Riku Kuismanen ja Henrik Martikainen

IP-verkon suorituskyvyn monitorointi heterogeenisessa ympäristössä

Tietotekniikan
pro gradu -tutkielma
19. joulukuuta 2006

Jyväskylän yliopisto

Tietotekniikan laitos

Jyväskylä

Tekijä: Riku Kuismanen ja Henrik Martikainen
Yhteystiedot: rtkuisma@cc.jyu.fi, henrik.martikainen@jyu.fi
Työn nimi: IP-verkon suorituskyvyn monitorointi heterogeenisessa ympäristössä
Title in English: Performance monitoring of the IP-network in a heterogeneous environment
Työ: Tietotekniikan pro gradu -tutkielma
Sivumäärä: 118
Tiivistelmä: IP-verkkojen suorituskyvyn mittaamiselle on olemassa lukuisia käytäntöjä. Tässä tutkielmassa käytiin läpi miten suorituskykyä tulisi mitata ja millä parametreilla. Lisäksi esiteltiin tekniikoita, joilla IP-verkkojen suorituskykyä voidaan mitata. Työn käytännön osuudessa tarkasteltiin esiteltyjen tekniikoiden toimivuutta heterogeenisessa IP-verkossa, ja arvioitiin kuinka hyvin tekniikat soveltuvat eri palveluiden ja liityntöjen suorituskyvyn valvontaan. Lisäksi toteutettiin NetFlow-rajapinta olemassa olevaan verkon visualisointiohjelmaan. Havaittiin että IP-verkon suorituskyvyn mittaaminen vaatii usean eri tekniikan käyttöä, jotta saadaan mahdollisimman hyvä kuva verkon suorituskyvystä.
English abstract: There exist several methods for measuring the performance of the IP-networks. This thesis explains how the performance should be measured and what parameters should be used. Several IP measuring techniques were also presented. These techniques were then studied in a heterogeneous IP-network in the experimental part of the thesis. Especially the suitability of the techniques for the performance monitoring of different services and interfaces was evaluated. Also a NetFlow interface was implemented to an existing network visualization tool. The practical evaluation showed that one technique can't provide comprehensive knowledge of the performance of the IP-network but several techniques are needed.
Avainsanat: IP, suorituskyky, monitorointi, mittaustekniikat, SNMP, NetFlow, OWAMP, IP SLA
Keywords: IP, performance, monitoring, measuring techniques, SNMP, NetFlow, OWAMP, IP SLA

Sisältö

Sanasto

3G *3rd Generation* eli kolmannen sukupolven matkapuhelinjärjestelmä. Yleensä tällä tarkoitetaan GSM:n jälkeen tulleita nopeaan datasiirtoon kykeneviä matkapuhelinverkkoja, kuten UMTS, CDMA2000 tai FOMA.

ADSL *Asymmetric Digital Subscriber Line* eli asynkroninen tilaajayhteys, jossa käytetään tiedonsiirtoon puhelinlinjoja. ADSL-yhteydelle on tyypillistä laskevan suunnan (engl. *downlink*) nousevaa suuntaa (engl. *uplink*) suurempi tiedonsiirtonopeus.

AS *Autonomous System* eli autonominen järjestelmä. Yhden organisaation hallinnoima verkkokokonaisuus, joka huolehtii omien IP-osoitteidensa reitityksestä perille.

ATM *Asynchronous Transfer Mode* on piirikytkentäinen siirtoyhteys- ja verkkokerroksen protokolla.

BGP *Border Gateway Protocol* on autonomisten järjestelmien välinen reititysprotokolla.

CLI *Command Line Interface* eli komentorivikäyttöliittymä.

DHCP *Dynamic Host Configuration Protocol* on verkkoprotokolla, joka mahdollistaa muun muassa IP-osoitteiden automaattisen jakamisen lähiverkon laitteille.

DLSw+ *Data-Link Switching* on Ciscon laajennus DLSw:hen. DLSw on tunnelointiprotokolla, joka mahdollistaa muiden kuin IP-pohjaisten protokollien tunneloinnin IP-protokollan päällä.

DNS *Domain Name System* eli nimipalvelinjärjestelmä mahdollistaa muunnokset verkkotunnusten ja IP-osoitteiden välillä.

DSL *Digital Subscriber Line* eli digitaalinen tilaajayhteys, jossa käytetään digitaaliseen tiedonsiirtoon puhelinlinjoja.

Ethernet Yleisesti käytetty pakettikytkentäinen lähiverkkotekniikka (*IEEE 802.3*).

Flash-OFDM *Fast Low-latency Access with Seamless Handoff Orthogonal Frequency-Division Multiplexing* eli langaton verkkoteknologia.

FTP *File Transfer Protocol* eli TCP-protokollan päällä toimiva tiedostonsiirtoprotokolla.

GPS *Global Positioning System* eli satelliittipaikannusjärjestelmä.

HTTP *Hypertext Transfer Protocol* eli hypertekstin siirtoprotokolla.

ICMP *Internet Control Message Protocol* on TCP/IP-pinon yhteydessä käytetty kontrolliprotokolla.

IETF *Internet Engineering Task Force* on organisaatio, jonka työryhmät kehittävät Internetin standardeja.

IOS *Internetwork Operating System* on Ciscon verkkolaitteiden käyttöjärjestelmä.

IP *Internet Protocol* on pakettikytkentäisissä verkoissa käytetty verkkokerroksen protokolla.

IP SLA *Internet Protocol Service Level Agreement* on Ciscon tietoverkkojen suorituskyvyn monitorointiohjelmisto.

IPFIX *Internet Protocol Flow Information eXport* on IETF:n ehdotus standardiksi vuovalvontatekniikaksi.

IPPM *Internet Protocol Performance Metrics* eli IP:n suorituskykyparametrit. IETF:n työryhmän määrittelemät standardit suorituskykyparametrit IP-verkoille.

IPTV *Internet Protocol Television* on IP-protokollan päällä lähetettävä televisiolähetys.

LAN *Local Area Network* eli lähiverkko. Sen toiminta-alue kattaa lähialueen, kuten toimiston tai muutaman rakennuksen.

LSP *Label Switched Path* on MPLS-verkon läpi kulkeva polku.

MAC *Media Access Control* on siirtoyhteyskerroksen protokolla.

MIB *Management Information Base* eli hallintatietokanta. Sisältää tiedon verkkolaitteen resursseista, joita kysellään SNMP-protokollan avulla.

MPLS *Multiprotocol Label Switching* emuloi piirikytkentäistä verkkoa pakettikytkentäisen verkon päällä. Mahdollistaa ATM-yhteyksien korvaamisen nopeilla pakettikytkentäisillä verkoilla.

OWAMP *One-Way Active Measurement Protocol* on IETF:n työryhmän kehittämä yhdensuuntaisen IP-verkon suorituskyvyn mittausprotokolla.

QoS *Quality of Service* eli tietoliikenneverkkojen palvelunlaatu.

NetFlow Ciscon vuovalvontatekniikka.

NTP *Network Time Protocol* on aikatiedon välitysprotokolla.

RFC *Request for Comments* ovat asiakirjoja jotka kuvaavat Internetin erilaisia käytäntöjä.

RMON *Remote Network Monitoring* on IP-verkkojen suorituskyvyn mittausprotokolla.

RTT *Round Trip Time* eli edestakainen viive.

SAA *Service Assurance Agent* on Ciscon tietoverkkojen suorituskyvyn monitorointiohjelmisto. Nykyisin nimeltään IP SLA.

SCTP *Stream Control Transmission Protocol* on luotettava yhteydellinen viestipohjainen kuljetuskerroksen protokolla.

SLA *Service Level Agreement* eli palvelutasosopimus, jolla sovitaan asiakkaalle toimitettavan palvelun taso.

SMI *Structure of Management Information* eli hallintatietokantojen rakenteen määrittely.

SNMP *Simple Network Management Protocol* on verkonhallintaprotokolla tai verkonhallintastandardien joukko.

TCA *Traffic Conditioning Agreement* eli liikenteen muotoilusopimus. Palvelusopimuksen teknisen toteutuksen kuvaus.

TCP *Transmission Control Protocol* on luotettava yhteydellinen kuljetuskerroksen protokolla.

TWAMP *Two-Way Active Measurement Protocol* on IETF:n työryhmän kehittämä IP-verkon edestakaisen liikenteen suorituskyvyn mittausprotokolla.

USM *User-Based Security Model* eli käyttäjäpohjainen tietoturvamalli. SNMP:n kolmannen version ominaisuus, jolla voidaan antaa jokaiselle käyttäjälle erilliset oikeudet

UDP *User Datagram Protocol* on yhteydetön kuljetuskerroksen protokolla, joka ei takaa pakettien perille menoa.

VACM *View-based Access Control Model* eli näkymäpohjainen pääsynhallintamalli. SNMP:n kolmannen version ominaisuus, jolla voidaan luoda valmiita näkymiä, joihin voidaan jakaa oikeuksia.

VLAN *Virtual Local Area Network* eli virtuaalilähiverkko. Mahdollistaa fyysisen lähiverkon jakamiseen loogisiin osiin.

VoD *Video on Demand* eli tilausvideo.

VoIP *Voice over Internet Protocol* on termi, jota käytetään kun kuljetetaan puheluita IP-protokollan päällä.

VPN *Virtual Private Network* on tapa jolla kahdesta tai useammasta lähiverkosta voidaan muodostaa julkisen verkon yli näennäisesti yksi yksityinen verkko.

WiMAX *Worldwide Interoperability for Microwave Access* eli MAN-alueen langaton verkkoteknologia (*IEEE 802.16*).

WLAN *Wireless Local Area Network* eli langaton lähiverkko (*IEEE 802.11*).

1 Johdanto

IP[1]-verkot ovat yleistyneet vauhdilla ja nykyään niitä löytyy miltei kaikkialta. Samalla kun IP-verkkojen kattavuus on kasvanut, on myös niiden käyttöaste lisääntynyt selvästi, ja niitä käyttävien sovellusten kirjo on kasvanut merkittävästi. Nämä kaikki kasvattavat ongelmatilanteiden mahdollisuuksia. Toisaalta uusien nopeiden verkkojen rakentaminen on kallista, joten olemassa olevia verkkoja tulisi käyttää entistä tehokkaammin hyväksi. IP-verkkojen suorituskyvyn mittaamisen ja valvonnan merkitys on siis lisääntynyt huomattavasti.

1.1 Palvelutasosopimus

Palvelutasosopimus[2] on sopimus, jossa palveluntarjoaja takaa tietyn tasoisen palvelun asiakkaalle. Palvelutasosopimus on osa palvelusopimusta, ja palvelutasosopimuksia voi olla palvelusopimuksessa useita tai ei yhtään.

Toisin kuin palveluntarjoajalle, asiakkaalle tietoverkoissa yleisesti käytetty *best-effort*-toimitustapa on sille riskitekijä. Palveluntarjoaja voi periaatteessa lisätä tulojaan liikennemääriä kasvattamalla, ilman uusia laiteinvestointeja. Tämä johtaa jossain vaiheessa väistämättä palvelutason heikkenemiseen. Palvelusopimuksilla pyritään rajoittamaan tätä verkon ylikäyttöä. Vaikka sopimuksen idea on periaatteessa molemmille osapuolille kohtuullinen, sen käytännön toteuttaminen ei ole yksinkertaista.

Sopimusta varten täytyy määrittää hyväksyttävä palvelutaso ja selvittää keinot mitata sitä. Tarkkoja rajoja on helpompi valvoa, eikä niistä synny helposti epäselvyyksiä, mutta ne voivat aiheuttaa verkon palvelutason tippumista. Toisaalta liian yleiset rajat voivat aiheuttaa tulkintaongelmia. Myös kolmannen osapuolen verkko vaikeuttaa palvelutason määrittämistä ja mittaamista. [1]

Liikenteen muotoilusopimus[3] on palvelutasosopimuksen tekninen osuus, jota käytetään hyväksi verkon konfiguroinnissa. Se sisältää yksityiskohtaiset parametrit jokaiselle palvelun tasolle ja se voi sisältää esimerkiksi palvelun tason odotetun läpipäästön (engl. *throughput*), pudotusten todennäköisyyden tai viiveen. [1]

[1]Internet Protocol
[2]Service Level Agreement (SLA)
[3]Traffic Conditioning Agreement (TCA)

1.2 Mittaaminen

Verkosta voidaan mitata useita suureita, jotka voidaan jakaa määrällisiin ja laadullisiin. **Määrälliset**: *Liikenteen määrä* (engl. *traffic volume*) sanelee kuinka verkko tulisi mitoittaa. *Vapaa kaistanleveys* (engl. *available bandwidth*) on liikenteen määrän ja *verkon kapasiteetin* erotus.

Laadulliset: *Liikenteen läpäisy* (engl. *throughput*) kertoo kuinka paljon hyötyliikennettä verkossa on liikkunut ja on liikenteen määrää pienempi arvo. Muun muassa uudelleen lähetykset ja verkon virheet sekä protokollien otsikot vaikuttavat liikenteen läpäisyyn. Mitä lähempänä liikenteen läpäisy on liikenteen määrää, sitä paremmin verkko suoriutuu. Pakettien *viiveet* ja *viiveiden vaihtelut* vaikuttavat suuresti käyttäjien tyytyväisyyteen. *Pakettien hävikkiin* on monia syitä. Verkon ruuhkautuessa paketit voivat vuotaa yli puskureista tai niitä voidaan tiputtaa tarkoituksella resurssien säästämiseksi. Toisaalta tietty määrä hävikkiä kuuluu joidenkin protokollien normaaliin toimintaan. [2]

1.2.1 Jakoperusteet aktiivisiin ja passiivisiin mittauksiin

Verkon mittaustavat voidaan jakaa monella tavalla. Yleisesti mittaukset on jaettu **aktiivisiin** ja **passiivisiin** mittaustapoihin. Jako näiden välillä voidaan kahdella tavalla:

- Tuottaako mittaustapa mittausliikennettä vai ei? Mikäli mittaamiseen kuuluu oleellisesti testiliikenteen generointi, mittaustapa määritellään aktiiviseksi. Ellei muuta mainita, tässä tutkielmassa jako aktiivisiin ja passiivisiin tehdään tällä perusteella.
- Tarvitseeko mittaajan aktiivisesti kysellä tuloksia vai lähettävätkö verkon laitteet tulokset automaattisesti mittaajalle?

Passiiviset mittaustavat mittaavat verkon olemassa olevaa liikennettä ja ne tuottavat liikennettä verkkoon ainoastaan testituloksia siirrettäessä. Tulokset voidaan kuitenkin toimittaa perille erillisen hallintaverkon kautta, jolloin varsinaista tuotantoverkkoa ei kuormiteta. Passiivista mittausta voivat suorittaa joko olemassa olevat verkon laitteet (esimerkiksi reitittimet ja kytkimet) tai erilliset mittaamista varten asennetut laitteet.

Passiivisella verkon mittauksella saadaan kattava näkymä verkkoon. Verkosta ei tarvitse tietää paljoakaan ennen mittauksia, joten passiivinen mittaaminen soveltuu

tuntemattomia verkkoja tutkittaessa. Kun verkosta on kerätty riittävä määrä mittaustietoa, voidaan sitä analysoida eri keinoin jälkikäteen. Kattavan mittaustiedon kerääminen tarkoittaa toisaalta sitä, että tietoa kertyy paljon. Kattava passiivinen mittaaminen vaatiikin mittaavalta ja analysoivalta laitteelta nopeaa tiedon tallentamista ja käsittelyä, sekä mittauksia varastoivalta laitteelta suurta tallennuskapasiteettia. [3]

Tässä tutkielmassa esiteltyjä passiivisia mittaustekniikoita ovat SNMP[4] luvussa 3.1, RMON[5] luvussa 3.2 ja NetFlow luvussa 3.5.

Aktiivinen verkon mittaaminen perustuu verkon tutkimiseen generoidun testiliikenteen avulla. Esimerkiksi käyttöjärjestelmistä yleisesti löytyvä *ping*-ohjelma lähettää *ICMP*[6] *echo* -paketteja, joilla se mittaa edestakaisen viiveen lähettäjän ja vastaanottajan välillä. Aktiivisilla mittauksilla tutkitaan yleensä verkkoa, passiivisella sen liikennettä. [3]

Aktiivinen mittaaminen tuottaa aina ylimääräistä liikennettä verkkoon. Ennen mittausten aloittamista tulisikin suunnitella paljonko ylimääräistä liikennettä tuotantoverkkoon saa syntyä. Tiheämmällä testaamisella saadaan tarkempia tuloksia, mutta samalla kuormitetaan verkkoa. Toisaalta suuri testiliikenteen määrä voi vääristää tuloksia. Joitain testejä voidaan ajaa hiljaisina hetkinä, kun taas toiset täytyy ajaa verkon ollessa kuormitettuina.

Aktiivinen mittaus ei yleensä vaadi paljoa suorituskykyä mittaavilta laitteilta. Mittaukset ovat yksinkertaisia, kuten ICMP-pakettien lähettämistä tai TCP[7]-yhteyksien avaamisia. Testit vaativat kuitenkin tarkkaa etukäteissuunnittelua, verkon tuntemusta ja tietoa siitä mitä halutaan tutkia. Jos jokin asia jää suunnitteluvaiheessa huomioimatta, voidaan kaikki testit joutua tekemään uudelleen. Lisäksi synteettiset testipaketit eivät välttämättä kuvaa hyvin sovelluksen lähettämää dataa.

Aktiivista mittausta voidaan suorittaa myös oikeilla sovelluksilla. Palveluiden testaaminen näin onkin suhteellisen helppoa, mutta nämäkin testaukset ovat vain suuntaa antavia, jotka pätevät vain kyseiselle ajanhetkelle ja testiasiakkaalle.

Tässä tutkielmassa esiteltyjä aktiivisia mittaustekniikoita ovat OWAMP[8] luvussa 4.1, TWAMP[9] luvussa 4.2 ja IP SLA[10] luvussa 4.3.

[4]Simple Network Management Protocol
[5]Remote Network Monitoring
[6]Internet Control Message Protocol
[7]Transmission Control Protocol
[8]One-Way Active Measurement Protocol
[9]Two-Way Active Measurement Protocol
[10]Internet Protocol Service Level Agreement

1.2.2 Muita mittausten jakoperusteita

Verkon mittaukset voivat erota toisistaan **ajallisesti**. Mittaukset voivat olla **jatkuvia**, jolloin mittaus kuvaa verkon tai laitteen tilaa koko mittausajalta. Esimerkiksi liitynnän läpimenneen liikenteen tavulaskureiden kirjaaminen on jatkuvaa mittaamista. Kaikkea liikennettä ei voida tai ei ole järkevää valvoa aktiivisesti. Silloin voidaan käyttää **näytteistystä** (engl. *sampling*), joka kuvaa verkon hetkellistä tilaa. Jotkin mittaukset sopivat luonnostaa näytteistämällä mitattavaksi, kuten viiveen mittaaminen. [2]

Mittaukset voivat erota toisistaan myös **paikan suhteen**. Mittauksia voidaan suorittaa **päästä-päähän**. Päästä-päähän mittaukset voivat olla aktiivisia, jolloin generoidaan testiliikennettä. Ne voivat olla myös passiivisia, jolloin paketeista lasketaan tunniste verkon eri pisteissä ja verrataan pakettien saapumisaikaa näissä pisteissä. Esimerkiksi *ping* ja *traceroute* suorittavat päästä-päähän mittauksia. **Yhdessä pisteessä** suoritetuilla mittauksilla saadaan näkymä vain yhdessä pisteessä. Nämä mittaukset ovat usein passiivisia ja esimerkkeinä näistä ovat muun muassa verkon kuuntelijat ja edellä mainitut SNMP ja NetFlow. [2]

1.2.3 Vuomittaukset

IP-verkon liikennettä voidaan mitata pakettikohtaisesti, mutta tällöin unohdetaan että nykyiset IP-verkot ovat luonteeltaan yhteyssuuntautuneita. TCP-yhteydet voidaan periaatteessa tunnistaa helposti yhteyksien aloitus- ja lopetuspaketeista, mutta käytännössä nämä paketit voivat jäädä havaitsematta. Yhteydettömillä protokollilla kuten UDP[11]-protokollalla ei ole varsinaista yhteyttä, vaikka niidenkin liikenne on yhteyssuuntautunutta. Tarvitaankin jokin muu määritelmä yhteydelle. Tietovuo on eräänlainen yhteyden määritelmä ja se määritellään yleensä 5-jonona, joka koostuu seuraavista IP-paketin ominaisuuksista:

- IP-kohdeosoite
- IP-lähdeosoite
- kohdeportti
- lähdeportti
- IP-protokolla

[11] User Datagram Protocol

Nämä ominaisuudet riittävät myös erottamaan TCP-yhteydet toisistaan. Aikakatkaisulla (engl. *timeout*) määritellään mitkä paketit kuuluvat samaan tietovuohon. Yleensä tietovuot käsitellään yksisuuntaisina, koska kaksisuuntaisuus voidaan aina muodostaa jälkikäteen. Lisäksi reititys voi erota eri suunnille ja liikenne on monessa tapauksessa hyvin erilaista eri suuntiin. Tässä tutkielmassa luvussa 3.5 esitelty NetFlow on eräs vuomittaustekniikka. [3]

1.2.4 Verkon kuuntelu

Kun verkkolaitteet mittaavat itse liikennettä varsinaisen liikenteen välittämisen ohessa, puhutaan itsenäisestä mittaamisesta. Tässä tutkielmassa esitellään pääasiassa itsenäisen mittauksen tekniikoita, mutta verkkoa voi mitata myös verkossa sijaitsevalla erillisellä kuuntelijalla (engl. *sniffer*). Kuuntelija voi olla sijoitettu linkin keskelle tai verkkolaitteen monitorointiporttiin. **Linkkiin sijoitettu kuuntelija** näkee vain sen linkin liikenteen. Kuuntelija ei kuormita verkkoa ja sillä voi tarkkailla liikennettä monipuolisesti. Kuuntelija voi olla passiivinen, jolloin linkki on fyysisesti monistettu väliltä, tai kuuntelija voi olla aktiivinen, jolloin verkkoliikenne menee kuuntelijan läpi. Kummassakin tapauksessa kuuntelijoita pitää sijoittaa jokaiselle valvottavalle linkille, joten kustannukset nousevat helposti suuriksi. Lisäksi kuuntelijan kytkeminen aiheuttaa pienen katkoksen verkkoliikenteeseen.

Halpa ratkaisu on **verkkolaitteen monitorointiporttiin kytketty** kuuntelija. Tällöin nähdään kaikki verkkolaitteeseen liitetyt yhteydet. Tämä kuitenkin kuormittaa verkkolaitetta. Monitorointiportin kapasiteetti voi olla riittämätön. Osa liikenteestä voi myös jäädä monistumatta. Ratkaisu on kuitenkin halpa ja toimiva pienillä liikennemäärillä. [3]

1.3 Tutkimusongelma

Tämän tutkielman keskeisin tutkimuksen aihe on IP-verkon suorituskyvyn mittaaminen. Ennen mittaamista tutkielmassa selvitetään mitä parametreja verkosta voidaan mitata ja kuinka ne määritellään. Suorituskyvyn mittaamiseen on olemassa lukuisia tekniikoita, jonka lisäksi uusia tekniikoita kehitellään koko ajan. Tässä tutkielmassa selvitetään mitä tekniikoita on olemassa ja kuinka nämä tekniikat toimivat.

Kun mitattavat parametrit ja mittaustekniikoiden toiminta ovat tiedossa, tutkielman käytännön osuudessa selvitetään kuinka nämä tekniikat toimivat käytännössä.

Tätä varten tietoliikennelaboratorioon rakennetaan heterogeeninen IPTV[12]-verkko, jossa tekniikoiden hyvät ja huonot puolet voidaan selvittää. Testiverkossa tutkitaan mitä niillä voidaan havaita IPTV-verkon suorituskyvystä. Lisäksi tutkitaan eri tekniikoiden soveltuvuus eri palveluille ja verkon eri osille. Testauksen ohessa käytännön osuudessa toteutetaan VizTool-verkonvalvontatyökaluun tuki uudelle mittaustekniikalle ja arvioidaan sen toimivuutta.

1.4 LaiLa

Tämä tutkielma tehtiin Jyväskylän yliopiston LaiLa[13]-projektille. Laila on tietotekniikan laitoksella sijaitseva TEKESin rahoittama tutkimusprojekti ja osa GIGA-teknologiaohjelmaa. Projekti keskittyy heterogeenisiin pääsyverkkoihin (engl. *Access Networks*) ja niiden hallintaan ja monitorointiin. Projektin pääpaino on IPTV, VoD[14] ja VoIP[15] -palveluissa sekä DSL[16], WLAN[17], WiMAX[18], 3G[19] ja Flash-OFDM[20] verkkotekniikoissa. LaiLan toiminnassa ovat mukana seuraavat yritykset:

- Digita
- Arena Partners
- SysopenDigia
- Nokia
- WTS Networks
- Ortikon Interactive
- Sofia Digital
- TEKES

[12]Internet Protocol Television
[13]Langattomien laajakaistapalveluiden hallinta multiaccess-verkoissa
[14]Video on Demand
[15]Voice over Internet Protocol
[16]Digital Subscriber Line
[17]Wireless Local Area Network
[18]Worldwide Interoperability for Microwave Access
[19]3rd Generation
[20]Fast Low-latency Access with Seamless Handoff Orthogonal Frequency-Division Multiplexing

1.5 Tutkielman rakenne

Luvussa 2 esitellään IPPM[21]-työryhmän määrittelemät suorituskykyparametrit. Luvussa 3 esitellään seuraavat passiiviset IP-verkon mittaustekniikat: SNMP, RMON1, RMON2 ja NetFlow. Luvussa 4 esitellään seuraavat aktiiviset mittaustekniikat: OWAMP, TWAMP ja IP SLA. Luvussa 5 kuvataan testiverkko ja tekniikoiden toimivuus käytännössä. Lisäksi luvussa 5 kerrotaan VizToolin NetFlow-toteutuksesta. Luvussa 6 on esitetty tutkielman yhteenveto.

Tutkielmalla on kaksi tekijää, joilla kummallakin on oma vastuualueensa. Riku Kuismanen on vastuussa luvuista 2 ja 3, sekä alaluvuista 5.4 ja 5.5. Henrik Martikainen on vastuussa luvuista 4 sekä alaluvuista 1.1, 1.2, 5.2 ja 5.3. Muut osat työstä on tehty yhteistyönä.

[21] Internet Protocol Performance Metrics

2 Suorituskykyparametrit

Tässä luvussa käsitellään IPPM:n määrittelemiä suorituskykyparametreja ja niiden mittaamista.

2.1 Saatavuus

Saatavuus (engl. *connectivity*)tarkoittaa sitä, että kaksi asiakaslaitetta voivat tavoittaa toisensa verkon yli ja näin ollen kommunikoida keskenään. Saatavuus onkin yksi internetin peruspilareista. IPPM-työryhmä määrittelee viisi eri suorituskykyparametria saatavuudelle. Nämä parametrit on esitelty dokumentissa 'IPPM Metrics for Measuring Connectivity' [4]. Näiden parametrien kuvauksessa käytetään lähettäjästä lyhennettä Src ja vastaanottajasta Dst. Dokumentissa 'Framework for IP Performace Metrics' [5] määritellään suorituskykyparametrien mittaukseen käytettävää pakettia type-P paketiksi, joka tarkoittaa että mittaustulos on riippuvainen käytetystä mittauspaketista. Käytettävää pakettia ei lyödä lukkoon vaan se jätetään määriteltäväksi suunniteltaessa mittaussovellusta tai suoritettaessa itse mittauksia.

2.1.1 Hetkellinen yhdensuuntainen saatavuus

Hetkellinen yhdensuuntainen saatavuus (eng. *instantaneous one-way connectivity*) -parametrin attribuutit:

Parametrin nimi: `Type-P-Instantaneous-Unidirectional-Connectivity`

- `Src` = Lähettäjän IP-osoite
- `Dst` = Vastaanottajan IP-osoite
- `T` = Aika

Parametri saa arvokseen tosi tai epätosi.

Src:llä on hetkellinen yhdensuuntainen saatavuus Dst:hen ajanhetkellä `T`, eli `Type-P-Instantaneous-Unidirectional-Connectivity` saa arvon Tosi, jos paketti `type-P`, joka on lähetetty ajanhetkellä `T` saapuu Dst:lle.

Yhdensuuntainen saatavuus ei ole kovin yleisesti tarpeellinen parametri käytännön sovelluksia silmälläpitäen. Monet käytännön sovellukset vaativat kaksisuuntaista saatavuutta ja usein jopa saatavuutta yli jonkun aikaintervallin. Tällä parametrilla voidaan kuitenkin testata verkkolaitteiden tietoturvallisuutta; esimerkiksi suodattaako palomuuri *Ping of Death* -hyökkäykset pois oikein.

2.1.2 Hetkellinen kaksisuuntainen saatavuus

Hetkellinen kaksisuuntainen saatavuus (engl. *instantaneous two-way connectivity*) -parametrin attribuutit:

Parametrin nimi: `Type-P-Instantaneous-Bidirectional-Connectivity`

- `A1` = Ensimmäisen asiakkaan IP-osoite
- `A2` = Toisen asiakkaan IP-osoite
- `T` = Aika

Parametri saa arvokseen tosi tai epätosi.

Osoitteiden `A1` ja `A2` välillä on hetkellinen kaksisuuntainen saatavuus ajanhetkellä `T`, eli `Type-P-Instantaneous-Bidirectional-Connectivity` saa arvon tosi, jos `A1`:llä on hetkellinen yhdensuuntainen saatavuus `A2`:een ja `A2`:lla on hetkellinen yhdensuuntainen saatavuus `A1`:een.

2.1.3 Yhdensuuntainen saatavuus

Yhdensuuntainen saatavuus (engl. *one-way connectivity*) -parametrin attribuutit:

Parametrin nimi: `Type-P-Interval-Unidirectional-Connectivity`

- `Src` = Lähettäjän IP-osoite
- `Dst` = Vastaanottajan IP-osoite
- `T` = Aika
- `dT` = Kesto

Parametri saa arvokseen tosi tai epätosi.

Src:llä on ajanjakson `[T, T + dT]` aikana yhdensuuntainen saatavuus Dst:hen, eli `Type-P-Interval-Unidirectional-Connectivity` saa arvon tosi, jos jollain arvolla `T'`, kun `T'` kuuluu väliin `[T, T + dT]`, Src:llä on hetkellinen yhdensuuntainen saatavuus Dst:hen.

2.1.4 Kaksisuuntainen saatavuus

Kaksisuuntainen saatavuus (engl. *two-way connectivity*) -parametrin attribuutit:

Parametrin nimi: `Type-P-Interval-Bidirectional-Connectivity`

- `A1` = Ensimmäisen asiakkaan IP-osoite
- `A2` = Toisen asiakkaan IP-osoite
- `T` = Aika
- `dT` = Kesto

Parametri saa arvokseen tosi tai epätosi.

Osoitteiden `A1` ja `A2` on ajanjakson `[T, T + dT]` aikana kaksisuuntainen saatavuus, eli `Type-P-Interval-Bidirectional-Connectivity` saa arvon tosi, jos `A1`:llä on yhdensuuntainen saatavuus `A2`:een ja `A2`:lla on yhdensuuntainen saatavuus `A1`:een ajanjakson `[T, T + dT]` aikana.

2.1.5 Kaksisuuntainen väliaikainen saatavuus

Kaksisuuntainen väliaikainen saatavuus (engl. *two-way temporal connectivity*) -parametrin attribuutit:

Parametrin nimi: `Type-P1-P2-Interval-Temporal-Connectivity`

- `Src` = Lähettäjän IP-osoite
- `Dst` = Vastaanottajan IP-osoite
- `T` = Aika
- `dT` = Kesto

Parametri saa arvokseen tosi tai epätosi.

Src:llä on kaksisuuntainen väliaikainen saatavuus Dst:hen ajanjakson `[T, T + dT]` aikana, eli `Type-P1-P2-Interval-Temporal-Connectivity` saa arvon tosi, jos on olemassa ajat `T1` ja `T2`, sekä intervallit `dT1` ja `dT2` siten että:

- `T1`, `T1 + dT1`, `T2`, `T2 + dT2` kuuluvat kaikki välille `[T, T + dT]`
- `T1 + dT1 <= T2`
- Src:llä on ajanhetkellä `T1` hetkellinen saatavuus `Type-P1` Dst:hen
- Dst:llä on ajanhetkellä `T2` hetkellinen saatavuus `Type-P2` Src:hen
- `dT1` on aika, joka paketilta kuluu matkalla Dst:hen, kun Src lähettää sen ajanhetkellä `T1`
- `dT2` on aika, joka paketilta kuluu matkalla Src:hen, kun Dst lähettää sen ajanhetkellä `T2`

Tämä parametri määrittelee yleisesti hyödyllisen saatavuuden. Dst vastaa Src:n lähettämään pakettiin. Paketit voivat olla eri tyyppiä kuten monissa käytännön sovelluksissakin.

2.1.6 Mittaaminen

IPPM-työryhmä määrittelee rungon mittaustavalle. Mittaustapaa ei voida määrittää yksikäsitteisesti, koska yksityiskohdat riippuvat käytetyistä paketeista. Työryhmä määrittelee syötteet ja algoritmin, jota käytetään.

Syötteet:

- Pakettityypit `P1` ja `P2`, osoitteet `A1` ja `A2`, sekä intervalli `[T, T + dT]`.
- Lähetettävien pakettien määrä `N`.
- Odotusaika `W`, joka määrää kuinka kauan odotetaan vastausta lähetetylle paketille. Vaaditaan: `W <= 255 s, dT > W`.
- Suositellut arvot: `dT = 60 s`, `W = 10 s` ja `N = 20`

Algoritmi: Lasketaan `N` lähetysaikaa väliltä `[T, T + dT - W]`, jotka on satunnaisesti hajautettu tuolle välille. Jokaisena lähetysaikana lähetetään hyvin muodostettu tyypin `P1` paketti `A1`:ltä `A2`:lle. Tutkitaan saapuvaa liikennettä `A1`:llä. Jos saadaan sallittu vastauspaketti (`P1` tai `P2`), mittauksen tulokseksi asetetaan tosi, ja mittaus voidaan lopettaa. Jos sallittua vastausta ei saada ajanhetkeen `T + dT` mennessä, mittauksen tulokseksi asetetaan epätosi.

Algoritmi ei ole täysin pitävä, sillä mittaus ei todista saatavuutta jokaisen ajanjakson `[T, T + dT]` hetkenä. Sopivan `N`:n arvon valinta on hankalaa; suurempi `N`:n arvo antaa tarkemman tuloksen, mutta generoi enemmän liikennettä verkkoon. Arvolle ei ole määritelty eksaktia arvoa.

2.2 Yhdensuuntainen viive

Tässä luvussa käsitellään yhdensuuntaista viivettä ja arvioidaan parametrin mittaukseen liittyviä virhetekijöitä.

2.2.1 Yleistä ajasta

Ennen kuin voidaan tarkastella itse viivettä, on tiedettävä muutamia asioita koskien aikaa. IPPM-työryhmä määrittelee neljä kelloihin ja aikaan liittyvää epätarkkuustekijää:

Tahdistus (engl. *Synchronization*) tarkoittaa, kahden eri laitteen kellojen olemista samassa ajassa. Esimerkiksi laitteen A kello voi olla kaksi millisekuntia laitteen B kelloa edellä.

Tarkkuus (engl. *Accuracy*) tarkoittaa, laitteen kellon samassa ajassa olemista koordinoituun yleisaikaan (engl. *coordinated universal time* = *UTC*) verrattuna.

Resoluutio (engl. *Resolution*) tarkoittaa kellon lyöntitiheyttä. Esimerkiksi kello voi lyödä kerran kymmenessä millisekunnissa, jolloin kellon resoluutio on kymmenen millisekuntia.

Vääristymä (engl. *Skew*) tarkoittaa tarkkuuden tai tahdistuksen muutosta ajan suhteen. Esimerkiksi kello saattaa saavuttaa UTC:tä kaksi millisekuntia joka tunti. Kello A on aluksi 22 millisekuntia UTC:tä jäljessä, ja tunnin päästä enää 20 millisekuntia. Tällöin sanotaan että kellolla A on kahden millisekunnin vääristymä verrattuna UTC:hen. Vääristymä voi olla myös suhteessa johonkin toiseen kelloon.

2.2.2 Yhdensuuntainen viive

Viive (engl. *delay*) on verkon ominaisuus, joka on erittäin merkittävässä roolissa monien sovellusten kannalta. Monet sovellukset vaativat toimiakseen, että viiveen on pysyttävä jonkun tietyn rajan alapuolella. Muutoin sovellusta ei voida käyttää tai se toimii huonosti. Tämän takia tietyn verkon läpi kulkevan polun (engl. *path*), tai verkon vierekkäiseltä laitteelta toiselle kulkevan linkin (engl. *link*) paketilta kuluvan ajan, eli viiveen, selvittäminen on tärkeää.

Yhdensuuntaista viivettä ja sen mittaamista verkossa on käsitelty dokumentissa [6]. Dokumentissa määritellään parametri Yhdensuuntainen viive (engl. *One-way Delay Metric*) [6].

Yhdensuuntaisen viiveen mittaaminen edestakaisen viiveen sijasta on perusteltua, koska nykyisissä verkoissa reitti lähettäjältä vastaanottajalle voi olla eri kuin reitti takaisin. Edestakainen viive mittaakin itse asiassa kahden eri reitin yhteistä viivettä. Kun mitataan yhdensuuntaisia viiveitä, korostuvat eri reittien suorituskykyerot paremmin. Vaikka reitit olisivat molempiin suuntiin symmetriset, voi viive vaihdella paljonkin reittien välillä, koska verkon laitteiden jonot voivat olla epäsymmetriset.

Yhdensuuntainen viive -parametrin attribuutit:

Parametrin nimi: `Type-P-One-way-Delay`

- `Src` = Lähettäjän IP-osoite
- `Dst` = Vastaanottajan IP-osoite
- `T` = Aika

Parametri saa arvokseen reaalisen määrän sekunteja tai määrittelemätön (ei voitu mitata).

Parametrin arvolla `dT Type-P-One-way-Delay`:lle tarkoitetaan sitä intervallia, kun Src on lähettänyt ensimmäisen bitin `Type-P` paketista ajanhetkellä `T` Dst:lle ja Dst on vastaanottanut paketin viimeisen bitin ajanhetkellä `T + dT`. Jos arvoksi saadaan määrittelemätön, Src lähetti paketin, mutta Dst ei saanut sitä.

2.2.3 Mittaustavoista

IPPM-työryhmä määrittelee algoritmin yhdensuuntaisen viiveen mittaamiselle. Yksityiskohtaisempi määrittely jätetään mittaussovelluksen kehittäjälle, koska yksi-

tyiskohdat algoritmissa riippuvat käytettävästä mittauspaketista.

Algoritmi: Huolehditaan että Src ja Dst on tahdistettu keskenään erittäin tarkasti ja, että kellot ovat mahdollisimman oikeassa ajassa. Src-laiteessa muodostetaan `type-P`-testipaketti ja asetetaan `Src`- ja `Dst` -kentiin lähettäjän ja vastaanottajan IP-osoitteet. Pakettiin laitetaan täytettä (engl. *padding*), jotta muodostettu paketti on tarpeeksi iso mittauksen oikeellisuuden varmistamiseksi. Liian pienet pakettikoot aiheuttavat pienemmän viiveen, johtuen polulla käytetyistä pakkausmenetelmistä. Dst-laitteessa valmistaudutaan vastaanottamaan mittauspakettia. Src:ssä asetetaan aikaleima pakettiin ja lähetetään se Dst:lle. Jos paketti saapuu järkevän ajan kuluessa Dst:hen, otetaan talteen mahdollisimman nopeasti paketin saapumisaika. Aikaleima ja saapumisaika vähennetään toisistaan ja näin saadaan arvio yhdensuuntaiselle viiveelle. Jos paketti ei saapunut Dst:hen järkevän ajan kuluessa, asetetaan arvoksi määrittelemätön. Järkevän ajan määritteleminen jätetään mittaussovelluksen kehittäjälle.

2.2.4 Virheanalyysistä

Viivettä mitattaessa täytyy ottaa huomioon monia virhelähteitä. Koska viiveen arvo on ajanjakso, täytyy aikaan liittyviin epätarkkuuksiin kiinnittää erityistä huomiota. Virhe voi liittyä kahteen eri epätarkkuustekijään. Src:n ja Dst:n kelloihin tai niiden väliseen epätarkkuuteen, tai mittausajan (engl. *host time*) ja siirtoajan (engl. *wire time*) väliseen erotukseen.

Kelloihin liittyvät virhelähteet: Virheanalyyseissä käytetään Src:n kellosta nimitystä lähettäjän kello ja Dst:n kellosta vastaanottajan kello. Paketin lähetysaikaa merkitään `Tsource`:lla ja vastaanottoaikaa `Tdest`:llä.

Virhe lähettäjän kellon ja vastaanottajan kellon tahdistuksessa aiheuttaa mittaustulokseen virheen. Sanotaan, että lähettäjän kellon ja vastaanottajan kellon välillä on tahdistusvirhe (engl. *synchronization error*) `Tsynch`, jos lähettäjän kello on arvon `Tsynch` verran vastaanottajan kelloa edellä. Jos tämä tiedettäisiin tarkasti, voitaisiin virhe korjata lisäämällä `Tsynch` arvoon `Tdest - Tsource`.

Kellojen tarkkuudella on merkitystä ainoastaan selvitettäessä, milloin viive on mitattu. Tarkkuudella ei ole vaikutusta viiveen arvoon.

Kellojen resoluutio tuo tulokseen epävarmuutta. Jos laitteen kellolla on 10 millisekunnin resoluutio, tuo se 10 millisekunnin epävarmuuden tulokseen. Tässä tutkielmassa kellojen resoluutioita merkitään seuraavasti: Lähettäjän kellon resoluutiota merkitään `Rsource`:lla ja vastaanottajan `Rdest`:llä.

Vääristymä vaikuttaa kellojen tahdistukseen ja arvoon `Tsynch`. Vääristymän

vaikutusta voidaan approksimoida tahdistuksen lineaarisena funktiona ajan suhteen, lisättynä joillakin korkeamman asteen termeillä. Merkitään tätä `Esynch`:llä siten, että `Esynch` on tahdistuksen epävarmuuden yläraja ajan funktiona, eli `|Tsynch(t)| <= Esynch(t).`

Kun huomioidaan nämä kellojen virhelähteet, saadaan naivi arvio tuloksen `Tdest - Tsource` virheelle `Tsynch(t) +/- (Rsource + Rdest)`. Käyttämällä edellä mainittua notaatiota saadaan arvio kelloihin liittyvästä kokonaisvirheestä `Esynch(t) + Rsource + Rdest`.

Mittausajan ja siirtoajan erotuksen tuomat virheet: Siirtoajalla tarkoitetaan paketin todellista siirtoaikaa, eli aikaa siitä hetkestä kun ensimmäinen bitti paketista lähetetään Src:stä, siihen hetkeen kun viimeinen bitti paketista on vastaanotettu Dst:ssä. Mittausaika eroaa tästä jonkin verran, sillä mittausaika on se aika kun Src on laittanut aikaleiman pakettiin ja Dst on tämän aikaleiman saanut paketista selvitettyä. Tämä erotus tunnetaan, joten päästään lähemmäksi haluttua siirtoaikaarvoa. Tämä erotus on kuitenkin epävarma, joten se on huomioitava virheanalyysissä. Merkitään `Hsource`:lla mittausajan ja todellisen siirtoajan erotuksen epävarmuuden ylärajaa lähetyspäässä, ja `Hdest`:llä vastaanottopäässä.

Yleisesti voidaan sanoa, että mittaustulos on seuraavanlainen: mitattu arvo = todellinen arvo + systemaattinen virhe + satunnainen virhe. Jos systemaattinen virhe voidaan selvittää, voidaan se poistaa ilmoitetusta tuloksesta: Ilmoitettu tulos = mitattu arvo - systemaattinen virhe. Tästä saadaan: Ilmoitettu tulos = todellinen arvo + satunnainen virhe.

Edellisistä kappaleista saadaan nyt kokonaisvirheelle arvio: `Esynch + Rsource + Hsource + Hdest`. Kelloihin liittyvät virheet ovat pieniä nykyään, koska kellot saadaan tahdistettua tarkasti GPS[1]:n avulla. Kellojen tahdistus voidaan hoitaa myös NTP[2]:n avulla. Mittausaikaan liittyvät virheet saadaan rajattua yhdistämällä kaksi laitetta nopealla LAN[3]-yhteydellä. Toistuvat mittaukset mittaavat tällöin samaa viiveen arvoa. Käyttämällä mahdollisimman pieniä testipaketteja viiveen voidaan katsoa olevan nolla, joten toistuvien mittausten keskiarvo on systemaattisen virheen arvo ja hajonta satunnainen virhe.

2.2.5 Mittaamisesta

Mittaukset suoritetaan yleensä useamman mittauksen sarjoissa. Mittaukset hajautetaan jollekin tietylle aikavälille mahdollisimman satunnaisesti. Hajautukseen käy-

[1]Global Positioning System
[2]Network Time Protocol
[3]Local Area Network

tetään yleensä Poisson-prosessia, koska se jäljittelee verkossa todellisuudessakin esiintyvää tilannetta todella hyvin.

2.2.6 Spatiaalinen viiveen mittaaminen

Dokumentissa [7] on määritelty spatiaaliset suorituskykyparametrit viiveelle, hävikille ja viiveen vaihtelulle. Suorituskykyparametrin sanotaan olevan spatiaalinen, jos sen mittaamiseen osallistuu muitakin osapuolia kuin lähettäjä ja vastaanottaja.

Spatiaalisessa mittauksessa mittaustuloksia kerätään koko polun matkalta lähettäjältä vastaanottajalle. Mittaukseen osallistuvat siis muutkin verkon aktiivilaitteet, kuin lähettävän ja vastaanottavan asiakkaan laitteet.

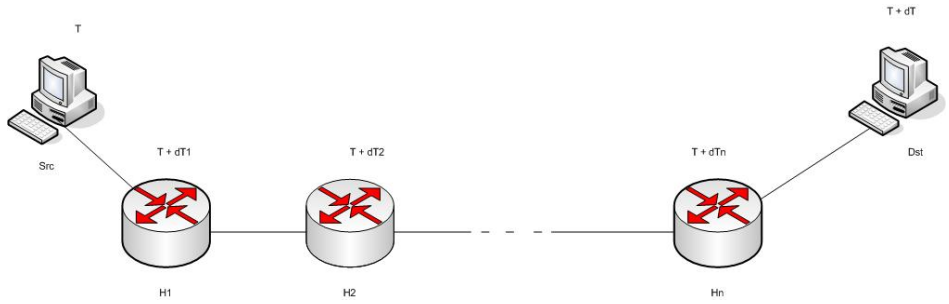
Spatiaalisen yhdensuuntaisen viiveen mittaaminen perustuu yhdensuuntaisen viiveen mittaamiseen, mutta mittaus suoritetaan kaikilla Src:n ja Dst:n välisen polun laitteilla. Parametrin attribuutit ovat seuraavanlaiset:

Parametrin nimi: `Type-P-Spatial-One-way-Delay-Stream`

- `Src` = Lähettäjän IP-osoite
- `Dst` = Vastaanottajan IP-osoite
- `i` = Polun laitteiden määrä
- `Hi` = Polun laite
- `T` = Aika
- `dT` = Yhdensuuntainen viive
- `dT1,...,dTn` = Yhdensuuntaisten viiveiden lista
- `P` = Pakettityyppi
- `<Src,H1,..,Hn,Dst>` = Kooste polusta

Parametri saa arvokseen sarjan aikoja.

Src lähettää `Type-P` -paketin Dst:lle ajanhetkellä `T` polkua `<H1,H2,...,Hn>` pitkin. Saadaan sarja aikoja `<T + dT1, T + dT2,...,T + dTn, T + dT>`, joista `dT` on yhdensuuntainen viive Src:stä Dst:hen ja jokaiselle polun laitteelle `Hi` vastaavat arvot `T + dTi`, jotka ovat ajanhetket jolloin paketti ohittaa laitteen `Hi`.



Kuva 2.1: Kuva spatiaalisen viiveen mittaamisesta

Parametrin `Type-P-Spatial-One-way-Delay-Stream` arvo polulle `<Src,H1,..,Hn,Dst>` on siis `T,dT1,...,dTn,dT`.

Tällä mittaustavalla voidaan selvittää myös yksittäisten linkkien ja alipolkujen (engl. *subpath*) viiveitä. Kun halutaan mitata jonkun polulla olevan välin viivettä, vähennetään viiveen arvot toisistaan. Esimerkiksi viive välin `[Ha, Hb]` välillä saadaan kun lasketaan `dTb - dTa`. Tässä `Hb`:n on oltava polulla myöhemmin, kuin `Ha`. Dokumentissa [7] ei oteta kantaa tapaan, jolla verkon laitteet tämän hoitavat.

2.3 Yhdensuuntainen hävikki

Viiveen lisäksi myös pakettien hävikki (engl. *packet loss*) vaikuttaa suuresti etenkin reaaliaikaisten sovellusten toimintaan. Joidenkin sovellusten toiminta on vaikeaa tai jopa mahdotonta, jos pakettien hävikki on liian suurta. Suuri hävikki vaikuttaa myös kuljetuskerroksen protokollien toimintaan.

Dokumentissa [8] on määritelty parametri yhdensuuntainen hävikki (engl. *one-way packet loss*) ja se, kuinka parametria voidaan mitata.

Yhdensuuntainen hävikki on monesti mielenkiintoisempi parametri kuin edestakaisin mitattu hävikki. Tähän pätevät pitkälti samat perusteet kuin aikaisemmin yhdensuuntaisen viiveen kanssa. Monesti toiseen suuntaan tapahtuva hävikki on paljon mielenkiintoisempaa sovelluksen toiminnan kannalta kuin toiseen. Esimerkiksi tiedonsiirrossa käyttäen TCP:tä on mielenkiintoisempaa tietää hävikin määrä tiedonsiirtosuuntaan, kuin suuntaan johon kuittausviestit (engl. *acknowledgement*) kulkevat.

2.3.1 Parametri

Yhdensuuntainen hävikki -parametrin attribuutit:

Parametrin nimi: `Type-P-One-way-Packet-Loss`

- `Src` = Lähettäjän IP-osoite
- `Dst` = Vastaanottajan IP-osoite
- `T` = Aika

Parametri saa arvokseen nollan tai ykkösen.

`Type-P-One-way-Packet-Loss` saa arvon nolla, jos Src:n ajanhetkellä `T` lähettämä paketti `Type-P` saapuu Dst:lle. `Type-P-One-way-Packet-Loss` saa arvon yksi, jos Src:n lähettää ajanhetkellä `T` paketin `Type-P` Dst:lle, mutta Dst ei saa kyseistä pakettia.

Parametrin arvot nolla ja yksi eivät ole aina välttämättä yksikäsitteisiä. Jos paketti saapuu pitkän ajan jälkeen perille, on paketti voitu jo tulkita hävinneeksi. Tällaisia tilanteita varten onkin mittausalgoritmiin sisällytettävä jonkinlainen yläraja saapumisajalle, jonka jälkeen saapuneet paketit ovat hävinneitä. Myös korruptoituneet paketit lasketaan hävinneiksi. Jos paketti on monistunut reitillä ja se saapuu useana ehjänä kopiona, lasketaan se saapuneeksi perille. Jos paketti on pilkottu, eikä sen uudelleenrakennus onnistu, lasketaan paketti hävinneeksi.

2.3.2 Mittaustavoista

Tässä, kuten viiveessä, algoritmin tarkempi määrittely riippuu käytetystä pakettityypistä. Algoritmin pääpiirteet ovat seuraavanlaiset:

Järjestetään että Src:n ja Dst:n kellot on tahdistettu. Tarkkuus riippuu käytettävästä raja-arvosta, jolla paketti määritellään hävinneeksi. Src:ssä määritellään lähettäjän ja vastaanottajan IP-osoitteet ja muodostetaan `Type-P`-paketti. Dst:ssä valmistaudutaan vastaanottamaan testipakettia. Src:ssä asetetaan pakettiin aikaleima ja lähetetään se Dst:lle. Jos paketti saapuu järkevän ajan kuluessa, on `one-way-packet-loss`:n arvo nolla. Jos paketti ei saavu perille järkevän ajan kuluessa, on `one-way-packet-loss`:n arvo yksi. Järkevä odotusaika on tässä algoritmin attribuutti.

2.3.3 Virheanalyysistä

Yhdensuuntaisen hävikin mittaamiseen liittyy kolme eri virhelähdettä. Src:n ja Dst:n kellojen tahdistus, raja-arvo paketin katoamiselle ja resurssien rajoitukset Dst:n verkkorajapinnassa tai ohjelmistossa. Näistä kaksi ensimmäistä liittyvät toisiinsa. Jos

kellojen tahdistus on pielessä, saattaa paketti, joka saapuu järkevässä ajassa, tulla lasketuksi hävinneeksi. Tai vastaavasti paketti, joka saapuu liian myöhään, saattaa tulla hyväksyttyä. Jos taas hyväksymisen raja-arvo asetetaan liian pieneksi, saattaa turhia paketteja tulla lasketuksi hävinneiksi. Tahdistukseen liittyvät virheet on käsitelty tarkemmin yhdensuuntaista viivettä käsiteltäessä.

Resurssien rajoitukset aiheuttavat virhettä mittaustuloksiin, jos paketti hylätään mittaavalla laitteella laitteen ollessa ruuhkainen. Verkko siis toimitti paketin perille asianmukaisesti, mutta laite hylkäsi paketin ja se katsottiin hävinneeksi. Tällaisen tapahtuman todennäköisyys on mittaavalla laitteella järjestettävä mahdollisimman pieneksi.

2.3.4 Spatiaalinen yksisuuntaisen hävikin mittaaminen

IPPM määrittelee myös yhdensuuntaiselle hävikille spatiaalisen mittaustavan. Spatiaalinen mittaus perustuu yhdensuuntaiseen hävikin mittaamiseen, mutta mittaukseen osallistuvat polun kaikki laitteet. Parametrin attribuutit:

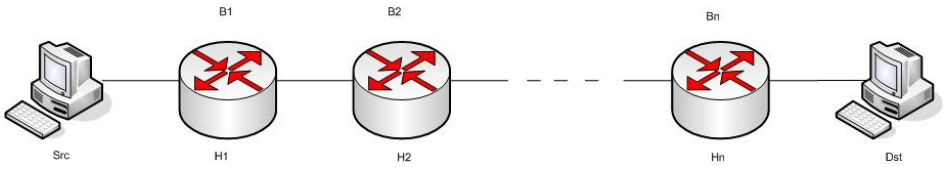
Parametrin nimi: `Type-P-Spatial-One-way-Packet-Loss-Stream`

- `Src` = Lähettäjän IP-osoite
- `Dst` = Vastaanottajan IP-osoite
- `i` = Polun laitteiden määrä
- `Hi` = Polun laite
- `T` = Aika
- `dT1,...,dTn` = Yhdensuuntaisten viiveiden lista
- `P` = Pakettityyppi
- `<Src,H1,..,Hn,Dst>` = Kooste polusta
- `B1,B2,..,Bn` = lista boolean-arvoista (tosi/epätosi)

Parametri saa arvokseen sarjan boolean-arvoja.

Src lähettää paketin ajanhetkellä `T` Dst:lle polkua `<H1,..,Hn>` pitkin. Saadaan sarja aikoja `T+dT1,...,T+dTn,T+dT`, kun paketti kulkee laitteiden `<H1,..,Hn,Dst>` kautta. `Type-P-Spatial-One-way-Packet-Loss-Stream`:lle

saadaan arvoksi sarja arvoja `B1,B2,..,Bn`, joissa jokaiselle laitteelle `Hi Bi`:n arvo 0 tarkoittaa, että `dTi`:n arvo on määrällinen ja `Bi`:n arvo 1 tarkoittaa, että `dTi`:n arvo on määrittelemätön. Dokumentti [7] ei sinänsä ota kantaa tapaan, jolla verkon laitteet tämän hoitavat.



Kuva 2.2: Kuva spatiaalisen hävikin mittaamisesta

2.4 Edestakainen viive

Edestakainen viive (engl. *Round-trip delay*) tarkoittaa sitä aikaintervallia, kun paketti lähtee lähettäjältä vastaanottajalle ja saapuu sitten takaisin lähettäjälle. Dokumentissa [9] määritellään parametri edestakaiselle viiveelle ja esitellään tapa sen mittaamiselle.

Aiemmin todettiin, että yhdensuuntainen viive on monessa suhteessa järkevämpi parametri mitattaessa IP-verkon viivettä. Vaikkakin polku ei ole molempiin suuntiin aina sama, on edestakaisella viiveellä etunsa yhdensuuntaiseen viiveeseen nähden. Edestakaisen viiveen mittaaminen on monesti helpompi järjestää, kuin yhdensuuntaisen viiveen mittaaminen. Vastaanottopään laitteeseen ei välttämättä tarvitse asentaa mitään mittaussovellusta. Voidaan käyttää esimerkiksi ICMP:n Echoa. Tulkinnan helppous on myös edestakaisen viiveen etu. Joskus edestakainen viive on juuri se mistä ollaan kiinnostuneita. Tällöin kahden yhdensuuntaisen viiveen mittaaminen ja yhdistäminen ei anna oikeaa kuvaa, koska vastaanottopään pakettien käsittelyaika ei ole sama, kuin se olisi edestakaista viivettä mitattaessa.

2.4.1 Parametri

Edestakainen viive -parametrin attribuutit:

Parametrin nimi: `Type-P-Round-trip-Delay`

- `Src` = Lähettäjän IP-osoite
- `Dst` = Vastaanottajan IP-osoite

- `T` = Aika

Parametri saa arvokseen reaalisen määrän sekunteja tai määrittelemätön (ei voitu mitata).

Edestakainen viive `Type-P-Round-trip-Delay` saa arvokseen aikaintervallin `dT`, kun Src lähettää ajanhetkellä `T` paketin `Type-P` Dst:lle ja Dst lähettää välittömästi `Type-P` paketin Src:lle takaisin, ja Src vastaanottaa tämän paketin. Jos `dT` saa arvon määrittelemätön, Dst ei ole saanut `Type-P` -pakettia, tai ei ole vastannut siihen, tai Src ei ole saanut `Type-P` -pakettia takaisin.

2.4.2 Mittaustavoista

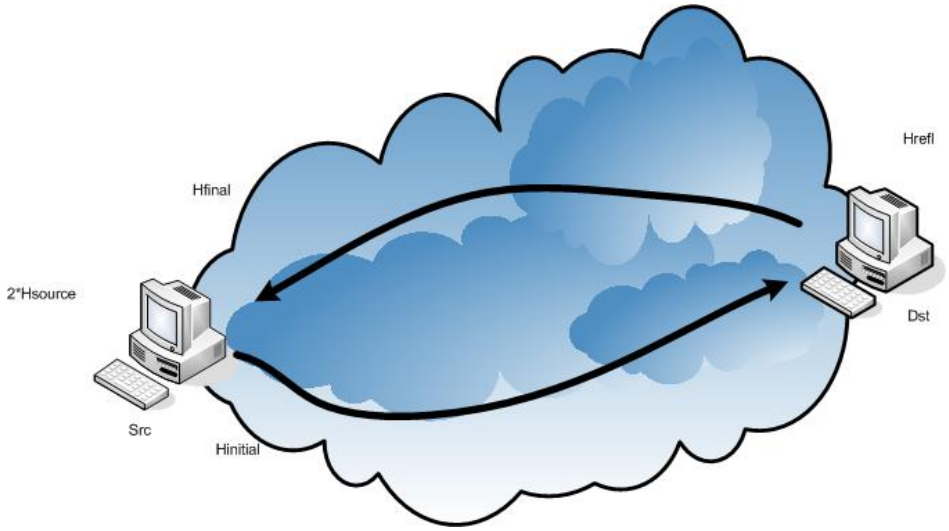
Kuten muissakin suorituskykyparametreissa, tässäkin tapauksessa algoritmin tarkka määrittely riippuu käytettävästä testipaketista. Algoritmin perusperiaate on kuitenkin seuraavanlainen:

Src määrittelee IP-osoitteet ja muodostaa `Type-P` -testipaketin, joka muodostetaan halutun kokoiseksi satunnaisella täytteellä. Paketissa täytyy olla jokin tunniste, jotta Dst:n lähettämä vastauspaketti voidaan tunnistaa. Dst:ssä valmistaudutaan vastaanottamaan paketti ja vastaamaan siihen mahdollisimman nopeasti. Src:ssä valmistaudutaan vastaanottamaan vastauspaketti. Src:ssä asetetaan pakettiin aikaleima ja lähetetään se Dst:lle. jos paketti saapuu Dst:lle, lähetetään vastauspaketti Src:lle mahdollisimman nopeasti. Jos vastauspaketti saapuu järkevän ajan kuluessa Src:lle, otetaan mahdollisimman nopeasti talteen paketin saapumisaika. Vähentämällä nämä aikaleimat keskenään saadaan edestakainen viive. Jos paketti ei saavu järkevän ajan kuluessa, saa edestakainen viive arvon määrittelemätön.

2.4.3 Virheanalyysistä

Virheanalyysi menee kutakuinkin samalla lailla kuin yhdensuuntaisessa viiveessä. Tahdistuksen kanssa ei tule samanlaisia ongelmia kuin yhdensuuntaisessa viiveessä, koska mittauksessa lähettäjä ja vastaanottaja ovat sama laite. Resoluutiosta koituva virhe voidaan naivisti laskea olevan `2*Rsource`. Mittausajan ja siirtoajan välinen erotus tuo virhettä tulokseen kuten yhdensuuntaisessakin viiveessä. Virhettä tulee Src:n lähettäessä ja vastaanottaessa pakettia. Merkitään `Hinitial`:lla ylärajaa virheelle lähetettäessä ja `Hfinal`:lla virheen ylärajaa vastaanotettaessa pakettia. Kokonaisvirhe, joka muodostuu mittausajan ja siirtoajan erotuksesta on siis: `Hinitial + Hfinal`. Virhettä tulokseen tuo myös Dst:n päässä tapahtuvat pake-

tin käsittelyt. Merkitään paketin käsittelystä tulevaa virhettä `Hrefl`:llä. Tästä saadaan kokonaisvirheelle arvioksi
`2*Rsource + Hinitial + Hfinal + Hrefl`.



Kuva 2.3: Edestakainen viive ja virheet.

2.5 Viiveen vaihtelu

Viiveen vaihtelu (engl. *Delay Variation*), eli ipdv, tarkoittaa nimensä mukaisesti sitä, kuinka paljon pakettien yhdensuuntainen viive vaihtelee lähettäjän ja vastaanottajan välisellä polulla. Viiveen vaihtelu vaikuttaa monien sovellusten toimintaan. Sovellukset, joille on tärkeää se, että paketit saapuvat tietyllä tahdilla, voivat asettaa vastaanottopuskurinsa koon sopivaksi, jos viiveen vaihtelu tiedetään.

Dokumentissa [10] määritellään suorituskykyparametri ja mittaustapa viiveen vaihtelulle. Viiveen vaihtelusta käytetään joissain yhteyksissä myös termiä *Jitter*. Tässä dokumentissa sitä ei käytetä, koska se saattaa aiheuttaa sekaannusta eri yhteyksissä saamansa merkityksen vuoksi.

2.5.1 Parametri

Viiveen vaihtelu -parametrin attribuutit:

Parametrin nimi: `Type-P-One-way-ipdv`

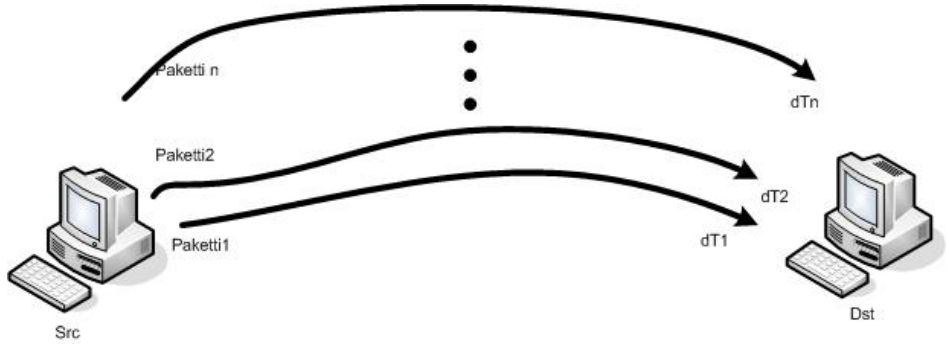
- `Src` = Lähettäjän IP-osoite

- `Dst` = Vastaanottajan IP-osoite
- `T1` = Aika
- `T2` = Aika
- `L` = Mitattavien pakettien pituus bitteinä
- `F` = Valintafunktio, jolla mittauspaketit valitaan yksiselitteisesti tietovirrasta
- `I1`, `I2` = Mittausaikajakson alku ja loppu
- `P` = Mittauspaketin tyyppi

Parametri saa arvokseen reaalisen määrän sekunteja (positiivinen, negatiivinen tai nolla) tai määrittelemätön (ei voitu mitata).

Tämän parametrin mittaukseen täytyy käyttää tietovirtaa (engl. *stream*). Olkoon meillä `Type-P` -pakettien tietovirta siten, että ensimmäinen mittauspaketti ohittaa mittauspaikan 1 ajanhetken `I1` jälkeen saaden indeksin `0` ja viimeinen paketti ohittaa mittauspaikan ennen ajanhetkeä `I2` saaden korkeimman indeksin arvon. Mittauspaikat ovat verkon laitteet, joiden välistä viivettä mitataan. Mittauspaikat voivat olla Src ja Dst, tai joku muu väli Src:n ja Dst:n väliseltä polulta.

`Type-P-One-way-ipdv` mitataan kahdesta Src:ltä Dst:lle kulkevasta paketista, jotka valitaan valintafunktiolla `F`. Parametrin arvo on `Type-P-One-way-Delay`:n arvon ajanhetkellä `T2` ja `Type-P-One-way-Delay`:n arvon ajanhetkellä `T1` erotus. `T1` ja `T2` ovat ensimmäisen ja toisen mittauspaketin lähetysajat. Ensimmäinen paketti vastaanotetaan ajanhetkellä T1 + dT1 ja toinen ajanhetkellä T2 + dT2. Näistä saadaan viiveenvaihtelulle arvo `ddT = dT2 - dT1`. Jos arvoksi saadaan määrittelemätön, Dst ei saanut jompaakumpaa tai kumpaakaan pakettia.



Kuva 2.4: Viiveen vaihtelu.

Jos kuvassa valittaisiin paketit `2` ja `n`, olisi viiveen vaihtelu `ddT = dTn - dT2`.

2.5.2 Mittaustavoista

Viiveen vaihtelun mittausalgoritmin tarkat yksityiskohdat riippuvat käytetystä mittauspaketista, mutta algoritmin periaate on seuraavanlainen. Tässä tapauksessa algoritmi mittaa viiveen vaihtelua reaaliajassa, mutta yhtä hyvin se voitaisiin määrittää aiemmin mitatuista viiveistä jälkeenpäin. Ajanhetken `I1` jälkeen valitaan Src:ssä lähettäjän ja vastaanottajan IP-osoitteet ja muodostetaan `Type-P` -paketit. Dst:ssä valmistaudutaan vastaanottamaan paketteja. Src:ssä laitetaan pakettiin aikaleima ja lähetetään se Dst:lle. Jos paketti saapuu järkevän ajan kuluessa Dst:lle, otetaan aikaleima paketista ja lasketaan arvio yhdensuuntaiselle viiveelle. Jos paketti täyttää valintafunktion `F` valintakriteerin, tallennetaan yhdensuuntaisen viiveen arvo. Muutoin jatketaan pakettien generoimista, kunnes kriteeri täyttyy tai ajanhetki `I2` ohitetaan. Src jatkaa pakettien lähettämistä. Jos paketti saapuu perille järkevän ajan kuluessa, lasketaan viiveelle arvio kuten yllä. Jos paketti täyttää valintakriteerin, tallennetaan sen viive ja lasketaan viiveen vaihtelulle arvo vähentämällä ensimmäinen viive jälkimmäisestä. Muutoin jatketaan kunnes kriteeri täyttyy tai ajanhetki `I2` ohitetaan.

2.5.3 Virheanalyysistä

Ensimmäinen oletus voisi olla, että kahden eri viiveen virheet kumoavat toisensa, kun viiveen vaihtelu lasketaan yhteen. Näin ei kuitenkaan ole, sillä osa virheistä muuttuu ajan suhteen. Tahdistusvirheet lasketaan kuten aikaisemminkin molemmille viiveille.

Virheitä jotka muuttuvat ajansuhteen ovat vääristymä ja pitkän aikavälin siirtymä (eng. *drift*). Ajatellaan, että vääristymä on ajan suhteen lineaarinen funktio. Tällöin myös kellojen poikkeama (engl. *offset*) on ajan funktio. Virhe on tällöin `e(t) = K*t + O`, jossa `K` on vakio ja `O` on poikkeama. Kun vähennetään kaksi aikaleimaa toisistaan, saadaan virhe `e(t2) - e(t1) = K*(t2 - t1)`, jossa `(t2 > t1)`. Koska pitkän aikavälin siirtymää ei voida jättää huomioimatta, mutta oletetaan että se on ajan suhteen lineaarinen funktio, tällöin vääristymä on $s(t)=M*(t^2) + N*t + s0$, jossa `M` ja `N` ovat vakioita ja `s0` on vääristymä ajanhetkellä `0`. Kun otetaan mukaan vääristymä ja pitkän aikavälin siirtymä, saadaan virheelle muoto `e(t) = O + s(t)`. Kahden aikaleiman vähennyslaskussa virhe on siis $e(t2) - e(t1) = N*(t2 - t1) + M*[(t2 - t1)^2]$.

2.5.4 Spatiaalinen viiveen vaihtelun mittaaminen

IPPM määrittelee myös viiveen vaihtelulle spatiaalisen mittaustavan. Spatiaalinen mittaus perustuu viiveen vaihtelun mittaamiseen, mutta mittaukseen osallistuvat polun kaikki laitteet. Parametrin attribuutit:

Parametrin nimi: `Type-P-Spatial-One-way-Jitter-Stream`

- `Src` = Lähettäjän IP-osoite
- `Dst` = Vastaanottajan IP-osoite
- `i` = Polun laitteiden määrä
- `Hi` = Polun laite
- `T1` = Ensimmäisen paketin lähetysaika
- `T2` = Toisen paketin lähetysaika
- `P` = Pakettityyppi
- `P1` = Ensimmäinen paketti
- `P2` = Toinen paketti
- `<Src,H1,..,Hn,Dst>` = Kooste polusta
- `<T1,dT1.1, dT1.2,..., dT1.n,dT1>` = Spatiaaliset yhdensuuntaiset viiveet ensimmäiselle paketille
- `<T2,dT2.1, dT2.2,..., dT2.n,dT2>` = Spatiaaliset yhdensuuntaiset viiveet toiselle paketille
- `L` = Pakettien pituus bitteinä

Parametri saa arvokseen sarjan aikoja.

Src lähettää paketin `P1` ajanhetkellä `T1` Dst:lle polkua `<H1,..,Hn>` pitkin. Src lähettää paketin `P2` ajanhetkellä `T2` Dst:lle samaa polkua pitkin. Saadaan sarjat aikoja `<T1,dT1.1, dT1.2,..., dT1.n,dT1>` ja `<T2,dT2.1, dT2.2,..., dT2.n,dT2>`, kun paketit kulkevat laitteiden `<H1,..,Hn,Dst>` kautta.

`Type-P-Spatial-One-way-Jitter-Stream`:lle saadaan arvoksi sarja arvoja `<T2-T1,dT2.1-dT1.1,dT2.2-dT1.2,...,dT2.n-dT1.n,dT2-dT1>`, joissa jokaiselle laitteelle `<H1, H2,..., Hn>` `dT2.i-dT1.i`:n arvo on määrällinen ,jos paketit `P1` ja `P2` saapuvat laitteelle `Hi` ajoissa `dT1.i` ja `dT2.i`. muutoin arvoksi saadaan määrittelemätön.

2.6 Hävikkimallit

Hävikkimallit (engl. *Loss patterns*) kuvaavat hävikin jakautumista. Hävikin jakautuminen vaikuttaa monien reaaliaikasovellusten, kuten VoIP-sovellusten suorituskykyyn huomattavasti. Hävikin määrän pysyessä samana, voi suorituskyky muuttua rajustikin hävikin erilaisella jakautumisella.

Dokumentissa [11] määritellään kaksi suorituskykyparametria: hävikin etäisyys ja hävikkijakso, sekä mittaustavat näille parametreille. Dokumentissa esitellään myös näiden parametrien avulla saatavia tilastotietoja hävikin jakautumisesta.

2.6.1 Parametrit

Parametreja on kaksi, ja niillä on molemmilla samat attribuutit.

Hävikin etäisyys ja hävikkijakso -parametrien attribuutit:

Parametrien nimet: `Type-P-One-way-Loss-Distance-Stream` ja `Type-P-One-way-Loss-Period-Stream`

- `Src` = Lähettäjän IP-osoite
- `Dst` = Vastaanottajan IP-osoite
- `T0` = Aika
- `Tf` = Aika
- `lambda` = Näytteenottotaajuus ajan käänteislukuna

Parametri hävikin etäisyys saa arvokseen sarjan lukupareja `<loss distance, loss>`, jossa `loss` on ykkönen tai nolla ja `loss distance` joko nolla tai positiivinen kokonaisluku.

Parametri hävikkijakso saa arvokseen sarjan lukupareja `<loss period, loss>`, jossa `loss` on kuten yllä ja `loss period` on kokonaisluku.

Hävikin etäisyys -parametrin arvo kertoo sen, kuinka monta pakettia on kulunut edellisestä hävinneestä paketista. Kun paketti todetaan hävinneeksi paketiksi luvussa 2.3 kerrotulla tavalla, otetaan paketista sen järjestysnumero ja verrataan sitä edellisen hävinneen paketin järjestysnumeroon. Hävikin etäisyys on näiden lukujen erotus. Ensimmäisen hävinneen paketin hävikin etäisyys on nolla.

Hävikkijaksoa mitattaessa aloitetaan laskuri `n` arvosta nolla. Kun paketti todetaan luvun yhdensuuntainen hävikki tavalla hävinneeksi, `loss period`:iin laitetaan `n`:n arvo. Jos paketti ei hävinnyt, `loss period`:n arvo on nolla. Laskurin `n` arvoa lisätään aina yhdellä, kun seuraava ehto täyttyy:

Olkoon `Pi` i:s paketti. `f(Pi)=1` , jos `Pi` on hävinnyt, muutoin `0`. Tällöin hävikkijakso alkaa, jos `f(Pi)=1` ja `f(P(i-1))=0`. Eli laskuria kasvatetaan, kun hävikkijakso alkaa. `loss period`:n arvo kertookin, mihin hävikkijaksoon hävinnyt paketti kuuluu.

Esimerkki: Olkoon meillä seuraavanlainen jono tuloksena yhdensuuntaisen hävikin mittaamisesta:

`<T1,0>,<T2,1>,<T3,0>,<T4,0>,<T5,1>`
`,<T6,0>,<T7,1>,<T8,0>,<T9,1>,<T10,1>`. Tässä siis paketit, jotka on lähetetty ajanhetkillä `T2, T5, T7, T9` ja `T10` on hävinneitä. Tästä saadaan jonot hävikkien etäisyyksille ja hävikkijaksoille.

```
   Type-P-One-way-Loss-Distance-Stream
= <0,0>,<0,1>,<0,0>,<0,0>,<3,1>,<0,0>,<2,1>,<0,0>,<2,1>,<1,1>
Type-P-One-way-Loss-Period-Stream
= <0,0>,<1,1>,<0,0>,<0,0>,<2,1>,<0,0>,<3,1>,<0,0>,<4,1>,<4,1>
```

2.6.2 Mittaustavoista

Mittaustapana voidaan käyttää samaa algoritmia, kuin yhdensuuntaisen hävikin mittaamiseen. Virheanalyysi on myös samanlainen kuin yhdensuuntaisen hävikin tapauksessa.

2.6.3 Tilastoja

Hävikkimallien parametreista on johdettu myös neljä muuta tilastollista parametria. Nämä parametrit ovat: huomattavuus, jaksojen määrä, jaksojen pituudet ja jaksojen välin pituus.

Huomattavuus (engl. *noticeable*) tarkoittaa, että jos kadonneen paketin ja edellisen kadonneen paketin väli on pienempi kuin delta, on katoaminen huomattava. Delta on tässä hävikkirajoite (engl. *loss constraint*).

Jaksojen määrä tarkoittaa nimensä mukaisesti hävikkijaksojen lukumäärää.
Jaksojen pituudet tarkoittavat hävikkijaksojen pituutta.
Jaksojen välin pituus tarkoittaa kahden hävikkijakson välistä välimatkaa.

2.7 Kaistan kapasiteetti

Kaistan kapasiteetin (engl. *Bandwidth Capacity*) mittaaminen kuulostaa yksinkertaiselta asialta, mutta on todellisuudessa melko monimutkainen asia. Asiaa vaikeuttaa myös yksikäsitteisen termistön puuttuminen. Kaistan kapasiteettia on käsitelty dokumentissa [12].

Jotta informaatio voitaisiin kuljettaa fyysisen median yli, täytyy se ensin koodata. Tämän jälkeen informaatio muunnetaan, mediasta riippuen, jollain systeemillä signaalijonoksi. Joissain medioissa signaalien maksimitaajuuden voidaan katsoa olevan kaistan kapasiteetti, mutta eri medioissa signaalien siirtotaajuus ja informaation kuljetuskapasiteetti on eri asia. Tässä tutkielmassa keskitytään informaation kuljetuskapasiteettiin. Informaation todellista kuljetuskapasiteettia pienentää hieman eri protokollakerroksien asettamat kehykset ja koodaus. Radiotiellä väliaineen koostumus asettaa siirrolle erityisvaatimuksia, jotka myös vähentävät todellista kapasiteettia.

2.7.1 Määritykset

Tässä luvussa käsitellään kapasiteetille asetettuja määrityksiä.

Fyysisen linkin nimellinen kapasiteetti (engl. *Nominal physical link capacity*): Tarkoittaa linkin teoreettista maksimiarvoa kuljetettavan datan määrälle. Tämä on fyysisen kerroksen parametri eikä IP-kerroksen. Tämä arvo on yleensä vakio ajan suhteen.

Loput määreet on ilmoitettu IP-kerroksen bitteinä (engl. *IP layer bits*), joka on kaikkien vastaanotettujen IP-pakettien oktettien, kehyksen ensimmäisestä kuorman viimeiseen, määrä kerrottuna kahdeksalla. Lisäksi määritellään polku `P`, joka on `n`:n pituinen ja koostuu linkeistä `(L1, L2,..,Ln)`. Linkit yhdistävät toisiinsa verkon laitteet `(N1, N2,.., Nn, Nn+1)`, joista Src on `N1` ja Dst `Nn+1`. Laitteiden `N2...Nn` ei välttämättä tarvitse olla reitittimiä. IP-kerroksen bitit tallennetaan Dst:ssä alkaen ajanhetkellä `T` ja päättyen ajanhetkellä `T+I`.

IP-kerroksen linkin kapasiteetti (engl. *IP layer link capacity*): Määritellään IP-kerroksen linkin kapasiteetti `C(L,T,I)` tarkoittamaan maksimimäärää IP-kerroksen bittejä, joka voidaan oikein siirtää Src:ltä Dst:lle linkin `L` yli aikajakson `[T, T+I]`

aikana, jaettuna `I`:llä.

IP-kerroksen polun kapasiteetti (engl. *IP layer path capacity*): Saadaan edellisestä siten, että koko polun kapasiteetti on sama kuin polulla olevan linkin, jolla on pienin kapasiteetti. Eli polun kapasiteetti on `C(P,T,I) = min 1..n C(Ln,T,I)`. Edellisten kapasiteettien mittausta varten tulisi mitattavan polun tai linkin resurssien olla vapaina. Käytännöllisempää on mitata ruuhkaisen linkin tai polun kapasiteettia.

IP-kerroksen linkin käytettävyys (engl. *IP layer link usage*): Tarkoittaa linkin `L` keskimääräistä käytettävyyttä. `Used(L,T,I)` on todellinen määrä IP-kerroksen bittejä, jotka on oikein siirretty linkin `L` yli aikajakson `[T, T+I]` aikana.

IP-kerroksen linkin keskimääräinen käyttöaste (engl. *Average IP layer link utilization*): Tarkoittaa sitä, kuinka suuri osa linkin kapasiteetista on käytössä. Käyttöaste saadaan jakamalla käytettävyys kapasiteetilla: `Util(L,T,I) = ( Used(L,T,I) / C(L,T,I) )`. Kertomalla tulos sadalla saadaan käyttöaste prosentteina.

IP-kerroksen linkin vapaana oleva kapasiteetti (engl. *IP layer available link capacity*): Kertoo sen, kuinka paljon linkin kapasiteetista on vapaana. Tämä saadaan edellisen tuloksen avulla: `Availcap(L,T,I) = C(L,T,I) * ( 1 - Util(L,T,I) )`.

IP-kerroksen polun vapaana oleva kapasiteetti (engl. *IP layer available path capacity*): Saadaan samoin, kuin aiemmin polun kapasiteetti. Polun vapaana oleva kapasiteetti on sama, kuin polulla olevan linkin, jolla on pienin vapaana oleva kapasiteetti. Eli `Availcap(P,T,I) = min 1..n Availcap(Ln,T,I)`. Vapaana oleva kapasiteetti on herkempi virheille kuin kapasiteetti, joten aika ja aikaintervalli täytyy määrittää tarkasti koska niillä on suuri vaikutus mittaustulokseen.

2.8 Hyötykuormakapasiteetti

Hyötykuormakapasiteetti (engl. *Bulk transport capacity*) kertoo verkon hyötykuorman kuljetuskyvyn kulutettuun aikaan nähden. Tähän parametriin ei lasketa mukaan kehyksiin kuluvia bittejä. IPPM-työryhmä ei ole määritellyt tähän suorituskykyparametria, mutta dokumentissa [13] selvitetään periaate parametrille.

Hyötykuormakapasiteetti tarkoittaa siis kuljetettua hyötykuormaa aikajaksossa. Kapasiteetin määrittämisestä tekee monimutkaisen se, että eri kuljetusprotokollilla on erilaiset ruuhkanhallintamekanismit. Hyötykuormakapasiteetti määrää joillekin sovelluksille, esim. www- tai FTP-sovelluksille, suoraan sovelluksen käyttöajan ja käyttäjälle näkyvän suorituskyvyn. Hyötykuormakapasiteetin mittauksen yhtey-

dessä on syytä kerätä muutakin informaatiota verkkoliikenteestä. Ideaalinen mittaustapa on siis `BTC = lähetetty data / kulunut aika`, jossa `lähetetty data` ei sisällä kehyksiin kuluvia bittejä. Hyötykuormakapasiteetin mittaus menetelmän täytyy tukea ruuhkanhallintamenetelmiä, joita TCP:kin tukee.

2.8.1 Muut mitattavat parametrit

Hyötykuormakapasiteettia mitattaessa voidaan mitata myös muita parametreja, jotka antavat lisätietoa verkosta ja käytettävästä ruuhkanhallintamenetelmästä.

Ruuhkan välttämiskapasiteetti (engl. *Congestion avoidance capacity*) tarkoittaa kuljetuskapasiteettia, joka ei vielä laukaise uudelleenlähetystä.

Tahdistuksen menetyksen välttäminen (engl. *Preservation of self-clock*) tarkoittaa sitä, että kuittausviestin katoaminen voi vaikuttaa hyötykuormakapasiteetin arvoon huomattavasti, koska katoamisesta toipuminen vie aikaa, kun uudelleentahdistus suoritetaan. Tämän vuoksi toipumiseen kulunut aika täytyy ottaa huomioon. Seuraavat neljä tilannetta voivat aiheuttaa tämän.

Menetetyt lähetysmahdollisuudet (engl. *Lost transmission opportunities*) ovat sellaisia tilanteita, joissa kuittausviestin saavuttua ajastin laukeaa ennen kuin seuraavaa pakettia ehditään lähettämään. Tämän vuoksi kapasiteetin tilasta on kerättävä tietoa, jotta tulevaisuudessa ruuhkanhallintamenetelmillä tällaisia tapauksia voitaisiin välttää paremmin.

Kokonaisen ikkunan katoaminen (engl. *Losing entire window*) voidaan havaita, jos viimeinen tapahtuma ennen ajastimen laukeamista oli lähetys. Tällaisessa tapauksessa käynnistyy uudelleenlähetys ja ikkunan koko alustetaan uudelleen. Tämä vaikuttaa merkittävästi hyötykuormakapasiteettiin. Mittausmenetelmien on tämän vuoksi raportoitava kokonaisten ikkunoiden menettämisistä.

Urhea tahdistuksen säilytys (engl. *Heroic clock preservation*) tarkoittaa joidenkin ruuhkanhallinta-algoritmien tapaa toimia tilanteissa joissa menetetään paljon paketteja. Algoritmit yrittävät saada itsensä uudestaan tahdistettua, kun toiset algoritmit eivät niin tekisi. Tämä vaikuttaa hyötykuormakapasiteetin arvoon, joten mittausmenetelmän täytyy raportoida katoamisten jakaumasta, jotta muiden ruuhkanhallinta-algoritmien suorituskykyä voitaisiin arvioida.

Ajastimen väärät laukeamiset (engl. *False timeouts*) ovat sellaisia, joissa uudelleenlähetysajastin laukeaa ennen kuin kuittausviesti saapuu jollekin aikaisemmin lähetetylle paketille. Tällaiset tilanteet täytyy ottaa huomioon mittausmenetelmässä.

Tietovirrasta riippuvat polun ominaisuudet (engl. *Flow based path properties*)

vaikuttavat myös hyötykuormakapasiteettiin, joten nämä ominaisuudet on otettava huomioon mittausmenetelmissä.

Lähetystahdin tarkastus on tärkeää, jottei käy niin että, mittausalusta on itse polun pullonkaula. Jos lähetyspäässä alkaa muodostua merkittävää jonoa lähetyspuskureihin, täytyy lähetystahtia muuttaa.

Kaksisuuntainen mittaaminen. Jos käytetään mittaustekniikkaa, jossa päätelaitteet eivät toimi yhteistyössä, eli voidaan mitata vain edestakainen kapasiteetti, täytyy molempien suuntien kapasiteetit erotella tuloksesta jollain muulla tavalla.

2.9 Pakettien uudelleenjärjestely

Pakettien uudelleenjärjestelyllä (engl. *Packet reordering*) tarkoitetaan sitä tilannetta, jossa paketit saapuvat perille eri järjestyksessä kuin ne on lähetetty. Pakettien uudelleenjärjestelyä ja sitä mittaava suorituskykyparametri on esitelty dokumentissa [14]. Dokumentissa määritellään myös muita parametreja liittyen uudelleenjärjestelyyn.

Pakettien uudelleenjärjestely on joillekin sovelluksille, etenkin reaaliaikaisille verkkosovelluksille, suuri haitta. Paketit jotka saapuvat myöhässä voidaan hylätä. Tämä näkyy esimerkiksi reaaliaikaisessa videolähetyksessä häiriönä. Pakettien saapumisjärjestys saattaa muuttua matkalla monesta syystä. Esimerkiksi jos kaksi eri reittiä eroavat vain hieman siirtoajassa, voivat pidempää reittiä kulkevat paketit tulla myöhemmin perille.

2.9.1 Parametri

Pakettien uudelleenjärjestely -parametrin attribuutit:

Parametrin nimi: `Type-P-Reordered`

- `Src` = Lähettäjän IP-osoite
- `Dst` = Vastaanottajan IP-osoite
- `SrcTime` = Paketin lähtöaika
- `s` = Pakettin vuoronumero
- `NextExp` = Seuraavana odotettava vuoronumero

Parametri saa arvokseen tosi tai epätosi.

Jos ajanhetkellä `SrcTime` lähetetty paketti `s` todetaan uudelleenjärjestellyksi vertaamalla vuoronumeroa `NextExp`-arvoon, on `Type-P-Reordered`:n arvo tosi. Muutoin arvo on epätosi. Uudelleenjärjestely todetaan, jos `s < NextExp`. Tällöin `NextExp`:n arvo ei muutu. Jos arvoksi saadaan epätosi, eli `s >= NextExp`, tulee `NextExp`:n arvoksi `s+1`. Seuraavana odotettavan paketin arvo ei voi pienentyä, koska käytetään kasvavaa vuoronumerointia. Tämä takaa edellisen menetelmän toimivuuden.

2.9.2 Muut parametrit

Dokumentissa [14] määritellään myös muita parametreja jotka liittyvät pakettien uudelleenjärjestelyyn. Parametrit kuvaavat muun muassa uudelleenjärjestelyn laajuutta. Seuraavassa on lueteltuna nämä parametrit.

Uudelleenjärjestelyn suhde (engl. *Reordering ratio*) kertoo, kuinka paljon paketteja on uudelleenjärjestelty suhteessa oikeassa järjestyksessä saapuneisiin. **Uudelleenjärjestelyn etäisyys** (engl. *Reorderin extent*) kuvaa sitä kuinka kaukana uudelleenjärjestelty paketti on omalta paikaltaan. Tällä voidaan arvioida esimerkiksi tarvittavien puskurien kokoa, jotta järjestys voidaan palauttaa. **Myöhästymisaika** (engl. *Late time*) kuvaa nimensä mukaan uudelleenjärjestellyn paketin etäisyyttä omalta paikaltaan ajallisesti. **Uudelleenjärjestelyn vastine tavuina** (engl. *Byte offset*) on vaihtoehtoinen tapa määrittää esimerkiksi tarvittava puskurien koko. **Epäjatkuvuusväli** (engl. *Gap between reordering discontinuities*) on parametri, jota mitataan ajallisesti ja paikkaan liittyen. Parametrilla mitataan uudelleenjärjestellyn paketin aiheuttaman epäjatkuvuuden kestoa. **Uudelleenohjausvapaat jaksot** (engl. *Reordering-free runs*) ovat jaksoja, jolloin paketit saapuvat oikeassa järjestyksessä.

3 Passiiviset mittaustekniikat

Tässä luvussa esitellään erilaisia passiivisia IP-verkon mittaustekniikoita. SNMP on esitelty luvussa 3.1, RMON luvussa 3.2 ja NetFlow luvussa 3.5. Passiivisilla tekniikoilla tarkoitetaan tässä yhteydessä niitä mittaustekniikoita, jotka eivät tuota mittauksia varten testiliikennettä tuotantoverkkoon.

3.1 SNMP

SNMP on TCP/IP -verkoissa yleisesti käytetty verkonhallintaprotokolla. Termillä SNMP tarkoitetaan yleensä joukkoa verkonhallintastandardeja, joka sisältää SNMP-protokollan, hallintatietokantojen (MIB[1]) määrittelyn ja hallintatietojen rakenteen määrittelyn (SMI[2]). SNMP:n kehitys aloitettiin 80-luvun lopulla täyttämään kasvavia verkonhallinnan tarpeita. SNMP kehittyi ja yleistyi nopeasti, se hyväksyttiin jo 1990 Internet-standardiksi, ja myös laitevalmistajat alkoivat nopeasti tukea sitä. SNMP onkin TCP/IP:n tavoin ylittänyt elinikäennusteensa moneen kertaan. Nykyään käytännössä kaikki verkkolaitteet (reitittimet, kytkimet, hubit, työasemat, palvelimet) tukevat ainakin jossain määrin SNMP:tä. [15]

3.1.1 Versiot

SNMP:stä on kehitetty useita versioita, joiden määritykset löytyvät taulukon 3.1 mukaisista RFC[3]-asiakirjoista. Ensimmäinen niistä on versio 1, joka kehitettiin 1980-luvun lopulla. Se oli kuitenkin liian rajoittunut täyttääkseen edes tuolloisia verkonhallinnan tarpeita. Lisäksi sen tietoturvallisuus oli pahasti puutteellinen. Tietoturvasta vastasivat vain kaksi yhteisötunnusta (engl. *community*), toinen lukuoikeuksille, toinen kirjoitusoikeuksille. Nämä yhteisöt olivat kaikille samat ja kulkivat verkon yli salaamattomana. [15]

Vuoden 1992 lopulla aloitettiin alkuperäisen SNMP:n laajennuksen määrittelyt. Tarkoituksena oli lisätä SNMP:hen parempi tietoturva ja lisätä sen toiminnallisuutta. Version 2 ehdotukset olivat valmiita 1993 keväällä. Vuonna 1996 standardin eh-

[1] Management Information Base
[2] Structure of Management Information
[3] Request for Comments

dotuksesta tietoturvaominaisuudet kuitenkin jätettiin erimielisyyksien ja kiireen takia pois. Näin syntynyttä versiota kutsutaan SNMP:n versioksi 2c tai yhteisöpohjaiseksi SNMP versio 2:ksi (engl. *Community-Based SNMPv2*). Tässä käytettiin edelleen ensimmäisen version tavoin yhteisöpohjaista tunnistusta. Vaikka versio 2c ei olekaan virallinen standardi, on se nykyään tuetuin ja käytetyin SNMP-versio. [15]

Version 2 julkistamisen jälkeen työ tietoturvaominaisuuksien parantamiseksi jatkui edelleen. Lopulta tammikuuhun 1998 mennessä IETF[4]:n SNMPv3-työryhmä oli julkaissut ehdotukset SNMP version 3 standardista. Lopulta SNMPv3 hyväksyttiin standardiksi vuonna 2002. SNMPv3 standardit on määritelty RFC:ssä 3411-3418. Kolmas versio toi mukanaan käyttäjäkohtaisen tietoturvamallin[5], jossa jokainen käyttäjä tunnistetaan erikseen. Viestien muuttumattomuus tarkistetaan MD5:n tai SHA-1:n avulla. Lisäksi viestit voidaan salata symmetrisesti DES:in avulla. Kolmas versio tarjoaa myös mahdollisuuden rajoittaa eri käyttäjien näkymiä hallintatietokantojen tietoihin [6]. Koska tulevat tietoturvaominaisuudet on otettu SNMP aiempien versioiden kohdalla huomioon, ei protokollan PDU:n määrityksiin tarvinnut koskea. [16]

Taulukko 3.1: SNMP:hen liittyvät RFC:t

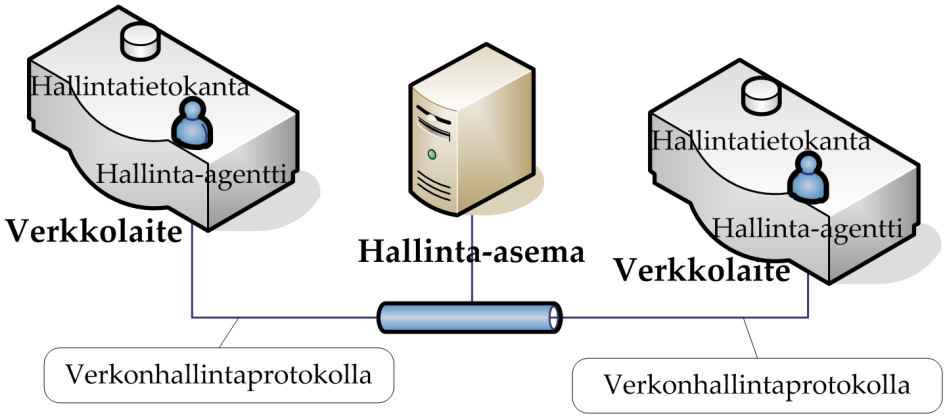
Versio	**RFC**
1	1155-1157
2	1441-1452
2c	1901-1908
3	3411-3412

3.1.2 Arkkitehtuuri

Kuvassa 3.1 on esitetty SNMP:n arkkitehtuuri, jossa ovat mukana seuraavat avainelementit:

- hallinta-asema
- hallinta-agentti
- hallintatietokanta (MIB)
- verkonhallintaprotokolla.

[4]Internet Engineering Task Force
[5]User-Based Security Model (USM)
[6]View-based Access Control Model (VACM)



Kuva 3.1: SNMP-arkkitehtuuri.

Hallinta-asema tarjoaa käyttäjälle työkalut, joiden avulla käyttäjä voi valvoa ja hallita verkkoa.

Hallinta-agentti taas sijaitsee verkon laitteilla (reititin, hubi, kytkin) ja mahdollistaa niiden hallinnan hallinta-asemalta. Hallinta-agentti vastaa hallinta-aseman kyselyihin ja toimintapyyntöihin. Agentti voi myös lähettää hallinta-asemalle hälytyksiä tärkeistä tapahtumista.

Hallintatietokanta kuvaa verkon resurssit joukkona muuttujia, joista jokainen edustaa jotain agentin ominaisuutta. Hallintatietokanta tarjoaa täten hallinta-asemalle liitynnän agentin ominaisuuksiin. Hallinta-asema valvoo verkkoa hakemalla hallintatietokannan muuttujien arvoja tai muuttaa verkon tilaa muokkaamalla näitä muuttujia.

Hallinta-asema ja hallinta-agentti keskustelevat keskenään **verkonhallintaprotokollan** avulla. Tämä protokolla on SNMP, jolla ovat seuraavat metodit:

- **get**: tällä metodilla hallinta-asema voi noutaa agentin muuttujan arvoja
- **set**: tällä metodilla hallinta-asema voi asettaa muuttujille uusia arvoja
- **trap**: tällä metodilla agentti voi viestiä hallinta-asemaa verkon merkittävistä tapahtumista

SNMP on sovellustason protokolla, joka toimii UDP-protokollan päällä. Täten hallinta-asemien ja -agenttien tulee tukea SNMP-, UDP- ja IP-protokollia. Koska UDP on yhteydetön, on myös SNMP yhteydetön protokolla. [15]

3.1.3 SNMP:n käyttö

SNMP:tä voidaan käyttää muun muassa seuraaviin tehtäviin:

- verkkolaitteen liityntöjen tietojen kerääminen
- verkkolaitteiden liityntöjen läpi kulkeneen liikenteen määrän valvonta
- verkkolaitteen saavutettavuuden valvonta
- verkkolaitteen resurssien valvonta

SNMP-kyselyt (engl. *SNMP polling*) sopivat hyvin laskureille, jotka kasvavat jatkuvasti (esimerkiksi liityntöjen liikennemäärät) tai laskureille jotka sisältävät keskiarvon joltain aikaväliltä (esimerkiksi prosessorikuorman keskiarvo viimeiseltä viideltä minuutilta). Sen sijaan laskurit, jotka sisältävät hetkellisiä arvoja (jonon pituus, hetkellinen prosessorikuorma), eivät sovellu hyvin kyseltäviksi.

SNMP-hälytykset (engl. *trap*) soveltuvat hyvin liityntöjen saavutettavuuden valvomiseen. Kun liitynnän tila muuttuu, SNMP-agentti lähettää hälytyksen. Hälytykset ovat suhteellisen rajoittuneita ja kiinteästi määriteltyjä, joten niillä ei voi valvoa laitteita kovin monipuolisesti. Tämän vuoksi on määritelty RMON-laajennus, josta lisää luvussa 3.2.

Koska SNMP tarkkailee liikennettä yhdessä pisteessä, päästä-päähän mittaukset eivät ole mahdollisia. Esimerkiksi viiveitä tai viiveen vaihteluita ei voi siis mitata SNMP:llä. Pakettien uudelleenjärjestymistä ei voi havaita SNMP:n avulla, sillä reitittimetkään eivät yleensä tunnista niitä. [17] Lisäksi SNMP valvoo verkkoliikennettä pääsääntöisesti liityntöjen tasolla, joten sillä ei voi eritellä liikennettä. Laitteet tarjoavat kuitenkin usein loogisia liityntöjä, joilta saa samat tiedot kuin fyysisiltä liitynnöiltä. Tällainen looginen liityntä voidaan muodostaa muun muassa seuraavien tekniikoiden avulla: VLAN[7], VPN[8] tai LSP[9].

Liityntöjä valvotaan niiden indeksinumeron perusteella. Laitteen liitynnät indeksoidaan laitteen käynnistyessä. Mikäli laitteeseen lisätään uusia liityntöjä lennossa, saavat uudet liitynnät indeksinumerot numeroinnin lopusta. Kun laite käynnistetään tämän jälkeen uudelleen, saattavat laitteen liityntöjen indeksoinnin muuttua täysin uusien liityntöjen johdosta. Tällaisen indeksoinnin siirtymisen voi estää ainakin Ciscon laitteissa ottamalla käyttöön `ifIndex Persistence` -ominaisuuden.

[7] Virtual Local Area Network
[8] Virtual Private Network
[9] Label Switched Path

3.2 RMON

RMON on oleellisesti vain hallintatietokannan (MIB) määritys, mutta tarjoaa silti merkittävää lisäarvoa muuttamatta itse SNMP-protokollaa. SNMP ei mahdollista hallinta-agentin käskemistä verkon yli, ainoastaan hallintatietokannan muuttujien arvojen muuttamisen. Hallinta-agentti voi kuitenkin tarkkailla hallintatietokannan muuttujien tiloja ja mukauttaa järjestelmän toimintaa näiden tilojen muuttuessa. RMON:in hallintatietokanta määrittelee joukon tällaisia tilamuuttujia. Järjestelmää, joka toteuttaa RMON-hallintatietokannan, kutsutaan RMON-tutkaimeksi (engl. *RMON probe*). Tutkain sisältää ohjelman, jota kutsutaan agentiksi. Tämä agentti vastaa sekä RMON:in, että SNMP:n toiminnallisuudesta.

RMON-hallintatietokantaan on mahdollista asettaa sääntöjä, joiden noudattamista tutkain valvoo. Mikäli sääntöjen rajat ylittyvät, tutkain lähettää SNMP-hälytyksen hallinta-asemalle. Koska tutkain lähettää hälytyksiä vain vikatilanteissa, verkon kuormitus on jatkuvaa SNMP-kyselyä pienempää. RMON soveltuu myös kyselyitä paremmin jatkuvasti vaihtelevien arvojen tarkkailuun. Esimerkiksi jonon koko poikkeaa vain hetkellisesti nollasta. SNMP-kyselyillä tämä jäisi todennäköisesti huomaamatta, mutta RMON-tutkain voi kuitenkin havaita tämän ja lähettää hälytyksen hallinta-asemalle.

RMON:in avulla voidaan myös kaapata paketteja. Tutkaimelle määritellään suodattimet, jotka määrittävät mitkä paketit tulee kaapata. Hallinta-asema voi noutaa nämä paketit myöhemmin SNMP:n avulla. Pakettien kaappausta tulee käyttää säästeliäästi ja vain tarvittaessa, sillä se voi kuormittaa paljon sekä tutkainta että verkkoa. Pakettien kaappaus sopiikin parhaiten ongelmatapausten ratkaisemiseen, kun suodattimet voidaan rajata tarkasti.

RMON valvoo liikennettä vain kahdella alimmalla tasolla, eli se voi tunnistaa ainoastaan oman verkkosegmenttinsä laitteet niiden MAC[10]-osoitteiden perusteella. Tätä puutetta korjaamaan on määritelty RMON2, josta lisää seuraavassa luvussa.

RMON:in hallintatietokanta sisältää kymmenen ryhmää [18]:

statistics -ryhmä sisältää perustilastot jokaisesta valvotusta aliverkosta.

history -ryhmään tallennetaan säännöllisesti tilastotietoa liitynnöistä.

alarm -ryhmä sisältää hälytysrajat, joiden perusteella SNMP-hälytykset lähetetään.

host -ryhmää käytetään tiettyjen lähiverkon laitteiden tilastotietojen keräämiseen.

[10]Media Access Control

hostTopN -ryhmä sisältää listan aliverkon kärkilaitteista. Esimerkiksi eniten tiettynä päivänä tietoa siirtäneet laitteet.

matrix -ryhmään tallennetaan MAC-parien välisten yhteyksien tiedot matriisimuodossa.

filter -ryhmä sisältää ehdot, joiden perusteella paketit jaetaan kanaviin.

capture -ryhmä määrittää mitkä filter-ryhmän määrittelemät kanavat kaapataan sekä kaappauksen yksityiskohdat.

event -ryhmä sisältää tapahtumien määritykset. Tapahtuma voi esimerkiksi lähettää hälytyksen tai laittaa kanavan päälle.

tokenRing -ryhmä sisältää Token Ring -verkolle ominaista lisätilastoa.

3.3 RMON2

Koska alkuperäinen RMON valvoi ainoastaan kahdella alemmalla tasolla, oli olemassa selvä tarve ylemmän tason valvonnalle. RMON2 standardi valmistui 1997 ja se toimii sovelluskerroksella asti (tasot 3-7). Tämä mahdollistaa valvonnan IP-osoitteiden ja sovellusprotokollien perusteella. RMON2 yhteydessä sovellusprotokollilla tarkoitetaan kaikkia tason 3-7 protokollia.

RMON2 hallintatietokanta sisältää yhdeksään uutta ryhmää [19]:

protocolDir (Protocol Directory) -ryhmä sisältää tiedot tutkaimen tuntemista protokollista.

protocolDist (Protocol Distribution) -ryhmä ryhmä sisältää tuettujen protokollien tavu- ja pakettilaskurit.

addressMap (Address Map) -ryhmä yhdistää verkko-osoitteet laitteen liityntöihin.

n1Host (Network-Layer Host) -ryhmä kerää tietoa laitteista verkkotason osoitteiden perusteella.

n1Matrix (Network-Layer Matrix) -ryhmä sisältää tiedot verkkotason laitteiden välisistä yhteyksistä.

a1Host (Application-Layer Host) -ryhmä sisältää tunnettujen protokollien tiedot jokaiselta havaitulta verkkotason laitteelta.

a1Matrix (Application-Layer Matrix) -ryhmä sisältää tiedot tunnettujen protokollien välisistä yhteyksistä.

usrHistory (User History Collection) -ryhmä mahdollistaa käyttäjän vapaasti määrittämien tilastojen keräämisen, toisin kuin RMON1:ssä, jossa kerättävät tilastotiedot olivat ennalta määrätyt.

probeConfig (Probe Configuration) -ryhmä sisältää tutkaimen asetuksia, mahdollistaen helpomman etähallinnan.

3.4 Esimerkkejä hallintatietokannoista

Tässä alaluvussa esitellään muutamia esimerkkejä hallintatietokannoista. Ensimmäiset viisi RFC-dokumenteissa esiteltyä hallintatietokantaa ovat pelkkiä määrityksiä, eikä niille löydy vielä toteutuksia. Näiden dokumenttien pohjalta voitaisiin kuitenkin laajentaa standardien mittaustekniikoiden ominaisuuksia. Osa näiden dokumenttien ominaisuuksista on jo toteutettu valmistajien omissa mittaustekniikoissa, kuten Ciscon IP SLA:ssa. Kaksi viimeistä hallintatietokantaa ovat Ciscon kehittämiä ja niille on tuki Ciscon laitteissa.

Taulukko 3.2: MIB-esimerkkejä

Dokumentti	**RFC**	**Selitys**
IP Performance Metrics (IPPM) Metrics Registry	4148	Suorituskykyparametrien rekisteri
Application Performance Measurement MIB	3729	Sovellusten suorituskyky -MIB
Transport Performance Metrics MIB	4150	Transaktioiden eritelty suorituskyky -MIB
Definition of Managed Objects for Synthetic Sources for Performance Monitoring Algorithms	4149	Aktiivisten tutkainten hallinta -MIB
Real-time Application Quality-of-Service Monitoring (RAQMON) Framework	4710	Reaaliaikaisten sovellusten suorituskyky -runko/MIB
Class-Based QoS MIB (CBQoSMIB)	-	Ciscon palvelunlaatu-MIB
RTTMON MIB	-	Ciscon päästä-päähän -MIB

'**IP Performance Metrics (IPPM) Metrics Registry**' [20] -dokumentti määrittää rekisterin suorituskykyparametreille, jotka esiteltiin luvussa 2. Esitelty rekisteri ei ole täydellinen hallintatietokannan määritys, sillä siinä määritetään ainoastaan parametreille identiteetit (`OBJECT IDENTITIES`) hallintatietokannassa. Dokumentti kertoo myös kuinka tulee toimia uusia parametreja tähän rekisteriin lisättäessä ja kehottaa kaikkia suorituskykyparametreja lisättäväksi tähän samaan rekisteriin.

Dokumentti '**Application Performance Measurement MIB**' [21] määrittelee hallintatietokannan loppukäyttäjien kokemalle sovellusten suorituskyvylle (APM-MIB). Tämä tarjoaa siis todellisen päästä-päähän näkymän verkon suorituskyvystä. Loppukäyttäjä havaitsee sovelluksen toimivuudesta vain kaksi asiaa: saavutettavuuden (engl. *availability*) ja palvelun nopeuden (engl. *responsiveness*). Jokaiselta transaktiolta mitataan edellä mainitut parametrit. Jokainen transaktion tunnistetaan sen sovelluksen, asiakkaan ja palvelimen perusteella. APM-MIB tarjoaa standardin tavan säilöä näitä transaktiovoita ja jatkaa siten RMON-arkkitehtuurin laajentamista. Se tarjoaa myös hälytysrajat, joiden avulla voidaan hälyttää liian suurista käyttäjien viiveistä.

'**Transport Performance Metrics MIB**' [22] -dokumentti määrittelee hallintatietokannan (TPM-MIB), joka laajentaa APM-MIB:iä tarjoamalla sovellustasoa tarkemman näkymän transaktioihin. Yksittäisen sovelluksen transaktio koostuu yleensä erillisistä käyttäjälle näkymättömistä osista. TPM-MIB mahdollistaa näiden osien erillisen valvonnan. Lisäksi se tarjoaa lisäparametreja ja -tilastoa ja mahdollistaa muiden määrittämien parametrien käytön viittauksilla.

Dokumentissa '**Definition of Managed Objects for Synthetic Sources for Performance Monitoring Algorithms**' [23] määritellään hallintatietokannan aktiivisten tutkainten (engl. *active probe, synthetic source*) etähallinnalle. Tämä tietokanta laajentaa siten RMON:in toiminnallisuutta aktiivisten mittausten suuntaan. Tämä tietokanta on kehitetty tiiviissä yhteistyössä APM-MIB:in [21] ja TPM-MIB:in [22] kanssa, jotka määrittelevät mitä parametreja aktiivisilla tutkaimilla voidaan mitata.

Dokumentti '**Real-time Application Quality-of-Service Monitoring (RAQMON) Framework**' [24] määrittää rungon reaaliaikaisten sovellusten palvelunlaadun mittaamiselle. RAQMON mahdollistaa päätelaitteiden ja sovellusten reaaliaikaisen palvelunlaadun raportoinnin. RAQMON-runko määrittelee mitattavat suorituskykyparametrit, raporttien tietomallin ja hallintatietokannan mittausten tallentamiseen.

Ciscon **Class-Based QoS MIB** (CBQoSMIB)[11] -hallintatietokanta sisältää tilastollisia muuttujia palvelunlaadusta. Siitä saadaan samat tiedot palveluluokittain kuin

[11] http://tools.cisco.com/Support/SNMP/do/BrowseMIB.do?local=en&mibName=CISCO-CLASS-BASED-QOS-MIB

`interfaces`-taulusta liityntäkohtaisesti. Lisäksi nämä tiedot saadaan ennen ja jälkeen palveluluokkiin jakamisen.

RTTMON MIB on Ciscon luoma hallintatietokanta, johon voidaan tallentaa mittaustietoja päästä-päähän -liikenteestä. Hallintatietokantaa käytetään IP SLA:n yhteydessä. Tämä hallintatietokanta sijaitsee IP-SLA mittauksia suorittavalla reitittimellä. IP-SLA:sta on kerrottu enemmän luvussa 4.3. [25]

3.5 NetFlow

IP-verkkojen valvonnassa käytettiin pitkään kaistanvalvontaan pelkkää SNMP:tä. Vaikka SNMP sopiikin hyvin kapasiteetin valvontaan, sillä ei voi tunnistaa verkkoa käyttäviä sovelluksia tai luokitella liikennettä. Tarkempi tieto IP-verkkojen kaistan käytöstä on kuitenkin nykyään tarpeen. Verkkoliityntöjen paketti- ja tavulaskurit ovat hyödyllisiä, mutta ymmärrys verkkoliikenteen lähde- ja kohdeosoitteista sekä sovelluksista on korvaamatonta [26].

Tähän tarkempaan verkkoliikenteen valvontaan Cisco kehitti NetFlow-teknologian vuonna 1996. Sitä voi käyttää muun muassa seuraaviin asioihin [27]:

- verkon käytön valvontaan
- sovellusten käytön valvontaan ja profilointiin
- verkon tuottavuuden ja verkon resurssien käytön valvontaan
- verkon muutoksien vaikutusten seurantaan
- verkon epänormaaliuksien ja haavoittuvuuksien havainnointiin
- verkon suunnitteluun
- pitkän ajan sääntöjen noudattamisen valvontaan
- käyttäjien valvotaan, profiloitiin ja laskutukseen.

Reitittimet ja kytkimet seuraavat niiden läpimenevää IP-liikennettä ja pitävät yllä tilastoa samaan tietovuohon kuuluvista paketeista. Ciscon määrittelee vuon (engl. *flow*) IP-paketeiksi, jotka jakavat seuraavat ominaisuudet [26]:

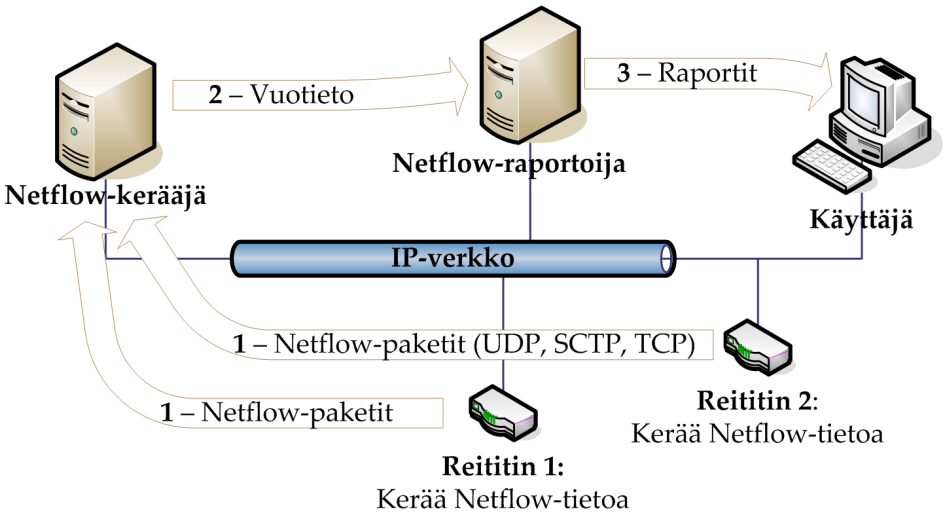
- IP-kohdeosoite
- IP-lähdeosoite

- kohdeportti
- lähdeportti
- IP-protokolla
- palveluluokka (COS)
- verkkolaitteen liityntä (interface).

Kaksi viimeistä kenttää ovat valinnaisia, eikä niitä käytetä aina luokittelussa.

3.5.1 Arkkitehtuuri

NetFlow ympäristössä on usein käytössä kolmitasoinen arkkitehtuuri, joka on esitetty kuvassa 3.2. Tässä arkkitehtuurissa ovat mukana sensori, kerääjä ja raportoija. Sensorina toimii yleensä reititin, joka kerää vuotiedot läpimenneestä IP-liikenteestä omaan välimuistiinsa. Yhteenvedon välimuistissa olevista vuotiedoista voi usein nähdä reitittimen komentorivikäyttöliittymän[12] kautta. Sensori lähettää määräajoin keräämänsä vuotiedot NetFlow paketteina kerääjälle. Kerääjä puolestaan lähettää usean sensorin vuotiedot raportoijalle, joka voi olla myös samalla koneella. Raportoija muodostaa vuotiedoista halutunlaisia raportteja.



Kuva 3.2: NetFlow-arkkitehtuuri.

[12]Command Line Interface (CLI)

3.5.2 Näytteistäminen ja tiivistäminen

Koska nykyisissä nopeissa tietoverkoissa tieto liikkuu nopeasti, kertyy siitä myös paljon mittaustietoa. Siksi kerättävää tietoa täytyy jotenkin rajoittaa. Yksi keino tähän on pakettien näytteistäminen (engl. *sampling*). Näytteistäessä sensori tarkistaa vain pienen osan paketeista ja muodostaa tämän perusteella arvion kokonaisliikennemääristä. Näytteistäminen voidaan tehdä deterministisesti, satunnaisesti tai aikaperusteisesti.

Deterministisessä näytteistyksessä valitaan järjestyksessä joka N:s paketti. Satunnaisessa näytteistyksessä valitaan järjestyksessä *keskimäärin* joka N:s paketti. Aikaperusteisessa näytteistyksessä taas valitaan N:n millisekunnin välein paketti mukaan laskentaan.

Satunnaisnäytteistämistä pidetään näistä parhaana [27]. Deterministinen näytteistys voi jättää tutkimatta tietyn vuon paketit kokonaan, jos molempien jaksot asettuvat sopivasti. Yleisesti näytteistäminen sopii hyvin tiivistämiskeinoksi verkon tai kapasiteetin suunnittelun yhteydessä, kun tieto yksittäisistä paketeista ei ole tärkeää. Esimerkiksi tietoturvauhkien analysoinnin yhteyteen näytteistys ei taas sovi. Näytteistäminen säästää sensorin laskentakuormaa selvästi, sekä hieman myös kaistaa tietovoiden lähetyksessä. Tuki näytteistykselle löytyy NetFlown versiosta viisi alkaen.

Sensorilla voidaan myös tiivistää (engl. *aggregate*) kerääjälle lähetettävää tietoa. Sensori voi yhdistää kerättyjä voita ja lähettää nämä tiivistetyt tiedot kerääjälle. Esimerkiksi vastaanottajan osoitteet ja portit voidaan jättää huomioimatta, jolloin ainoastaan lähettäjä ja protokolla erottavat vuot toisistaan. Näin säästetään kaistaa sekä kerääjän levytilaa ja laskentataakkaa. Koska NetFlow-vuotietoa ei säilötä lähettämisen jälkeen, on näin sensorin tiivistämistä voista vaikea saada jälkikäteen tarkempaa tietoa. Sensori voi kuitenkin kerätä tietoa voista erityiseen NetFlow-hallintatietokantaan. Lisäksi komentoriviltä voi nähdä jotain yhteenvetoja. Tiivistämisen tuki löytyy NetFlown versioista kahdeksan ja yhdeksän.

Tiivistäminen voidaan tehdä myös kerääjällä tai raportoijalla. Tällöin tiivistämisen hallinta on helpompaa ja monipuolisempaa. Alkuperäinen vuotieto voidaan myös halutessa säilyttää. Ongelmaksi vain muodostuu suuri levytilan ja laskentatehon tarve. Lisäksi verkkoyhteys sensorin ja kerääjän välillä kuormittuu enemmän.

Näiden keinojen lisäksi vuotietojen lähettämistiheyttä harventamalla voidaan säästää verkkoyhteyden kuormitusta sekä kerääjän laskentakapasiteettia ja levytilan käyttöä. Tällöin vuotietojen lähettämisviive kasvaa ja aikatarkkuus kärsii.

3.5.3 Vientiprotokolla

Versio 1 oli alkuperäinen NetFlow-versio, jota ei enää käytetä. Versio 5 on tämän laajennus joka lisää BGP[13]-protokollan -tuen ja sekvenssinumeroinnin. Versio 7 on Cisco Catalyst 5000 sarjan kytkimille tarkoitettu laajennus, eikä ole Ciscon reitittimien kanssa yhteensopiva. Versio 6 on versio 7 kaltainen, eikä sitä enää tueta uusissa Ciscon IOS[14]-versioissa. Versio 8 lisäsi reititinpohjaisen tiivistysmallin. Versioita 2, 3 ja 4 ei ole julkaistu. [28]

Versio 9 on NetFlown uusin versio ja sen vientiprotokolla on kokonaan mallipohjainen (engl. *template based*). Se lisäsi NetFlowhun MPLS-[15] ja ryhmälähetystuen (engl. *multicast*). Aiemmissakin versioissa ryhmälähetykset laskettiin mukaan tavu- ja pakettimääriin, mutta yhdeksännessä versiossa ryhmälähetysten tavu- ja pakettimäärät on mahdollista saada erikseen. Mallipohjaisena vientiprotokollaa on helppo laajentaa ohjelmia uudelleen kääntämättä. Versio 9 on määritelty myös RFC-dokumentissa [29]. Versiot 5 ja 9 ovat käytetyimpiä NetFlown versioita. Esimerkkejä NetFlown versio 9 mahdollistamista kentistä [30]:

- lähdeosoite ja -portti
- kohdeosoite ja -portti
- protokolla,
- sensorin liityntä
- palveluluokka
- paketti- ja tavulaskuri,
- seuraava kohde
- lähde- ja kohde-AS[16].

Vuotiedot lähetetään UDP-paketissa, jonka jälkeen sensori poistaa vuotiedot muististaan. Jos paketti häviää matkalle, ei sen tietoja saa enää mitenkään tietoon. NetFlown sekvenssinumeroinnin perusteella voidaan kuitenkin laskea monenko vuon tiedot puuttuvat. Mikäli reititin on ylikuormitettu, se saattaa myös jättää vuotiedot kirjaamatta tai lähettämättä, jolloin virhetilannetta ei voida edes huomata.

[13] Borger Gateway Protocol
[14] Internetwork Operating System
[15] Multiprotocol Label Switching
[16] Autonomous System

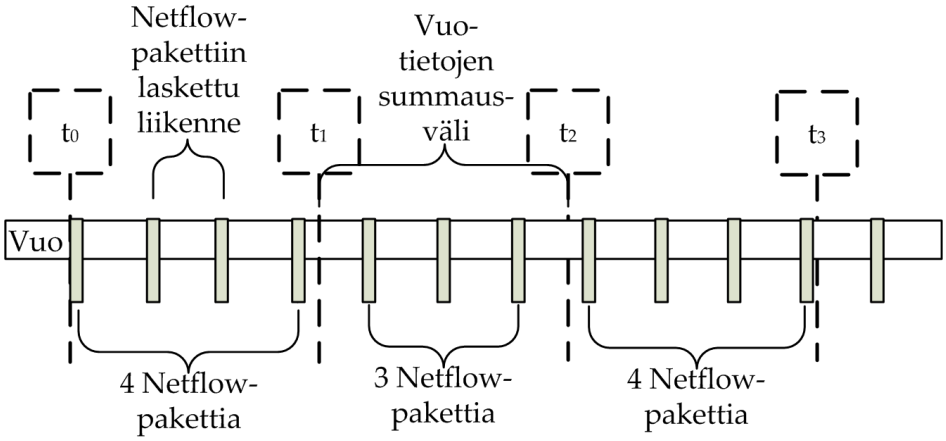
Ciscon reitittimet kirjaavat jatkuvasti vuotietoja välimuistiinsa ja tiedot lähetetään NetFlow-pakettina, jos joku seuraavista ehdoista täyttyy. Näiden ehtojen täyttyminen tarkistetaan reitittimellä sekunnin välein [27]:

- Kun vuovälimuisti täyttyy. Tällöin sovelletaan useaa heuristiikkaa, joilla voita vanhennetaan aggressiivisesti.
- Kun vuo on ollut passiivilaskurin määräämän ajan ilman liikennettä. Ciscon passiivilaskurin oletusaika on 15 sekuntia (valittavissa 10 s – 600 s).
- Kun TCP-yhteys loppuu (FIN) tai nollautuu (RST).
- Kun aktiivisen vuon kesto ylittää aktiivilaskurin ajan. Ciscon oletus on 30 minuuttia (valittavissa 1 min – 60 min).

3.5.4 Käyttökohteet ja -tavat

NetFlowta käytetään usein portti- tai palvelinkohtaisten tavumäärien laskemiseen. NetFlow-paketit lähetetään kuitenkin suhteellisen harvaan, joka saattaa muodostua ongelmaksi. Kuvassa 3.2 on esitetty yhden jatkuvan tietovuon tilanne, jossa summausvälin aikana vastaanotetaan 3–4 NetFlow-pakettia kyseisestä vuosta. Jos oletetaan vuon tietojen kuuluvan kokonaan edelliselle summausvälille, vuon tavulaskurin virhe on enimmillään 25 prosenttia. Jos tässä vuossa liikkuu paljon dataa, vääristyvät myös kokonaislaskennat selvästi. Tavumäärät voivat jopa hetkellisesti ylittää liitynnän fyysisen kapasiteetin. Virhettä voidaan pienentää lyhentämällä aktiivisten voiden aikakatkaisua ja pidentämällä summausaikaa. Lisäksi summaus voitaisiin suorittaa älykkäämmin jakamalla tavut useamman summausvälin kesken. Summausajan pidentäminen ja älykäs jakaminen lisäävät kuitenkin vuotietojen saamisen viivettä. Joka tapauksessa NetFlown avulla ei voi saada mittauksia reaaliajassa, vaan vähintään minuutin viive on väistämätön. [31]

TopN-analyysi kertoo verkon laitteet, jotka tuottavat eniten liikennettä. Liikenteen mittana voidaan käyttää joko yhteyksien määrää (Top N Session) tai siirrettyjen tavujen määrää (Top N Data). TopN-analyysillä etsitään usein matojen saastuttamia koneita. TopN-analyysia voidaan käyttää myös verkon muutosten kartoittamiseen. Tätä varten tarvitsee kuitenkin määrittää ensin **vertailukohta** (engl. *baseline*). Se on historiatietojen perusteella määritelty normaalin liikenteen malli. Liikenne joka ylittää vertailukohdan asettamat rajat, määritellään epänormaaliksi liikenteeksi.



Kuva 3.3: NetFlown aikaikkunaongelmat.

Asioita joita NetFlowlla ei voi tai kannata valvoa:

- Päästä-päähän mittaukset. NetFlow mittaa liikennettä yhdessä pisteessä eikä siten viiveiden, viiveiden vaihteluiden tai pakettihävikkien mittaaminen ole mahdollista.
- Ylempien kerrosten valvonta. NetFlow näkee ainoastaan IP-otsikot, ei ylempien kerrosten dataa. Tätä varten Cisco onkin kehittänyt Flexible NetFlow -tekniikan, josta lisää luvussa 3.6.
- Muun kuin IP-liikenteen tarkkailu. NetFlow ei tue muita protokollia, kuten NetBIOSia, AppleTalkia tai IPX:ää.
- Laitteen liityntöjen tilan valvonta. Mikäli verkkotason yläpuolisilla tiedoilla ei ole merkitystä, sopivat SNMP ja RMON paremmin liityntöjen tilan valvontaan.

Jos NetFlow-mittauksia summataan useammalta reitittimeltä, täytyy tiivistämisperusteiden valinnassa olla huolellinen. Samat paketit voivat kulkea usean reitittimen läpi, joka täytyy ottaa huomioon, jotta vältettäisiin tavumäärien monistuminen. NetFlowta ei olekaan järkevää ottaa käyttöön jokaisella verkon reitittimellä, vain verkon avainkohdissa sijaitsevilla reitittimillä.

3.5.5 Työkalut ja tuki verkkolaitteissa

Koska Cisco kehitti NetFlown, löytyy sen verkkolaitteista sille laaja tuki, IOS-versioista riippuen. Lisäksi Ciscon NetFlowta tukevia verkkolaitteita löytyy ainakin

seuraavilta valmistajilta: Foundry Networks, Enterasys Secure Networks ja Extreme Networks.

Taulukossa 3.3 on lueteltu testattuja NetFlow-kerääjiä ja raportoijia. Windows-puolella ohjelmat ovat enimmäkseen kaupallisia ja niissä on mukana myös raportointiominaisuuksia. Linuxin ohjelmat eivät pääsääntöisesti sisällä monipuolisia raportointiominaisuuksia vaan tukeutuvat muiden ohjelmien raportointiominaisuuksiin. Ne tarjosivatkin paremmat rajapinnat ja mahdollisuuden muiden sovelluksien kanssa integrointiin. Luvussa 5.3 on kerrottu enemmän flowd:n käytöstä.

Taulukko 3.3: Testattuja NetFlow sovelluksia

Ohjelma	**Käyttöjärjestelmä**	**Käyttötarkoitus**
ntop	Linux, BSD, Windows	sensori, kerääjä ja raportoija
NetFlowAnalyzer 5	Windows	kaupallinen kerääjä ja raportoija
pmacct	Linux	sensori ja kerääjä
flow-tools	Linux	sensori, kerääjä ja edelleenlähettäjä
nfdump	Linux	kerääjä ja edelleenlähettäjä
flowd	Linux	kerääjä

Ciscon NetFlown lisäksi ovat myös muut valmistajat kehitelleet omia vuopohjaisia mittaustekniikoita.

Cflowd Juniper Networksin kehittämä vuopohjainen mittaustekniikka.

CLEAR flow Extreme Networksin kehittämä vuopohjainen mittaustekniikka.

SFlow InMon Corporationin kehittämä vuopohjainen mittaustekniikka. Tuki löytyy seuraavilta verkkolaitevalmistajilta: AlaxalA Networks, Alcatel, Allied Telesis, Comtec Systems, Extreme Networks, Force10 Networks, Foundry, Hitachi, HP ja Nec.

3.6 Flexible NetFlow

Flexible NetFlow on Ciscon kehittämä laajennus. Se mahdollistaa monipuolisemman tiedon tiivistämisen (engl. *aggregating*) reitittimellä ja vientipakettiin mukaan enemmän tietoa, muun muassa 1200 tavua sovellustason paketin sisältöä.

Flexible NetFlow vuotietojen vientiprotokollan versio on valittavissa, se voi olla esimerkiksi versio 5 tai 9. Viidettä versiota käytettäessä, voidaan kerääjälle viedä vain viidennen version tukemia kenttiä. Sen sijaan yhdeksännellä versiolla voidaan

hyödyntää täysin Flexible NetFlown tarjoamat mahdollisuudet. Flexible NetFlow suunniteltiin protokollariippumattomaksi, joten vientiprotokollana voidaan käyttää UDP-protokollan lisäksi esimerkiksi SCTP[17]-protokollaa.

Flexible NetFlowssa vientikohteiden määrää ei ole keinotekoisesti rajoitettu, toisin kuin tavallisessa NetFlowssa, jossa vientikohteita oli enimmillään kaksi. Lisäksi Flexible mahdollistaa usean vuovälimuistin käytön, joille jokaiselle voi määrittää omat tiivistyssääntönsä ja vientikohteet.

Flexible NetFlow tarjoaa kolme erilaista määritystä vuovälimuistille [32]:

- **Tavallinen** (engl. *normal*): Vuovälimuisti käyttäytyy kuten tavallisen NetFlown tapauksessa. Välimuisti tyhjennetään ja viedään aikalaskureiden umpeutuessa, välimuistin täyttyessä tai TCP-yhteyden päättyessä.
- **Pysyvä** (engl. *permanent*): Vuovälimuistin koko on määriteltävissä eikä sitä tyhjennetä lainkaan. Kun välimuisti on täysi, ainoastaan olemassa oleviin voihin päivitetään tiedot.
- **Välitön** (engl. *immediate*): Jokainen paketti luo oman vuonsa. Tämä mahdollistaa sovellusdatan liittämisen vuon yhteyteen.

3.7 IPFix

NetFlow on Ciscon käyttämä tekniikka, eivätkä kaikki verkkolaitteet tue sitä. IETF onkin perustanut IPFIX[18]-työryhmän kehittämään standardia vuoprotokollaa. Työryhmä kehittää tietomallin, joka määrittää IP-vuon sekä IPFIX-vientiprotokollan, joka siirtää vuotiedon sensorilta kerääjälle. Määritystyö on vielä kesken, vaatimukset on kuitenkin määritelty RFC3917-dokumentissa. [33]

IPFIX-protokollan kehittelyn perustaksi valittiin Ciscon NetFlown versio 9. IPFIX-vientiprotokolla perustuu NetFlow yhdeksännen version tavoin mallipohjiin ja tarjoaa saman laajennettavuuden. IPFIX-toteutuksien täytyy tukea SCTP-protokolla vientiprotokollan yhteydessä, UDP- ja TCP-protokollien tukeminen ei ole välttämätöntä.

[17]Stream Control Transmission Protocol
[18]Internet Protocol Flow Information eXport

4 Aktiiviset mittaustekniikat

Tässä luvussa esitellään erilaisia aktiivisia IP-verkon suorituskyvyn mittaustekniikoita. OWAMP on esitelty luvussa 4.1, TWAMP luvussa 4.2 ja IP SLA luvussa 4.3. Aktiivisilla mittaustekniikoilla tarkoitetaan tekniikoita, jotka käyttävät synteettistä liikennettä mittauksissaan.

4.1 OWAMP

Tässä luvussa esitellään OWAMP-mittausprotokollan rakenne ja toiminta. OWAMP-Pin eräs käytännön toteutus esitellään luvussa 5.

OWAMP on IP-verkon suorituskyvyn mittaamiseen kehitetty aktiivinen mittausprotokolla. Aktiivisuus tarkoittaa sitä, että protokolla käyttää mittausdatana synteettisiä lähetyksiä, joten se generoi verkkoon liikennettä. Sen on kehittänyt IETF:n IPPM-työryhmä ja se on määritelty dokumentissa [34]. OWAMP-protokollalla voidaan mitata yksisuuntaisia suorituskykyparametreja kuten luvussa 2 kuvatut yhdensuuntainen viive ja -hävikki. Protokollan määrittelyn tarkoituksena on asettaa suorituskyvyn mittaussovelluksille standardoitu protokollarakenne, jolloin eri sovelluksilla saadut mittaustulokset voisivat olla vertailukelpoisia.

OWAMP sisältää kaksi eri protokollaa, OWAMP-Test- ja OWAMP-Control-Protokollan. Protokollilla on eri tehtävät testausprosessissa. Tehtävät esitellään tarkemmin myöhemmin. OWAMP-protokolla jakautuu loogisesti viiteen eri komponenttiin. Komponenttien tehtävät ja toiminta esitellään myöhemmin.

4.1.1 OWAMP-protokollan tavoitteita

OWAMP-protokollan suunnitellussa on asetettu monia tavoitteita, jotka liittyvät protokollan tietoturvallisuuteen, toiminnallisuuteen ja yleiskäyttöisyyteen. Ajanhetken määrittäminen onnistuu nykyään verkossa erittäin tarkasti esimerkiksi NTP:n tai GPS:n avulla, joten yhdensuuntaisten mittausten tarkkuus saadaan sitä kautta nostettua tarpeeksi korkealle. Työryhmä pyrkii näin ollen luomaan yhdensuuntaisten parametrien mittaamisesta yhtä yleistä kuin esimerkiksi edestakaisen viiveen mittausta ICMP-protokollan Ping-työkalulla. OWAMP-protokollasta on myös pyritty tekemään mahdollisimman tietoturvallinen. Protokollan mittausdata on ainoas-

taan UDP-tietovuo, jossa lähde ja kohde portit voidaan erikseen neuvotella. Näin ollen testaustapahtumaa on vaikea havaita ja testausdataan ei ole helppo päästä käsiksi. Testaustuloksia ei siis päästä muuttelemaan. OWAMPin tietoturvallisuutta takaa myös mahdollisuus salattuun tiedonsiirtoon ja autentikointiin. Tarvittaessa sekä protokollan hallintaliikenne että mittausliikenne salataan. Jos OWAMP-liikenne olisi helposti havaittavissa ja muuteltavissa, voitaisiin esimerkiksi pakettien aikaleimoja muuttaa ja mittaustulokset vääristyisivät.

4.1.2 Looginen malli

OWAMP-protokolla jakautuu loogisesti viiteen erilliseen komponenttiin, joilla on eri tehtävät mittauksessa. Komponentit voivat sijaita verkossa fyysisesti eri laitteilla, tai päästä-päähän -mittauksissa mittausvälin päissä olevilla laitteilla. Tällöin useampi looginen osa on toteutettu samalle laitteelle.

Protokollan loogiset osat ja niiden tehtävät:

Session-Sender Testauksessa lähettäjänä toimiva laite.

Session-Receiver Testauksessa vastaanottajana toimiva laite.

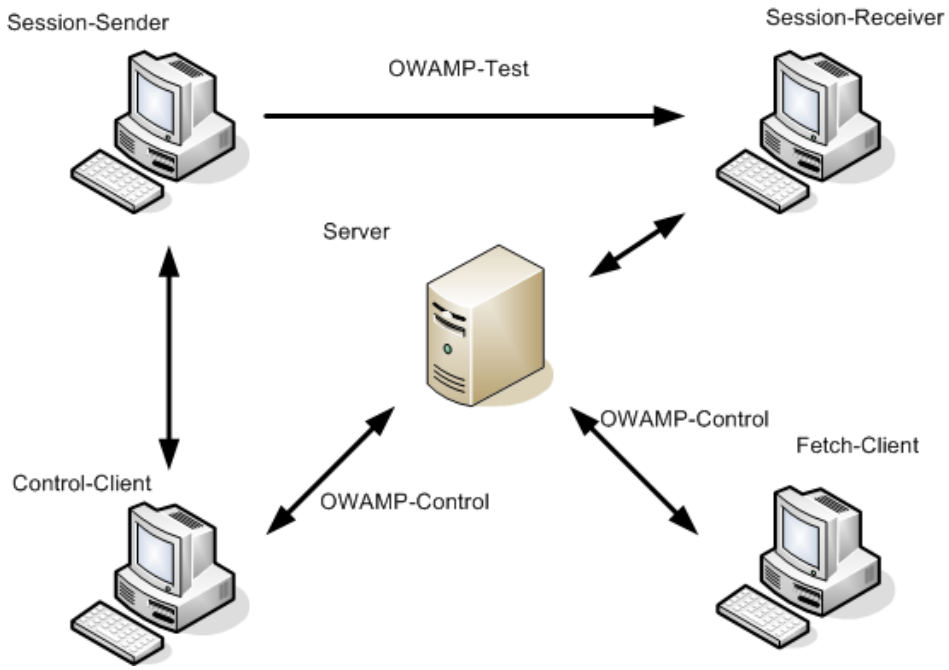
Server Palvelin, joka hallinnoi yhtä tai useampaa testausistuntoa. Tallentaa testausistuntojen tulokset.

Control-Client Laite, jolla tehdään pyynnöt testausistuntojen aloituksista ja lopetuksista.

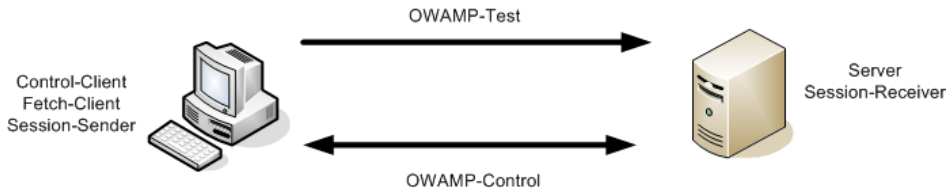
Fetch-Client Laite, jolla noudetaan päättyneiden testausistuntojen tulokset.

Kuvassa 4.1 on esitetty OWAMP-protokollan looginen malli. Kuvan nuolet esittävät eri osien keskinäisen kommunikoinnin. Nuolien kohdalla lukee myös protokolla jota kommunikointiin käytetään. Nuolet, joiden kohdalla ei lue protokollan nimeä, ovat määrittelemättömiä. Toisin sanoen näiden osien väliseen kommunikointiin voidaan käyttää jotain sopivaa protokollaa. Kuvan 4.1 mukaisessa mallissa protokollan eri osat on hajautettu verkon eri laitteille. Kuvassa 4.2 on esitetty perinteisen server-client -mallin mukainen toteutus. Server-client -mallissa palvelin ja testausistunnon vastaanottaja -osat ovat samassa laitteessa, ja testausistunnon lähettäjä, testitulosten noutaja ja testausten hallintaosa ovat samassa laitteessa. Hajautettu malli sopisi suurempien verkkojen mahdollisesti pitempikestoiseen monitorointiin. IPPM-työryhmän ajatus olisi, että tulevaisuudessa verkossa voisi olla julkisia OWAMP-palvelimia, joiden avulla yhdensuuntaisen viiveen mittaaminen olisi yhtä

yleistä kuin edestakaisen viiveen mittaaminen PING:llä. Kuvan 4.2 mukainen malli sopii taas pienempiin verkkoihin ja yksittäisiin testaustapahtumiin.



Kuva 4.1: OWAMP-protokollan looginen malli.



Kuva 4.2: OWAMP-protokollan looginen server-client -malli.

4.1.3 Hallinta- ja testausprotokollat

OWAMP-protokolla siis jakautuu itse asiassa kahteen protokollaan: Hallinta- ja testausprotokollaan. Molempia käytetään eri tehtäviin testauksen aikana. Hallintaprotokollaa käytetään testausistuntojen alustukseen, aloittamiseen ja lopettamiseen, sekä testausistuntojen tulosten hakemiseen. Hallintaprotokolla käyttää kuljetuskerroksen protokollista TCP:tä [35]. Testausprotokollaa käytetään testauspakettien kuljetukseen. Testausprotokolla käyttää testiliikenteen kuljetukseen UDP-protokollaa [36].

4.1.4 Protokollan toiminta lyhyesti

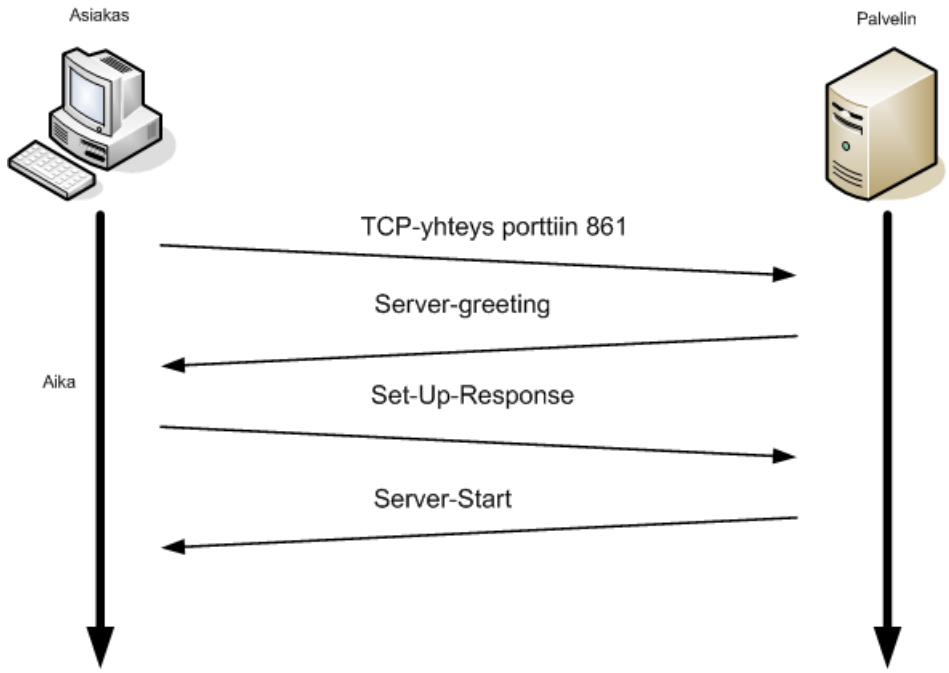
Seuraavassa esitellään OWAMP-protokollan toiminta lyhyesti. Kun halutaan suorittaa OWAMP-testi joidenkin verkon laitteiden välillä, lähetetään aluksi palvelimelle testauspyyntö. Tämän pyynnön lähettää hallintalaite. Tämän jälkeen palvelin käskee verkon laitteita suorittamaan testausistunto pyydetyillä parametreilla, ja lähettämään istunnon tulokset palvelimelle. Tulokset voidaan noutaa palvelimelta tulostennouto-laitteella. Tarkemmin tämä on selitetty seuraavissa alaluvuissa 4.1.5, 4.1.6 ja 4.1.8.

4.1.5 Testausistunnon alustaminen (OWAMP-Control)

Ennen kuin Control-Client voi pyytää palvelimelta testausistuntoa tai Fetch-Client testausistunnon tuloksia, täytyy luoda yhteys palvelimen ja asiakkaan (Control- tai Fetch-Client) välille. Ensimmäiseksi luodaan TCP-yhteys palvelimeen. Yhteys muodostetaan porttiin 861. Palvelin vastaa tervehdysviestillä (`Server-greeting`). Palvelimen tervehdysviestissä on lueteltuna tuetut yhteysmuodot, joita voivat olla autentikoimaton, autentikoitu ja salattu. Tähän viestiin asiakaslaite vastaa yhteydenalustus-viestillä (`Set-Up-response`). Viestissä asiakas ilmoittaa palvelimelle myös valitsemansa yhteyden muodon. Palvelin vastaa asiakkaalle aloitusviestillä (`Server-Start`). Tällä viestillä palvelin ilmoittaa, onko se halukas jatkamaan kommunikointia asiakaslaitteen kanssa. Viestissä on hyväksyntä-kenttä (engl. *Accept*), jonka arvo nolla tarkoittaa hyväksyntää. Muut arvot tarkoittavat, että palvelin ei hyväksy autentikointia, tai ei ole jostain muusta syystä halukas jatkamaan kommunikointia. Testausistunnon alustaminen on kuvattu sekvenssikaaviona kuvassa 4.3. Mikäli molemmat osapuolet toimivat edellä kuvatulla tavalla, on yhteys muodostettu.

4.1.6 Testausistunto (OWAMP-Control)

Kun yhteys palvelimeen on muodostettu, voidaan palvelimelta pyytää testausistuntojaa suoritettaviksi. Testausistunnon pyytäminen tapahtuu siten, että asiakaslaite lähettää palvelimelle testauspyyntö-viestin (`Request-Session`). Pyyntöviesti pitää sisällään kaikki testaukseen liittyvät parametrit, kuten lähettäjän ja vastaanottajan osoitteet. Pakettirakenne ja kenttien merkitys on selitetty lähtessä [34]. Tämän jälkeen lähetetään aikataulutukseen liittyviä lähetysaikapaketteja (yksi tai useampia), joiden määrä on ilmoitettu testauspyyntö-viestissä. Aikataulutuspaketit sisältävät muun muassa tiedon siitä, miten lähetysajat satunnaistetaan. Lähetyksessä käytetään satunnaisuutta, jotta se kuvastaisi normaalia verkkoliikennettä mah-

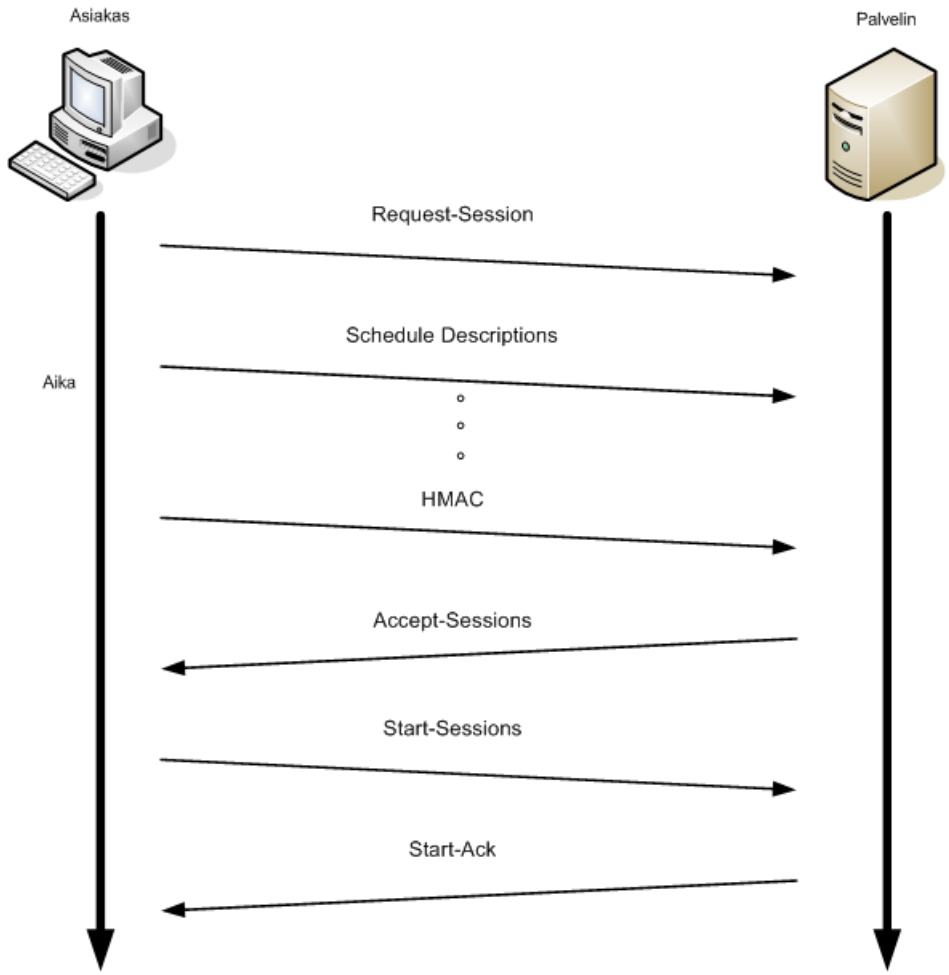


Kuva 4.3: OWAMP-protokollan testausistunnon alustaminen.

dollisimman hyvin. Tähän voidaan käyttää esimerkiksi Poisson-jakaumaa [37] tai jotain jaksoittaista lähetystä [38], joka kuvaa jonkun oikean sovelluksen käyttäytymistä. Tämän jälkeen lähetetään HMAC-autentikointikoodi. Palvelin vastaa viesteihin istunnonhyväksyntä-viestillä (`Accept-Session`), jossa testauspyyntö voidaan myös evätä. Hyväksyntä viestissä on `accept`-kenttä, jonka arvo nolla tarkoittaa hyväksyntää. Muut arvot tarkoittavat, että testausta ei hyväksytä syystä tai toisesta. `Accept`-kentän eri arvot ja merkitykset on selitetty tarkemmin lähteessä [34]. Palvelin järjestää testauksen suorittamisen lähettäjälaitteen ja vastaanottajalaitteen välillä pyydetyillä parametreilla. Saatuaan hyväksynnän testauspyyntöön, voi asiakaslaite pyytää palvelinta käynnistämään testausistunnon. Tämä tapahtuu lähettämällä istunnon aloitus -viesti (`Start-Session`) palvelimelle. Palvelin vastaa tähän mahdollisimman nopeasti kuittausviestillä (`Start-Ack`), jonka jälkeen palvelin käynnistää testauksen mahdollisimman nopeasti. Jos testausistunnolle oli määritelty aloitusaika testauspyyntö-viestissä, aloitetaan testaus vasta silloin. Istunto aloitetaan näistä kahdesta ajankohdasta myöhempänä. Asiakaslaite käynnistää testauksen lähettäjälaitteen osalta samoilla ehdoilla kuin palvelin edellä. Testausistunnon pyytäminen ja aloitus on esitetty sekvenssikaaviona kuvassa 4.4.

Testausistuntoja voidaan haluttaessa pysäyttää kesken. Pysäytyksen voi suorittaa joko asiakaslaite tai palvelin. Pysäyttäminen tapahtuu lähettämällä ensiksi pysäytyskäsky-viesti (`Stop-Sessions`), joka pitää sisällään tiedon, montako istuntoa

halutaan pysäyttää. Heti perään lähetetään jokaista pysäytettävää istuntoa kohden istunnon tunnus -viesti. Tämän jälkeen lähetetään yksi tai useampia pakettien hylkäys -viestejä (`Skip Range`). Näiden viestien lukumäärä on ilmoitettu istunnon tunnus -viestissä. Hylättävät paketit ilmoitetaan lukupareilla, jotka kuvaavat lähetysväliä paketista k pakettiin k+n, jotka jätetään lähettämättä. Viimeisenä lähetetään vielä HMAC-autentikointikoodi. Viestien rakenne ja otsikkokenttien merkitys on selitetty tarkemmin lähteessä [34].



Kuva 4.4: OWAMP-protokollan testausistunnon pyytäminen.

4.1.7 Testausistunto(OWAMP-Test)

Kun testausistunto on saatu alustettua ja käynnistettyä, alkaa varsinaisten testauspakettien lähettäminen. Tähän käytetään OWAMP-Test -protokollaa, joka käyttää kuljetuskerroksen protokollista hyväkseen UDP:tä. Testausprotokolla käyttää samo-

ja tietoturvamekanismeja kuin hallintaprotokollakin. Testausistunto voidaan siis hoitaa salattuna, ja käyttäjän autentikointi on mahdollista. Testausistunnon lähettäjälaite lähettää sovitun määrän testauspaketteja vastaanottajalle sovitussa lähetysaikataulussa. On erittäin tärkeää, että molemmat, sekä lähettäjä, että vastaanottaja saavat laskettua saman lähetysaikataulun mahdollisimman tarkasti. Lähettäjän täytyy myös noudattaa tätä aikataulua mahdollisimman tarkasti. Vastaanottajan täytyy tietää aikataulu siksi, että se tietää milloin kukin paketti lähtee lähettäjältä. Näin voidaan määritellä esimerkiksi viiveelle raja, jolloin kyseinen paketti todetaan hävinneeksi. Lähettäjän ei tule lähettää pakettia, jos se on yli raja-arvon verran myöhässä. Tällöin paketti todettaisiin joka tapauksessa hävinneeksi. Lähetysaikataulu lasketaan istunnon alustuksen yhteydessä vaihdettujen ajastustietojen perusteella. Jokainen testauspaketti sisältää lähetysaikaleiman ja virhearvion. Testauspakettien jälkeen lähettäjä lähettää istunnon pysäys -viestin, joka sisältää tiedon paketeista, joita se ei lähettänyt. Tieto lähettämättä jääneistä paketeista lähetetään myös tulosten nouto -laitteelle.

4.1.8 Tulosten noutaminen

Tulosten noutaminen palvelimelta tapahtuu seuraavasti: Asiakaslaite lähettää palvelimelle tulostennoutamis-viestin (Fetch-Session). Viesti sisältää tiedot istunnosta, jonka tulokset halutaan. Palvelin vastaa viestiin hyväksyntäviestillä (Fetch-Ack), joka sisältää tiedon hyväksynnästä ja siitä, onko istuntoo suoritettu loppuun, ja istunnon yhteydessä lähettämättä jääneistä testipaketeista (lukupari(t), kuten pysäytyksessäkin). Jos istunto on kesken, tulosten nouto kielletään. Tulosten ollessa valmiita lähetettäviksi, palvelin lähettää tulokset pyydetystä istunnosta asiakkaalle. Tarkemmin tämä tapahtuma ja sen mahdolliset erikoistilanteet on kerrottu lähteessä [34].

4.1.9 HOTS

HOTS on OWAMP-prorokollaa tukeva lisälaite, joka tuo lisätarkkuutta OWAMP-mittauksiin. HOTS:in ovat kehittäneet Zhang Shu ja Katsushi Kobayashi ja he esittelevät laitteen etuja lähteessä [39].

HOTS on PCI-väylään liitettävä lisälaite, joka tuo lisätarkkuutta pakettien aikaleimojen liittämiseen ja vastaanottamiseen. Aikaleimojen laittamisessa paketteihin tai niiden poimimisessa paketeista voi syntyä huomattavia virheitä mittaustuloksiin. Aikaleiman laittamisen ja paketin lähettämisen välillä tapahtuu yleensä prosessointiviivettä, jos se toteutetaan ohjelmallisesti. Sama ongelma ilmenee myös vastaanottopäässä. Ongelma pahenee silloin, kun laitteet ovat ruuhkautuneita. HOTS

tarjoaa tähän ongelmaan ratkaisun. HOTS-laiteessa on optinen ulostulo ja erilliset liitännät, joihin voidaan kytkeä esimerkiksi GPS-laite, jota käytetään aikalähteenä. HOTS ei aiheuta mittauksiin prosessointiviivettä. HOTS:ia on testattu mittauksissa ja on huomattu, että se tuo aikaleimoihin huomattavasti tarkkuutta ja vähentää virheen määrää. Tarkemmat tiedot HOTS:ista ja mittauksista löytyy lähteestä [39].

4.1.10 Yhteenveto

OWAMP on suorituskyvyn mittausprotokolla, joka mittaa yhdensuuntaisia parametreja. OWAMP sopii erittäin hyvin päästä-päähän -mittauksiin. OWAMPilla mitattava parametrit ovat IETF:n kehittämiä standardeja, joten mittaustulokset ovat keskenään vertailukelpoisia. OWAMP sopisi hyvin verkon laitteisiin sisäänrakennetuksi tekniikaksi, jolloin se yleistyisi yhtä yleiseksi menetelmäksi kuin ICMP:n Ping. OWAMPista ei ole vielä kovin montaa käytännön toteutusta. Tässä tutkielmassa esiteltävä OWPing on toteutus, joka tukee ainoastaan server-client -mallia. OWPingin huono puoli on että, mittausdatan tallennustiedostot nimetään huonolla tavalla. Tiedostonimissä esiintyy pitkä numerosarja ja sen lisänä satunnainen numerosarja. Eri mittausistuntojen tulostennouto on siis erittäin vaikeaa, koska tiedostonimeä ei voi etukäteen tietää. Tulostennoudossa käytetään salattua yhteyttä, joten tällainen tiedostonnimeäminen ei ole järkevää. Järkevämpi tapa olisi nimetä tiedostot esimerkiksi mittauspäivämäärän ja kellonajan mukaan. Tulokset voitaisiin myös tallettaa reitittimen tai kytkimen hallintatietokantaan, jolloin niitä voitaisiin noutaa esimerkiksi SNMP:n avulla. OWAMP tukee salattua yhteyttä ja käyttäjän autentikointia. OWAMP on hyvin skaalautuva. Se sopii laajoihin ja suppeisiin verkkoihin. Käytännön toteutuksessa on vielä kehittelemisen varaa, mutta OWAMPissa on aineksia yleiseksi mittausprotokollaksi.

4.2 TWAMP

Tässä luvussa esitellään TWAMP-protokolla. Protokolla pohjautuu OWAMP-protokollaan. Luvussa esitellään TWAMP-protokollan eroavaisuudet OWAMP-protokollaan.

4.2.1 Yleistä

TWAMP-mittausprotokolla on suunniteltu mittaamaan edestakaisia suorituskykyparametreja, kuten esimerkiksi edestakaista viivettä 2.4.1. TWAMP-protokollan on

kehittänyt IPPM-työryhmä ja protokolla on esitetty dokumentissa [40]. TWAMP-protokollan toiminta pohjautuu paljolti OWAMP-protokollan toimintaan. TWAMP-protokollan loogisessa mallissa on pieniä muutoksia OWAMP-protokollan vastaavaan nähden. Protokollaan on tehty myös toiminnallisia muutoksia OWAMP-protokollaan verrattuna.

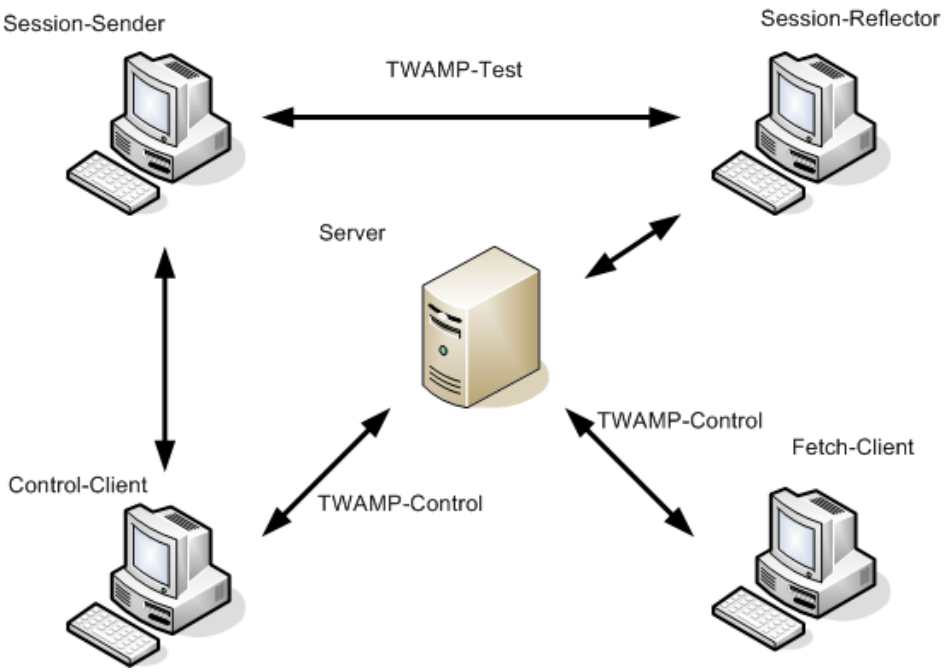
4.2.2 TWAMP-protokollan Looginen malli

TWAMP-protokollasta voidaan piirtää useampia loogisia malleja riippuen siitä, käytetäänkö protokollaa ainoastaan kahdensuuntaisiin mittauksiin, vai mitataanko sillä myös yhdensuuntaisia suureita. Ensimmäisessä vaihtoehdossa ei käytännössä ole kovin paljoa järkeä. Seuraavassa esitellään kolme erilaista loogista mallia TWAMP-protokollalle ja kerrotaan, millaiseen tilanteeseen kukin sopii.

Kuvassa 4.5 on esitetty TWAMP-protokollan malli, joka vastaa OWAMP-protokollan mallia. Tämä malli sopii tilanteeseen, jossa protokollalla mitataan myös yhdensuuntaisia suorituskykyparametreja. OWAMP-protokollan testipakettien vastaanottaja on tässä mallissa vaihdettu testipakettien heijastajaan (engl. *Reflector*). Heijastajan tehtävät poikkeavatkin testipakettien vastaanottajan tehtävistä. Heijastajan tehtävä on aikaleimata testipaketit, ja lähettää ne takaisin lähettäjälle. Tässä mallissa heijastaja toimii lisäksi OWAMP-protokollan vastaanottajan roolissa. Tämä on ainoa malli jossa tarvitaan tulosten noutaja -laitetta, koska mitataan myös yhdensuuntaisia parametreja. Edestakaisissa mittauksissa ei tarvita erillistä tulostennoutoistuntoa vastanottajalta, koska testipaketit palautuvat lähettäjälle. Toimintamalli on tässä mallissa samanlainen kuin OWAMP-protokollan hajautetussa mallissa.

Kuvassa 4.6 on TWAMP-protokollan malli, jossa ei ole tulosten kerääjä -laitetta. Tämä malli soveltuu tilanteeseen, jossa mitataan ainoastaan edestakaisia parametreja. Tällainen tilanne ei käytännössä ole kovinkaan järkevää, koska samallahan voitaisiin mitata myös yhdensuuntaisia parametreja. Tämän mallin etu on se, ettei heijastajana toimivalla laitteella tarvitse olla niin paljoa älyä, kuin esimerkiksi OWAMP-protokollan tapauksessa.

Kuvassa 4.7 on client-server -malli TWAMP-protokollasta. Tässä mallissa on loogiset osat yhdistetty samoihin laitteisiin. Lähettävässä laitteessa toimivat testipakettien lähettäjä, palvelin ja hallintalaite. Vastaanottopäässä on ainoastaan heijastaja. Vastaanottopäässä ei tarvita muita osia, koska mittauspaketit palaavat lähettävään laitteeseen, ja tulosten koonti tapahtuu siellä. Tämä malli soveltuu myös ainoastaan edestakaisten parametrien mittaamiseen.



Kuva 4.5: TWAMP-protokollan looginen malli.

4.2.3 Yhteenveto

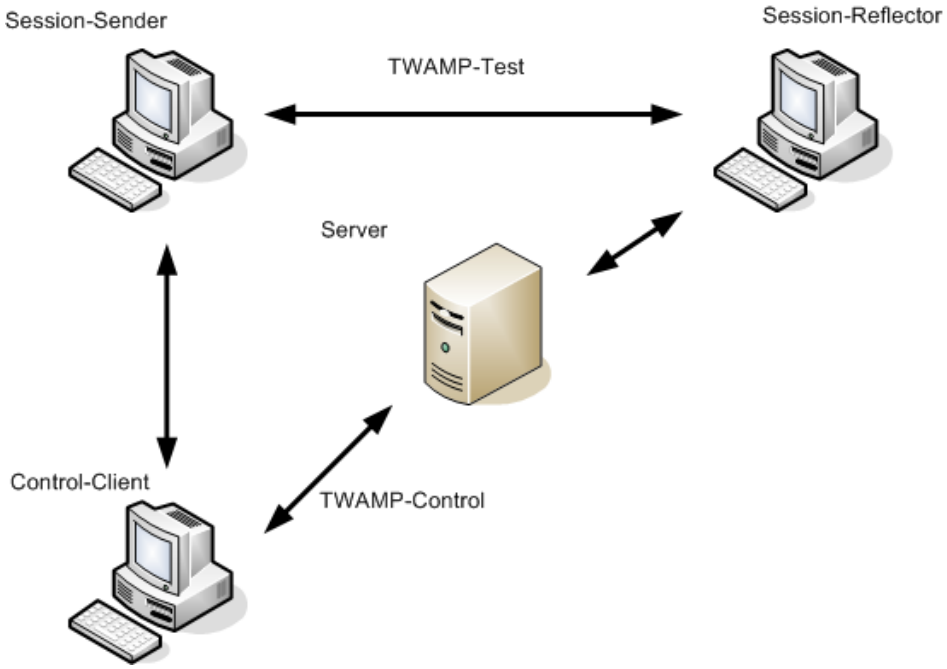
TWAMP-protokolla on OWAMP-protokollan muunnos, joka mittaa edestakaisia parametreja. TWAMP-protokollan muutokset OWAMP-protokollaan nähden ovat niin pieniä, ettei olisi ollut välttämätöntä toteuttaa uutta protokollaa tähän tarkoitukseen. Järkevämpää olisi lisätä tämä ominaisuus OWAMP-protokollaan. Toteutuksen yhteydessä edestakaisista mittauksista voitaisiin tehdä valinnainen optio mittaussovellukseen.

4.3 Cisco IOS IP SLA

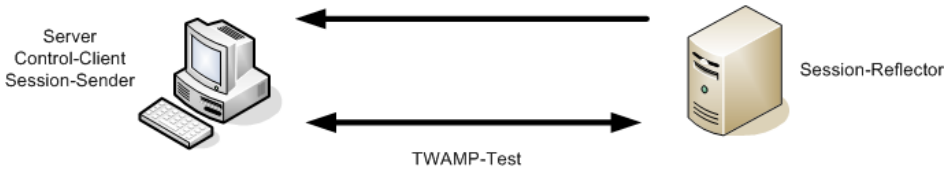
Cisco IOS IP SLA on Ciscon kehittämä tietoverkon suorituskyvyn monitorointiin soveltuva ohjelmisto. Tässä luvussa esitellään Cisco IOS IP SLA:n toimintaperiaate ja mitä kaikkea sillä voi mitata.

4.3.1 Yleistä

Cisco IOS IP SLA esiintyi aiemmin nimellä Service Assurance Agent. IP SLA on Ciscon verkkolaitteisiin sisäänrakennettu verkon suorituskyvyn mittaussovellus. Mittaamiseen vaaditaan siis vähintään yksi Ciscon verkkolaite. Mittaustilanteesta riip-



Kuva 4.6: TWAMP-protokolla ilman tulostennoutajaa.



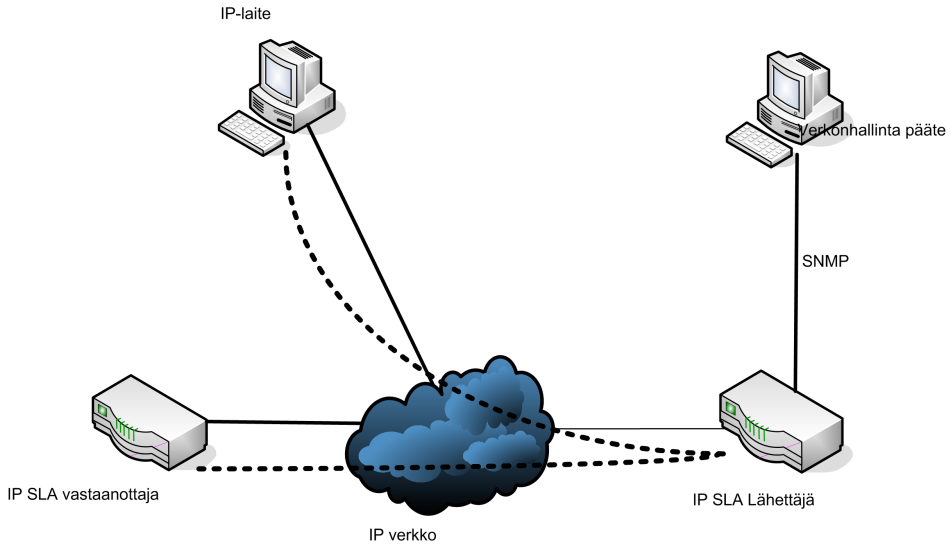
Kuva 4.7: TWAMP-protokollan client-server -malli.

puen saatetaan tarvita myös useampi Ciscon laite, jotta mittaus olisi mahdollinen. IP SLA on aktiiviseen mittaukseen erinomaisesti soveltuva ohjelmisto. Sillä voi mitata oikeastaan kaikkia suorituskykyparametreja, sekä yhdensuuntaisia että edestakaisia. Huono puoli on se, että mittaukset vaativat Ciscon laitteistoa, joten mittauksia ei voida toteuttaa joka verkossa. IP SLA:lla voidaan mitata normaalien suorituskykyparametrien lisäksi myös eri verkkopalveluiden saatavuutta, ohjelmistojen suorituskykyä ja palvelimien vasteaikoja [25].

4.3.2 Toimintaperiaate

IP SLA:lla voidaan siis mitata päästäpäähän-mittauksia eri verkkolaitteiden välillä, tai voidaan suorittaa kyselyjä eri palvelimille tai IP-laitteille. Mittaukset voivat olla myös spatiaalisia, eli tuloksia kerätään hyppy kerrallaan koko polun matkalta. Tällaisella mittauksella voidaan löytää esimerkiksi pullonkaulalinkkejä huonos-

ti toimivalta verkon polulta. Mittaus alustetaan verkkolaiteen, esimerkiksi reitittimen hallintakäyttöliittymässä. Mittaukselle voidaan antaa paljon erilaisia määrityksiä. Jotkut operaatiot voidaan alustaa myös jollain verkonhallintasovelluksella erilliseltä päätteeltä käyttäen SNMP-protokollaa [15]. Tulosten noutaminen tapahtuu joko suoraan verkkolaitteen hallintakäyttöliittymästä, tai noutamalla ne RTTMON MIB:eistä (luku 3.4) käyttämällä SNMP-protokollaa. Kaikki operaatiot eivät tallenna tuloksiaan MIBeihin. Kuvassa 4.8 on esitetty IP SLA -mittauksen arkkitehtuuri. Mittausoperaatiot on merkitty kuvaan katkoviivoilla.



Kuva 4.8: IP SLA -mittaus.

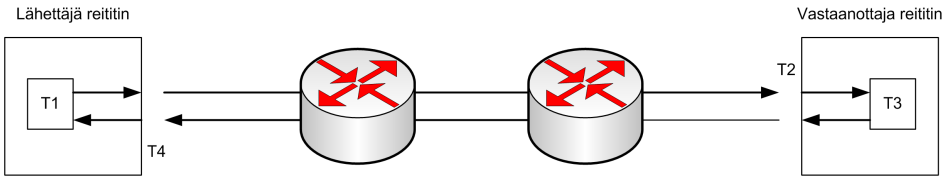
Seuraavassa käydään läpi mittauksen alustamisen vaiheet lyhyesti.

- Käynnistetään vastaajaprosessi (Responder), jos halutaan yhdensuuntaisia tuloksia. Tämä ei ole pakollinen kaikissa mittauksissa.
- Määritellään haluttu mittaustyyppi.
- Määritellään mittaustyypille halutut määritykset.
- Määritellään raja-arvo -tilanteet, jos halutaan sellaisia käyttää.
- Määritellään mittaukselle aikataulutus.
- Tarkastellaan mittauksen tuloksia. Mittauksen ei tarvitse olla päättynyt.

Mittauksia voidaan aloittaa samalta laitteelta useita, ja ne voivat kohdistua eri laiteisiin ja mitata eri parametreja. Mittauksia voi olla käynnissä yhtä aikaa useita.

Suuria verkkoja käsiteltäessä olisi turhan työlästä alustaa jokainen mittaus yksitellen yllä kerrotulla tavalla. Tällaisia tapauksia varten voidaan tehdä suuria operaatioryhmiä, jolloin voidaan alustaa useita mittauksia yhdellä kertaa. Tästä on kerrottu tarkemmin dokumentissa [41]. Operaatioille voidaan myös asettaa raja-arvoja, joiden ylittäminen aiheuttaa jatkotoimenpiteitä. Jos esimerkiksi mittausoperaatio huomaa viiveen vaihtelun olevan poikkeuksellisen runsasta, laite voi lähettää tiedotteen esimerkiksi verkonhallintapäätteelle. Tämä onnistuu esimerkiksi SNMP-hälytyksellä. Raja-arvon ylittymisellä voidaan laukaista myös muita operaatioita, jotka alkavat kerätä lisää dataa poikkeuksellisesta käytöksestä. Raja-arvoista ja niiden konfiguroinnista on kerrottu dokumentissa [42].

Kun mitataan edestakaista viivettä IP SLA -operaatioissa, on otettu huomioon aiemmin esitelty prosessointiviive ongelma. Reitittimillä saattaa kestää kymmeniä millisekunteja prosessoida saapuvia paketteja. IP SLA -mittauksissa tämä ongelma on pyritty poistamaan siten, että edestakaisen viiveen mittaamisessa tulos muodostetaan neljän eri aikaleiman avulla. Ensimmäinen aikaleima laitetaan, kun paketti lähtee lähettävän reitittimen prosessorilta. Toinen aikaleima laitetaan, kun paketti saapuu vastaanottavan reitittimen liityntään. Kolmas aikaleima laitetaan, kun paketti lähtee takaisin vastaanottavan reitittimen prosessorilta. Neljäs aikaleima laitetaan, kun paketti saapuu lähettävän reitittimen liityntään. Kuvassa 4.9 on esitetty aikaleimojen laittamishetket. Kuvan mukaisessa tilanteessa edestakainen viive laskettaisiin: `viive = T4 - T1 - (T3 - T2) = T4 - T1 - T3 + T2`. Tällöin tuloksesta jää pois reitittimien saapuvan liikenteen prosessoinnista aiheutuva virhe. Lähettävän ja vastaanottavan laitteen kellojen on oltava synkronoitu keskenään mahdollisimman tarkasti.



Kuva 4.9: Aikaleimat edestakaisen viiveen mittauksessa.

4.3.3 Mittausoperaatiot

Tässä alaluvussa käydään läpi mittausoperaatiot, joita Ciscon laiteilla voidaan ajaa. Tarkemmin tarkastellaan UDP Jitter nimistä prosessia, joka on kaikkein yleisin mittausprosessi. Mittausoperaatiot sisältävät kukin useita mitattavia parametreja, riippuen niiden käyttötarkoituksesta. Seuraavassa on listattu eri mittausoperaatiot ja

selitetty niiden mittaamat parametrit.

UDP Jitter on yleisin mittausoperaatio. Se käyttää mittauksissa nimensä mukaisesti UDP-protokollaa. UDP Jitterin mittaamat parametrit ovat: Edestakainen viive, yhdensuuntainen viive, yhdensuuntainen viiveen vaihtelu, yhdensuuntainen hävikki ja saatavuus. UDP Jitter suunniteltiin alun perin, jotta voitaisiin tutkia verkon sopivuutta reaaliaikaisille sovelluksille kuten VoIP ja videoneuvottelut. UDP Jitterin konfigurointi on esitetty dokumentissa [43]. UDP Jitteriä tarkastellaan tarkemmin esimerkin avulla myöhemmin.

ICMP Path Jitter -operaatiolla mitataan spatiaalista viiveen vaihtelua, hävikkiä ja viivettä. Spatiaalinen mittaus on joissain tapauksissa informatiivisempi kuin pelkkä yhdensuuntainen mittaus. Spatiaalisella mittauksella voidaan löytää esimerkiksi pullonkaulalinkit joltain tietyltä polulta verkossa. ICMP Path Jitter toimii siten, että aluksi selvitetään polku ICMP:n Traceroute -mekanismilla. Tämän jälkeen mitataan polun linkit ICMP:n Echolla. Arvot ovat siis arvioita, koska ICMP mittaa vain edestakaisia arvoja. ICMP:n Path Jitteriä ei tueta RTTMON MIB:eissä, joten konfigurointi ja tulosten nouto voidaan tehdä ainoastaan laitteen käyttöliittymässä. ICMP Path Jitter on esitelty tarkemmin dokumentissa [44].

UDP Jitter for VoIP on UDP Jitter -operaation modifikaatio. Sillä voidaan mitata parametreja erityisesti VoIP:in kannalta. Operaatiolle määritellään VoIP-kodekki jota simuloidaan. Operaatio muodostaa VoIP:ia muistuttavaa UDP-liikennettä. Liikenteestä lasketaan kaksi eri arvoa, MOS (Mean Opinion Score) ja ICPIF (Impairment Calculated Planning Impairment Factor), jotka kuvaavat VoIP-liikenteen hyvyyttä. Operaatio mittaa myös yhdensuuntaista ja edestakaista viivettä, sekä yhdensuuntaista hävikkiä. UDP Jitter for VoIP ja MOS sekä ICPIF on esitelty tarkemmin dokumentissa [45].

UDP Echo on operaatio, jolla mitataan UDP-liikenteen edestakaista viivettä sekä saatavuutta. UDP Echo -operaatio täytyy konfiguroida Ciscon laitteella, mutta vastaanottajalaitteena voi olla mikä tahansa IP-laite. UDP Echo on esitelty dokumentissa [46].

ICMP Echo -operaatiolla mitataan edestakaista viivettä, käyttäen ICMP:n Echo Request- ja Echo Reply -viestejä. Tässäkin vastaanottajalaite voi olla mikä tahansa IP-laite. ICMP Echo -operaatio on esitelty dokumentissa [47].

ICMP Path Echo on vastaavanlainen operaatio kuin ICMP Path Jitter. Operaatiossa selvitetään polku samalla tavalla kuin ICMP Path Jitterissä, jonka jälkeen mitataan polun jokaisen linkin edestakainen viive käyttäen Pingiä. Operaatiota käytetään saatavuuden mittaamiseen ja pullonkaulalinkkien etsimiseen. Vastaajalaitteena voi toimia mikä tahansa IP-laite. ICMP Path Echo:n tarkempi määrittely ja konfigu-

rointi on esitetty dokumentissa [48].

HTTP[1]-operaatiolla mitataan HTTP-palvelimen suorituskykyä. HTTP-operaatio noutaa pyydetyn HTML-dokumentin HTTP-palvelimelta ja mittaa tähän kuluneen ajan. HTTP-operaatiosta on myös olemassa HTTP RAW -operaatio, jossa käyttäjä saa määritellä monipuolisemmat parametrit operaatiolle. HTTP-operaatio mittaa muun muassa TCP-yhteyden luontiaikaa ja HTML-dokumentin noutamiseen kuluvaa kokonaisaikaa. HTTP-operaatio on esitelty tarkemmin dokumentissa [49].

TCP Connect -operaatiolla mitataan TCP-yhteyden luontiaikaa. Operaatio luo TCP-yhteyden toiseen Ciscon laiteeseen, tai johonkin IP-laitteeseen. Jos yhteys luodaan kahden Ciscon laitteen välille, voidaan porttinumero valita vapaasti. Jos kohdelaite on joku muu IP-laite, täytyy käyttää porttia, jota vastaajalaite kuuntelee. TCP Connect -operaatiota voidaan käyttää palvelimien ja sovellusten suorituskyvyn mittaamiseen. Tarkemmin operaatio on esitelty dokumentissa [50].

FTP[2]-operaatiolla voidaan mitata FTP-palvelimen suorituskykyä. FTP-operaatio lataa FTP-palvelimelta tiedoston ja mittaa siihen kuluneen ajan. FTP-operaatiota käytettäessä on varottava käyttämästä isoja tiedostoja testaukseen, koska ne ruuhkauttavat verkkoa. FTP-operaatio on esitelty dokumentissa [51].

DHCP[3] on operaatio, jolla mitataan DHCP-palvelimen suorituskykyä. Operaatiolla on kaksi toimintatapaa. Perusasetuksilla DHCP-kysely lähetetään yleislähetyksenä kaikkialle. Halutessa kysely voidaan lähettää vain jollekin tietylle DHCP-palvelimelle ja mitata juuri kyseisen palvelimen suorituskykyä. Operaatio mittaa, kuinka kauan aikaa kuluu kun tehdään DHCP-kysely ja saadaan käyttöön uusi IP-osoite. DHCP-operaatio on esitelty dokumentissa [52].

DNS[4]-operaatiolla mitataan WWW- tai DNS-palvelimen suorituskykyä. Operaatio mittaa aikaa, joka kuluu DNS-kyselyn lähettämisestä vastauksen saamiseen. Operaatiolle määritellään IP-osoite, jonka verkkotunnus halutaan tietää. Kysely toimii myös toisinpäin. DNS-operaatio on esitetty dokumentissa [53].

DLSw+[5]-operaatiolla mitataan vasteaikaa käytettäessä DLSw+ -tunnelointia. Mittaus suoritetaan etäpisteiden välillä. DLSw+ -operaatio on esitetty tarkemmin dokumentissa [54].

Frame Relay on operaatio, jolla mitataan WAN-verkkojen suorituskykyä ja toteuttaako se voimassa olevan SLA-sopimuksen.

[1]Hypertext Transfer Protocol
[2]File Transfer Protocol
[3]Dynamic Host Configuration Protocol
[4]Domain Name System
[5]Data-Link Switching

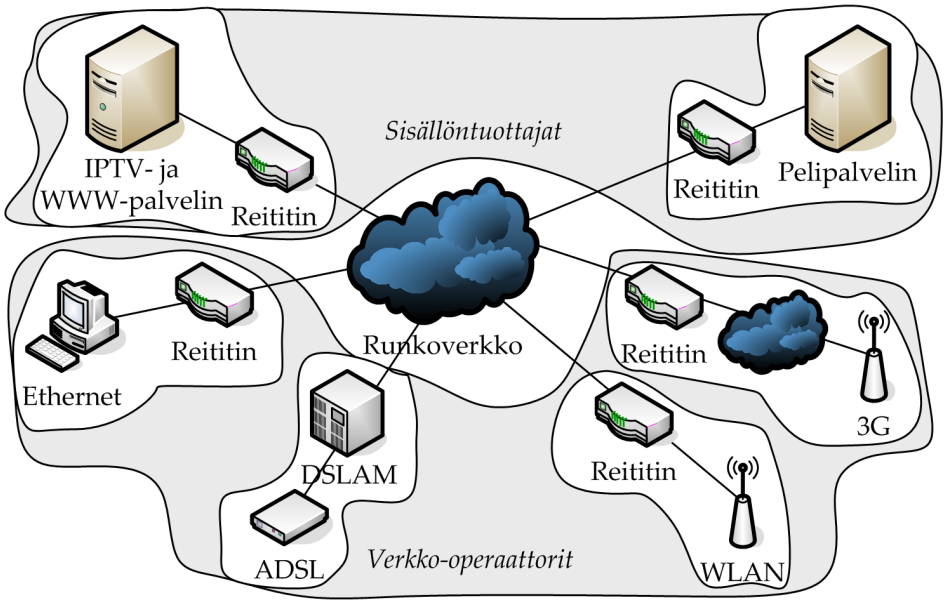
4.3.4 Yhteenveto

Ciscon IP SLA on hyvä monitorointitekniikka laajoihin verkkoihin, koska se on valmiiksi toteutettuna Ciscon laitteisiin. Ei siis välttämättä tarvita mitään erillistä monitorointisovellusta. IP SLA -operaatiot mittaavat useita suorituskykyparametreja samanaikaisesti. Tämä helpottaa monitorointia, koska ei tarvitse määritellä jokaista parametria erikseen. Laajoihin verkkoihin tarkoitetut operaatioryhmät helpottavat ja nopeuttavat mittausten konfigurointia. IP SLA:ssa on myös valmiita operaatioita eri palvelimien ja verkkosovellusten suorituskyvyn monitorointiin. Esimerkiksi nykyään suositun VoIP:in monitorointiin on valmiit, juuri kyseiselle liikenteelle räätälöidyt operaatiot. Huono puoli IP SLA:ssa on se, että käytössä täytyy olla Ciscon verkkolaitteistoa. IP SLA:sta voisi tehdä yleisen standardin ja se voitaisiin sulauttaa muidenkin valmistajien verkkolaitteisiin. Tällöin mittauksia voisi tehdä vielä monipuolisemmin. OWAMP on esimerkki tällaisen yleisen standardin kehittelystä. Tämä vaikuttaisi tietysti valmistajien väliseen kilpailuun ja vaatisi paljon muitakin muutoksia valmistajilta. Verkoissa, jotka koostuvat Ciscon laitteista, IP SLA on hyvä suorituskyvyn monitorointitekniikka monipuolisuutensa vuoksi.

5 Käytännön testit

Tutkielman empiirisessä osuudessa keskitytään siihen, miten eri mittaustekniikat soveltuvat IPTV-verkon suorituskyvyn mittaamiseen. Tässä keskitytään erityisesti siihen, mikä tekniikka sopii mihinkin verkon osaan parhaiten. Tässä luvussa esitellään myös teorialuvuissa esiteltyjen mittaustekniikoiden käytännön toteutuksia ja niiden käyttöesimerkkejä. TWAMP-protokollalle ei ole vielä olemassa toteutusta, joten sitä ei testattu. Myöskään RMON:ia ei testattu, koska testauksessa käytetty VizTool-sovellus ei testivaiheessa tukenut SNMP-hälytysten vastaanottoa.

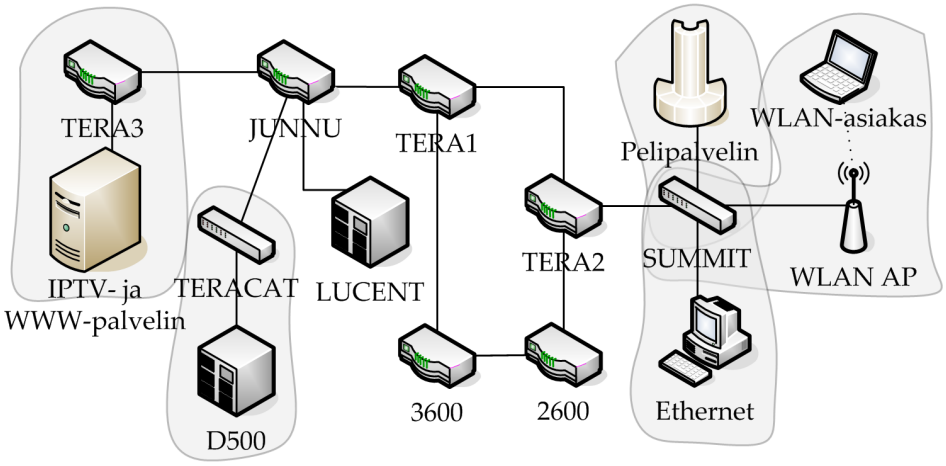
5.1 Testausympäristö



Kuva 5.1: Testiverkon lähtökohta.

Testausympäristön lähtökohtana käytetään IPTV-verkkoa, jossa toimii eri osapuolia, kuten runkoverkko-operaattori ja sisällöntuottajat. Testausympäristön perustana käytetään kuvan 5.1 mukaista tilannetta. Verkossa toimii siis erillinen runkoverkko-operaattori, kaksi eri sisällöntuottajaa ja eri liityntöjä tarjoavia verkko-operaattoreita. Tätä kuvaa vastaava verkko toteutetaan laboratorio-olosuhteissa, jossa eri testejä voidaan ajaa vapaasti. Testauksissa on otettava kuitenkin huomioon,

ettei mitä tahansa testejä voida ajaa missä tahansa. Operaattorit eivät pääse toisten operaattorien laitteisiin käsiksi oikeassa tilanteessa, vaikka tämä laboratorioverkossa onkin mahdollista.



Kuva 5.2: Testiverkon runkolaitteet.

Taulukko 5.1: Verkon aktiivilaitteet.

Laitteen nimi	Laite	Tyyppi
SUMMIT	Kytkin	Extreme Summit 48s
WLAN AP	WLAN tukiasema	Cisco Aironet 1200
TERA2	Reititin	Cisco 7200
TERA1	Reititin	Cisco 7200
JUNNU	Reititin	Juniper M5
TERA3	Reititin	Cisco 2651
2600	Reititin	Cisco 2610
3600	Reititin	Cisco 3640
TERACAT	Kytkin	Cisco Catalyst 2950
D500	DSLAM	Nokia D500
LUCENT	DSLAM	Lucent DSLAM

Laboratorioon on rakennettu kuvan 5.2 mukainen verkko, jossa testaukset suoritetaan. Tässä runkoverkkoa kuvastavat laitteet JUNNU, TERA1, TERA2, 2600 ja 3600. Sisällöntuottajina ovat IPTV-palvelin, WWW-palvelin ja Pelipalvelin. TERA3 toimii IPTV-sisällöntuottajan oman verkon reunareitittimenä. Taulukossa 5.1 on esitetty verkon aktiivilaitteiden nimet ja mallit. Tähän verkkoon sovelletaan kuvan 5.1 mukaista ajatusta. Mittaukset suoritetaan siis samalla tavalla kuin ne suoritettaisiin

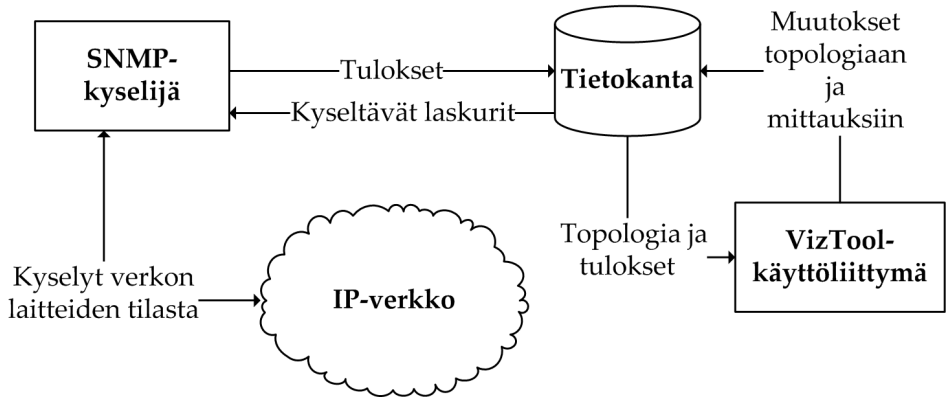
oikeassakin verkossa. Runkoverkon laitteita ei voida monitoroida esimerkiksi suoraan asiakaslaitteella. Ainoastaan palvelut näkyvät muille operaattoreille.

5.2 SNMP

SNMP-mittauksia valvottiin pääasiassa kahdella sovelluksella: Net-SNMP:llä ja VizToolilla. **Net-SNMP** on avoin komentorivipohjainen sovelluskokoelma SNMP-toiminnallisuuksien toteuttamiseen. Testauksessa käytettiin `snmpget` ja `snmpwalk` -ohjelmia. `Snmpget` hakee yhden laskurin arvon ja `snmpwalk` useita. Esimerkki `snmpwalk`-komennon käytöstä:

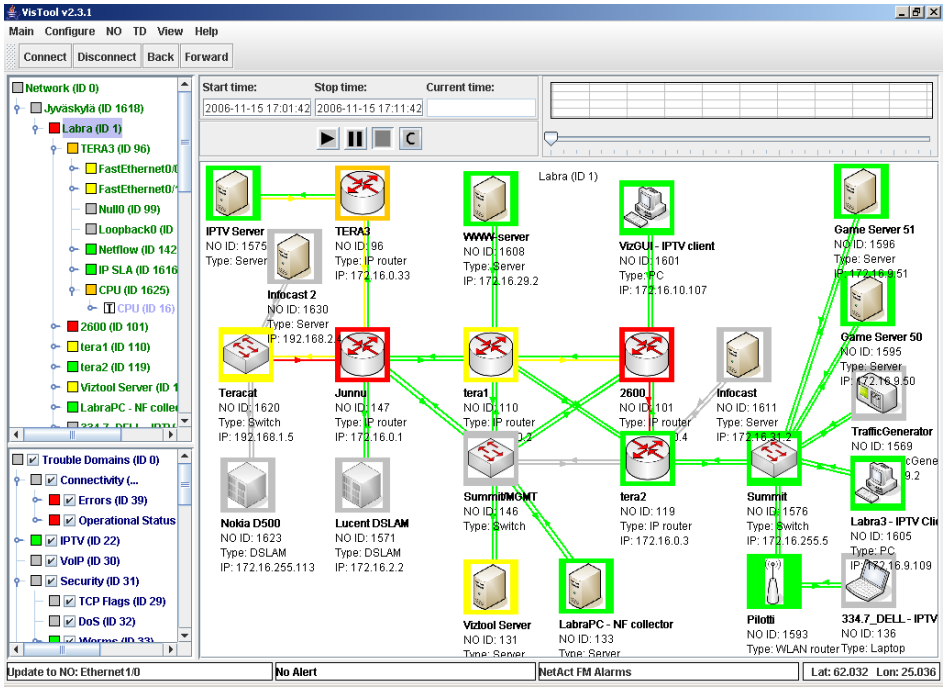
```
snmpwalk -c public -v 2c 172.16.0.1 interfaces
```

Komento hakee kohteen `172.16.0.1` `interfaces`-hallintatietokannasta kaikki arvot käyttäen avainta `public` ja SNMP-versiota `2c`. `Snmpget`-komento hakee vain yhden laskurin arvon tai ilmoittaa virheestä, ellei laskuria ole olemassa. Molemmat ohjelmat ovat hyviä ja nopeita apuvälineitä verkon laitteiden tilan pikaiseen tarkistukseen.



Kuva 5.3: VizTool-visualisointiohjelman arkkitehtuuri.

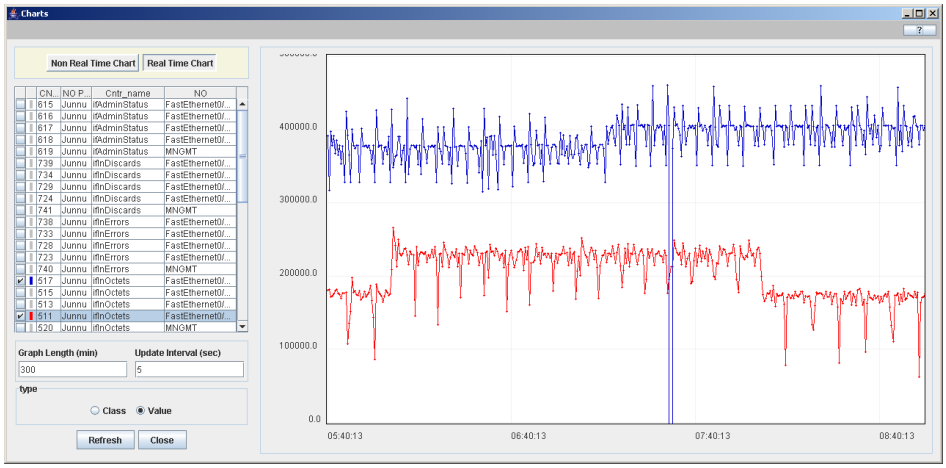
Pääsääntöisesti verkon valvontaa suoritettiin **VizTool**-sovelluksen avulla, jonka arkkitehtuuri on esitetty kuvassa 5.3. Erillinen SNMP-kyselijä (engl. *poller*) kerää verkon laitteilta tietoja SNMP:n *get*-metodilla ja kirjoittaa sen jälkeen tulokset tietokantaan. VizTool on Javalla kehitetty verkon visualisointisovellus, jolla näitä SNMP-kyselijän tietokantaan keräämiä tietoja voidaan tarkastella.



Kuva 5.4: VizTool-visualisointiohjelman topologianäkymä.

Kuvassa 5.4 on esitetty miltä VizToolin avulla esitetty verkon topologiakuva näyttää. Laskureille määrätään VizToolissa luokittelijat, joiden mukaan verkon objektit luokitellaan niiden tilaa kuvaavilla väreillä. Värit siirtyvät verkon syvemmältä tasolta ylimmälle tasolle, jolloin virhetilanteet voidaan havaita välittömästi. Kuvassa näkyvät VizToolissa käytetyt värit, jotka ovat normaalitilanteesta lähtien: vihreä, keltainen, oranssi ja punainen.

VizToolin avulla voidaan myös katsoa laskureiden luokkien ja arvojen historiatiedot kaavioina. Kuvassa 5.5 on esitetty kahden Fast Ethernet -liitynnän läpäisy (engl. *throughput*). Kuvan ylemmässä kuvaajassa näkyy ajan hetkellä 7.30, kuinka kuvaaja käy hetkellisesti nollassa. Tämä johtuu 32-bittisten `counter`-tyyppisten SNMP-laskureiden liian pienestä koosta [15]. Tavulaskuri laskee läpi menneiden pakettien kokonaismäärää, joista läpäisy saadaan kaavalla vähentämällä mittaushetken tavumäärästä edellisen mittaushetken tavumäärä ja jakamalla erotus kyselyvälillä. Kun laskuri saavuttaa arvon $2^{32} \approx 4 * 10^9$, se nollautuu ja läpäisyksi saadaan negatiivinen arvo, joka tulkitaan kuitenkin nollaksi. 100 Mbps nopeudella 2^{32} tavun siirtyminen kestää noin 6 minuuttia ja 10 Mbps nopeudella noin 57 minuuttia. Tätä ongelmaa korjaamaan MIB-II:een on kehitetty laajennus, jossa käytetään 64-bittisiä lukuja tavu- ja pakettilaskureissa. [55]



Kuva 5.5: VizTool-visualisointiohjelman kaavionäkymä.

5.2.1 Käytännön havaintoja SNMP:n toiminnasta

SNMP-testauksissa mitattiin seuraavia MIB-II hallintatietokantastandardin [56] mukaisia laskureita:

- *ifAdminStatus*, *ifOperStatus*
- *ifInOctets*, *ifInUcastPkts*, *ifInNUcastPkts*
- *ifInDiscards*, *ifInErrors*
- *ifOutOctets*, *ifOutUcastPkts*, *ifOutNUcastPkts*
- *ifOutDiscards*, *ifOutErrors*, *ifOutQLen*.

Lisäksi mitattiin seuraavia epästandardeja laskureita:

- *hrProcessorLoad* Windowsin `host-resources-mib`[1] -hallintatietokannasta (minuutin keskiarvo).
- *cpmCPUTotal5secRev*, *cpmCPUTotal1minRev* ja *cpmCPUTotal5minRev* `CISCO-PROCESS-MIB`[2]-hallintatietokannasta Ciscon 7200-sarjan reitittimiltä.
- *avgBusy1* ja *avgBusy5* `OLD-CISCO-CPU-MIB`[3]-hallintatietokannasta Ciscon 2600-sarjan reitittimiltä.

[1] <http://net-snmp.sourceforge.net/docs/mibs/host.html>
[2] <http://tools.cisco.com/Support/SNMP/do/BrowseMIB.do?local=en&step=2&mibName=CISCO-PROCESS-MIB>
[3] <http://tools.cisco.com/Support/SNMP/do/BrowseMIB.do?local=en&step=2&mibName=OLD-CISCO-CPU-MIB>

SNMP-hälytyksiä (engl. *trap*) ei testattu, koska VizTool ei tue niiden vastaanottoa. Testauksen aikana havaittiin, että SNMP-paketit voivat hävitä ruuhkaisessa verkossa. Testiverkon hallintaverkko oli 10 Mbps nopeuksinen Ethernet, joka kovan kuormituksen aikana hävitti UDP:n päällä kulkevia SNMP-paketteja. SNMP-pakettien perille menoa voidaan kuitenkin parantaa antamalle niille korkea prioriteetti. Myös ylikuormitettu reititin voi aiheuttaa ongelmia. Varsinkin Ciscon 2600-sarjan reititin on helppo ylikuormittaa esimerkiksi jatkuvilla *ping*-kutsuilla, jolloin reititin ei ehdi vastaamaan SNMP-kyselyihin. Kyseinen reititin on kuitenkin suhteellisen vanha ja uudemmissa reitittimissä on varauduttu paremmin tällaisiin palvelunestohyökkäyksiin (engl. *Denial of Service*). SNMP-kyselyt eivät kuitenkaan ole joka tilanteessa kovin luotettavia saavutettavuuden mittaajia.

SNMP-kyselyt soveltuvat hyvin liityntöjen liikennemäärien (engl. *throughput*) mittaamiseen. Pientä ongelmaa aiheuttavat 32-bittisten laskureiden nollaantuminen. 64-bittiset laskurit korjaavat nämä ongelmat ja tietoliikennelaboratorion laitteista ainakin 7200-sarjan reitittimet ja Summit-kytkin tukevat näitä.

Jatkuvasti kasvavien arvojen lisäksi jonkin ajan keskiarvot sopivat hyvin SNMP-kyselyille, olettaen että keskiarvot ovat kyselyväliä pidemmältä aikaväliltä. Esimerkiksi *hrProcessorLoad* ja *cpmCPUTotal1minRev* -laskurit kertovat prosessorikuorman keskiarvon viimeisen minuutin ajalta. Jos kyseisiä laskureita kysellään minuutin välein, saadaan kattava tieto laitteen prosessorin käyttöasteen historiasta. Sen sijaan *cpmCPUTotal5secRev*-laskuri minuutin välein kyseltynä ei yksinään kerro kovin hyvin laitteen prosessorinkäytöstä. Lisäksi hetkellisten arvojen, kuten *ifOutQLen*-laskurin kysely ei kerro paljoakaan.

Ryhmälähetysten (engl. *multicast*) erittelyyn ei MIB-II hallintatietokanta sovellu kovin hyvin. Sen avulla näkee vain niiden pakettien määrät, jotka eivät ole täsmälähetyksiä (engl. *unicast*). *IfOutNUcastPkts* ja *ifInNUcastPkts* -laskurit sisältävät siten ryhmälähetysten lisäksi myös yleislähetysten (engl. *broadcast*) paketit. Pakettien kokoa eli menneen liikenteen määrää ei pysty MIB-II:n avulla tietämään. MIB-II ei myöskään tarjoa minkäänlaista tietoa käytetyistä sovelluksista tai liikenteen osapuolista.

5.2.2 SNMP:n soveltuvuus

Runkoverkossa SNMP on ollut pitkään perusvalvontatyökalu. SNMP-kyselyillä saadaan hyvin tieto laitteiden tiloista ja liikenteen määrästä. Kun VizToolin kaltaiseen työkaluun valitaan järkevät laskurit valvottavaksi, huomataan eri puolilla verkkoa syntyvät ongelmatilanteet nopeasti. Standardista MIB-II:sta `interfaces` on tär-

kein valvottava taulu. Sen avulla voidaan havaita liitynnät jotka muodostavat verkkoon pullonkaulan tai jotka eivät ole päällä. SNMP soveltuu siinäkin mielessä hyvin yleisvalvontaprotokollaksi, koska tuki sille löytyy käytännössä kaikkialta.

Verkko-operaattorit voivat käyttää SNMP:tä runkoverkko-operaattoreiden tavoin. Luonnollisestikaan valvontaa ei voi ulottaa oikeille asiakkaille asti. SNMP:llä voidaan kuitenkin valvoa operaattorin viimeistä omaa laitetta, kuten DSLAMia, WLAN-tukiasemaa tai kytkintä. Esimerkiksi WLAN-tukiasemalla suuri määrä tippuneita lähteneitä paketteja (*ifOutDiscards*) ja samanaikaisesti suuret siirtomäärät ulospäin (*ifOutOctets*) saattavat osoittaa verkon ruuhkautumista.

Sisällöntuottajat voivat valvoa SNMP:llä omien laitteidensa toimivuutta ja liityntöjensä kapasiteettia. Kovin tarkkaa kuvaa tarjotun palvelun laadusta ei standardi SNMP:llä voi saada. Pakettien tippuminen tai liityntöjen ruuhkautuminen voidaan kuitenkin havaita.

5.3 NetFlow

Ennen kuin NetFlowta pystyttiin testaamaan, täytyi reitittimet asettaa keräämään ja lähettämään NetFlow-tietoja. Testeissä käytettiin seuraavia yleisiä NetFlow-asetuksia:

```
ip flow-cache timeout active 1
ip flow-cache timeout inactive 15
ip flow-export source Loopback0
ip flow-export version 5
ip flow-export destination 172.16.255.103 5004
```

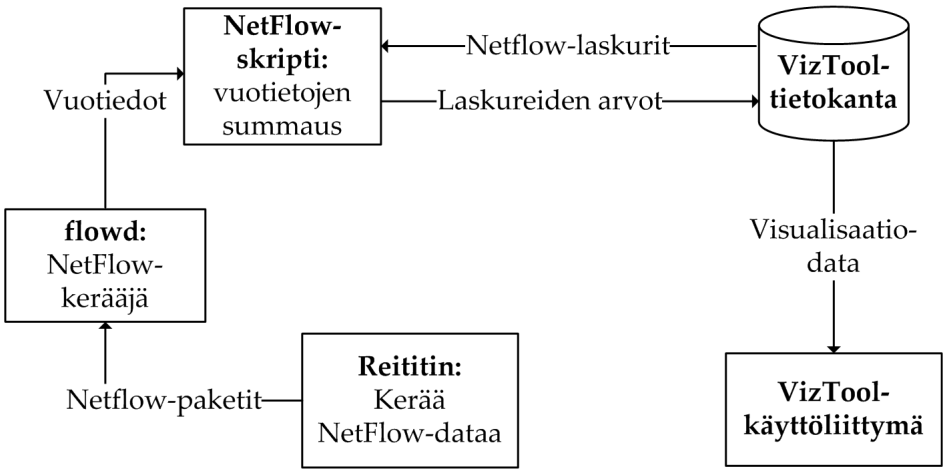
Tämän jälkeen reitittimen aktiivisten voiden aikakatkaisu oli 1 minuutti ja passiivisten 15 sekuntia. Vuotiedot lähetettiin version 5 protokollalla osoitteeseen `172.16.255.103` ja porttiin `5004` lähettäjän ollessa `Loopback0`. Lisäksi jokaiselle liitynnälle, jolta NetFlow-tietoa haluttiin saada, täytyi asettaa NetFlow-tietojen kerääminen erikseen päälle:

```
ip route-cache flow
```

NetFlown testaamista varten toteutettiin NetFlow-tietojen vastaanotto ja tietojen syöttö VizToolin tietokantaan. Kuvassa 5.6 on esitetty NetFlow-toteutuksen rakenne. *Flowd*[4] on unix-pohjaisille järjestelmille kehitetty NetFlow-kerääjä. Se vastaanottaa NetFlow-paketteja ja tallentaa niiden tiedot omaan binäärimuotoonsa. Flowd

[4]http://www.mindrot.org/projects/flowd/

tarjoaa sen lisäksi monipuoliset rajapinnat, joiden avulla NetFlow-tietoihin pääsee käsiksi. *Flowd*:n Perl-rajapintaa käyttäen toteutettiin skripti, joka noutaa aika ajoin laskureiden tiedot VizToolin tietokannasta, laskee laskureita vastaavista *flowd*:n tarjoamista vuotiedoista tavumäärät ja kirjoittaa laskureiden tavumäärät VizToolin tietokantaan.



Kuva 5.6: NetFlow-toteutuksen rakenne.

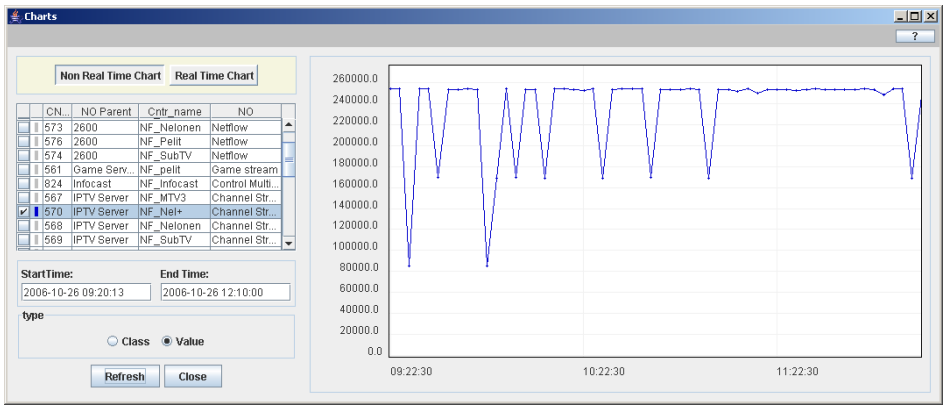
Testiverkossa NetFlow-tuki oli vain Ciscon reitittimillä. Sen lisäksi JUNNU-reititin tuki Juniperin omaa *cflowd*-vuoprotokollaa, jota *flowd* ei kuitenkaan osannut vastaanottaa. Reitittimien lisäksi Ciscolla on tarjolla kytkimiä, joissa on NetFlow-tuki, mutta tällaisia kytkimiä ei ollut testiverkossa käytettävissä.

5.3.1 Käytännön havaintoja NetFlown toiminnasta

NetFlown avulla mitattiin läpimenneen liikenteen määrää (engl. *throughput*). Saadut arvot olivat kuitenkin suuntaa antavia, eivät ehdottoman tarkkoja. Heittoa NetFlow-laskureiden arvoihin aiheuttivat tai voivat aiheuttaa seuraavat seikat:

- Näytteistys
- NetFlow-pakettien hukkuminen
- NetFlow-pakettien lähettämättä jättäminen
- Vuotietojen kirjaaminen väärälle aikavälille
- Vuotietojen saapumisen viive

Näytteistys aiheuttaa epävarmuutta arvoihin, koska kaikkia paketteja ei tutkita. Näytteistystä ei kuitenkaan käytetty testiverkossa. NetFlow-pakettien lähettämättömyys tai matkalle hukkuminen aiheutti välillä suurta heittoa arvoihin. NetFlow-paketit liikkuvat verkossa UDP:n päällä, joten ne voivat tippua, eikä NetFlow-protokollan toimintaan kuulu uudelleenlähetykset. Toisaalta kuormitettu reititin välttämättä kirjaa kaikkien havaitsemiensa pakettien tietoja vuovälimuistiinsa tai lähetä kaikkia NetFlow-paketteja. Kuvassa 5.7 näkyy kuinka tasaisen bittivirran omaavan vuon kaavio heittelee, koska ruuhkaisessa hallintaverkossa kaikki NetFlow-paketit eivät saapuneet kerääjälle asti. Tässä tapauksessa laskurin tiedot summattiin kolmen minuutin välein ja reititin lähetti vuon tiedot minuutin välein. Yhteen kaavion mittaustulokseen laskettiin siis kolmen NetFlow-paketin tiedot. Yhden paketin hukkuminen näkyy kuvassa selvästi laskurin arvon tippuessa $2/3$-osaan todellisesta.

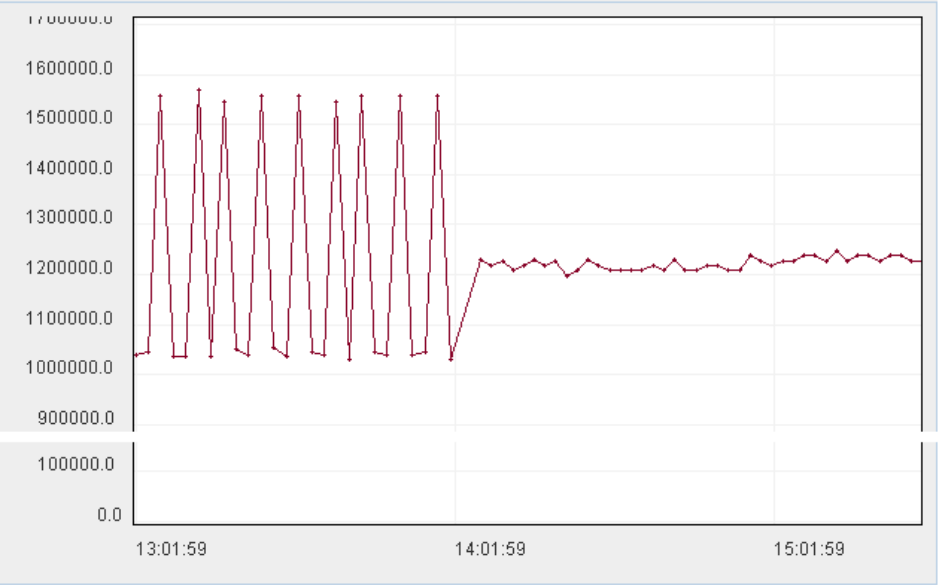


Kuva 5.7: NetFlow-pakettien hävikin aiheuttama kaavion heittely.

NetFlow-laskurien arvot saattavat heitellä, vaikka mitattava vuo olisikin tasaista bittivirtaa ja kaikki paketit tulisivat kerääjälle asti. Kuvassa 5.8 näkyy kuinka ajan hetkellä 14.00 kaavion heittely tasaantuu. Ennen sitä vuotietoja summattiin laskuriin 140 sekunnin välein ja reititin lähetti NetFlow-paketteja kyseisestä vuosta minuutin välein. Tällöin laskurin arvoon tuli mukaan vaihtelevasti kahden tai kolmen NetFlow-paketin tiedot. Summausvälin muuttaminen kahteen minuuttiin vähensi kuvion heittelyä huomattavasti. Ilmiöstä on kerrottu myös luvussa 3.5.4. Summausvälin kasvattaminen ja useamman vuon tiivistäminen samaan laskuriin pienentää heittelyä, koska yhden vuon tiedot vaikuttavat vähemmän laskurin arvoon.

NetFlow-laskureiden aikaleimaksi kirjattiin se hetki, kun arvot kirjattiin VizToolin tietokantaan. Tätä ennen kerääjä summasi tavut, joka kesti pahimmillaan puoli minuuttia. Koska kyseisellä summauskierroksella käsiteltävien vuotietojen keräys lopetettiin kierroksen alussa, viimeisinä käsiteltäviin laskureihin tuli pahimmillaan

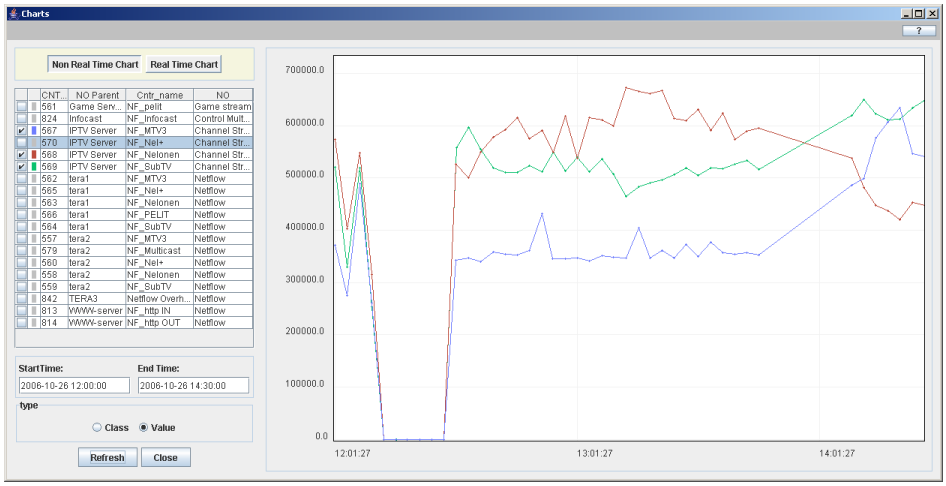
puolen minuutin viive. Tämä aikavirhe olisi kuitenkin mahdollista poistaa antamalla jokaiselle laskurille sama summauskierroksen aloitushetken aika. Tämä edellyttäisi kuitenkin VizToolin tietokannan ja NetFlow-kerääjän järjestelmien kellonaikojen säännöllistä synkronointia. Tutkielman kannalta toteutettu versio on kuitenkin riittävän tarkka, eivätkä muutamien sekuntien vaihtelut arvojen ajoissa aiheuttaneet mitään ongelmia.



Kuva 5.8: NetFlow-mittausten vaihtelut huonosti valitulla keräämisvälillä.

Kuvassa 5.9 näkyy kuinka testiverkon IPTV-lähetyksien bittivirtaa pystyttiin seuraamaan VizToolin ja NetFlown avulla. Jokainen IPTV-kanava lähetettiin omana ryhmälähetyksenä, eli jokainen kanava sai oman ryhmälähetysosoitteensa. Jokaiselle kanavalle voitiin siten muodostaa NetFlow-laskuri tämän ryhmälähetysosoitteen perusteella.

Artikkelissa [57] on testattu NetFlown käytännön suorituskykyä Ciscon reitittimillä. Testissä lähetettiin 65 000 eri vuohon kuuluvaa 64 tavun pakettia minuutin aikana. Tällöin kaistan bittivirta oli noin 600 kbps. Testin liikennemäärä oli pientä liityntöjen mahdollistamiin nopeuksiin nähden (esimerkiksi Fast Ethernet -liitynnällä 100 Mbps), joten siinä mielessä testi ei kuvaa kovin hyvää ruuhkaisen reunareitittimen kuormituksesta. Toisaalta artikkelin mukaan pienet paketit kuormittavat reititintä enemmän kuin suuremmat, vaihtelevan kokoiset paketit. Lisäksi reititin joutui luomaan oman rivin välimuistiinsa jokaiselle paketille, joka luonnollisesti kuormittaa reititintä enemmän kuin sama liikennemäärä vähemmillä erilaisilla voilla.



Kuva 5.9: NetFlow-mittauksia VizToolin kaavionäkymässä.

Testiverkossa käytetyistä reitittimistä Ciscon testissä olivat mukana 2600-, 3640- ja 7200-sarjan reitittimet. Yhteenveto testin tuloksista on esitetty taulukossa 5.2. Tuloksista nähdään kuinka vanhemmalla 2600-sarjan reitittimellä maksimaalinen NetFlown käyttö kuormittaa reititintä huomattavasti. Tuloksia tulkittaessa täytyy muistaa, että ne on ajettu Ciscon toimesta laboratorio-olosuhteissa. Tulokset kertovat kuitenkin että NetFlowlla voi olla suuri vaikutus reitittimien suorituskykyyn, joten ennen NetFlown käyttöönottoa täytyy sen vaikutus testata tapauskohtaisesti.

Taulukko 5.2: Ciscon reitittimien NetFlow-suorituskyky

Reititin	**Lähtötaso**	**NetFlown aiheuttama maksimikuormitus**
2600	16	69
3640	6	40
7200NPE300	11	51

Toteutettu NetFlow-skripti pystyi käsittelemään noin 2000 vuota sekunnissa eli 120000 vuota minuutissa. Kaikki vuot täytyy kuitenkin käydä jokaiselle laskurille erikseen läpi. Skripti pyöri CentOS 4 Linuxilla, jossa laitteistona oli Celeron 600 MHz 256Mt muistilla. Suorituskyky paranisi siis selvästi jo laitteiston päivityksellä. Flowd tarjoaa Perlin lisäksi rajapinnat Pythonille ja C++:lle, joista varsinkin jälkimmäisen käyttäminen saattaisi parantaa suorituskykyä huomattavasti. Joka tapauksessa vuotietojen jatkokäsittely ei vaikuta verkon suorituskykyyn. Testiverkossa täydellisten, pakkaamattomien, vuotietojen säilöminen vei levytilaa noin gigatavun kuukaudessa. Erilaisten voiden lisääntyessä levytilan määrä kuitenkin kasvaa.

Yksi *flowd*:n säilömä vuo vie noin 100 tavua levytilaa. Tällöin edellä esitetyn Ciscon testitilanteen mukaisella 65 000 vuon minuuttivauhdilla ja minuutin välein tapahtuvalla vuotietojen tallennuksella, saadaan vaadituksi levytilaksi jo 8,7 Gt ($100\,t * 65000 * 60 * 24 = 9360000000\,t \approx 8,7\,Gt$) per vuorokausi.

5.3.2 NetFlown soveltuvuus

Runkoverkossa NetFlown avulla saa hyvän yleiskuvan runkoverkon liikennemääristä, protokollista, sovelluksista ja sen osapuolista. Tätä tietoa voi sitten käyttää avuksi verkon suunnittelussa. Nämä tilastot nähdään vielä suhteellisen vähällä vaivalla.

Verkko-operaattorit voivat käyttää reunareitittimiensä NetFlow-tietoa laskutusta varten. NetFlown avulla saadaan tavumäärien lisäksi tieto liikenteen kohteesta, jolloin voidaan laskuttaa esimerkiksi vain operaattorin verkon ulkopuolelle suuntautuvasta liikenteestä. Loppukäyttäjän kokonaiskaistan valvontaan NetFlow ei tarjoa mitään juuri mitään lisäarvoa SNMP-kyselyihin nähden. Sen sijaan käytettyjä portteja tai protokollia voidaan valvoa NetFlown avulla. Lähinnä loppukäyttäjää olevilta verkkolaitteilta eli WLAN-tukiasemilta, DSLAM:eilta tai kytkimiltä ei yleensä saada NetFlow-tietoa. NetFlow-valvonta täytyy siis suorittaa lähimmällä reitittimellä, jolloin loppukäyttäjän verkkosegmentti jää NetFlow-valvonnan ulkopuolelle. Lisäksi verkko-operaattorit voivat tunnistaa matojen saastuttamia käyttäjiä, estää hyökkäysten leviäminen ja siten parantaa verkon hyötykuormaa.

Sisällöntuottajat voivat valvoa eri palveluiden kaistankäyttöä NetFlown avulla, mikäli palvelut sijaitsevat eri osoitteissa tai porteissa. Lisäksi voidaan valvoa mistä palveluita käytetään. Palvelun laatua ei voida valvoa NetFlown avulla kovin hyvin. Verkko-operaattoreiden tavoin myös sisällöntuottajat voivat käyttää NetFlowta matohyökkäysten havainnointiin.

5.4 OWAMP

OWAMPia voidaan käyttää päästä-päähän -mittauksiin. Mittauksia voidaan tehdä esimerkiksi IPTV-asiakkaan ja IPTV-palveluntarjoajan välillä. Mitattaessa koko väliä saadaan tietoon suorituskykyparametrit koko tiedonsiirtomatkalta. Toisaalta tällainen mittaustapa ei ongelmatapauksissa anna mitään tietoa ongelman sijainnista. Spatiaalinen mittaustapa olisi parempi. Ongelmien olemassaolo voidaan kuitenkin todeta tällä tavalla. Toinen tapa voisi olla, että esimerkiksi runkoverkossa olisi OWAMP-palvelin ja molemmat osapuolet voisivat monitoroida itsensä ja runkover-

kon väliä. Tutkielman esimerkkitapauksessa OWAMP-palvelin voisi olla esimerkiksi IPTV- tai Pelipalvelimen yhteydessä, koska OWAMP-palvelin ei vaadi paljoa laitteistoresursseja. Runkoverkon monitorointiin OWAMPia ei kannata käyttää, koska siihen on monipuolisempia ja järkevämpiä tekniikoita. OWAMPilla voitaisiin esimerkiksi selvittää, mistä johtuu jos IPTV:n katselussa ilmenee jotain häiriötä. Selvitetään, johtuuko se viiveestä, hävikistä vai jostain muusta. Tämän jälkeen paikallistetaan vika jollain muulla tekniikalla. Toisaalta, jos OWAMPista tulisi verkkolaitteisiin sisäänrakennettu tekniikka, voisi sitä käyttää myös runkoverkon suorituskyvyn mittauksissa. OWAMPista voisi tulla Ciscon IP SLA:n korvaaja, jos sen implementoisivat kaikki laitevalmistajat.

5.4.1 One-way Ping

One-way Ping on Internet2- työryhmän tekemä OWAMP-protokollan käytännön toteutus. One-way Ping on toteutettu OWAMP-protokollan client-server -mallilla. One-way Pingillä voidaan mitata yhdensuuntaista viivettä, viiveen vaihtelua, pakettien uudelleenjärjestelyä ja hävikkiä. One-way Ping mittaa nämä parametrit molempiin suuntiin [58].

One-way Ping on toteutettu kuvan 5.10 mukaisesti. Palvelimelle käynnistetään kolme prosessia, joista kuvassa ylimpänä oleva hallintaprosessi hoitaa yhteyden luonnin ja tietoturvaan liittyvät käytänteet asiakaslaitteen hallintaprosessin kanssa. Palvelimen toinen hallintaprosessi hoitaa testausistunnon parametrien sopimisen asiakaslaiteen hallintaprosessin kanssa. Mittausprosessit hoitavat mittauspakettien lähetyksen ja vastaanottamisen. One-way Ping ei vielä tue hajautettua mallia, vaikka prosessit on toteutettu erillisinä. Sovellus on toteutettu sekä Linux- että Windows-käyttöjärjestelmille. Windows-versio on toteutettu Javalla ja vaatii erillisen tietokannan sekä Web-palvelimen pystyttämisen. Tämä on työlästä verrattuna Linux-versioon. Linux-versiossa palvelinkoneelle täytyy käynnistää *owampd*-prosessi. Asiakaskoneella ajetaan *owping*-prosessi, jolle annetaan palvelinkoneen IP-osoite ja halutessa myös muita parametreja. Palvelinlaitteella täytyy olla käynnissä *owampd*-prosessi. Ennen *owpingin* ajamista kannattaa laitteiden kellot synkronoida. Tämä onnistuu Linuxissa *ntpd -q* -käskyllä, jolloin koneen kello päivitetään määritellyltä NTP-palvelimelta.

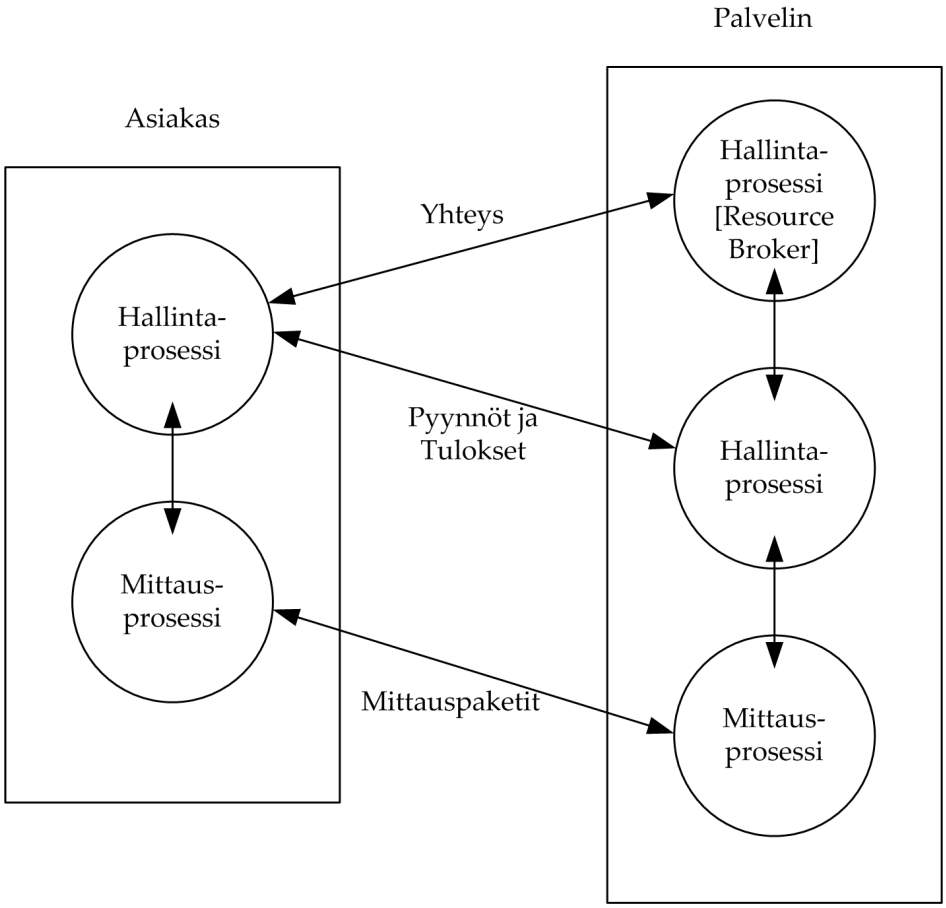
Seuraavassa on esimerkkiajo owpingistä:

```
./owping 130.234.169.76
Approximately 12.9 seconds until results available

--- owping statistics from [130.234.169.71]:33039 to [130.234.169.76]:32771 ---
SID: 82eaa94cc9055690ec78c86824749eab
```

```
100 sent, 0 lost (0.000%), 0 duplicates
one-way delay min/median/max = 0.196/0.3/0.286 ms, (err=250 ms)
one-way jitter = 0 ms (P95-P50)
TTL not reported
no reordering


--- owping statistics from [130.234.169.76]:32772 to [130.234.169.71]:33040 ---
SID: 82eaa947c9055690f1e95b789450a9a9
100 sent, 0 lost (0.000%), 0 duplicates
one-way delay min/median/max = -0.124/-0.1/2.66 ms, (err=250 ms)
one-way jitter = 0 ms (P95-P50)
TTL not reported
no reordering
```



Kuva 5.10: One-way Ping sovelluksen arkkitehtuuri.

Tässä ajossa laite lähettää 100 mittauspakettia palvelimelle ja palvelin 100 pakettia takaisin. Näistä mitataan minimi, maksimi ja keskimääräinen viiveen arvo ja hävikki molempiin suuntiin. Viiveen vaihtelu mitataan paketeista 50-95. Esimerkkiajo ei ole kovin informatiivinen, koska se suoritettiin liian hyvissä olosuhteissa.

Verkossa ei ollut ruuhkaa ja laitteiden välimatkat pieniä. Esimerkkiajosta käy kuitenkin ilmi, miten One-Way Ping esittää tulokset. One-Way Pingissä on myös erillinen tulostennouto-prosessi, mutta sen käyttäminen tulosten noutoon myöhempinä ajanhetkinä on työlästä johtuen aiemmin mainitusta tiedostojen nimeämiskäytännöstä. Esimerkkiajossa on viiveellä negatiivisia arvoja. Tämä johtuu siitä, ettei testauslaitteiden kelloja saatu synkronoitua riittävän tarkasti. Mitatut viiveet ovat niin pieniä että kellojen tuoma virhe vaikuttaa tuloksiin liikaa.

5.4.2 Eri liityntöjen näkökulmasta

Käytettäessä **Ethernet**-liittymää, on käytössä yleensä melko nopea yhteys verkkoon. Tällöin liittymän kaistanleveys riittää helposti esimerkiksi IPTV-palveluihin. Asiakas voi mitata OWAMPin avulla itsensä ja palveluntarjoajan väliä. Palveluntarjoaja ei taasen voi mitata samalla tavalla tätä väliä, koska asiakkaalla ei varmastikaan ole OWAMP-palvelinta. Palveluntarjoaja voisi monitoroida tällaista tilannetta siten, että sillä olisi testiasiakaslaite kytkettynä esimerkiksi samaan kytkimeen kuin sen asiakas, ja suorittamalla mittauksia sieltä palvelimeen. Testiverkossa suoritetuissa mittauksissa oli OWAMP-palvelin asennettu kuvan 5.2 IPTV-palvelimen yhteyteen, ja asiakaslaite oli kuvan 5.2 labra3-laite. Mittauksissa saatiin viiveelle keskiarvoksi noin kolme millisekuntia ja viiveen vaihtelulle kuusi millisekuntia. Hävikkiä eikä pakettien uudelleenjärjestelyä Ethernet-liittymällä tullut lainkaan. Yhdensuuntaisen viiveen maksimiarvo oli noin 20 millisekuntia. Viiveen vaihtelun ja yhdensuuntaisen viiveen arvot eivät ole toisiinsa nähden järkeviä. Tämä johtuu kellojen virheistä. Mittauksissa käytettiin NTP-protokollaa kellojen synkronointiin ennen mittausta. Viiveiden ollessa näin pieniä, aiheutuu virhettä jo kellojen vääristymisestä mittauksen aikana.

WLAN-liittymää käytettäessä kaistanleveys on yleensä pienempi kuin Ethernetissä. Lähetyksiin tulee myös herkemmin virheitä, koska käytetään radiotietä. WLAN-liittymilläkin pystyy käyttämään IPTV-palveluita. Monitorointi onnistuu samalla tavalla kuin Ethernet-liittynälläkin. Eli asiakaslaite voi mitata itsensä ja palvelimen väliä. Viiveet ovat WLAN-liittymää käytettäessä palvelimen väliä. Viiveet ovat WLAN-liittymää käytettäessä suurempia ja hävikkiä esiintyy enemmän kuin Ethernet-liittymällä. Palveluntarjoaja voi mitata tätä väliä samoin kuin Ethernetissä, tuomalla oman asiakaslaitteensa WLAN:iin ja mittaamalla sen ja palvelimen väliä. Laboratoriossa suoritetuissa mittauksissa palvelimena toimi sama laite kuin Ethernet-liittymääkin mitattaessa eli IPTV-palvelin. WLAN-tukiasema oli kytkettynä samaan kytkimeen kuin labra3-laite (kuva 5.2), joten Ethernetin ja WLAN:in ero-

ja on helppo verrata keskenään. Mittauksissa havaittiin, että yhdensuuntainen viive ja hävikki kasvoivat reilusti asiakkaalta palvelimelle päin mitattaessa. Toiseen suuntaan viive ja hävikki pysyivät kutakuinkin samassa kuin Ethernetin kohdalla. Erot hävikissä ja viiveessä asiakkaan ja palvelimen välillä mitattaessa eri suuntiin johtuvat lähetystehojen erosta tukiaseman ja verkkokortin välillä. Tästä voidaan päätellä että, viive ja hävikki sekä viiveen vaihtelu ovat korkeampia kun käytetään WLAN-liittymää, ja varsinkin kun samanaikaisia asiakkaita on useampia. Suorituskykyä on siis järkevää monitoroida, ja se onnistuu OWAMPin avulla helposti. Koska radiotie aiheuttaa selvästi enemmän viivettä ja hävikkiä kuin kiinteä yhteys, olisi järkevää sijoittaa OWAMP-palvelin WLAN-tukiaseman yhteyteen, jolloin voitaisiin monitoroida pelkän WLAN:in suorituskykyä. WLAN-linkki on tässä tapauksessa todennäköisin pullonkaula.

ADSL[5]-operaattori voi suorittaa mittauksia oman verkkonsa laitteilla. OWAMP-palvelin voisi sijaita esimerkiksi operaattorin reunareitittimien ja DSLAM:in välissä, jolloin voitaisiin suorittaa mittauksia ADSL-testiasiakkaalta palvelimelle. Voitaisiin mitata esimerkiksi, riittääkö DSLAM:in suorituskyky, kun siihen liitetään lisää asiakkaita, jotka käyttävät esimerkiksi IPTV-palveluita.

Käytettäessä **3G**-liittymää viive ja hävikki kasvavat verraten edellisiin tekniikoihin. 3G-liittymässä kaista ei riitä IPTV:n katseluun, mutta muita reaaliaikasovelluksia voidaan käyttää. OWAMPilla voidaan suorittaa mittauksia testiasiakkaalla samoin kuin WLAN:in tapauksessakin.

Sisällöntuottajat voivat käyttää OWAMPia esimerkiksi monitoroimaan omaa linkkiään runkoverkkoon. Tämä vaatii sen, että runkoverkossa on OWAMP-palvelin. Vaikka OWAMP olisi valmiiksi asennettuna verkon laitteille, ei mittauksia voitaisi suorittaa, koska runkoverkon laitteisiin ei päästä suoraan käsiksi. Toisaalta, jos kaikissa laitteissa olisi valmiina OWAMP-tuki, voisi jotkut runkoverkon laitteet toimia OWAMP-palvelimina, joihin olisi rajoitettu pääsy. Sisällöntuottajat voivat suorittaa mittauksia testiasiakkailla aiemmin kuvatulla tavalla esimerkiksi viemällä oma testilaite WLAN:iin, jossa palvelua käytetään ja suorittaa sieltä mittauksia omaan palvelimeen.

Runkoverkossa OWAMP olisi hyvä tekniikka, jos sille löytyisi tuki kaikilta laitteilta. Ciscon IP SLA on huomattavasti monipuolisempi suorituskyvyn mittaussovellus, mutta se toimii ainoastaan Ciscon laitteilla. OWAMP olisi hyvä vastine Ciscon tekniikalle, jos se olisi yleinen standardi ja kaikki laitevalmistajat tukisivat sitä. OWAMPissa olisi tietysti paljon kehitettävää ja sen ominaisuuksia olisi monipuolistettava paljon. Tällä hetkellä OWAMP ei sovi runkoverkon monitorointiin,

[5] Asymmetric Digital Subscriber Line

mutta yleistyessään se poistaisi eri valmistajien laitteiden välisiä ongelmia. Kaikkien laitevalmistajien saaminen tällaiseen mukaan olisi tietysti melko haastava tehtävä, koska esimerkiksi Cisco menettäisi yhden valttikorteistaan valmistajien välisessä kilpailussa.

5.5 IP SLA

IP SLA on Ciscon oma suorituskyvyn monitorointisovellus, joka on sisäänrakennettuna Ciscon reitittimille ja isoimmille kytkimille. IP SLA on erittäin hyvä sovellus runkoverkon monitorointiin, jos käytössä on pelkkiä Ciscon laitteita. IP SLA:lla voidaan tehdä automaattisia mittauksia, ja ne voidaan ajastaa tapahtumaan milloin tahansa tai vaikka aina tiettyyn kellonaikaan joka päivä. IP SLA:ssa on myös mahdollisuus asettaa erilaisia raja-arvoja, jotka ylittyessään laukaisevat muita mittauksia tai vaikka raportin lähetyksen verkonhallintalaitteelle. IP SLA:ta voidaan konfiguroida ja tuloksia noutaa Ciscon laitteiden omasta käyttöliittymästä tai SNMP:n avulla. Ciscon IP SLA:lla voidaan myös mitata eri palvelimien suorituskykyä. Asiakaslaitteita IP SLA:lla voisi teoriassa monitoroida, koska joillekin operaatioille kelpaa vastaanottajaksi mikä tahansa IP-laite. Käytännössä tämä ei ole kuitenkaan mahdollista, koska verkkolaitteet eivät näe asiakaslaitteita.

5.5.1 IP SLA:lla mittaaminen

Tässä tutkielmassa käytettävään mallitilanteeseen 5.1 IP SLA sopii erittäin hyvin. Runkoverkko-operaattori voi monitoroida omaa suorituskykyään ja IPTV -palveluntarjoaja voi monitoroida IPTV-palvelimensa suorituskykyä IP SLA -operaatioilla. Testiverkkoon konfiguroitiin tätä tilannetta vastaavat IP SLA -operaatiot. Laboratorion verkkolaitteissa on Ciscon IP SLA:ta edeltänyt suorituskyvyn mittaussovellus Cisco Service Assurance Agent eli SAA. SAA on IP SLA:n vanha versio. SAA:ssa operaatiot eroavat hieman IP SLA:han verrattuna. IP SLA:ssa on lisäksi operaatioita, joita SAA:ssa ei ole. Perusoperaatiot ovat kuitenkin sisällöltään samoja. Runkoverkon monitorointiin konfiguroitiin UDP Jitter -operaatio TERA1:ltä TERA2:lle, sekä UDP Jitter -operaatio ja ICMP Path Echo TERA2:lta TERA3:lle. TERA3 ei ole runkoverkkoa, mutta ajatellaan, että operaattoreilla olisi olemassa sopimus joka oikeuttaa tällaiseen mittaukseen. TERA3:lta tehdään myös HTTP -kyselyjä IPTV-palvelimelle. Sisällöntuottajana IPTV-operaattori valvoo palvelimensa tilaa seuraamalla vastaako se kyselyihin. WLAN-operaattoria varten konfiguroitiin ICMP Echo -operaatio TERA2:lta WLAN-testiasiakaslaitteelle. Mittaukset on esitetty taulukossa 5.3.

Taulukko 5.3: IP SLA -mittaukset.

Lähettävä laite	Vastaanottava Laite	IP SLA -operaatio
TERA3	IPTV -palvelin	HTTP
TERA1	TERA2	UDP Jitter
TERA2	TERA3	ICMP Path Echo
TERA2	TERA3	UDP Jitter
TERA2	WLAN-asiakas	ICMP Echo

IP SLA -operaatiot alustettiin ottamalla Telnet-yhteys reitittimiin, konfiguroimalla operaatiot reitittimien käyttöliittymästä. Operaatiot alustetaan käskyllä `rtr N`, jossa N on operaation tunnukseksi annettu kokonaisluku. Seuraavassa esimerkkinä UDP Jitter -operaation konfigurointi TERA1:n ja TERA2:n välille:

```
tera2#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
tera2(config)#rtr responder
tera2(config)#exit
```

Tässä asetettiin IP SLA -vastaajaprosessi TERA2:lle, jotta se vastaa saapuviin UDP Jitter -lähetyksiin.

```
tera1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
tera1(config)#rtr 65
tera1(config-rtr)#type jitter dest-ipaddr 172.16.6.2 dest-port 100
tera1(config-rtr-jitter)#exit
tera1(config)#rtr schedule 65 start-time now
tera1(config)#exit
```

Tässä asetettiin UDP Jitter -operaatio, jonka kohdeosoitteena on TERA2:n IP-osoite, ja kohdeportti 100. Operaatio käynnistettiin heti. Operaation asetuksia voidaan tarkastella jälkeenpäin komennolla `show rtr configuration 65`. Komento antaa tulosteena seuraavanlaisen näkymän:

```
tera1#show rtr configuration 65
Entry number: 65
Owner:
Tag:
Type of operation to perform: jitter
Target address: 172.16.6.2
Source address: 0.0.0.0
Target port: 100
Source port: 0
Request size (ARR data portion): 32
Operation timeout (milliseconds): 5000
```

```
Number of packets: 10
Interval (milliseconds): 20
Type Of Service parameters: 0x0
Verify data: No
Vrf Name:
Control Packets: enabled
Operation frequency (seconds): 60
Next Scheduled Start Time: Start Time already passed
Life (seconds): 3600
Entry Ageout (seconds): never
Status of entry (SNMP RowStatus): Active
Connection loss reaction enabled: No
Timeout reaction enabled: No
Verify error enabled: No
Threshold reaction type: Never
Threshold (milliseconds): 5000
Threshold Falling (milliseconds): 3000
Threshold Count: 5
Threshold Count2: 5
Reaction Type: None
Number of statistic hours kept: 2
Number of statistic distribution buckets kept: 1
Statistic distribution interval (milliseconds): 20
Enhanced History:
```

Konfiguraatiosta näkyy mittaukselle asetetut parametrit. Nämä ovat oletusarvoja. Tästä nähdään, että operaatio lähettää 10 pakettia 20 millisekunnin väleillä kohdeosoitteeseen. Tulokset voidaan myös noutaa reitittimen käyttöliittymästä. Seuraavassa on esimerkkinä haettu UDP Jitter -operaation tulokset TERA1:ltä.

```
tera1#show rtr collection-statistics 65
Entry number: 65
Start Time Index: *18:08:34.520 eest Tue Nov 28 2006
Number of successful operations: 60
Number of operations over threshold: 0
Number of failed operations due to a Disconnect: 0
Number of failed operations due to a Timeout: 0
Number of failed operations due to a Busy: 0
Number of failed operations due to a No Connection: 0
Number of failed operations due to an Internal Error: 0
Number of failed operations due to a Sequence Error: 0
Number of failed operations due to a Verify Error: 0
RTT Values:
NumOfRTT: 600   RTTAvg: 1       RTTMin: 1       RTTMax: 3
RTTSum: 603     RTTSum2: 611
Packet Loss Values:
PacketLossSD: 0 PacketLossDS: 0
PacketOutOfSequence: 0  PacketMIA: 0    PacketLateArrival: 0
InternalError: 0        Busies: 0
Jitter Values:
MinOfPositivesSD: 1     MaxOfPositivesSD: 2
NumOfPositivesSD: 19    SumOfPositivesSD: 21    Sum2PositivesSD: 25
MinOfNegativesSD: 1     MaxOfNegativesSD: 2
NumOfNegativesSD: 20    SumOfNegativesSD: 22    Sum2NegativesSD: 26
```

```
MinOfPositivesDS: 1     MaxOfPositivesDS: 1
NumOfPositivesDS: 42    SumOfPositivesDS: 42    Sum2PositivesDS: 42
MinOfNegativesDS: 1     MaxOfNegativesDS: 1
NumOfNegativesDS: 39    SumOfNegativesDS: 39    Sum2NegativesDS: 39
Interarrival jitterout: 0       Interarrival jitterin: 0
One Way Values:
NumOfOW: 0
OWMinSD: 0      OWMaxSD: 0      OWSumSD: 0      OWSum2SD: 0
OWMinDS: 0      OWMaxDS: 0      OWSumDS: 0      OWSum2DS: 0
```

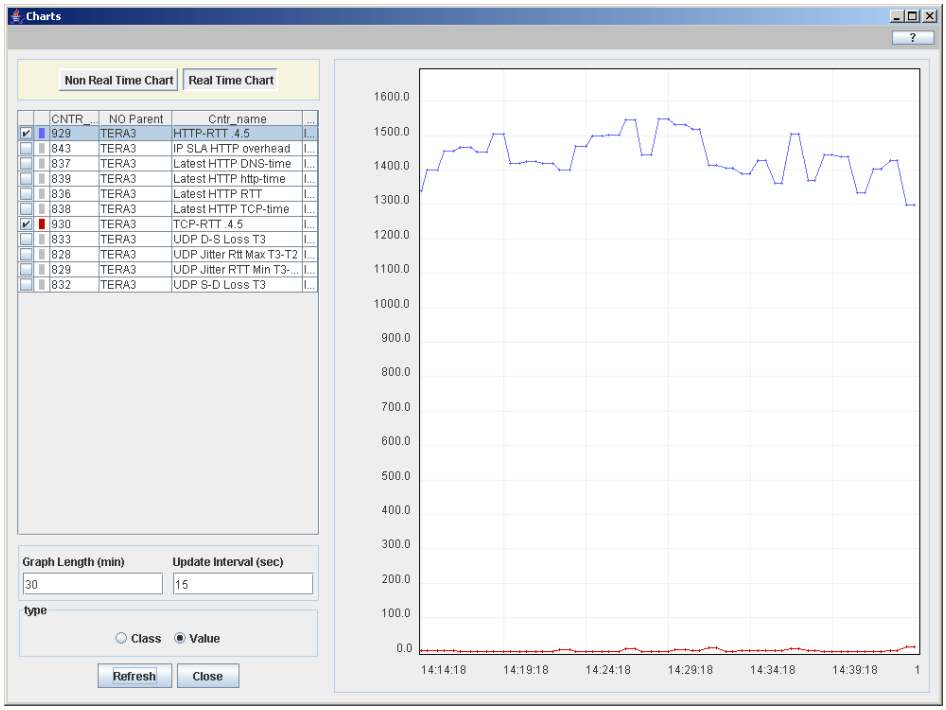
Tuloksista nähdään operaation aloitusaika, onnistuneet ja epäonnistuneet operaatiot ja edestakaisen viiveen arvosta erilaisia tilastollisia arvoja. Lisäksi nähdään muun muassa hävikin arvo, sekä viiveen vaihtelusta useita eri arvoja. Viiveen vaihtelun arvojen merkitys selviää dokumentista [59]. Kaikki IP SLA -konfiguraatiot sekä tulokset on esitetty liitteessä.

5.5.2 Eri liityntöjen näkökulmasta

WLAN-operaattori voi suorittaa IP SLA -mittauksia oman verkkonsa laitteiden välillä. Lisäksi operaattorilla voi olla testiasiakaslaite WLAN-verkossa, jota operaattori voi käyttää mittauksissa. Asiakaslaitetta voidaan mitata esimerkiksi ICMP Echo -operaatiolla, kuten tässä tapauksessa tehtiin. Oikeita asiakkaita ei voida käyttää mittauksissa.

ADSL-operaattori voi, samoin kuin WLAN-operaattorikin, suorittaa mittauksia oman verkkonsa laitteiden välillä. Operaattorilla voi olla myös testiasiakas, jota voidaan käyttää esimerkiksi ICMP Echo tai UDP Echo -mittauksissa kohdelaitteena. UDP Echolla voitaisiin seurata viiveiden käyttäytymistä, kun DSLAM:iin liitetään uusia asiakkaita. Samalla tavalla voidaan myös käyttää Ethernet-liittymää.

Sisällöntuottajille IP SLA on tehokas mittaustekniikka. Sisällöntuottajat, kuten IPTV-lähetysten tarjoaja, voi monitoroida oman verkkonsa laitteita sekä suorittaa mittauksia palvelimille. Palvelimien tilaa voidaan seurata eri IP SLA -operaatioilla. Esimerkkinä käytettiin HTTP-palvelinta, joka on testiverkossa oleva IPTV-palvelin. IPTV-palvelimelle lähetettiin HTTP-kyselyjä TERA3:lta ja seurattiin kyselyjen vasteaikoja. Samalla tavalla voitaisiin seurata myös esimerkiksi FTP tai DNS -palvelimien tilaa. Kuvassa 5.11 on esitetty HTTP-operaation vasteajat TCP-yhteyden luonnille ja koko operaatiolle 30 minuutin aikajaksolla. Palvelimelta haettiin HTML-dokumentti, joka oli yksinkertainen testisivu. HTML-dokumentin noutoaika vaihtelee 1300:n ja 1550 välillä. TCP-yhteyden luonti kestää vain muutamia millisekunteja.



Kuva 5.11: HTTP-operaation vasteajat.

Runkoverkko-operaattoreille IP SLA on erittäin hyvä tekniikka, jos käytössä on Ciscon laitteistoa. IP SLA sopii hyvin myös laajoihin verkkoihin aiemmin mainittujen operaatioryhmiensä vuoksi. Runkoverkosta löydetään myös pullonkaulalinkit IP SLA -operaatioilla, joissa mitataan polkua linkki kerrallaan. Tällainen on esimerkiksi UDP Path Jitter -operaatio. Testiverkkoon konfiguroitiin esimerkkimittaukset taulukon 5.3 mukaisesti. UDP Path Jitter -operaatiota ei voitu testata, koska testiverkon laitteistossa ei löydy siihen riittävää tukea. Vaihtoehtona UDP Path Jitter -operaatiolle, verkkoon konfiguroitiin UDP Jitter ja ICMP Path Echo -operaatiot TERA2:lta TERA3:lle. ICMP Path Echo -operaation tuloksista nähdään, että Ciscon reitittimet tulkitsevat polun olevan kahden linkin mittainen, vaikka se on oikeasti kolmen linkin mittainen. Tämä johtuu matkalla olevasta Juniper M5 -reitittimestä. Ciscon laitteet jättävät sen pois laskuista, koska se ei tue IP SLA -operaatioita. Operaattori voi valvoa verkkonsa tilaa esimerkiksi UDP Jitter -operaatioilla verkon keskiosassa jatkuvasti, koska se antaa monipuolista tietoa runkoverkon kannalta tärkeimmistä suorituskykyparametreista kuten viiveestä, viiveen vaihtelusta ja hävikistä. Reunalaitteilta voisi olla esimerkiksi UDP Path Jitter -operaatioita verkon toiselle reunalle, jolloin saadaan tietoa myös koko verkon läpi kulkevista poluista.

5.6 Yhteenveto soveltuvuuksista

Taulukossa 5.4 on esitetty kuinka eri suorituskyvyn mittaustekniikat (SNMP, RMON, NetFlow, OWAMP, TWAMP ja IP SLA) soveltuvat eri parametrien mittaamiseen. Parametrit on pääsääntöisesti esitelty luvussa 2. Resursseilla tarkoitetaan laitteen resurssien, kuten muistinkäytön tai prosessorikuorman, valvonnan mahdollisuutta. Taulukon luvut merkitsevät seuraavaa:

- **0** = mittaaminen ei ole mahdollista
- **1** = mittaaminen onnistuu puutteellisesti tai ominaisuus on toteuttamatta
- **2** = mittaaminen on mahdollista

Taulukko 5.4: Tekniikoiden soveltuvuus eri parametreille.

	SNMP	**RMON**	**NetF**	**OW**	**TW**	**IPS**
Kaksisuuntainen saatavuus	1	2	2	1	2	2
Pakettien uudelleenjärjestely	0	0	0	2	1	1
Yhdensuuntainen viive	0	0	0	2	0	2
Yhdensuuntainen viiveen vaihtelu	0	0	0	2	0	2
Yhdensuuntainen hävikki	0	0	0	2	0	2
Edestakainen viive	0	0	0	0	2	2
Edestakainen viiveen vaihtelu	0	0	0	0	2	2
Edestakainen hävikki	0	0	0	0	2	2
Kaistan läpipäästö	2	2	2	0	0	0
Verkkolaitteen resurssit	2	2	0	0	0	0

Taulukosta 5.4 nähdään kuinka käsitellyt tekniikat ovat erilaisia. Yleisesti aktiivisilla mittauksilla (OWAMP, TWAMP ja IP SLA) nähdään päästä-päähän mittaustietoa verkon tilasta ja passiivisilla (SNMP, RMON) paikallista tai päästä-päähän (RMON, NetFlow) tietoa verkon liikenteestä. Lisäksi SNMP:llä ja RMON:illa voidaan valvoa verkkolaitteiden resursseja. Saatavuutta voidaan valvoa jollain tavalla jokaisella esitellyllä tekniikalla.

Taulukossa 5.5 on esitetty mittaustekniikoiden soveltuvuus eri palvelujen suorituskyvyn mittaamiseen. IPTV ja HTTP eroavat VoIP:sta siten, että niillä palvelua tuottaa aina jokin palvelin. Tätä palvelinta voidaan sitten valvoa passiivisin menetelmin tai simuloida liikennettä synteettisillä paketeilla. VoIP-liikenteen molemmat osapuolet ovat yleensä asiakkaita, joten niiden saatavuutta tai resursseja ei voida

mitata. Sen sijaan voidaan arvioida VoIP-liikenteen suorituskykyä synteettistä VoIP-liikennettä tai oikeaa VoIP-liikennettä valvomalla.

Taulukko 5.5: Tekniikoiden soveltuvuus eri palveluille.

	SNMP	**RMON**	**NetF**	**OW**	**TW**	**IPS**
IPTV - saatavuus	1	2	1	2	1	1
IPTV - yhdensuuntaiset mittaukset	0	0	0	1	0	1
IPTV - edestakaiset mittaukset	0	0	0	0	1	1
IPTV - kaistan läpipäästö	1	2	2	1	0	1
IPTV - resurssit	2	2	0	0	0	0
VoIP - yhdensuuntaiset mittaukset	0	0	0	1	0	2
VoIP - edestakaiset mittaukset	0	0	0	0	1	2
VoIP - kaistan läpipäästö	1	2	2	0	0	0
HTTP - saatavuus	1	2	1	1	1	2
HTTP - yhdensuuntaiset mittaukset	0	0	0	1	0	1
HTTP - edestakaiset mittaukset	0	0	0	0	1	2
HTTP - kaistan läpipäästö	1	2	2	0	0	0
HTTP - resurssit	2	2	0	0	0	0

SNMP:llä, RMON:illa ja NetFlowlla ei voi mitata ollenkaan päästä-päähän mittauksia, kuten viiveitä tai hävikkejä. Ne kertovat ainoastaan kyseisen verkon pisteen tilasta ja ohi menneestä liikenteestä. RMON2 ja NetFlow kertovat lisäksi mistä osoitteesta paketit ovat lähteneet ja minne ne ovat menossa. Toisaalta NetFlow ei sovellu lainkaan laitteiden valvontaan.

OWAMP ja TWAMP voisivat yhdessä muodostaa hyvän päästä-päähän yleisvalvontatyökalun. OWAMP:in nykyinen toteutus ja tuki on kuitenkin puutteellinen, eikä TWAMPille ole vielä lainkaan toteutusta. IP SLA tarjoaa suunnilleen samat ominaisuudet kuin edelliset, mutta on tuettu vain Ciscon laitteissa. Sen lisäksi siitä löytyy suoraan tuki monen eri palvelun (esimerkiksi HTTP ja VoIP) testaamiseen. Nämä mittaustekniikat kertovat kuitenkin vain verkon tilasta, eivät sen liikenteestä tai laitteista.

Yksikään edellisistä tekniikoista ei yksinään kerro kattavasti IP-verkon suorituskyvystä, liikenteestä, palveluista ja laitteiden tilasta. Tarvitaan siis useamman tekniikan käyttöä. Mittauksia tehdessä tulisi ehdottomasti käyttää standardeja suorituskykyparametreja, jolloin mittaukset ovat vertailukelpoisia.

5.7 Jatkokehitysideoita

Tehdyt testit eivät suinkaan olleet kattavia, vaan lukuisia asioita jäi tekemättä. Tässä luvussa on kerrottu, mitä asioita voisi vielä testata ja missä ympäristössä. Lisäksi kerrotaan mitä parannettavaa VizToolissa ja esitellyissä tekniikoissa vielä olisi.

5.7.1 Tekniikoiden testausideoita

Tutkielmassa kyseltiin lähinnä MIB-II -hallintatietokantastandardin laskureita. Tämän lisäksi voitaisiin testata ainakin `interfaces`-taulun laajennusta [55]. Lisäksi voitaisiin tutkia mitä muita hallintatietokantoja on olemassa ja mitä tietoa näistä saataisiin. SNMP:n hälytyksiä (engl. *trap*) tai RMON:ia ei käytetty lainkaan, joten niitäkin tulisi testata.

NetFlowsta jäi lukuisia ominaisuuksia testaamatta, kuten näytteistys, reitittimellä tiivistäminen tai TCP-lippujen analysointi. Lisäksi Flexible NetFlow tarjoaa lukuisia uusia ominaisuuksia, jotka voisivat tuoda lisäarvoa NetFlow-mittauksiin. Myös muut vuomittaustekniikat, tuleva IPFIX-standardi mukaan lukien, olisivat mielenkiintoinen tutkimusaihe.

IP SLA:sta on olemassa uudempia versioita, joita ei laboratoriossa ollut käytettävissä. Nämä tarjoavat valmiit mittaukset muun muassa VoIP-liikenteelle. Olisi mielenkiintoista ja hyödyllistä arvioida, kuinka hyvin nämä valmiit mittauskeinot kertovat käyttäjien kokemasta palvelun laadusta.

Verkon haistelijoita ei varsinaisesti ollut testeissä käytössä. Sovelluspohjaisia haistelijoita käytettiin kuitenkin hyväksi ongelmatilanteita ratkottaessa. Rautapohjaisia haistelijoita ei ollut käytettävissä. Näiden soveltuvuutta suorituskykymittauksiin voisi siten myös tutkia tarkemmin.

5.7.2 Testattavia ympäristöjä

Erinäisten käytännön ongelmien takia suorituskykymittauksia ei tehty lainkaan 3G-verkossa. Tällä saralla olisikin vielä runsaasti tutkittavaa. Matkapuhelinoperaattorit siirtyvät runkoverkoissaan tulevaisuudessa yhä enemmän IP-verkkoihin, jolloin niiden suorituskykyä on kyettävä mittaamaan ja valvomaan.

Kaikki testit tehtiin laboratorion testiverkossa, jossa liikenne ja verkkolaitteiden määrä oli vähäistä. Seuraavaksi näitä tekniikoita tulisikin testata mahdollisimman oikeassa ympäristössä, jotta saataisiin oikea kuva niiden mahdollisuuksista ja toimivuudesta.

5.7.3 VizToolin kehitysideoita

Ideaalitapauksessa kaikkia tekniikoita voitaisiin valvoa mahdollisimman vähillä sovelluksilla. VizTool tukee tällä hetkellä SNMP-kyselyitä ja NetFlowta. SNMP-kyselyillä voidaan valvoa SNMP:n omien hallintatietokantojen lisäksi myös RMON:in ja IP SLA:n keräämiä tuloksia. Suunnitteilla oleva SNMP-hälytysten tuki mahdollistaisi myös RMON:in ja IP SLA:n täysimittaisen käytön. Tällöin VizToolilla voitaisiin valvoa lähes kaikkia esiteltyjä tekniikoita. *Set*-metodin tuki mahdollistaisi lisäksi mittaustekniikoiden osittaisen hallinnan yhdellä käyttöliittymällä.

5.7.4 Mittaustekniikoiden kehitysideoita

OWAMPissa ja TWAMPissa on määritelty paljon hyviä mittauksia, mutta niille kaikille ei löydy vielä toteutusta. Protokollat on lisäksi turhaan eroteltu, joka hidastaa niiden tuen leviämistä. Joka tapauksessa OWAMP ja TWAMP tarvitsevat kunnollisen toteutuksen verkkolaitteille, jotta siitä tulisi varteen otettava mittausmenetelmä.

Jos vain mahdollista, kaikkien mittaustekniikoiden tulisi käyttää tuloksia varten standardeja hallintatietokantoja. Myös tekniikoiden konfigurointi tulisi olla mahdollista SNMP:n *set*-metodin avulla. Tällöin mittaustekniikoiden käyttö ja tulosten noutaminen olisi mahdollisimman helppoa ja joustavaa. Lisäksi mittausten tulisi käyttää standardeja suorituskykyparametreja, jolloin eri mittaustekniikoiden tarjoamat mittaukset olisivat keskenään vertailukelpoisia.

6 Yhteenveto

IP-verkkojen käyttö on lisääntynyt, ja lisääntyy edelleen. Yhä useammat sovellukset käyttävät IP-verkkoa hyväkseen. Tämä aiheuttaa sen, että IP-verkkojen kuormitus kasvaa ja verkkojen suorituskyky heikkenee. Suorituskykyä täytyy pystyä monitoroimaan, jotta verkon tilaa voidaan seurata ja ruuhkautumisesta aiheutuvia ongelmatilanteita havaita ja ennaltaehkäistä.

IP-verkkojen monitorointiin tarkoitetut tekniikat voidaan jaotella monella eri tavalla. Mittauksissa käytettävän liikenteen mukaan ne jaotellaan aktiivisiin ja passiivisiin tekniikoihin. Aktiiviset tekniikat käyttävät mittauksissaan itse generoitua liikennettä. Passiiviset tekniikat suorittavat mittaukset verkon olemassa olevalle liikenteelle. Eri tekniikat soveltuvat erilaisiin tilanteisiin. Jotkut tekniikat soveltuvat runkoverkko-operaattorin tarpeisiin paremmin kuin esimerkiksi sisällöntuottajan. Erilaiset tarpeet johtavat siihen, että yksi tekniikka ei riitä kaikkien tarpeisiin. Tarvitaan useita standardoituja tekniikoita, joita tukevat kaikki laitevalmistajat ja operaattorit. Useat eri tekniikat luovat tarpeen työkalulle, jolla näitä kaikkia voidaan hallita ja visualisoida. Tällainen on esimerkiksi tutkielmassa mainittu VizTool.

Tutkielman teoriaosuudessa esiteltiin erilaisia suorituskyvyn mittaamistekniikoita. Tekniikoista esiteltiin niiden toimintaperiaate ja suorituskykyparametrit, joita niillä voidaan mitata. Käytännön osuudessa testattiin tekniikoiden soveltuvuutta verkon eri osiin laboratorioverkossa. Tutkielmassa pohdittiin, mitkä tekniikat ovat käyttökelpoisia ja mihin verkon osaan ne sopivat parhaiten. Tutkielmassa pyrittiin löytämään kunkin tekniikan hyvät ja huonot puolet. Tutkielmassa havaittiin, että tekniikoissa on monia keskeneräisiä ominaisuuksia ja yksi tekniikka ei riitä koko verkon suorituskyvyn monitorointiin. Tulevaisuudessa suorituskyvyn monitorointiin täytyy kiinnittää paljon huomiota, koska IP-verkon käyttö kasvaa erilaisten reaaliaikasovellusten yleistyessä. Monitorointitekniikoihin täytyy kehittää monipuolisia, kullekin reaaliaikasovellukselle sopivia mittausoperaatioita.

Lähteet

[1] Huston, Geoff, *Internet Performance Survival Guide - QoS Strategies for Multiservice Networks*, John Wiley & Sons, Inc., 2000.

[2] Peuhkuri, Markus, *Internet traffic measurements - aims, methodology, and discoveries*, Tietoliikenteen lisensiaatintutkimus, saatavilla WWW-muodossa <URL: `http://www.netlab.tkk.fi/u/puhuri/publications/li.shtml`>, 2002.

[3] Viipuri, Timo, *Traffic Analysis and Modeling of IP Core Networks*, Tietoliikenteen diplomityö, saatavilla WWW-muodossa <URL: `http://www.netlab.tkk.fi/julkaisut/tyot/diplomityot/1039/`>, 2005.

[4] J. Mahdavi, V. Paxson, *IPPM Metrics for Measuring Connectivity*, IETF RFC2678, 1999.

[5] V. Paxson, G. Almes, J. Mahdavi, M. Mathis, *Framework for IP Performance Metrics*, IETF RFC2330, 1998.

[6] G. Almes, S. Kalidindi, M. Zekauskas, *A One-way Delay metric for IPPM*, IETF RFC2679, 1999.

[7] E. Stephan, L. Liang, A. Morton, *IP Performance Metrics (IPPM) for spatial and multicast*, Internet Draft, 2005.

[8] G. Almes, S. Kalidindi, M. Zekauskas, *A One-way Packet Loss Metric for IPPM*, IETF RFC2680, 1999.

[9] G. Almes, S. Kalidindi, M. Zekauskas, *A Round-trip Delay Metric for IPPM*, IETF RFC2681, 1999.

[10] C. Demichelis, P. Chimento, *IP Packet Delay Variation Metric for IP Performance Metrics (IPPM)*, IETF RFC3393, 2002.

[11] R. Koodli, R. Ravikanth, *One-way Loss Pattern Sample Metrics*, IETF RFC3357, 2002.

[12] P. Chimento, J. Ishac, *Defining Network Capacity*, Internet Draft, 2006.

[13] R. Koodli, R. Ravikanth, *A Framework for Defining Empirical Bulk Transfer Capacity Metrics*, IETF RFC3148, 2001.

[14] A. Morton, L. Ciavattone, G. Ramanchandran, S Shalunov, J. Perser, *Packet Reordering Metric for IPPM*, Internet Draft, 2006.

[15] Stalling William, *SNMP, SNMPv2, SNMPv3 and RMON1 and RMON2*, Third Edition, Addison Wesley Longman, Inc., 2000.

[16] Case J., Mundy R., Partain D., Stewart B., *RFC 3410 - Applicability Statements for SNMP*, IETF RFC 3410, 2002.

[17] Huston, Geoff, *Measuring IP Network Performance*,
The Internet Protocol Journal - Volume 6, Number 1, saatavilla WWW-muodossa
<URL: `http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_6-1/measuring_ip.html`, 2003.

[18] S. Waldbusser, *Remote Network Monitoring Management Information Base*, IETF RFC 2819, 2000.

[19] S. Waldbusser, *Remote Network Monitoring Management Information Base Version 2*, IETF RFC 4502, 2006.

[20] E. Stephan, *IP Performance Metrics (IPPM) Metrics Registry*, IETF RFC 4148, 2005.

[21] S. Waldbusser, *Application Performance Measurement MIB*, IETF RFC 3729, 2004.

[22] R. Dietz, R. Cole, *Transport Performance Metrics MIB*, IETF RFC 4150, 2005.

[23] C. Kalbfleisch, R. Cole, D. Romascanu, *Definition of Managed Objects for Synthetic Sources for Performance Monitoring Algorithms*, IETF RFC 4149, 2005.

[24] A. Siddiqui, D. Romascanu, E. Golovinsky*Real-time Application Quality-of-Service Monitoring (RAQMON) Framework*, IETF RFC 4710, 2006.

[25] Cisco Systems, *Cisco IOS IP SLAs Overview*, saatavilla WWW-muodossa
<URL: `http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cg/hsla_c/hsoverv.pdf`>, viitattu 2.11.2006.

[26] *Introduction to Cisco IOS NetFlow - A Technical Overview*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/en/US/products/ps6601/products_white_paper0900aecd80406232.shtml>, viitattu 28.9.2006.

[27] *NetFlow Services Solutions Guide*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/en/US/products/sw/netmgtsw/ps1964/products_implementation_design_guide09186a00800d6a11.html>, 2004.

[28] *NetFlow Export Datagram Format*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/univercd/cc/td/doc/product/rtrmgmt/nfc/nfc_3_0/nfc_ug/nfcform.htm>, 2003.

[29] B. Claise, Ed., *Cisco Systems NetFlow Services Export Version 9*, IETF RFC 3954, 2004.

[30] *NetFlow Version 9 Flow-Record Format*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/en/US/products/ps6601/products_white_paper09186a00800a3db9.shtml>, viitattu 2.10.2006.

[31] Sommer Robin, Feldmann Anja, *NetFlow: Information loss or win?*, saatavilla WWW-muodossa
<URL: *citeseer.ist.psu.edu/article/sommer02netflow.html*>, 2002.

[32] *Cisco IOS Flexible NetFlow Technology White Paper*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/en/US/products/ps6601/products_white_paper0900aecd804be1cc.shtml>, 2006.

[33] T. Zseby, B. Claise, S. Zander, *Requirements for IP Flow Information Export (IPFIX)*, IETF RFC 3917, 2004.

[34] S. Shalunov, B. Teitelbaum, A. Karp, J. Boote, M. Zekauskas, *A One-way Active Measurement Protocol (OWAMP)*, IETF RFC4656, 2006.

[35] J. Postel, *Transmission Control Protocol*, IETF RFC793, 1981.

[36] J. Postel, *User Datagram Protocol*, IETF RFC768, 1980.

[37] Andres Koropecki, *Planet Math: Poisson random variable*, saatavilla WWW-muodossa
<URL: http://planetmath.org/encyclopedia/PoissonDistribution.html/>, viitattu 24.10.2006.

[38] V. Räisänen, G. Grptefeld, A. Morton, *Network performance measurement with periodic streams*, IETF RFC3432, 2002.

[39] Z. Shu, K. Kobayashi, *HOTS: An OWAMP-Compilant Hardware Packet Timestamp Recorder*, saatavilla WWW-muodossa
<URL: http://ieeexplore.ieee.org/iel5/10434/33128/01559884.pdf>, viitattu 25.10.2006.

[40] J. Babiarz, K. Hedayat, R. Krzanowski, K. Yum, *A Two-way Measurement Protocol (TWAMP)*, Internet Draft, 2006.

[41] Cisco Systems, *IP SLAs-Multiple Operation Scheduling*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cg/hsla_c/hsmulti.pdf>, viitattu 8.11.2006.

[42] Cisco Systems, *IP SLAs-Proactive Threshold Monitoring*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cg/hsla_c/hsthresh.pdf>, viitattu 8.11.2006.

[43] Cisco Systems, *IP SLAs-Analyzing IP Service Levels Using the UDP Jitter Operation*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cg/hsla_c/hsjitter.pdf>, viitattu 3.11.2006.

[44] Cisco Systems, *IP SLAs-Analyzing IP Service Levels Using the ICMP Path Jitter Operation*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cg/hsla_c/hspthjit.pdf>, viitattu 6.11.2006.

[45] Cisco Systems, *IP SLAs-Analyzing IP Service Levels Using the VoIP UDP Jitter Operation*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cg/hsla_c/hsvoipj.pdf>, viitattu 6.11.2006.

[46] Cisco Systems, *IP SLAs-Analyzing IP Service Levels Using the UDP Echo Operation*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cg/hsla_c/hsudpe.pdf>, viitattu 6.11.2006.

[47] Cisco Systems, *IP SLAs-Analyzing IP Service Levels Using the ICMP Echo Operation*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cg/hsla_c/hsicmp.pdf>, viitattu 6.11.2006.

[48] Cisco Systems, *IP SLAs-Analyzing IP Service Levels Using the ICMP Path Echo Operation*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cg/hsla_c/hspaecho.pdf>, viitattu 7.11.2006.

[49] Cisco Systems, *IP SLAs-Analyzing IP Service Levels Using the HTTP Operation*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cg/hsla_c/hshttp.pdf>, viitattu 7.11.2006.

[50] Cisco Systems, *IP SLAs-Analyzing IP Service Levels Using the TCP Connect Operation*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cg/hsla_c/hstcpc.pdf>, viitattu 7.11.2006.

[51] Cisco Systems, *IP SLAs-Analyzing IP Service Levels Using the FTP Operation*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cg/hsla_c/hsftp.pdf>, viitattu 7.11.2006.

[52] Cisco Systems, *IP SLAs-Analyzing IP Service Levels Using the DHCP Operation*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cg/hsla_c/hsdhcp.pdf>, viitattu 7.11.2006.

[53] Cisco Systems, *IP SLAs-Analyzing IP Service Levels Using the DNS Operation*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cg/hsla_c/hsdns.pdf>, viitattu 8.11.2006.

[54] Cisco Systems, *IP SLAs-Analyzing IP Service Levels Using the DLSw+ Operation*, saatavilla WWW-muodossa
<URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios124/124cg/hsla_c/hsdlsw.pdf>, viitattu 8.11.2006.

[55] K. McCloghrie, F. Kastenholz, *Evolution of the Interfaces Group of MIB-II*, IETF RFC 1573, 1994.

[56] K. McCloghrie, M. Rose, *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*, IETF RFC 1213, 1991.

[57] *NETFLOW PERFORMANCE ANALYSIS*, saatavilla WWW-muodossa <URL: http://www.cisco.com/en/US/tech/tk812/technologies_white_paper0900aecd802a0eb9.shtml>, viitattu 28.9.2006.

[58] Internet2, *One-way Ping*, saatavilla WWW-muodossa <URL: http://e2epi.internet2.edu/owamp/details.html>, viitattu 14.11.2006.

[59] Cisco Systems, *Configuring Cisco IP SLAs UDP Jitter operation*, saatavilla WWW-muodossa <URL: http://www.cisco.com/application/pdf/en/us/guest/products/ps6602/c1244/cdccont_0900aecd804fb392.pdf>, viitattu 29.11.2006.

A Flowd-skripti

```
#!/usr/bin/perl

# Copyright (c) 2004 Damien Miller <djm@mindrot.org>
#
# Permission to use, copy, modify, and distribute this software for any
# purpose with or without fee is hereby granted, provided that the above
# copyright notice and this permission notice appear in all copies.
#
# THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
# WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
# MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
# ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
# WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
# ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
# OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

use strict;
use warnings;
use DBI;
use Flowd;
use Time::Local;
use Net::Netmask;
use Number::Range;

my $min_looptime = 10;
my $start_error = "\nUsage: perl flowd-mysql-insert.pl flowd-log looptime_seconds".
                  " database:ip-address:port db-username db-password [--debug]\n".
                  "Example: perl flowd-mysql-insert.pl flowd 150 ".
                  "visdemo2:127.0.0.1:3306 root pass\n\n";
die $start_error unless (@ARGV && $#ARGV >= 4);
die "looptime must be at least ". $min_looptime unless ($ARGV[1] >= $min_looptime);

# Database settings
my $DBI_DRIVER = "mysql"; # or one of "Pg" "mysql" "mysqlPP"
my $DB = $ARGV[2]; #"visdemo2:172.16.255.105:3306";
my $TABLE = "netflow_counter";
my $USER = $ARGV[3]; #"root";
my $PASS = $ARGV[4]; #"viztool";

my $flow_log = $ARGV[0];
my $loop_time = $ARGV[1];
my $flow_templog = $flow_log.".temp";

my $debug_mode = 0;
if ($#ARGV >= 5){
  if ($ARGV[5] eq "--debug"){
      $debug_mode = 1;
  }
}
```

```
sub createMask {
  if (!defined($_[0])){
    return undef;
  }
  return new2 Net::Netmask($_[0]);
}

#creates a new range
#returns undefined if input is not ok
sub createRange {
  if (!defined($_[0])){
    return undef;
  }
  if ($_[0] eq ""){
    return undef;
  }
  my $range = Number::Range->new();
  eval{
    $range->addrange($_[0]);
  };
  if ($@){
    undef($range);
  }
  return $range;
}

#Checks if integer[1] is defined and is out of range[0],  $_[0] Number:Range, $_[1] Integer
sub numberOutRange{
  if (defined($_[0]) && !$_[0]->inrange($_[1]) ){
    return 1;
  }
  return 0;
}
sub getFormattedTime{
  my ($second, $minute, $hour) = localtime();
  return "$hour:$minute:$second";
}

my $time_difference;
my $loop_count=0;
my @log_times;
$log_times[0] = time();

#Script structure:
#infinte_loop {
#    while(rulerows_from_visdemo_db){
#        while(flowrows_from_flowd_logs){
#        }
#    }
#}
#Every rule from Visdemo db will be read once per program loop and
#every flow record as many times as there are rules in the visdemo db

while (1){
  #print getFormattedTime().": Loop started.\n";
```

```
#rename log to templog (this won't stop logging) and then signal flowd to reopen
#its logfile then process the flows in the templog and delete it afterwards
system("mv ".$flow_log." ".$flow_templog);
system("killall -s SIGHUP flowd");
$log_times[1] = time();
$time_difference = $log_times[1] - $log_times[0];

if ($debug_mode == 1){
    print "Last loop took: ".$time_difference." seconds.\n";
}
my $db = DBI->connect("dbi:$DBI_DRIVER:dbname=$DB", $USER, $PASS)
  or die "DBI->connect error: " . $DBI::errstr;

my $flow_handle = Flowd->new($flow_templog);
my $select_query = ("select CNTR_ID, agent_addr, PROTOCOL, TOS,".
  "TCP_FLAG_0, TCP_FLAG_1, TCP_FLAG_2, TCP_FLAG_3, TCP_FLAG_4,TCP_FLAG_5,".
  "TCP_FLAG_6,TCP_FLAG_7,IPV4_SRC_PORT, IPV4_SRC_ADDR, INPUT_SNMP, ".
  "IPV4_DST_PORT, IPV4_DST_ADDR,".
  "OUTPUT_SNMP from ".$TABLE.";");

my $f_duration;
my $agent_addr_block;
my $protocol_range;
my $tos_range;
my $ipv4_src_port_range;
my $ipv4_src_addr_block;
my $input_snmp_range;
my $ipv4_dst_port_range;
my $ipv4_dst_addr_block;
my $output_snmp_range;
my $total_bytes=0;
my $Bps;
my $call_query;
my $sth = $db->prepare($select_query) or
  die "db->prepare failed: " . $DBI::errstr;
$sth->execute()  or die "sth->execute failed: " . $DBI::errstr;

while (my $row_href = $sth->fetchrow_hashref){
  $agent_addr_block = new2 Net::Netmask($row_href->{'agent_addr'});
  $protocol_range = &createRange($row_href->{'PROTOCOL'});
  $tos_range = &createRange($row_href->{'TOS'});
  #$tcp_flags_range = &createRange($row_href->{'TCP_FLAGS'});
  $ipv4_src_port_range = &createRange($row_href->{'IPV4_SRC_PORT'});
  $ipv4_src_addr_block = &createMask($row_href->{'IPV4_SRC_ADDR'});
  $input_snmp_range =  &createRange($row_href->{'INPUT_SNMP'});
  $ipv4_dst_port_range = &createRange($row_href->{'IPV4_DST_PORT'});
  $ipv4_dst_addr_block = &createMask($row_href->{'IPV4_DST_ADDR'});
  $output_snmp_range =  &createRange($row_href->{'OUTPUT_SNMP'});

  my $match_count=0;
  my $flow_count=0;
  while (my $flow = $flow_handle->read_flow()) {

    #match agent_addr
    if ( defined($agent_addr_block) &&
       !$agent_addr_block->match($flow->{agent_addr}) ){ next; }
```

```
  #match PROTOCOL
  if ( numberOutRange($protocol_range, $flow->{protocol}) ){ next; };
  #match TOS
  if ( numberOutRange($tos_range, $flow->{tos} ) ){ next; };
  #match TCP_FLAGS
  #if ( numberOutRange($tcp_flags_range, $flow->{tcp_flags}) ){ next; };
  #match IPV4_SRC_PORT
  if ( numberOutRange($ipv4_src_port_range, $flow->{src_port}) ){ next; };
  #match IPV4_SRC_ADDR
  if ( defined($ipv4_src_addr_block) &&
     !$ipv4_src_addr_block->match($flow->{src_addr}) ){ next; }
  #match INPUT_SNMP
  if ( numberOutRange($input_snmp_range, $flow->{if_index_in}) ){ next; }
  #match IPV4_DST_PORT
  if ( numberOutRange($ipv4_dst_port_range, $flow->{dst_port}) ){ next; };
  #match IPV4_DST_ADDR
  if ( defined($ipv4_dst_addr_block) &&
    !$ipv4_dst_addr_block->match($flow->{dst_addr}) ){ next; }
  #match OUTPUT_SNMP
  if ( numberOutRange($output_snmp_range, $flow->{if_index_out}) ){ next; }

  $total_bytes = $total_bytes + $flow->{flow_octets};
  $match_count++;
}
#Count Bps
if ($time_difference > 0){
    $Bps = $total_bytes/$time_difference;
}
else{
    $Bps = 0;
}

#Print debugging information to console
if ($debug_mode == 1){
    print getFormattedTime(). ": Id: ". $row_href->{'CNTR_ID'} .
          " - Flow count: ". $match_count." Total bytes: ";
    print $total_bytes ." Average: ".$Bps." B/s \n";
    print "Rule: ";
    if (defined($agent_addr_block)) {
      print "Agent: ". $agent_addr_block }
    if (defined($input_snmp_range)) {
      print " INPUT_SNMP: ".$input_snmp_range->range; }
    if (defined($ipv4_src_port_range)) {
      print " SRC_PORT: ".$ipv4_src_port_range->range; }
    if (defined($ipv4_dst_port_range)) {
      print " DST_PORT: " .$ipv4_dst_port_range->range; }
    if (defined($ipv4_src_addr_block)) {
      print " SRC: ". $ipv4_src_addr_block }
    print "\n\n";
}

#Won't write data to db on first round
if ( $loop_count > 0 ) {
  $call_query = "CALL Classifier(".$row_href->{'CNTR_ID'}.", ".$Bps.")";
  my $sth2 = $db->prepare($call_query) or die "db->prepare failed: " . $DBI::errstr;
  $sth2->execute()  or die "sth2->execute failed: " . $DBI::errstr;
```

```
    }
      $flow_handle = Flowd->new($flow_templog);
  }

  $flow_handle->finish();
  shift @log_times;
  system("rm " . $flow_templog);
  $loop_count++;

  my $proc_time = time() - $log_times[0];
  if ($debug_mode == 1){
     print "Flow processing took ". $proc_time . " seconds.\n\n";
  }

  if ( ($loop_time-$proc_time) > 0 ){
    sleep($loop_time-$proc_time);
  }
}
```

B IP SLA konfiguraatiot ja tulokset

TERA3:n konfiraatiot ja mittaustulokset. Konfiguraatiot nähdään käskyllä `TERA3#show rtr configuration 50` ja tulokset käskyllä `TERA3#show rtr collection-statistics 50`.

```
Entry Number: 50
Owner:
Tag:
Type of Operation to Perform: http
Reaction and History Threshold (milliseconds): 5000
Operation Frequency (seconds): 60
Operation Timeout (milliseconds): 5000
Verify Data: FALSE
Status of Entry (SNMP RowStatus): active
Protocol Type: httpAppl
Target Address:
Source Address: 0.0.0.0
Target Port: 0
Source Port: 0
Request Size (ARR data portion): 1
Response Size (ARR data portion): 1
Control Packets: enabled
Loose Source Routing: disabled
LSR Path:
Type of Service Parameters: 0x0
HTTP Operation: get
HTTP Server Version: 1.0
URL: http://172.16.4.5/index.html
Raw String(s):




Cache Control: enabled
Life (seconds): 3600
Next Scheduled Start Time: Start Time already passed
Entry Ageout: never
Connection Loss Reaction Enabled: FALSE
Timeout Reaction Enabled: FALSE
Threshold Reaction Type: never
Threshold Falling (milliseconds): 3000
Threshold Count: 5
Threshold Count2: 5
Reaction Type: none
Verify Error Reaction Enabled: FALSE
Number of Statistic Hours kept: 2
Number of Statistic Paths kept: 1
Number of Statistic Hops kept: 1
```

```
Number of Statistic Distribution Buckets kept: 1
Statistic Distribution Interval (milliseconds): 20
Number of History Lives kept: 0
Number of History Buckets kept: 15
Number of History Samples kept: 1
History Filter Type: none


Tulokset:

Entry Number: 50
HTTP URL: http://172.16.4.5/index.html
Start Time: .15:03:26.000 eest Wed Nov 29 2006

             Comps: 35             RTTMin: 1143
             OvrTh: 0              RTTMax: 1393
        DNSTimeOut: 0              RTTSum: 42612
        TCPTimeOut: 0             RTTSum2: 51935938
        TraTimeOut: 0              DNSRTT: 0
          DNSError: 0           TCPConRTT: 146
         HTTPError: 2            TransRTT: 42466
          IntError: 0            MesgSize: 5954130
            Busies: 0
```

TERA1:n konfiguraatiot ja mittaustulokset.

```
Entry number: 65
Owner:
Tag:
Type of operation to perform: jitter
Target address: 172.16.6.2
Source address: 0.0.0.0
Target port: 100
Source port: 0
Request size (ARR data portion): 32
Operation timeout (milliseconds): 5000
Number of packets: 10
Interval (milliseconds): 20
Type Of Service parameters: 0x0
Verify data: No
Vrf Name:
Control Packets: enabled
Operation frequency (seconds): 60
Next Scheduled Start Time: Start Time already passed
Life (seconds): 3600
Entry Ageout (seconds): never
Status of entry (SNMP RowStatus): Active
Connection loss reaction enabled: No
Timeout reaction enabled: No
Verify error enabled: No
Threshold reaction type: Never
Threshold (milliseconds): 5000
Threshold Falling (milliseconds): 3000
Threshold Count: 5
```

```
Threshold Count2: 5
Reaction Type: None
Number of statistic hours kept: 2
Number of statistic distribution buckets kept: 1
Statistic distribution interval (milliseconds): 20
Enhanced History:

Tulokset:

Entry number: 65
Start Time Index: *18:08:34.520 eest Tue Nov 28 2006
Number of successful operations: 60
Number of operations over threshold: 0
Number of failed operations due to a Disconnect: 0
Number of failed operations due to a Timeout: 0
Number of failed operations due to a Busy: 0
Number of failed operations due to a No Connection: 0
Number of failed operations due to an Internal Error: 0
Number of failed operations due to a Sequence Error: 0
Number of failed operations due to a Verify Error: 0
RTT Values:
NumOfRTT: 600   RTTAvg: 1       RTTMin: 1       RTTMax: 3
RTTSum: 603     RTTSum2: 611
Packet Loss Values:
PacketLossSD: 0 PacketLossDS: 0
PacketOutOfSequence: 0  PacketMIA: 0    PacketLateArrival: 0
InternalError: 0        Busies: 0
Jitter Values:
MinOfPositivesSD: 1     MaxOfPositivesSD: 2
NumOfPositivesSD: 19    SumOfPositivesSD: 21    Sum2PositivesSD: 25
MinOfNegativesSD: 1     MaxOfNegativesSD: 2
NumOfNegativesSD: 20    SumOfNegativesSD: 22    Sum2NegativesSD: 26
MinOfPositivesDS: 1     MaxOfPositivesDS: 1
NumOfPositivesDS: 42    SumOfPositivesDS: 42    Sum2PositivesDS: 42
MinOfNegativesDS: 1     MaxOfNegativesDS: 1
NumOfNegativesDS: 39    SumOfNegativesDS: 39    Sum2NegativesDS: 39
Interarrival jitterout: 0       Interarrival jitterin: 0
One Way Values:
NumOfOW: 0
OWMinSD: 0      OWMaxSD: 0      OWSumSD: 0      OWSum2SD: 0
OWMinDS: 0      OWMaxDS: 0      OWSumDS: 0      OWSum2DS: 0
```

TERA2:n konfiguroinnit ja mittaustulokset.

```
Entry number: 60
Owner:
Tag:
Type of operation to perform: pathEcho
Target address: 172.16.1.2
Source address: 0.0.0.0
Request size (ARR data portion): 28
Operation timeout (milliseconds): 5000
Type Of Service parameters: 0x0
Verify data: No
Loose Source Routing: Disabled
Vrf Name:
```

```
LSR Path:
Operation frequency (seconds): 60
Next Scheduled Start Time: Start Time already passed
Life (seconds): 3600
Entry Ageout (seconds): never
Status of entry (SNMP RowStatus): Active
Connection loss reaction enabled: No
Timeout reaction enabled: No
Verify error enabled: No
Threshold reaction type: Never
Threshold (milliseconds): 5000
Threshold Falling (milliseconds): 3000
Threshold Count: 5
Threshold Count2: 5
Reaction Type: None
Number of statistic hours kept: 2
Number of statistic paths kept: 5
Number of statistic hops kept: 16
Number of statistic distribution buckets kept: 1
Statistic distribution interval (milliseconds): 20
Number of history Lives kept: 0
Number of history Buckets kept: 15
Number of history Samples kept: 16
History Filter Type: None

Entry number: 70
Owner:
Tag:
Type of operation to perform: echo
Target address: 172.16.9.106
Source address: 0.0.0.0
Request size (ARR data portion): 28
Operation timeout (milliseconds): 5000
Type Of Service parameters: 0x0
Verify data: No
Vrf Name:
Operation frequency (seconds): 60
Next Scheduled Start Time: Start Time already passed
Life (seconds): 3600
Entry Ageout (seconds): never
Status of entry (SNMP RowStatus): Active
Connection loss reaction enabled: No
Timeout reaction enabled: No
Verify error enabled: No
Threshold reaction type: Never
Threshold (milliseconds): 5000
Threshold Falling (milliseconds): 3000
Threshold Count: 5
Threshold Count2: 5
Reaction Type: None
Number of statistic hours kept: 2
Number of statistic distribution buckets kept: 1
Statistic distribution interval (milliseconds): 20
Number of history Lives kept: 0
Number of history Buckets kept: 15
History Filter Type: None
```

```
Enhanced History:

Entry number: 75
Owner:
Tag:
Type of operation to perform: jitter
Target address: 172.16.1.2
Source address: 0.0.0.0
Target port: 100
Source port: 0
Request size (ARR data portion): 32
Operation timeout (milliseconds): 5000
Number of packets: 10
Interval (milliseconds): 20
Type Of Service parameters: 0x0
Verify data: No
Vrf Name:
Control Packets: enabled
Operation frequency (seconds): 60
Next Scheduled Start Time: Start Time already passed
Life (seconds): 3600
Entry Ageout (seconds): never
Status of entry (SNMP RowStatus): Active
Connection loss reaction enabled: No
Timeout reaction enabled: No
Verify error enabled: No
Threshold reaction type: Never
Threshold (milliseconds): 5000
Threshold Falling (milliseconds): 3000
Threshold Count: 5
Threshold Count2: 5
Reaction Type: None
Number of statistic hours kept: 2
Number of statistic distribution buckets kept: 1
Statistic distribution interval (milliseconds): 20
Enhanced History:

Tulokset:

Entry number: 60
Start Time Index: *11:35:12.863 eest Wed Nov 29 2006
Path Index: 1
Hop in Path Index: 1
Number of successful operations: 60
Number of operations over threshold: 0
Number of failed operations due to a Disconnect: 0
Number of failed operations due to a Timeout: 0
Number of failed operations due to a Busy: 0
Number of failed operations due to a No Connection: 0
Number of failed operations due to an Internal Error: 0
Number of failed operations due to a Sequence Error: 0
Number of failed operations due to a Verify Error: 0
Target Address 172.16.6.1

Start Time Index: *11:35:12.863 eest Wed Nov 29 2006
Path Index: 1
```

```
Hop in Path Index: 2
Number of successful operations: 60
Number of operations over threshold: 0
Number of failed operations due to a Disconnect: 0
Number of failed operations due to a Timeout: 0
Number of failed operations due to a Busy: 0
Number of failed operations due to a No Connection: 0
Number of failed operations due to an Internal Error: 0
Number of failed operations due to a Sequence Error: 0
Number of failed operations due to a Verify Error: 0
Target Address 172.16.1.2


Entry number: 70
Start Time Index: *11:18:05.462 eest Wed Nov 29 2006
Number of successful operations: 31
Number of operations over threshold: 0
Number of failed operations due to a Disconnect: 0
Number of failed operations due to a Timeout: 29
Number of failed operations due to a Busy: 0
Number of failed operations due to a No Connection: 0
Number of failed operations due to an Internal Error: 0
Number of failed operations due to a Sequence Error: 0
Number of failed operations due to a Verify Error: 0
RTT Values:
RTTAvg: 5       RTTMin: 0       RTTMax: 48
NumOfRTT: 31    RTTSum: 163     RTTSum2: 4161


Entry number: 75
Start Time Index: *11:40:29.763 eest Wed Nov 29 2006
Number of successful operations: 60
Number of operations over threshold: 0
Number of failed operations due to a Disconnect: 0
Number of failed operations due to a Timeout: 0
Number of failed operations due to a Busy: 0
Number of failed operations due to a No Connection: 0
Number of failed operations due to an Internal Error: 0
Number of failed operations due to a Sequence Error: 0
Number of failed operations due to a Verify Error: 0
RTT Values:
NumOfRTT: 600   RTTAvg: 3       RTTMin: 1       RTTMax: 18
RTTSum: 2313    RTTSum2: 14967
Packet Loss Values:
PacketLossSD: 0 PacketLossDS: 0
PacketOutOfSequence: 0  PacketMIA: 0    PacketLateArrival: 0
InternalError: 0        Busies: 0
Jitter Values:
MinOfPositivesSD: 1     MaxOfPositivesSD: 16
NumOfPositivesSD: 236   SumOfPositivesSD: 939   Sum2PositivesSD: 5285
MinOfNegativesSD: 1     MaxOfNegativesSD: 14
NumOfNegativesSD: 210   SumOfNegativesSD: 821   Sum2NegativesSD: 4569
MinOfPositivesDS: 1     MaxOfPositivesDS: 10
NumOfPositivesDS: 108   SumOfPositivesDS: 209   Sum2PositivesDS: 935
MinOfNegativesDS: 1     MaxOfNegativesDS: 17
NumOfNegativesDS: 110   SumOfNegativesDS: 212   Sum2NegativesDS: 1106
```

```
Interarrival jitterout: 0       Interarrival jitterin: 0
One Way Values:
NumOfOW: 0
OWMinSD: 0      OWMaxSD: 0      OWSumSD: 0      OWSum2SD: 0
OWMinDS: 0      OWMaxDS: 0      OWSumDS: 0      OWSum2DS: 0
```