Inteli x86 arhitektuuriga arvutid ning operatsioonisüsteem Linux. Platvormi valiku tingis tema kättesaadavus. Teistele platvormidele protimine ei ole antud töö skoobis, kuid disaini juures on seda võimalust silmas peetud.

Tööriista on programmeeriskeeleks on C++. Võimalikeks alternatiivideks oleks mõni interpreteeritav keel (ilmselt Python) või Java, mis pakuvad võrreldavat jõudlust, sobivaid teke ning süntaktilist mugavust. Kuna ükski keel kolmest ei oma teiste ees olulisi eeliseid, osutus valituks C++. Valiku põhjuseks on autori kõige suurem kogemus just selles keeles programmeerimises.

Tööriist kasutab peale standardsete C++ teekide veel libxml2 teeki XML seadistuste parsimiseks ning libpcap teeki võrguliikluse jälgimiseks. Mõlemad teegid on saadaval erinevatel platvormidel.

Disain

Tööriist koosneb kolmest komponendist: SeireServer, PaketiGeneraator ning Klient. Komponentideks jaotus tuleneb töö eesmärgi püstituses ära toodud võrguliikluse kujundamise lahenduse hindamiseks vajalikest operatsioonidest: SeireServer-i ülesandeks on liikluse jälgimine, PaketiGeneraator-i ülesandeks nii liiklusvoogude tekitamine kui ka viidete mõõtmine, Kliendi ülesanne ülejäänud komponentide töö kordineerimine.



Joonis 6: Tööriista komponendid võrgutopoloogias

Komponentide ülevaade

Järgnevad alampeatükid kirjeldavad SeireServer-i, PaketiGeneraator-i ning Kliendi ehitust. Nad annavad piisavalt täpse pildi, et mõista iga komponendi rolli ning üldist arhitektuuri, kuid ei lasku detailidesse komponendi täpse sisemise ehituse kohta.

SeireServer

SeireServer-i ülesanne on ettemääratud paketiühikute väärtustele vastavalt jagada võrguliiklus klassidesse ning pidada arvet kõigi määratud klasse läbivate pakettide suuruste kohta, mis katse käigus SeireServer-i võrguliideseid läbivad.

Vastavalt seadistatud, valmistub ta salvestama andmeid defineeritud võrguliikluse klasside kohta. Peale käivitamist võtab vastu kõik tema võrguliideseni jõudnud pakettid ning salvestab klassifitseerija suunamisel iga sobiva klassi andmekogu juurde paketi saabumise aja ning paketi suuruse. Eksperimendi lõppedes väljastab ta kogutud info.

Pakettide klassikuuluvuse kontroll toimub päise väärtuste ja oodatud päise väärtuste loogilise tehte alusel. Iga klassi ära tundmiseks saab määrata suvalise keerulisusega loogilise tehte võrdlusoperatsioonidest, kus võrreldakse paketi päises määratud

sügavusel paiknevat väärtust ette antud väärtuse või väärtuste vahemikuga.

SeireServer peab võrgutopoloogias paiknema sääraselt, et kõik võrku sisenevad või talt väljuvad paketid tema võrguliidestele kättesaadavad oleksid. See tähendab kas piirderuuteri ette-taha paigutamist või piirderuuteri ühe liidese peegeldamist SeireServer-i liidesele.

SeireServer-i peamised moodulid on:

- ServerInterface – Liides komponentidevaheliseks suhtluseks
- XMLConfigurator – Seadistusinfo parsimine
- Sniffer – võrguliikluse jälgija
- Classifier – paketi klassikuuluvuse tuvastaja
- DataStorage – kogutud info säilitamine väljastamiseks katse lõpus

Paketigeneraator

PaketiGeneraator-il on kaks ülesannet. Esimeseks ülesandeks on erinevate tunnuste ning mahtudega võrguliikluse genereerimine. Teiseks ülesandeks on *ping* kombel paketi ja vastuspaketi vahelise viite mõõtmine.

Ta loob ühendused temaga paaris oleva PaketiGeneraator-iga ning eksperimendi alates alustab nii võrguliikluse kui ka *pingide* genereerimist (ning *pingidele* vastamist). Talletanud *pingide* saatmise ning vastuse saamise ajad, väljastab ta need eksperimendi lõppedes.

PaketiGeneraator suudab luua ühendusi kasutades TCP ja UDP protokolle, kuid sisemine arhitektuur kasutab transporditasemena abstraktset transpordiklassi. See võimaldab transpordiprotokollide lisamist ilma suuremate muutusteta komponendi ehituses.

PaketiGeneraator-il on piiratud võimekus võrguliiklust tekitada väliste rakenduste abiga, käivitades regulaarselt ette antud käsurida.

TALLINNA TEHNIKAÜLIKOOL

PaketiGeneraator-eid peab eksperimendi jaoks olema üldiselt vähemalt kaks (see ei tähenda eraldi riistvara, vaid kaht masinat, kus tarkvara käia saaks). Üks peab asuma kohtvõrgust väljas ning mängima võrguliikluse kujundaja jaoks anonüümse Interneti serveri rolli ning teine (või ülejäänud) kohtvõrgus, mängides kohaliku kliendi rolli. Kohtvõrku võib olla sobilik paigutada mitu Paketigeneraatorit juhul, kui võrguliiklust kujundatakse kohtvõrgu masinate vahel erinevalt.

PaketiGeneraator-i peamised moodulid on:

- ServerInterface – Liides komponentidevaheliseks suhtluseks
- XMLConfigurator – Seadistusinfo parsimine
- ConnectionsHandler – Võrguühenduste loomine ning andmeedastus nende kaudu
- Transport – Abstraktne klass, mille realisatsioonid on tegelikud kasutatavad transpordiprotokollid.
- DataStorage – kogutud info säilitamine väljastamiseks katse lõpus

Klient

Kliendi ülesanne on vastavalt kasutaja soovile seadistada SeireServer ja PaketiGeneraator-id ning koguda neilt eksperimendi tulemused.

Ta seadistab ülejäänud komponendid vastavalt kasutaja nõuetele ning annab märku eksperimendi alustamiseks. Eksperimendi lõppedes kogub ülejäänud katse tulemused ning talletab need ühtlustatud vormis, valmis väliste tööriistadega visualiseerimiseks ning analüüsiks.

Võrgutopoloogiliselt on ainsaks kriteeriumiks ligipääs kõigile ülejäänud komponentidele, mis ilmselt paljudel juhtudel tähendab, et asukohaks on kohtvõrgus asetsev masin.

Klient kasutab vaid üht üldist moodulit:

- ServerInterface – Liides komponentidevaheliseks suhtluseks

Komponentidevaheline suhtlusprotokoll

Tööriista komponendid suhtlevad omavahel IP võrgus lihtsa protokolliga abil. Protokoll kasutab transporditasemena TCP ühendust, mis algatatakse alati Kliendist - SeireServer ja PaketiGeneraator võtavad vastu ühenduse, loevad käsu, annavad sellele täpselt ühe vastuse ning sulgevad ühenduse.

Protokolliga sõnumi vorming on lihtne:

Sõnumi tüüp (16 bitti)	Andmete pikkus (16 bitti)	Andmed...
------------------------	---------------------------	-----------

Joonis 7: Komponentidevahelise suhtlusprotokolliga sõnum

Käsu tüüp on defineeritud järgnevalt:

Tabel 2: Komponentidevahelise suhtlusprotokolliga sõnumi tüüp

Identifikaator	Tüübi nimi
1	SET_CONF
2	RUN
3	GET_RESULTS
4	RESULTS
5	ERROR
6	OK

Sõnumitüübi SET_CONF puhul on sõnumiga kaasas seadistusinfo komponendile, RESULTS puhul on kaasas katsetulemused: jada struct datapoints andmetüüpe:

```

struct dpoint {
    long value;
    timeval tv;
};

struct datapoints {
    int num_points;
    dpoint points[];
};

```

Joonis 8: Katsetulemuste vorm (C kood)

Teist tüüpi sõnumitega koos andmeid ei edastata.

Protokolli sõnumivahetuse näide

Tüüpiline protokollile vastav sõnumivahetus näeb välja nõnda (K – Klient, S – SeireServer):

Tabel 3: Komponentidevahelise suhtluse näide

Saatja	Saaja	Sõnumi tüüp	Andmete pikkus	Andmed	Kommentaar
K	S	SET_CONF	n	n baiti	Klient saadab seadustusinfo SeireServer-ile
S	K	OK	0	-	SeireServer kinnitab info kättesaamist
K	S	RUN	0	-	Klient annab märku katse alustamiseks
S	K	OK	0	-	SeireServer alustab katset
Möödub katse kestuses määratud aeg...					
K	S	GET_RESULTS	0	-	Klient küsib katsetulemusi
S	K	RESULTS	n	n baiti	SeireServer saadab mõõdetud liiklusklasside läbilasked

Samasugune suhtlus toimub iga katse juures ka Kliendi ja iga PaketiGeneraator-i vahel.

Seadistused

Tööriista komponentide seadustusinfo on XML vormingus, et ta oleks korraka nii inimloetav (ja kirjutatav) kui ka masinloetav. Ühtlasi jätab see võimaluse vajadusel seadistust standardisel viisil automaatselt genereerida.

Seadistuste XML skeem on kirjeldatud XSD (XML Schema Definition) vormingus, kuid tegelik skeem erineb veidi formaalsest kirjeldusest. Erinevused on välja toodud vaba tekstiga formaalse kirjelduse järel.

SeireServer

Süntaks

```
<!DOCTYPE SeireServer [
<!ELEMENT SeireServer (Experiment,TrafficClass*)>
<!ELEMENT Experiment (EMPTY)>
<!ELEMENT TrafficClass (And*,Or*,Match*)>
<!ELEMENT And (Or*,Match*)>
<!ELEMENT Or (And*,Match*)>
<!ELEMENT Match (EMPTY)>

<!ATTLIST Experiment duration CDATA #REQUIRED>
<!ATTLIST TrafficClass name CDATA #IMPLIED>
<!ATTLIST Match
type (is|isnot|isbetween|isnotbetween|isset|isnotset) #REQUIRED
size (char|short|long) #REQUIRED
offset CDATA #REQUIRED
value CDATA #REQUIRED
value2 CDATA "0x0" #IMPLIED>
]>
```

Joonis 9: SeireServer-i seadistuse definitsioon (XSD)

Sildi *Match* atribuut *value2* on nõutav atribuudi *type* väärtuste *isbetween* ning *isnotbetween* puhul.

Semantika

SeireServer on seadistuse väline silt ning temas sisaldub üks *Experiment* ning suvaline arv *TrafficClass* silte.

Experiment sildil on vaid üks atribuut *duration*, milles sisaldub katse kestus sekundites.

TrafficClass sildi sees paiknevad *And*, *Or* ning *Match* sildid, mis määravad sellesse klassi kuuluvatele pakettide omadused (*TrafficClass* laste tulemustele rakendatakse loogilist *and* funktsiooni); tühi *TrafficClass* tunnistab omaks kõik paketid. Seadistuse loetavuse huvides on võimalik määrata atribuut *name*, kirjeldamaks liikluse klassi.

And ning *Or* sildid võimaldavad vajadusel määrata keerulisema struktuuriga võrguliikluse klasse – nende sisse võib omakorda paigutada suvalisel hulgal *And*, *Or* või *Match* silte, mille tulemusele rakendatakse vastavalt loogilist *and* või *or* funktsiooni. Sel viisil saab luua võrdluste hierarhia.

Match silt *TrafficClass* klassimääratluse puu terminaalne leht, mis määratleb ühe kontrolloperatsiooni paketi päistel. Atribuudiga *offset* määratakse võrdlusoperatsiooni sügavus paketi päises, atribuudiga *size* võrdluse andmevälja suurus (*char* – 8 bitti, *short* – 16 bitti, *long* – 32 bitti), atribuudid *value* ning *value2* on võrdluses kasutatavad suurused.

is ning *isnot* võrdlevad *value*'t otse paketi päise väärtusega.

isbetween ning *isnotbetween* kontrollivad paketi päise väärtuse kuuluvust *value* ning *value2* vahelisse piirkonda.

isset ning *isnotset* tuvastavad, kas paketi päise väärtuses on püsti *value*'s määratud bitid

Näide

Antud näide kirjeldab 60-sekundilist eksperimenti, mille jooksul tehakse mõõtmisi kolme võrguliikluse klassi kohta: klass, mis hõlmab kõiki pakette; klass, mis hõlmab IP protokollki kapseldatud TCP pakette, mille lähteport on 10008; sarnane klass, kuid TCP lähteport piirkonnas 10005-10007.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SeireServer>
  <Experiment duration="60"/>
  <TrafficClass name="all">
  </TrafficClass>
  <TrafficClass name="IP TCP port 10008">
    <Match type="is" size="short" offset="12" value="0x0800"/>
    <Match type="is" size="char" offset="23" value="0x06"/>
    <Match type="is" size="short" offset="34" value="0x2718"/>
  </TrafficClass>
  <TrafficClass name="IP TCP port 10005-10007">
    <Match type="is" size="short" offset="12" value="0x0800"/>
    <Match type="is" size="char" offset="23" value="0x06"/>
    <Match type="isbetween" size="short" offset="34"
value="0x2715" value2="0x2717"/>
  </TrafficClass>
</SeireServer>
```

Joonis 10: SeireServer-i seadistuse näide

PaketiGeneraator

Süntaks

```

<!DOCTYPE PaketiGeneraator [
<!ELEMENT PaketiGeneraator (Experiment,Connection*,Exec*)>
<!ELEMENT Experiment (EMPTY)>
<!ELEMENT Connection (Transport,Volume*,RTT*,Receiver?)>
<!ELEMENT Transport (#CDATA)>
<!ELEMENT Volume (EMPTY)>
<!ELEMENT RTT (EMPTY)>
<!ELEMENT Exec (EMPTY)>
<!ELEMENT Receiver (EMPTY)>

<!ATTLIST Experiment duration CDATA #REQUIRED>
<!ATTLIST Transport type CDATA #REQUIRED>

<!ATTLIST Volume
datalen CDATA #REQUIRED
start CDATA #IMPLIED
interval CDATA #REQUIRED
end CDATA #IMPLIED>

<!ATTLIST RTT
role (record|mirror) #REQUIRED
start CDATA #IMPLIED
interval CDATA #REQUIRED
end CDATA #IMPLIED>

<!ATTLIST Exec
command CDATA #REQUIRED
start CDATA #IMPLIED
interval CDATA #REQUIRED
end CDATA #IMPLIED
>
]>

```

Joonis 11: PaketiGeneraatori seadistuse definitsioon (XSD)

Sildi *Transport* sees nõutavad sildid sõltuvad tema atribuudi *type* väärtusest ning on kirjeldatud sildi *Transport* semantika kirjelduse juures.

Semantika

PaketiGeneraator on seadistuse väline silt ning sisaldab ühte *Experiment* silti ning suvalist arvu *Connection* ning *Exec* silte.

Experiment sildil on vaid üks atribuut *duration*, milles sisaldub katse kestus sekundites.

Connection sildi sees on täpselt üks *Transport* silt, suvaline arv *Volume* ning *RTT* silte ning valikuliselt *Receiver* silt.

Transport sildil on atribuut *type*, mille väärtusest sõltub tema alamsiltide tähendus. Praegusel hetkel on implementeeritud *type TCP* ning *UDP* - mõlema alla võivad olla sildid *SIP* (lähteaddress), *DIP* (sihtaaddress), *sport* (lähteport) ning *dport* (sihtport). *TCP* puhul lisaks ka silt *role*, mille väärtuseks kas *server* või *client*, määramaks ühenduse algatajat ja vastuvõtjat.

Volume, *RTT* ning *Exec* sildid on oma loomult sarnased. Kõik nad vajavad üldiselt atribuute *start*, *interval* ning *end*, millega määratakse nende käivitumise algus, intervall ja lõpp (eksperimenti suhtes). Lisaks sellele vajab *Volume* atribuuti *datalen*, millega määratakse regulaarselt saadetava paketi suurus; *RTT* atribuuti *role*, mille väärtusega *record* või *mirror* määratakse, kas antud instants genereerib viite mõõtmiseks pakette ja salvestab tulemusi, või ainult „peegeldab“ neid vastaspoolele tagasi; *Exec* atribuuti *command*, millega antakse ette regulaarselt käivitav käsuriid.

RTT sildi atribuudi *role* väärtuse *mirror* korral pole vaja määrata ülejäänud atribuute.

Receiver silt on kasutamiseks „tühjade“ *TCP Transport*ide jaoks, et saadetud paketid operatsioonisüsteemilt vastu võetaks ning *TCP* ühenduse vastuvõtuaken vabaneks.

Näide

Antud näiteks toodud seadistus kirjeldab 10-sekundilist katset, mille käigus luuakse kaks küllatki suuremahulist ühendust (üks *UDP*, teine *TCP*) kohtvõrgu aadressi 192.168.10.1 pihta. Ühtlasi käivitatakse iga sekundi tagant käsuriid `„/usr/bin/wget http://ttu.ee/“`, mis laeb alla Tallinna Tehnikaülikooli veebi esilehe.

TALLINNA TEHNIKAÜLIKOOL

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<PaketiGeneraator>
  <Experiment duration="10"/>
  <Connection>
    <Transport type="tcp">
      <Role>client</Role><DIP>192.168.10.1</DIP>
      <DPort>10005</DPort><SPort>10005</SPort>
    </Transport>
    <Volume datalen="5000" start="2" interval="0.101" end="8"/>
  </Connection>
  <Connection>
    <Transport type="udp">
      <DIP>192.168.10.1</DIP>
      <DPort>10006</DPort><SPort>10006</SPort>
    </Transport>
    <Volume datalen="5000" start="4" interval="0.112" end="6"/>
  </Connection>
  <Exec cmd="/usr/bin/wget http://ttu.ee/" start="1" interval="1"
end="9"/>
</PaketiGeneraator>
```

Joonis 12: PaketiGeneraator-i seadistuse näide

Katse ja analüüs

Käesolev peatükk kirjeldab lihtsa võrguliikluse kujundamise ülesande lahendamist loodud tööriista abil ning analüüsib seda protsessi.

Katse kirjeldus

Katse eesmärk on luua näitlik võrguliikluse kujundamise lahendus, kus suuremahulised andmeedastused ei suurenda märgatavalt vähe andmeid edastava kõrvalise ühenduse pakettide viiteaegu.

Katse käigus käivitatakse 5 TCP ühendust (portidel 10006-10010), mille kaudu liigutatakse nii palju andmeid, kui Internetiühendus võimaldab. Ühtlasi käivitatakse üks prioriteetne TCP ühendus (pordil 10005), mille kaudu edastatakse vaid nii palju pakette, kui palju on tarvis viite mõõtmiseks.

Võrguliikluse kujundamise lahendus loetakse sobivaks, kui prioriteetse ühenduse pakettide viiteaeg püsib 20ms piires.

Võrgu seadistus

Katsest võtavad osa kolm masinat:

1. „OpenWrt“ on Broadcom BCM947XX protsessoril põhinev marsruuter, kus käib Linuxi 2.4 kernelit kasutav väike operatsioonisüsteem OpenWRT, mis võimaldab võrguliikluse kujundamist. Internetti on „OpenWrt“ ühendatud autori korrusmajas pakutava ühenduse kaudu, mille parameetrid on teadmata.
2. „windolik“ on Inteli x86 arhitektuuriga lauaarvuti, kus käib Linux 2.6 kernelit kasutav operatsioonisüsteem Debian ning mis on ühendatud „OpenWrt“ külge. Selles masinas käivad katse jooksul PaketiGeneraator ja SeireServer ning samuti käivitatakse siit Klient. SeireServer-i marsruuteri asemel lauaarvutisse paigutamise tingis marsruuteri piiratud operatsioonisüsteem, mille jaoks oleks keeruline SeireServerit kompileerida. SeireServer-i võrguliikluse jälgimise võimekust see mööndus ei mõjuta, kuna katsete jooksul kohtvõrgus ükski teine

masin välisvõrku ei kasuta.

3. „tsee-diees“ on Inteli x86 arhitektuuriga server, kus käib Linux 2.6 kernelit kasutatav operatsioonisüsteem Debian. Internetti on ta ühendatud Elisa ADSL ühendusega, mis võimaldab allalaadimiskiirust 5Mbit/s ning üleslaadimiskiirust 1Mbit/s. Selles masinas käib katse ajal PaketiGeneraator.

„windolik“ ning „tsee-diees“ vahel on katse päeval viiteaeg (paketi ja vastuspaketi vahel) 11ms ümbruses.

Tööriista seadistus

Võrguliikluse mahtu jälgib SeireServer „windolik“ peal. Tema seadistus defineerib jälgimiseks kaks võrguliikluse klassi:

1. TCP/IP paketid, mille lähteport on 10005.
2. TCP/IP paketid, mille lähteport on vahemikus 10006-10010.

Võrguliiklust genereerivaid ning pakettide viiteid mõõtvaid PaketiGeneraator-eid on katses kaks. Sisevõrgus asub paketi generaator „windolik“ peal. Tema seadistus defineerib 6 TCP ühendust, millest:

- Üks on pordil 10005, mida kasutatakse mõõtmaks viidet ühenduse sees iga 100ms tagant
- Üks on pordil 10006, mida kasutatakse mõõtmaks viidet ühenduse sees iga 100ms tagant ning suuremahuliseks andmeedastuseks
- Neli on portidel 10007-10010, mida kasutatakse suuremahuliseks andmeedastuseks

Välisvõrgus asub PaketiGeneraator „tsee-diees“ peal. Tema seadistus on paariline „windolik“ peal asuva PaketiGeneraator-i seadistusele, võttes vastu TCP ühendused, vastates viite mõõtmistele ning võttes vastu edastatud andmed.

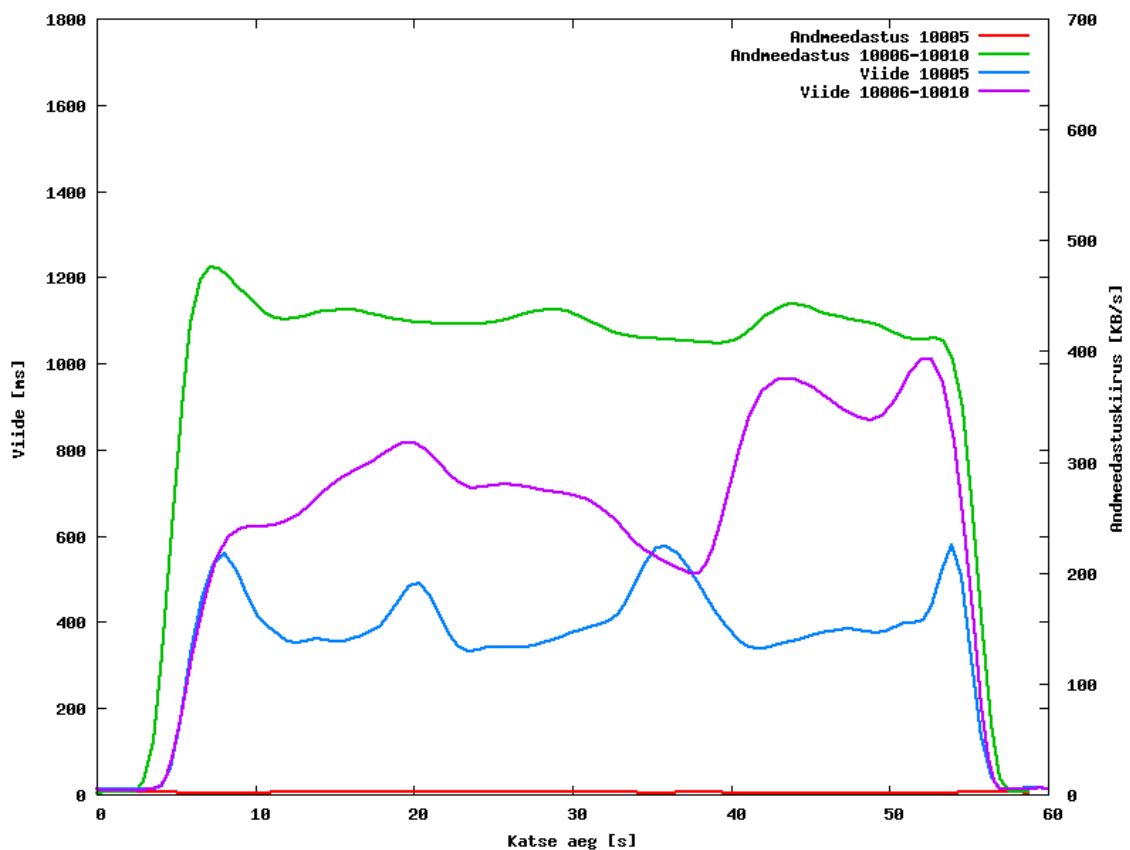
Seadistused on täispikkuses ära toodud töö lisas.

Tööriista kasutamine

Iga katse korduse juures toimub tööriista käivitamine ühel viisil: Komponentidesse laetakse seadistus ning käivitatakse katse. Peale katse möödumist loetakse komponentidest katsetulemused ning salvestatakse hilisemaks võrdluseks. Nende operatsioonide sooritamiseks on kasutatud kahte skripti, mis on ära toodud töö lisas.

Lahenduste katsetamine

Veendumaks, et suuremahulised andmeedastused mõjutavad kõrvaliste ühenduste viiteaegu on kõigepealt käivitatud tööriist ilma marsruuteri liikluse kujundamise võimekust sisse lülitamata. Katse tulemustest genereeritud graafik kinnitab eeldust, et suuremahulised andmeedastused põhjustavad viite kasvu ka kõrvalise ühenduse jaoks.

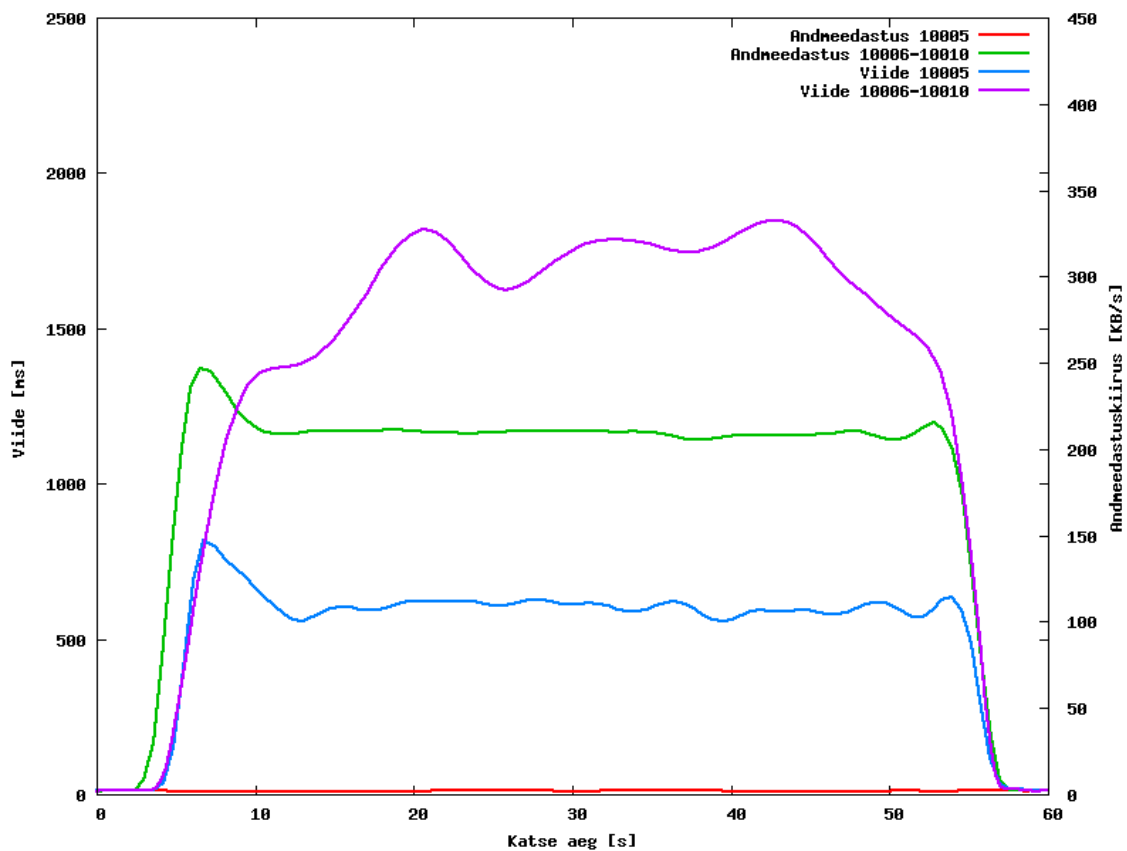


Joonis 13: Katse tulemused ilma võrguliikluse kujundamiseta

Ehkki suuremahuline andmeedastus (kokku ~400KB/s) toimub vaid TCP portidel 10006-10010, on pakettide ja vastuspakettide vaheline viiteaeg kasvanud ka minimaalselt andmeid edastaval ühendusel TCP pordil 10005, olgugi et tõus on märksa

väiksem (~10ms pealt ~400ms peale), kui andmeid edastaval ühendusel (~ 600-700ms peale).

Esimese katsena vähendada mõju prioriteetsele ühendusele (sellele, mis andmeid ei edasta) on piiratud Internetiühenduse kiirus 200KB/s peale, oletades et kui Internetiühendust vaid osaliselt kasutada ei lähe ta „umbe“. Katse tulemused olulist paranemist siiski ei näita.

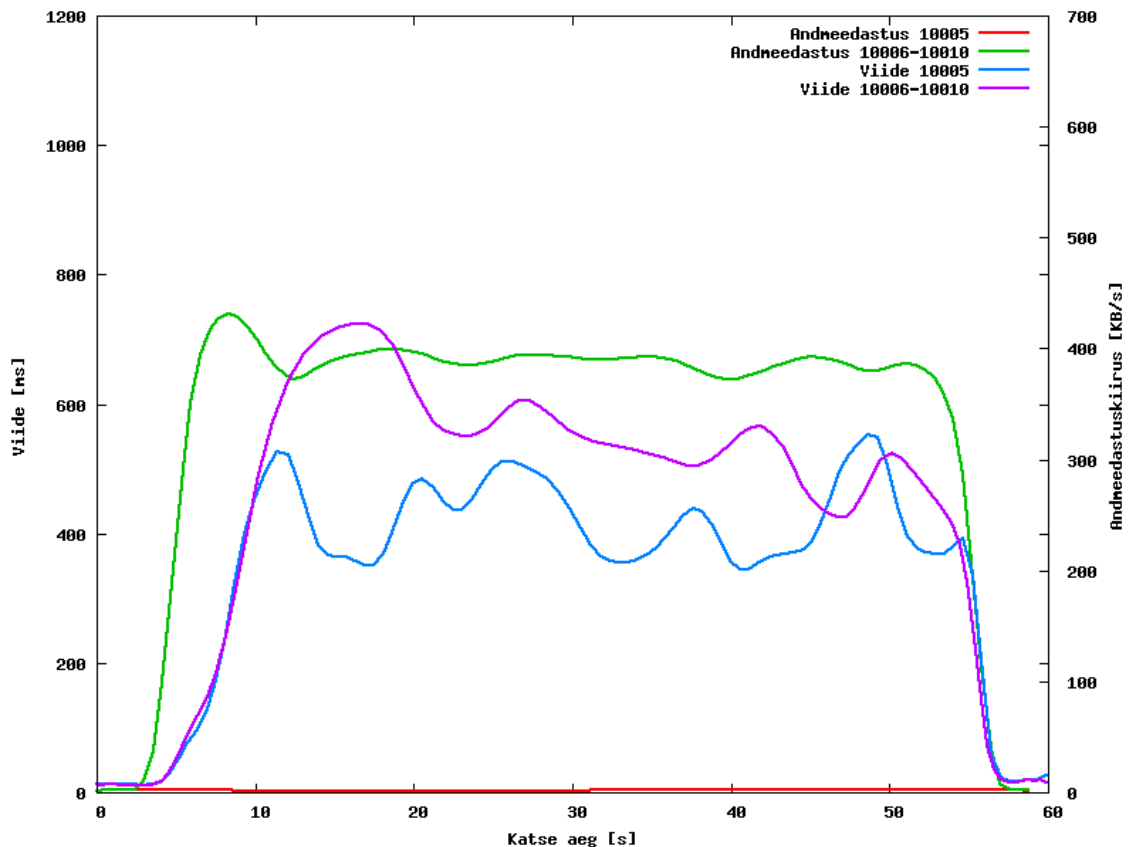


Joonis 14: Katse tulemused piirates Internetiühendust 200KB/s

Graafikult paistab, et ühenduste viited on kasvanud. Ühtlasi on märgata ka kasvanud läbilaskevõime ning viidete stabiilsust, mille võib olla põhjustanud täpselt määratud ühenduse tippkiirus. Kui piiramata kiirusega ühenduse maksimaalne läbilase sõltus igal ajahetkel kõikvõimalikest tingimustest (liinimüra, teised kasutajad teenusepakkujate võrkudes), siis piirkiirus 200KB/s oli igal hetkel kättesaadav, mis võimaldas TCP-l jääda stabiilselt leitud maksimaalse kiiruse juurde.

Teise katsena vähendada andmeedastuse mõju prioriteetsele ühendusele on seadistatud

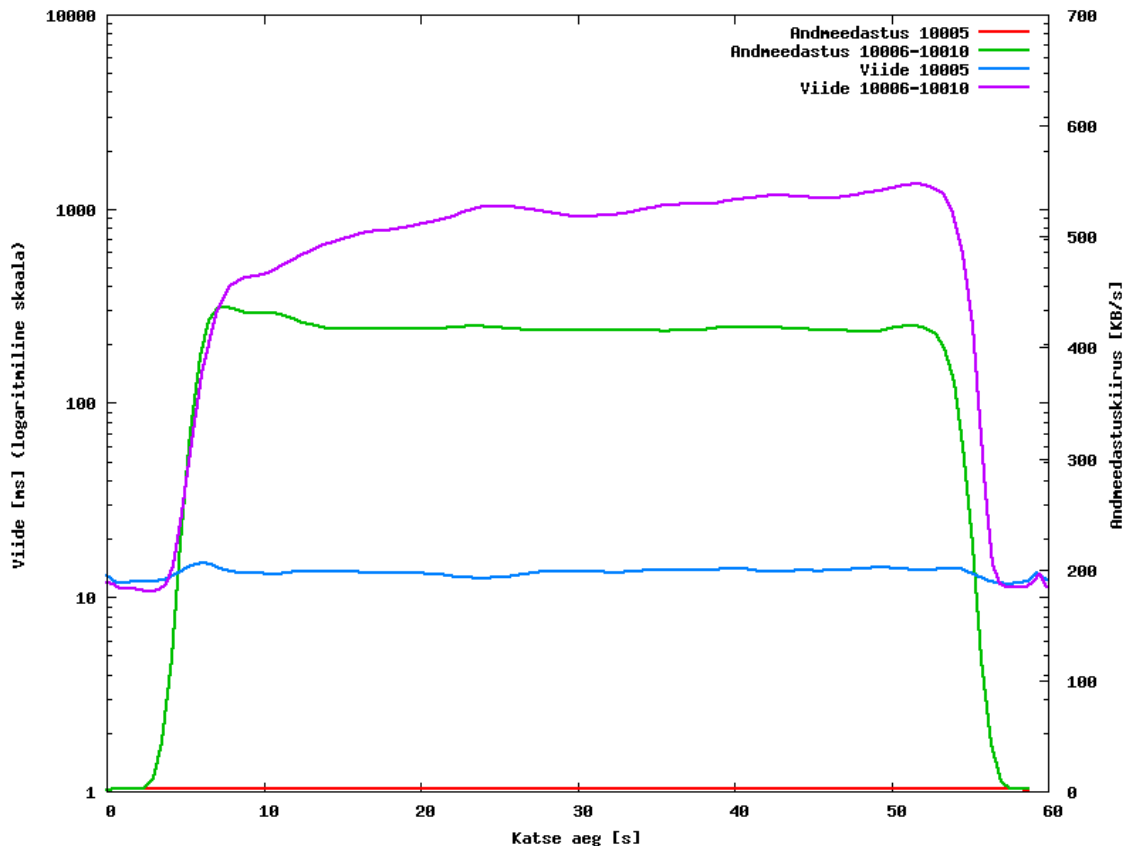
marsruuter „OpenWrt“ eelistama prioriteetse ühenduse pakette – kui Internetti ühendatud liidesel ootab väljasaatmist mitu paketti, saadetakse alati enne ära kõik prioriteetse ühenduse paketid ning ülejäänud pakette ainult siis, kui esimesi enam puhvrites ei ole. Ka see lahendus ei anna soovitud tulemust.



Joonis 15: Katse tulemused prioritseerimisega

Graafik on sarnane esimese katse graafikuga – viited on kasvanud sarnasele tasemele ning sarnane on ka andmeedastuskiirus. Oodatus tulemus jäi ilmselt saamata, kuna Internetiühenduse „pudelikael“ ei asu liiklust kujundavas marsruuteris „OpenWrt“ vaid mõne järgmise võrgulüüsi sees. „OpenWrt“ saadab saabunud paketid kohe edasi, kus need kogunevad mõne aeglasema lüli puhvritesse, kus neid aga ei prioritseerita.

Kolmanda katsena vähendada mõju prioriteetsele ühendusele on ühendatud prioritseerimine (enne saadetakse välja prioriteetse ühenduse paketid) ning pudelikaela toomiseks „OpenWrt“ juurde on ühenduse kiirus piiratud. Piirkiiruseks on valitud 400KB/s, kuna eelmiste katsete põhjal tundub see olevat reaalse piirkiiruse lähedal.



Joonis 16: Katse tulemused prioritseerimise ja Internetiühenduse piiramisega

Graafikult on näha, et selline lahendus on edukas ning prioriteetse ühenduse viite tõus andmedastuse alates on minimaalne ning viide alla eduka katse kriteeriumiks määratud 20ms.

Katse analüüs

Võrguliikluse kujundamise lahenduse hindamise keerukus asub katsekeskkonna ehitamises, katsete defineerimises, katse tulemuste analüüsis ning lahenduses muudatuste tegemises. Katse läbiviimine, kui eelpoolmainitu on tehtud, on triviaalne.

Esimeste edutute lahenduste puuduste kõrvaldamine nõudis teadmisi võrkude tööst. Katse tulemustest ei väljendunud kuidagi, miks lahendused ei toimunud nagu (naivselt) oodatud oli. Loodud tööriist ei aita kuigi palju kaasa kujundamise lahenduse probleemide mõistmisele või nende kõrvaldamisele, vaid ainult annab märku probleemide olemasolust.

TALLINNA TEHNIKAÜLIKOOL

Ehkki viimane katse osutus edukaks, ei garanteeri see, et lahendus igas olukorras püstitatud eesmärgi täidaks. On võimalik, et jagatud ühenduse kasutuse muutus (näiteks ööpäevane kõikumine) vähendab oluliselt „OpenWrt“ kättsaadavat piirkiirust ning ühenduse pudelikael nihkub tagasi järgmiste võrgulüüside juurde. Katse tulemused väljendavad ühe hetke seisu – seega on lahenduse toimivuse kinnitamiseks ikkagi vaja kogemustega katsetajat.

Tööriista kasulikkus tuleneb võimalusest kulutada vähem aega ning tähelepanu katse läbiviimisele ning rohkem katsete loomisele, katsete tulemuste ning lahenduse üldisele analüüsile.

Kokkuvõte

Loodud tööriist täidab oma eesmärgi. Võrguliikluse kujundamise lahenduse hindamiseks vajalikud operatsioonid on keskselt juhitavad ja korratavad ning katse andmed ühtsustatud ning salvestatud – kõik tööriista loomist motiveerinud puudused lahenduse hindamisel on kõrvaldatud. Katsetaja saab keskenduda katse väljamõtlemisele ning tulemuste analüüsile, ega pea pühenduma katse korduvale läbiviimisele.

Vajalikuks võib osutada ka uue (näiteks ICMP) transporditaseme protokollide lisamine või olemasolevate protokollide täpsema seadistamise võimaldamine. Modulaarse arhitektuuri kasutamise tõttu peaks need võimalikud arendused mahult koosnema ainult XML parseri täiendamisest ning transpordimooduli lisade implementeerimises või uue transpordimooduli kirjutamises. Kumbki arendus vajab vaid triviaalseid muudatusi olemasolevas lähtekoodis.

Resume

The purpose of this paper is to design and implement an effective application for testing traffic shaping solutions. Such application would not have any novel functionality, but rather lessen some of the hardships that the author has endured testing traffic shaping solutions with general purpose network analysis software.

Most prominent problems with general purpose applications are difficulties with timing and simultaneous execution when reproducing similar tests for a number of times. Another problem lies in parsing and unifying the output of those tools and having to transfer the results around to store them for future reference. The application is designed specifically to overcome these problems.

Testing the implemented application reveals that it fulfills its goals thus allowing an experimentator to concentrate less on the actual running of an experiment and more on designing experiments and analysing the results.

The application is very specific in its use cases and would not elicit wide usage and is also unlikely to need further development. Possible developments could be porting the application to Windows operating system and refining or adding transport modules for the packet generation component. These possible developments have been foreseen and the architecture of the application allows them to be made with minimal changes to current code.

Kasutatud kirjandus

1. List of international submarine communications cables [WWW] http://en.wikipedia.org/wiki/List_of_international_submarine_communications_cables (27.11.2007)
2. Tallinn Internet Exchange Members [WWW] <http://tix.estpak.ee/?members> (26.11.2007)
3. STV Shared Access tootekirjeldus [WWW] <http://stv.ee/index.php?act=net&t=tallinn&p=6&id=10&pid=37&lng=est> (27.11.2007)
4. The BitTorrent Effect [WWW] <http://www.wired.com/wired/archive/13.01/bittorrent.html> (27.11.2007)
5. Internet Protocol RFC791 [WWW] <http://www.ietf.org/rfc/rfc0791.txt> (27.11.2007)
6. Transmission Control Protocol RFC793 [WWW] <http://www.ietf.org/rfc/rfc0793.txt> (27.11.2007)
7. IANA Port Numbers [WWW] <http://www.iana.org/assignments/port-numbers> (27.11.2007)

LISA 1 – katse seaded ning skriptid

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<SeireServer>
  <Experiment duration="59"/>
  <TrafficClass name="TCP port 10005">
    <Match type="is" size="short" offset="12" value="0x0800"/>
    <Match type="is" size="char" offset="23" value="0x06"/>
    <Match type="is" size="short" offset="34" value="10005"/>
  </TrafficClass>
  <TrafficClass name="TCP port 10005-1007">
    <Match type="is" size="short" offset="12" value="0x0800"/>
    <Match type="is" size="char" offset="23" value="0x06"/>
    <Match type="isbetween" size="short" offset="34" value="10006"
value2="10010"/>
  </TrafficClass>
</SeireServer>
```

Joonis 17: Katse SeireServer-i seadistus

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<PaketiGeneraator>
  <Experiment duration="60"/>
  <Connection>
    <Transport type="tcp">
      <Role>client</Role><DIP>10.0.0.1</DIP>
      <DPort>10005</DPort><SPort>10005</SPort>
    </Transport>
    <RTT role="record" interval="0.1"/>
  </Connection>
  <Connection>
    <Transport type="tcp">
      <Role>client</Role><DIP>10.0.0.1</DIP>
      <DPort>10006</DPort><SPort>10006</SPort>
    </Transport>
    <RTT role="record" interval="0.1"/>
    <Volume datalen="18000" start="5" interval="0.077" end="55"/>
  </Connection>
  <Connection>
    <Transport type="tcp">
      <Role>client</Role><DIP>10.0.0.1</DIP>
      <DPort>10007</DPort><SPort>10007</SPort>
    </Transport>
    <Volume datalen="18000" start="5" interval="0.073" end="55"/>
  </Connection>
  <Connection>
    <Transport type="tcp">
      <Role>client</Role><DIP>10.0.0.1</DIP>
      <DPort>10008</DPort><SPort>10008</SPort>
    </Transport>
    <Volume datalen="18000" start="5" interval="0.071" end="55"/>
  </Connection>
  <Connection>
    <Transport type="tcp">
```

TALLINNA TEHNIKAÜLIKOOL

```
<Role>client</Role><DIP>10.0.0.1</DIP>
  <DPort>10009</DPort><SPort>10009</SPort>
</Transport>
  <Volume datalen="18000" start="5" interval="0.071" end="55"/>
</Connection>
<Connection>
  <Transport type="tcp">
    <Role>client</Role><DIP>10.0.0.1</DIP>
    <DPort>10010</DPort><SPort>10010</SPort>
  </Transport>
  <Volume datalen="18000" start="5" interval="0.071" end="55"/>
</Connection>
</PaketiGeneraator>
```

Joonis 18: Katse esimese PaketiGeneraatori seadistus

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<PaketiGeneraator>
  <Experiment duration="60"/>
  <Connection>
    <Transport type="tcp">
      <Role>server</Role><SPort>10005</SPort>
    </Transport>
    <RTT role="mirror"/>
  </Connection>
  <Connection>
    <Transport type="tcp">
      <Role>server</Role><SPort>10006</SPort>
    </Transport>
    <RTT role="mirror"/>
  </Connection>
  <Connection>
    <Transport type="tcp">
      <Role>server</Role><SPort>10007</SPort>
    </Transport>
    <Receiver/>
  </Connection>
  <Connection>
    <Transport type="tcp">
      <Role>server</Role><SPort>10008</SPort>
    </Transport>
    <Receiver/>
  </Connection>
  <Connection>
    <Transport type="tcp">
      <Role>server</Role><SPort>10009</SPort>
    </Transport>
    <Receiver/>
  </Connection>
  <Connection>
    <Transport type="tcp">
      <Role>server</Role><SPort>10010</SPort>
    </Transport>
    <Receiver/>
  </Connection>
```

```
</Connection>
</PaketiGeneraator>
```

Joonis 19: Katse teise PaketiGeneraatori seadistus

```
#!/bin/sh
# Run experiment

klient=../Klient

# clear old results
echo "Removing old results"
rm *.csv
rm *.png

# conf
echo "Sending configuration"
$klient -d 127.0.0.1:10000 -c pg-tsee-diees.xml
$klient -d 127.0.0.1:10001 -c pg-windolik.xml
$klient -d 127.0.0.1:10002 -c ss-windolik.xml

# run
echo "Sending RUN"
$klient -d 127.0.0.1:10000 -r
$klient -d 127.0.0.1:10001 -r
$klient -d 127.0.0.1:10002 -r

# sleep
echo "Sleep for 65 sconds"
sleep 65

# results
echo "Fetching results"
$klient -d 127.0.0.1:10001 -f rtt
$klient -d 127.0.0.1:10002 -f bw
```

Joonis 20: Katse käivitamise skript

```
#!/bin/sh
# Save experiment results in separate directory for reference

name=$1

if [ -z "${name}" ]; then
    echo "Usage: $0 <experiment name>"
fi

./result.gnuplot
rm -rf $name
mkdir $name
cp *.csv $name
cp *.xml $name
cp result.png $name
```

Joonis 21: Katse tulemuste salvestamise skript

```
#!/usr/bin/gnuplot
# Generate graphs out of experiment results

set macros
set xlabel "Katse aeg [s]";
set xrange [0:60];
set ylabel "Viide [ms]";
set y2label "Andmeedastuskiirus [KB/s]";
set y2tics;
#set logscale y;
set terminal png size 800, 600;
set output "result.png"
bw="smooth bezier axis xly2 w lines lw 2"
rtt="smooth bezier axis xly1 w lines lw 2"

plot "bw_0.csv" u ($1):($2/1000*4) @bw t 'Andmeedastus 10005', \
"bw_1.csv" u ($1):($2/1000*4) @bw t 'Andmeedastus 10006-10010', \
"rtt_0.csv" u ($1):($2/1000) @rtt t 'Viide 10005', \
"rtt_1.csv" u ($1):($2/1000) @rtt t 'Viide 10006-10010'
```

Joonis 22: Katse tulemuste visualiseerimise skript