Ville Niskanen

IP-verkon suorituskyvyn mittaamisjärjestelmä

Insinööri (AMK)

Tieto- ja viestintätekniikka

Kevät 2020

KAMK • University of Applied Sciences

Tiivistelmä

Tekijä: Niskanen Ville

Työn nimi: IP-verkon suorituskyvyn mittaamisjärjestelmä

Tutkintonimike: Insinööri (AMK), tieto- ja viestintätekniikka

Asiasanat: IP-verkko, suorituskyky, mittaus, Linux, Python

Opinnäytetyö tehtiin toimeksiantona Bittium Wireless Oy:lle, jonka tuotteisiin lukeutuvat taktisen tiedonsiirron tuotteet, kuten taktinen runkoverkko. Työn tavoitteena oli kehittää kyseisen verkon suorituskykyä mittaava järjestelmä. Suorituskykyä oli kyettävä mittaamaan sekä kertaluontoisilla mittauksilla että toistuvilla mittauksilla. Mittaamisjärjestelmä sulautuisi osaksi taktisen runkoverkon valmiita ohjelmistokomponentteja.

Toimeksiantajan vaatimusten mukaisten mittaustyyppien perusteella valittiin järjestelmän käyttämät mittaustyökalut. Mittaustyökaluja ohjataan mittausparametrien avulla, jotka ovat luonteeltaan joko välttämättömiä tai vapaaehtoisia. Mittaustyökalujen palauttamat tulokset palautetaan kertaluontoisessa mittauksessa suoraan käyttäjälle, toistuvan mittauksen tulokset tallennetaan puolestaan erilliseen tulostiedostoon, josta ne ovat luettavissa käyttäjärajapinnan kautta. Koko mittaamisjärjestelmän hallinta tapahtuu kyseisen rajapinnan kautta.

Mittaamisjärjestelmä on toteutettu Python-ohjelmointikielellä. Koska järjestelmän oli tuettava useita yhtäaikaisia mittauksia, on järjestelmän toteutuksessa hyödynnetty säikeistystä ja aliprosesseja. Järjestelmässä toimii socket-palvelin, joka kommunikoi käyttäjärajapinnan kanssa. Kertaluontoiset mittaukset käynnistetään socket-palvelimen kautta. Toistuvat mittaukset määritetään mittaamisjärjestelmän konfiguraatiotiedostoon taktisen runkoverkon erillisen konfiguraationhallinnan kautta. Järjestelmä vastaanottaa muun muassa mittausparametrit JSON-tiedostomuodossa, jonka sisältö varmennetaan konfiguraationhallinnan toimesta.

Opinnäytetyön tuloksena syntynyt mittaamisjärjestelmä vastasi suurimmilta osin toimeksiantajan vaatimuksia. Järjestelmä mahdollistaa suorituskykymittaukset kertaluontoisina ja toistuvina. Toistuvat mittaukset voi ajastaa ja niiden tuloksia voi lukea ja myös poistaa reaaliaikaisesti.

KAMK • University of Applied Sciences

Abstract

Author: Niskanen Ville

Title of the Publication: IP Network Performance Measurement System

Degree Title: Bachelor of Engineering, Information and Communication Technology

Keywords: IP, network, performance, measurement, Linux, Python

The commissioner of this thesis, Bittium Wireless Ltd, provides products for tactical communication purposes. The aim of the thesis was to develop a performance measurement system for one of their products, Bittium Tactical Wireless IP Network (TAC WIN). The performance of the network should be able to be measured with one-time "ad hoc" measurements as well as recurring measurements. The measurement system would be integrated as part of the existing software components of TAC WIN system.

The measurement tools used by the measurement system were selected on the basis of the measurement types required by the commissioner. The tools are controlled with measurement parameters, which are either obligatory or optional. Results of the ad hoc measurement given by the tools are instantly returned back to the user. Recurring measurement results are saved in a separate result file, which can be read via user interface. Management of the whole measurement system is performed using the user interface.

The measurement system is written with Python programming language. Due to a required support for multiple simultaneous measurements, implementation of the system utilizes threading and subprocessing. The measurement system contains a socket server communicating with the user interface. Ad hoc measurements are launched via the socket server. Recurring measurements are written as part of the measurement system's configuration file via separate TAC WIN configuration management. The JSON-formatted measurement parameters are validated by the configuration management.

The measurement system created as a result of the thesis met most of the requirements of the commissioner. The system enables both one-time and recurring performance measurements, which can be preset. The results of the recurring measurements are possible to be read and deleted in real time.

Sisällys

Lyhenneluettelo

HTTP	Hypertext Transfer Protocol. Selainten ja WWW-palvelimien käyttämä tiedonsiirtoprotokolla.
ICMP	Internet Control Message Protocol. Verkon diagnosointiin ja virheilmoitusten välittämiseen käytettävä protokolla.
IP	Internet Protocol. Internet-verkon pakettiliikenteestä vastaava tiedonsiirtoprotokolla.
JSON	JavaScript Object Notation. Tiedonsiirrossa käytettävä avain-arvo-pareihin perustuva tiedostomuoto.
MANET	Mobile Ad Hoc Network. Langaton itseohjautuva dynaaminen tietoverkko.
REST	Representational State Transfer. Ohjelmointirajapintojen arkkitehtuurimalli.
SMTP	Simple Mail Transfer Protocol. Sähköpostipalvelimien käyttämä tiedonsiirtoprotokolla.
SNMP	Simple Network Management Protocol. Verkon hallintaan ja monitorointiin käytettävä tietoliikenneprotokolla.
TAC WIN	Tactical Wireless IP Network. Bittium Oyj:n tuotteista koostuva sotilas- ja viranomaiskäyttöön tarkoitettu langaton laajakaistaverkko.
TCP	Transmission Control Protocol. Yhteysperustainen luotettavaan tiedonsiirtoon tarkoitettu protokolla.
ToS	Type of Service. Osa IP-otsikkoa, jossa määritellään toivottu palvelun laatu IP-pakettia käsiteltäessä.
TTL	Time To Live. Datan elinaika tietoverkossa.
UDP	User Datagram Protocol. Yhteydetön yksinkertainen tiedonsiirtoprotokolla.

1 Johdanto

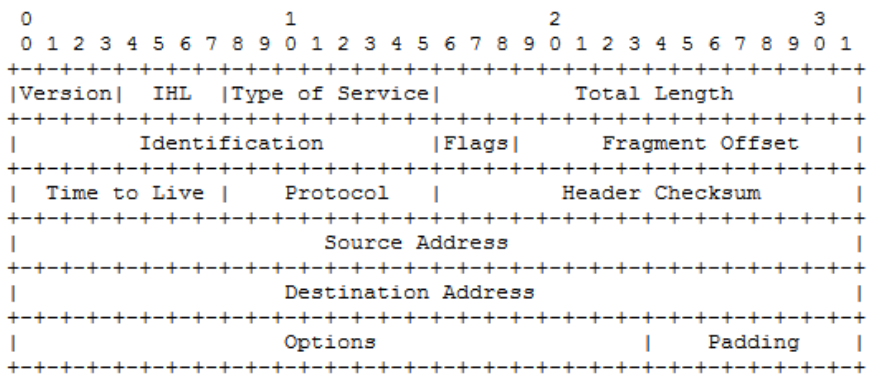
Bittium Oyj on Oulussa vuonna 1985 perustettu luotettavia ja turvallisia viestintä- ja liitettävyysratkaisuja kehittävä yritys [1]. Bittium on tunnettu etenkin tuotteistaan taktisen tiedonsiirron osa-alueella, mutta viime vuosina yritys on laajentanut toimintaansa myös terveydenhuollon teknologian puolelle. Opinnäytetyön toimeksiantaja, Bittium Wireless Oy, on Bittium Oyj:n tytäryhtiö ja vastaa pääasiassa taktisen tiedonsiirron tuotteista ja palveluista.

Yksi näistä tuotteista on taktinen runkoverkko eli Bittium Tactical Wireless IP Network™ (TAC WIN). Se on langaton laajakaistaverkko, joka on tarkoitettu sotilas- ja viranomaiskäyttöön. Bittium TAC WIN mahdollistaa MANET-, linkki- ja liityntäverkkojen muodostamisen, joista se kytkee yhden loogisen IP-verkon. Järjestelmän ytimenä toimiva taktinen reititin sisältää useita eri liitäntärajapintoja langalliseen tiedonsiirtoon. Langattomien verkkojen muodostaminen on mahdollista TAC WIN -järjestelmään sisältyvillä kolmen eri taajuusalueen radioyksiköillä. [2.]

Toimeksiantajalla ilmeni tarve mitata TAC WIN -verkon ja sen yhteyksien laatua ja kapasiteettia erilaisilla suorituskykymittauksilla. Mittaukset tulisi toteuttaa estämättä tai häiritsemättä verkon muuta toimintaa. Opinnäytetyöni aiheeksi tarjottiin suunnitella suorituskyvyn mittausjärjestelmä, joka toteuttaisi mittaukset ja muut niihin liittyvät toimenpiteet, kuten mittaustulosten tallentamisen, toimeksiantajan vaatimusten mukaisesti. TAC WIN -tuotteissa oli jo ennestään osana Bittium Network Manager -työkalua pienimuotoisia mittauksia tekevä komponentti, mutta tavoitteena oli laajentaa sen toiminnallisuutta uusien vaatimusten mukaiseksi.

2 IP-verkko

Tietokoneverkolla tarkoitetaan verkkoa, jossa tietokoneet ovat yhteydessä toisiinsa ja jakavat informaatiota keskenään. Tietokoneiden välinen tiedonsiirto on mahdollista yhteisen protokollan avulla, joka määrittää säännöt laitteiden väliselle tietoliikenteelle, jotta ne ymmärtävät toisiaan [3]. Yksi yleisimmästä tietoliikenneprotokollista on IP (Internet Protocol), joka toimii nimensä mukaisesti Internet-verkon protokollana. Siinä laitteiden välillä liikkuva tieto jaetaan ja pakataan paketteihin, jotka välitetään halutulle vastaanottajalle yksilöllisen IP-osoitteen perusteella. IP-paketti koostuu välitettävästä tiedosta sekä kuvassa 1 havainnollistettavasta IP-otsikosta, joka sisältää muun muassa lähettäjän ja vastaanottajan IP-osoitteen sekä käytettävän kuljetusprotokollan. [3.]



Kuva 1. Esimerkki IP-otsikon rakenteesta [4]

Kuljetusprotokolla vastaa nimensä mukaisesti pakettien kuljettamisesta lähettäjältä vastaanottajalle. Yleisimpiä kuljetusprotokollia ovat TCP (Transmission Control Protocol) ja UDP (User Datagram Protocol). TCP on näistä kahdesta luotettavampi, koska se varmistaa, että lähetetty data myös saapuu vastaanottajalle ja oikeassa järjestyksessä. TCP vaatii myös yhteyden päätelaitteiden välille ennen lähettämistä, joka tapahtuu IP-paketeista muodostettuna katkeamattomana ”datavirtana”. UDP ei vaadi laitteiden välille yhteyttä, mutta se ei myöskään varmista datan perille pääsyä ja järjestystä. UDP lähettää ja vastaanottaa datan paketteina eli datagrammeina, mistä myös protokollan nimi tulee. [5.]

Internet-verkossa kommunikointia varten on kehitetty TCP/IP-viitemalli, jossa protokollat on jaettu neljään eri kerrokseen:

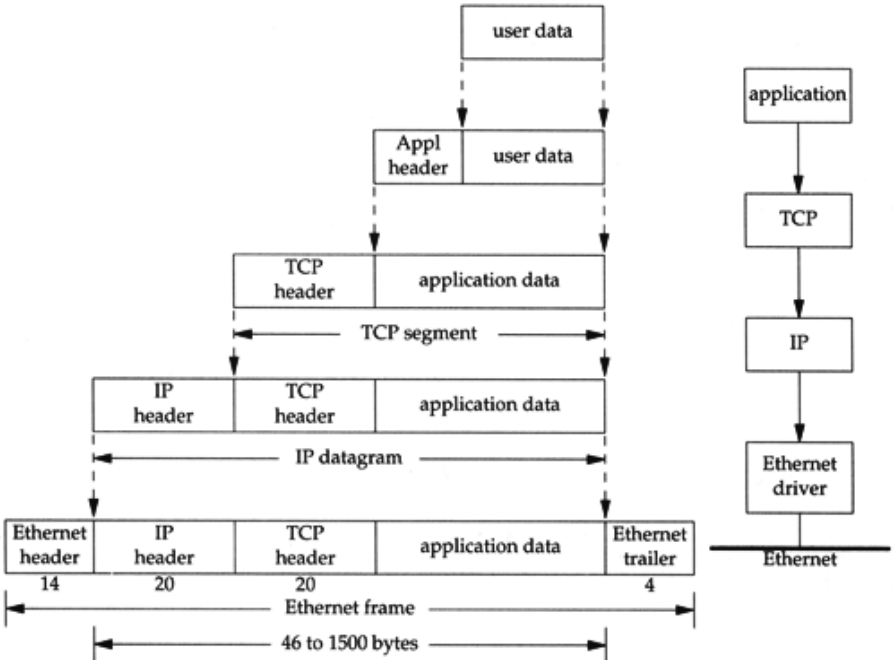
- Sovelluskerros

- Kuljetuskerros
- Verkkokerros
- Fyysinen kerros

Sovelluskerros on mallin ylin kerros, ja sen käsittämät protokollat tarjoavat palveluitaan suoraan käyttäjille, kuten esimerkiksi sähköpostien välittämiseen tarkoitettu SMTP (Simple Mail Transfer Protocol). Sovelluskerrokseen lukeutuu myös tukiprotokollia yleisiä järjestelmätoimintoja varten, kuten verkon hallinnoimiseen käytettävä SNMP (Simple Network Management Protocol). [6.]

Kuljetuskerros käsittää jo aiemmin esitellyt kuljetusprotokollat ja viitemallin nimen mukaisesti kuljetusprotokolla TCP ja verkkokerroksen IP muodostavat perustan internet-verkolle. Mallin alin kerros eli fyysinen kerros on tarkoitettu verkon fyysinen tiedonsiirrolle, eli miten tieto fyysisesti luodaan ja miten se siirtyy fyysisesti päätelaitteiden välillä esimerkiksi verkkokaapelin välityksellä tai langattomasti. Fyysisen kerroksen protokollista yleisin on Ethernet. [7.]

Jokainen viitemallin protokollista lisää paketin liikkumisen kannalta olennaista tietoa pakettiin luomalla oman otsikkonsa ylemmästä kerroksesta saamansa datan eteen. Kuvassa 2 havainnollistetaan, kuinka käyttäjän antamasta datasta muodostuu kerros kerrokselta fyysistä tiedonsiirtoa varten valmis paketti, tässä tapauksessa Ethernet-paketti. [8.]



Kuva 2. Esimerkki paketin rakentumisesta protokollakerroksittain [8]

Verkko toimii silloin, kun verkon sisäiset yhteydet toimivat, eli paketit liikkuvat onnistuneesti kahden päätelaitteen välillä. Yleinen tapa testata yhteyden toimivuutta on ping-työkalun avulla, jolla voi mitata aikaa, joka kuluu kyselypaketin lähettämisestä vastauspaketin vastaanottamiseen. Tätä aikaa voidaan kutsua myös latenssiksi. Ping-työkalusta kerrotaan tarkemmin tämän opinnäytetyön luvussa 3.2.1.

Jos lähetetty paketti ei saavuta vastaanottajaa, tapahtuu pakettihäviötä. Yleinen syy tälle on se, että yhteys on kuormittunut liikaa, joko ruuhkasta tai yhteyden siirtokapasiteetin rajallisuudesta johtuen. Pakettihäviöksi voidaan katsoa myös tilanne, jossa paketti ei todellisuudessa häviä, mutta saapuu perille suurella viiveellä. [9.] Pakettien latenssien vaihdellessa suuresti voivat paketit päätyä väärään järjestykseen tai hitaampia paketteja joudutaan odottamaan, jolloin yhteyden laatu kärsii. Yhdensuuntaisten latenssien keskinäisten erojen keskiarvo eli jitter kertoo siis paljon yhteyden tasaisuudesta. [10.]

Yhteyden nopeus riippuu siitä, kuinka paljon dataa se kykenee välittämään tietyssä ajassa, eli mikä on yhteyden siirtokapasiteetti. Siirtokapasiteetti ilmaistaan yleensä bitteinä sekunnissa, tosin nykyisillä internetin yhteysnopeuksilla käytetään yleisemmin sen kerrannaisarvoja eli kilobittejä sekunnissa (1 000 bittiä) tai megabittejä sekunnissa (1 000 000 bittiä). [11.]

3 Suunnittelu

Mittaamisjärjestelmän tulisi pystyä suorittamaan mittauksia sekä kertaluontoisina että toistuvina. Kertaluontoisessa mittauksessa mittaus tehdään välittömästi yhden kerran, jonka jälkeen mittaustulokset palautetaan käyttäjälle. Toistuvassa mittauksessa sama mittaus suoritetaan yhden tai useamman kerran ja mittaukset voi aikatauluttaa. Toistuvan mittauksen aikana suoritettujen kertamittausten tulokset tallennetaan tiedostoon, josta niitä voi hakea mittauksen aikana ja myös mittauksen loppumisen jälkeen.

Mittaamisjärjestelmän alustana toimii sulautettuna käyttöjärjestelmänä Linux, joten myös käytettävien mittaustyökalujen on oltava yhteensopivia Linuxin kanssa.

3.1 Mittaustyypit

Järjestelmän tulisi toimeksiantajan alkuperäisten vaatimusten mukaan tukea seuraavia mittaustyyppejä:

- Latenssi eli IP-paketin edestakaiseen matkaan kuluva aika
- Pakettihäviö eli vastaanotettujen ja lähetettyjen pakettien suhde
- Jitter eli peräkkäin mitattujen yhdensuuntaiseen matkaan kuluvien aikojen erojen keskiarvo [12]
- Siirtokapasiteetti eli lähetettyjen pakettien nopeus yhdensuuntaisella matkalla
- Reittijäljitys eli IP-paketin kulkema reitti lähettäjältä vastaanottajalle
- Linkkikuorma

Linkkikuorman mittaamiseen pitäisi käyttää SNMP-protokollaa. Aikataulusyistä SNMP-protokollaan perehtyminen ja linkkikuorman mittaaminen päätettiin kuitenkin siirtää myöhemmäksi ja toteuttaa ensimmäinen versio mittausjärjestelmästä ilman linkkikuorman mittaamista.

Kaikki mittaustyypit ovat käytettävissä kertaluontoisia mittauksia suoritettaessa. Toistuvia mittauksia suoritettaessa mahdollisia mittaustyyppejä ovat latenssi, jitter ja pakettihäviö.

3.2 Mittaustyökalut

Latenssin, pakettihäviön sekä reittijäljityksen mittaamiseen päätettiin käyttää Linuxista valmiina löytyviä ping- ja traceroute-työkaluja. Kyseisiä työkaluja oli jo aiemmin käytetty osana nyt laajennettavaa komponenttia. Jitterin ja siirtokapasiteetin mittaamiseen valittiin avoimen lähdekoodin iPerf-työkalu, jota oli myös käytetty aiemmin TAC WIN -järjestelmän ulkoiseen testaamiseen.

3.2.1 Ping

Ping on työkalu, joka lähettää ICMP-protokollan ECHO_REQUEST-paketteja sille parametrina annettuun IP-osoitteeseen. Paketin saatuaan vastaanottaja lähettää takaisin ECHO_RESPONSE-paketin. [13.] Ping mittaa paketin lähettämisen ja vastaanottamisen välisen ajan eli latenssin ja tulostaa sen reaaliajassa. Mittaamisen päätyttyä ping tulostaa myös pienimmän ja suurimman mitatun latenssin, latenssien keskiarvon ja pakettihäviön. Kuvassa 3 ping-työkalulla on suoritettu viiden paketin mittaus Googlen osoitteeseen.

```
ville@VM1:~$ ping google.com -c 5
PING google.com (172.217.20.46) 56(84) bytes of data.
64 bytes from arn11s01-in-f14.1e100.net (172.217.20.46): icmp_seq=1 ttl=52 time=31.8 ms
64 bytes from arn11s01-in-f14.1e100.net (172.217.20.46): icmp_seq=2 ttl=52 time=32.9 ms
64 bytes from arn11s01-in-f14.1e100.net (172.217.20.46): icmp_seq=3 ttl=52 time=33.1 ms
64 bytes from arn11s01-in-f14.1e100.net (172.217.20.46): icmp_seq=4 ttl=52 time=32.2 ms
64 bytes from arn11s01-in-f14.1e100.net (172.217.20.46): icmp_seq=5 ttl=52 time=32.7 ms

--- google.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4014ms
rtt min/avg/max/mdev = 31.853/32.572/33.134/0.470 ms
```

Kuva 3. Ping-työkalun terminaalinäkymä

Käytettäessä ping-työkalua mittaamiseen ei vastaanottajalta vaadita muita ennakkotoimenpiteitä kuin palomuurin aukaisu, koska työkalu käyttää ICMP-protokollaa, joka on puolestaan kiinteä osa IP-protokollaa [14]. Jos mittauksen molemmat päätepisteet ovat samassa IP-verkossa, niin vastaanottaja lähettää vastauspaketin automaattisesti.

3.2.2 Traceroute

Traceroute on työkalu, joka jäljittää IP-pakettien kulkeman reitin lähettäjältä vastaanottajalle. Tracerouten toimintaperiaate perustuu Time to Live -arvon (TTL) hyödyntämiseen [15]. TTL määrää, monenko reitittimen kautta paketti saa kulkea, eli se määrittää paketin eliniän. Jokainen välittäjänä toimiva reititin vähentää TTL-arvoa yhdellä ennen sen välittämistä eteenpäin. Jos TTL:n arvo vähenee nollaan, paketti hylätään ja alkuperäiselle lähettäjälle lähetetään virheviesti.

Traceroute lähettää IP-paketteja vastaanottajalle aloittaen TTL-arvosta yksi ja kasvattaen niiden TTL-arvoa yhdellä jokaisen lähetetyn paketin jälkeen. Traceroute voi käyttää pakettien lähettämiseen joko ICMP- tai UDP-protokollaa. Vastaanottamalla TTL-arvon nollautumisesta aiheutuvia virheviestejä traceroute pystyy keräämään listan kaikista laitteista, jotka välittävät IP-pakettia eteenpäin kohti vastaanottajaa. Traceroute tulostaa virheviestin lähettäneiden laitteiden IP-osoitteet ja laitekohtaiset latenssit. Kuvassa 4 on suoritettu traceroute-työkalulla reittijäljitys Googlen osoitteeseen. [15.]

```
ville@VM1:~$ traceroute google.com
traceroute to google.com (216.58.207.206), 30 hops max, 60 byte packets
 1  * * *
 2  10.235.228.253 (10.235.228.253)  13.123 ms  13.048 ms  12.938 ms
 3  10.237.229.69 (10.237.229.69)  13.275 ms  13.165 ms  13.254 ms
 4  185.171.178.210 (185.171.178.210)  13.344 ms  13.249 ms  13.294 ms
 5  185.171.178.211 (185.171.178.211)  13.348 ms  13.236 ms  13.352 ms
 6  62-44-205-60.bb.dnainternet.fi (62.44.205.60)  23.772 ms  23.995 ms  23.874 ms
 7  hel5-tr1.dnaip.fi (62.78.112.137)  24.057 ms  24.126 ms  24.271 ms
 8  migbgp1.dnaip.fi (62.78.107.253)  24.396 ms  24.218 ms  24.099 ms
 9  hel5-tr1.dnaip.fi (62.78.117.22)  32.607 ms  32.477 ms  32.385 ms
10  tur4-tr1.dnaip.fi (62.78.110.216)  32.251 ms  32.463 ms  32.320 ms
11  tuk2-sr1.dnaip.fi (62.78.107.168)  32.550 ms  32.588 ms  32.405 ms
12  google.o.dnaip.fi (217.78.199.105)  32.124 ms  32.030 ms  32.046 ms
13  * * *
14  108.170.253.177 (108.170.253.177)  32.277 ms 209.85.242.82 (209.85.242.82)  33.430 ms  33.272 ms
15  108.170.253.182 (108.170.253.182)  32.491 ms arn11s04-in-f14.1e100.net (216.58.207.206)  32.165 ms  32.014 ms
```

Kuva 4. Traceroute-työkalun terminaalinäkymä

3.2.3 iPerf

iPerf on työkalu, joka luo lähettäjän ja vastaanottajan välille tietoliikennettä joko TCP- tai UDP-protokollalla. iPerf mittaa luomansa pakettiliikenteen avulla yhteyden siirtokapasiteettia, jonka lisäksi UDP-yhteyttä käytettäessä se mittaa myös jitterin ja pakettihäviön. [16.]

iPerf-työkalun toiminta perustuu palvelin-asiakas-malliin, jossa yhteyden toisen pään on toimittava palvelimena toisen toimiessa asiakkaana. Asiakaslaite lähettää paketteja palvelimelle

sille annettujen parametrien mukaisesti, joita voivat olla esimerkiksi lähettämisen kesto tai pakettien määrä ja koko. Toisin kuin ping- ja traceroute-työkalut, iPerfin käyttäminen vaatii siis toimenpiteitä mittauksen molemmilta päätepisteiltä. Kuvassa 5 on aluksi luotu palvelin daemon-prosessina, jonka jälkeen käynnistetty asiakkaana mittaus siihen UDP-protokollalla kolmen sekunnin ajaksi.

```
ville@VM1:~$ iperf3 -s -D
ville@VM1:~$ iperf3 -c 127.0.0.1 -u -t 3
Connecting to host 127.0.0.1, port 5201
[  4] local 127.0.0.1 port 51735 connected to 127.0.0.1 port 5201
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   1.00-2.00   sec   128 KBytes  1.05 Mbits/sec  16
[  4]   2.00-3.00   sec   128 KBytes  1.05 Mbits/sec  16
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  4]   0.00-3.00   sec   384 KBytes  1.05 Mbits/sec  0.108 ms  0/47 (0%)
[  4] Sent 47 datagrams

iperf Done.
```

Kuva 5. iPerf3-työkalun asiakasterminaalinäkymä

Työkalusta on kaksi aktiivista versiota, iPerf2 ja iPerf3, joita kehitetään täysin itsenäisinä, eivätkä ne ole keskenään yhteensopivia [17]. Mittaamisjärjestelmän käyttöön päätettiin valita iPerf3-versio, koska sen palvelin tukee sekä TCP- että UDP-protokollaa yhtä aikaa. Lisäksi iPerf3-versiossa on mahdollista määrittää parametrina asiakkaalle sen lähettämien pakettien määrä. [12.] Tämä mahdollistaa myös mittaamisjärjestelmän parametrien yhdenmukaistamisen työkalujen välillä.

3.3 Mittausparametrit

Ennen mittauksen käynnistämistä tulee sille määrittää parametreja, jotka ohjaavat mittauksen kulkua ja toimintaa. Parametrit ovat luonteeltaan pakollisia, joita ilman mittausta ei voi suorittaa, tai vapaaehtoisia, joilla annetaan niiden puuttuessa oletusarvo tai jotka mahdollisesti voi jättää kokonaan pois.

Mittausjärjestelmän tulisi toimeksiantajan vaatimusten mukaisesti tukea seuraavia pakollisia mittausparametreja:

- Mittaustyyppi

- Lähettäjän IP-osoite
- Vastaanottajan IP-osoite
- Lähetettävän paketin koko
- Palvelun tyyppi eli type of service (ToS)

Toistuvalle mittaukselle pakollisia parametreja ovat lisäksi aloitusaika, lopetusaika ja aikaväli, jolloin kertamittauksia suoritetaan aloitusajasta alkaen aikavälin välein lopetusaikaan asti.

Mittaustyypistä riippuen mittaukselle tulisi pystyä määrittämään myös muita parametreja:

- Lähetettävien pakettien määrä, muut mittaustyypit paitsi reittijäljitys
- Pakettien lähetysväli, mitattaessa latenssia
- Aikakatkaisu, mitattaessa latenssia tai reittijäljitystä
- Fragmentoitumisen esto, mitattaessa latenssia tai reittijäljitystä
- Protokolla, mitattaessa reittijäljitystä

3.4 Mittaustulokset

Mittauksen päätyttyä mittaamisjärjestelmän mittaamat tulokset tulisi joko palauttaa käyttäjälle, jos mittaus oli kertaluontoinen, tai tallentaa tiedostoon, jos mittaus oli toistuva.

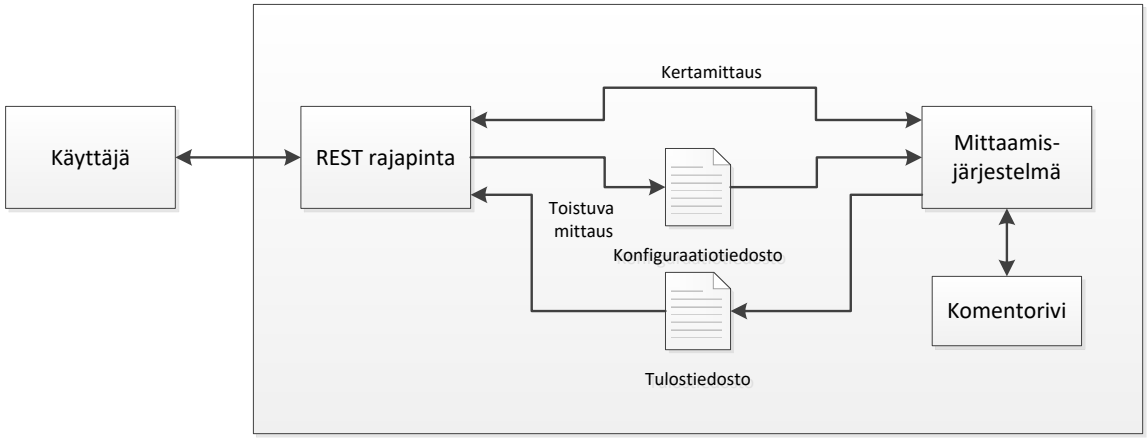
Ping-työkalu palauttaa tuloksina samalla sekä latenssin että pakettihäviön. iPerf-työkalu palauttaa puolestaan tuloksina UDP-protokollaa käytettäessä sekä jitterin että pakettihäviön. Jos mittaustyypiksi määriteltäisiin pelkästään latenssi, jitter tai pakettihäviö, jäisi käytettävän mittaustyökalun joka tapauksessa palauttamista tuloksista silloin osa hyödyntämättä. Tästä johtuen päätettiin mittaustyypeiksi vaihtaa *latenssi ja pakettihäviö* sekä *jitter ja pakettihäviö*. Latenssin yhteydessä mitataan ICMP-pakettien häviötä ja jitterin yhteydessä UDP-pakettien häviötä.

Järjestelmän tulisi toimeksiantajan vaatimusten mukaisesti saada seuraavia tuloksia mittaustyypistä riippuen:

- Mittaustyyppi: latenssi ja pakettihäviö
    - Tulokset: pakettikohtaiset latenssit, pienin latenssi, suurin latenssi, keskiarvolatenssi ja pakettihäviö. Latenssit ilmoitetaan millisekunteina ja pakettihäviö prosentteina. Toistuvassa mittauksessa tallennetaan vain keskiarvolatenssi ja pakettihäviö.
- Mittaustyyppi: jitter ja pakettihäviö
    - Tulokset: jitter millisekunteina ja pakettihäviö prosentteina
- Mittaustyyppi: siirtokapasiteetti
    - Tulokset: siirtokapasiteetti bitteinä sekunnissa
- Mittaustyyppi: reittijäljitys
    - Tulokset: jokaista hyppyä eli paketin välittäjää kohden numero, IP-osoite ja latenssi millisekunteina. Lisäksi status-tieto siitä, onnistuiko jäljittäminen täysin vai ilmenikö mittauksen suorituksessa virheitä.

3.5 Ohjelmistoarkkitehtuuri

TAC WIN -järjestelmän laitteita ja koko verkkoa voi hallita joko verkkoselaimessa toimivan käyttöliittymän tai REST-rajapinnan kautta. Mittaamisjärjestelmän käyttäminen ja hallinta suunniteltiin toteutettavaksi vain REST-rajapinnan kautta, kuten kuvassa 6 on havainnollistettu, mutta tulevaisuudessa myös verkkokäyttöliittymän tukeminen voisi olla mahdollista.



Kuva 6. Kuvaus mittaamisjärjestelmästä osana Linux-alustaista ohjelmistoa

Kertamittauksia suorittaessa rajapinta lähettäisi mittauskäskyn suoraan mittaamisjärjestelmälle ja jäisi odottamaan mittauksen tuloksia. Suoritettuaan mittauksen mittaamisjärjestelmä palauttaa tulokset rajapinnan kautta käyttäjälle. Tällä menetelmällä mittaukseen liittyvien toimenpiteiden viiveet ovat mahdollisimman vähäiset, kun toimeksiantajan vaatimusten mukaisesti kertamittaukset tulisi suorittaa välittömästi ja tulokset tulisi myös palauttaa viipymättä.

REST-rajapinta kirjoittaa suoritettavat toistuvat mittaukset parametreineen mittaamisjärjestelmän konfiguraatiotiedostoon. Mittaamisjärjestelmä lukee tiedostosta mittauksien parametrit ja käynnistää mittaukset parametrien mukaiseen aikaan. Mittauksien valmistuttua järjestelmä kirjoittaa mittaustulokset tulostiedostoonsa. Rajapinnan kautta käyttäjä voi lukea mittaustuloksia tiedostosta haluamanaan ajankohtana.

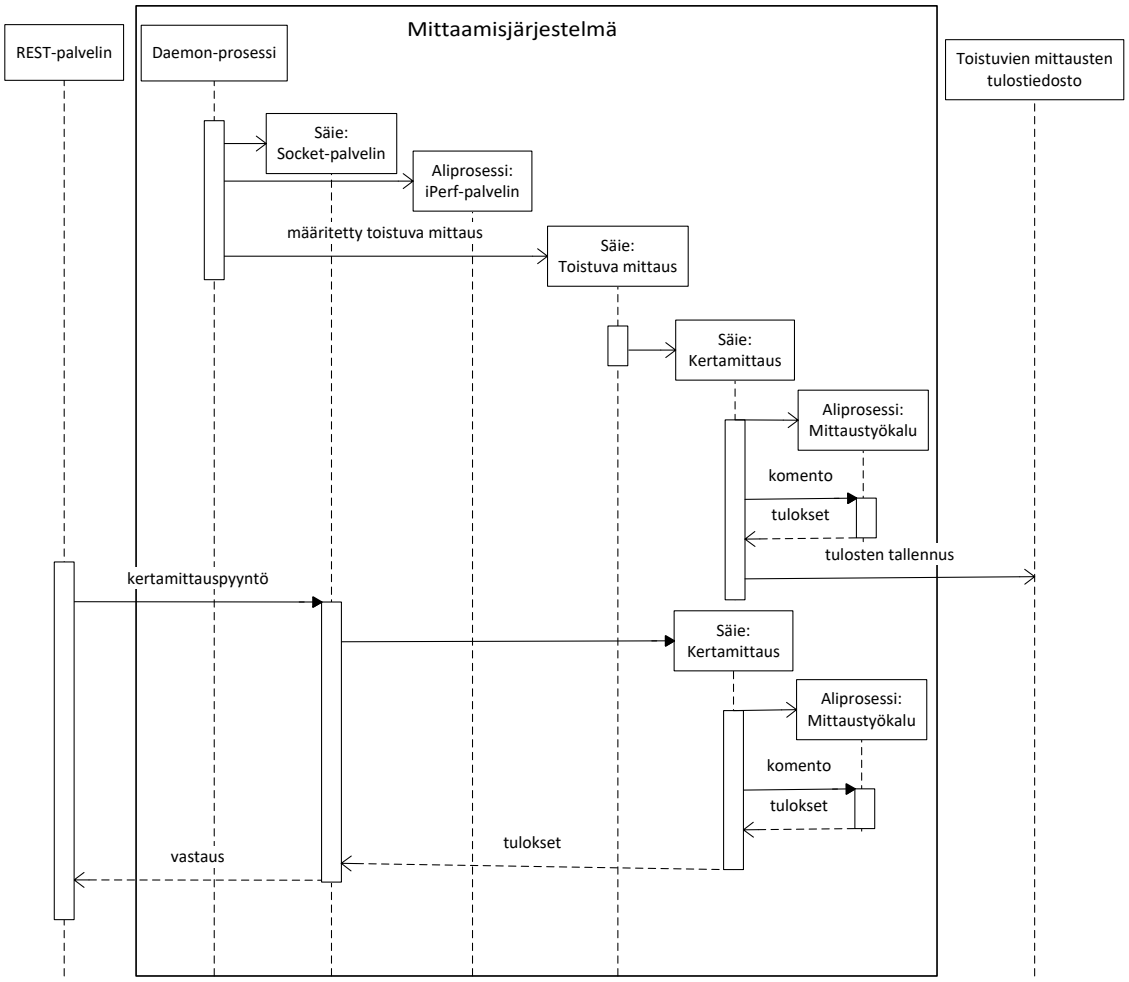
4 Mittaamisjärjestelmän toteutus

Mittaamisjärjestelmä toteutettiin osaksi jo ennestään olemassa olevia ohjelmistokomponentteja. Järjestelmän myötä laajennetut komponentit on toteutettu Python-ohjelmointikielellä.

Järjestelmän ytimenä toimii Python-skripti, joka luo daemon-prosessin. Daemonilla tarkoitetaan prosessia, joka suoriutuu taustalla itsenäisesti ilman käyttäjän ohjaamista [18]. Mittaamisjärjestelmässä daemon-prosessi vastaa mittausten suorittamisesta ja muista järjestelmän ydintehtävistä.

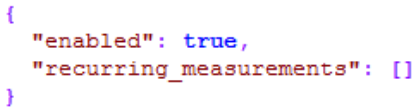
Prosessi luo socket-palvelimen, joka käsittelee socketin kautta tulleet mittauspyynnöt ja muut järjestelmälle tulevat komennot. Socket-yhteys mahdollistaa prosessien välisen paikallisen tietoliikenteen laitteen sisällä [19]. Socket-palvelin on toteutettu tässä järjestelmässä Pythonin SocketServer-kirjastolla.

Järjestelmän pitää tukea useita yhtäaikaisia mittauksia. Tästä syystä socket-palvelin toimii omassa säikeessään ja jokaiselle mittaukselle luodaan myös aina oma säie. Säikeistämisellä tarkoitetaan prosessin jakamista pienempiin yhtäaikaisesti suoriutuviin osiin, jotka jakavat keskenään prosessin suorituskyvyn ja resurssit [20]. Prosessin säikeistys on toteutettu tässä järjestelmässä Pythonin threading-kirjastolla. Kuvan 7 sekvenssikaaviossa kuvataan mittaamisjärjestelmän päätoimintoja ja havainnollistetaan, kuinka järjestelmä jakautuu eri tilanteissa useisiin eri säikeisiin ja aliprosesseihin.



Kuva 7. Sekvenssikaavio mittaamisjärjestelmän päätoiminnoista

Mittaamisjärjestelmällä on oma JSON-muotoinen konfiguraatiotiedosto, jonka sisältöä päivitetään konfiguraationhallinnan kautta. Tiedostossa on määritetty toistuvat mittaukset, jotka järjestelmän tulee suorittaa, mikäli järjestelmä on määritetty käyttöön. Myös järjestelmän käyttöönotto on määritetty tiedostossa. Tiedoston rakenne on esiteltynä kuvassa 8. Järjestelmän daemon-prosessi lukee käynnistyessään tiedostoa, ja jos järjestelmän tulee olla käytössä, niin daemon käynnistää iPerf-palvelimen ja ajastaa tiedostossa määritetyt toistuvat mittaukset. Järjestelmän on siis oltava määritetty käyttöön myös siinä tilanteessa, jos laite on vastaanottavana päätepisteenä iPerf-työkalua käyttävässä mittauksessa.



Kuva 8. Esimerkki konfiguraatiotiedoston rakenteesta, kun järjestelmä on määritetty käyttöön ja toistuvia mittauksia ei ole määritetty

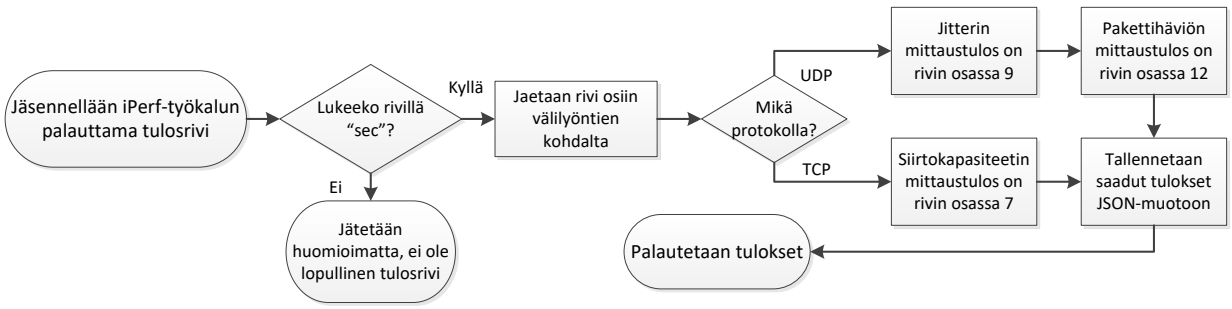
4.1 Mittaustyökalujen järjestelmäohjaus

Jokaiselle mittaustyökalulle on toteutettu oma luokka Python-skriptissä. Luokka määrittää siitä luotavan olion tyypin, rakenteen ja toiminnan [21]. Jokaisessa luokassa on määritetty siten työkalusta riippuen sen suorittamiseen ja tulosten talteen ottamiseen vaadittavat toimenpiteet ja tiedot, kuten kiinteät komentojen nimet ja tuloksien esittämismuodot. Ping- ja traceroute-työkaluille luokat olivat jo valmiina aiemman toteutuksen myötä. Niissä kuitenkin työkalujen suoritus oli jatkuvaa ja tuloksia palautettiin käytännössä reaaliaikaisesti, joten luokat vaativat muokkaamista palvellakseen myös mittaamisjärjestelmän vaatimuksia.

Kun luokasta luodaan olio, sille voidaan antaa oliokohtaisia parametreja, kuten tässä tapauksessa mittausparametrit. Olio suoriutuu luokan määrityksien ja omien parametriensa mukaisesti. Jokaista erillistä mittausta, joko kertamittausta tai toistuvan mittauksen osamittausta kohden luodaan siis oma olio. Luomisensa jälkeen mittausolio tallentaa parametrinsa muuttujiksi itsensä ja funktioidensa käyttöön self-avainsanan avulla, sekä luo mittausta varten oman säikeen.

Oliot ohjaavat mittaustyökaluja luomalla aliprosessin, jota ne hallitsevat ja kuuntelevat putkien (pipe) kautta. Nämä toimenpiteet toteutetaan Pythonin subprocess-kirjaston Popen-luokan avulla. Aiemmin tallennetuista muuttujista kootaan työkalun käynnistämiskomento, joka annetaan aliprosessille suoritettavaksi.

Mittaustyökalun tuottamia tuloksia luetaan aliprosessin tuloputken kautta rivi kerrallaan. Mittaustyökalun otsikkorivit ja muut haluttujen tulosten kannalta merkityksettömät rivit jätetään huomioimatta. Mahdolliset virhetilanteet, kuten yhteyden muodostamisen epäonnistuminen, pyritään tunnistamaan myös rivien lukemisen aikana. Tulosriveiksi tunnistetuista riveistä jäsennellään mittauksesta riippuen halutut tulokset talteen. Esimerkiksi iperf-luokassa tulosrivi annetaan argumenttina kuvassa 9 esitellylle jäsentelyfunktiolle, joka palauttaa rivistä jäsennellyt tulokset.



Kuva 9. iPerf-työkalun tuloksien jäsentelyfunktion rakenne

4.2 Kertamittaukset

Kertamittausten käynnistämispyynnöt tulevat socket-yhteyden kautta. Mittaamisjärjestelmän socket-palvelin välittää pyynnöt eteenpäin mittauksen käynnistävälle funktiolle. Koska mittauksella ei ole yksilöllistä tunnistetta, funktio tunnistaa mittauksen kertamittaukseksi ja luo sitä varten jonon, joka mahdollistaa mittaustulosten välittämisen mittaussäikeeltä palvelinsäikeelle. Tämän jälkeen funktio luo mittaustyypin perusteella oikean mittaustyökalun luokasta olion, joka saa parametrinaan mittausparametrien lisäksi myös funktion luoman jonon. Koska mittaus on kertaluontoinen, funktio jää odottamaan, kunnes olio on suorittanut mittauksen ja asettanut saamansa mittaustulokset jonoon. Lopuksi funktio lukee tulokset jonosta ja palauttaa tulokset JSON-muodossa socket-palvelimelle, joka puolestaan lähettää tulokset takaisin socketiin. Jonoon liittyvät toimenpiteet on toteutettu Pythonin queue-kirjaston avulla. [22.]

Kertamittausten suorittamista varten mittaamisjärjestelmän tulee olla konfiguroitu käyttöön. Jos järjestelmää ei ole määritetty käyttöön, palauttaa socket-palvelin kertamittausten käynnistämispyyntöihin vastaukseksi suoraan virheviestin.

4.3 Toistuvat mittaukset

Toistuvat mittaukset määritetään osaksi mittaamisjärjestelmän konfiguraatiotiedostoa. Mittaukset määritetään JSON-muotoisen tiedoston sisältämään listaan, jokainen mittaus omaksi itsenäiseksi objektikseen. Järjestelmän daemon-prosessi lukee käynnistyessään tiedoston

listalta objektit yksi kerrallaan ja lähettää objektin toistuvien mittauksien luomisesta vastaavalle funktiolle, jos mittaus on määritetty aktiiviseksi.

Luontifunktio muuttaa mittausparametreina saamansa mittauksen aloitus- ja lopetusajan tekstimuodosta vertailtavaan päivämäärän ja kellonajan muotoon erillisen funktion avulla, joka käyttää päivämäärän ja kellonajan muodostamiseen Pythonin datetime-kirjastoa. Luontifunktio laskee ajat sekunteina, jotka kuluvat funktion suorittamishetkestä mittaukselle määritettyihin aloitus- ja lopetusaikaan. Funktio tallentaa laskemansa ajat muuttujiin ”viive” ja ”kesto”.

Jos mittauksen aloitusajan ja funktion suorittamishetken välinen aika eli muuttuja ”viive” on negatiivinen eli mittauksen aloitusaika on jo mennyt, niin voidaan päätellä syyn olevan jompikumpi seuraavista:

1. Mittaamisjärjestelmä ei ole ollut käynnissä ja/tai käytössä mittauksen aloitusaikana.

2. Mittaamisjärjestelmä on käynnistynyt uudestaan mittauksen aloitusajan jälkeen.

Näistä syistä numero 2 on todennäköisempi vaihtoehto, koska konfiguraationhallinta käynnistää järjestelmän uudelleen aina konfiguraatiotiedostoa päivitettäessä. Tästä syystä funktio lukee ensin mittaustulokset sisältävää tiedostoa. Jos tiedostosta ei löydy mittauksen tunnisteen perusteella tuloksia, niin syy on vaihtoehto 1, ja funktio asettaa ”viive”-muuttujan arvoksi nollan. Muussa tapauksessa syy on vaihtoehto 2 ja funktio laskee tuloksista löytyvän viimeisimmän kertamittauksen ja funktion suorittamishetken välisen ajan. Funktio vertaa laskemaansa aikaa mittaukselle määritettyyn aikaväliin. Jos aikaväli on pienempi kuin laskettu aika, niin viimeisimmästä kertamittauksesta on kulunut aikaväliä suurempi aika, ja funktio asettaa ”viive”-muuttujan arvoksi nollan. Muussa tapauksessa funktio laskee aikavälin ja lasketun ajan välisen eron ja asettaa sen ”viive”-muuttujan arvoksi, jolloin kertamittausten aikaväli säilyy määritetyn mukaisena.

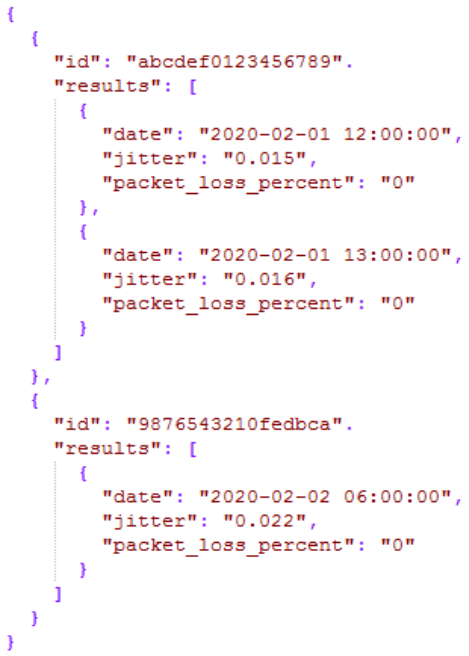
Jos ”viive”-muuttujan arvo on nolla, niin luontifunktio käynnistää mittauksen välittömästi lähettämällä mittausparametrit toistuvan mittauksen aloittavalle funktiolle. Muussa tapauksessa luontifunktio luo ajastinsäikeen, joka käynnistää aloitusfunktion ”viive”-muuttujassa olevan ajan jälkeen. Funktio luo myös mittauksen lopettamista varten ajastimen, joka käynnistää toistuvan mittauksen lopettavan funktion ”kesto”-muuttujassa olevan ajan jälkeen. Ajastinsäikeiden luominen on toteutettu Pythonin threading-kirjaston Timer-aliluokan avulla.

Koska mittauksen yksilöllinen tunniste löytyy mittausparametreista, niin aloitusfunktio tunnistaa mittauksen olevan toistuva. Funktio toteuttaa toistuvuuden luomalla uuden ajastimen, joka käynnistää aloitusfunktion uudestaan mittausparametreissa määritetyn aikavälin jälkeen. Myös mittausoliot tunnistavat mittauksen toistuvaksi yksilöllisen tunnisteen perusteella ja tallentavat saamansa tulokset tiedostoon. Mittauksen lopetusajan koittaessa lopetusfunktio peruuttaa mittauksen uudelleen käynnistävän ajastimen ja määrittää mittauksen konfiguraatiotiedostoon epäaktiiviseksi. Mittauksen voi myös lopettaa ennen lopetusaikaa määrittämällä mittaus epäaktiiviseksi konfiguraationhallinnan kautta.

4.3.1 Mittaustulosten tallentaminen

Mittaamisjärjestelmä luo käynnistyessään mittaustyypeille omat JSON-muotoa olevat tulostiedostot ja lukot. Kun jokin säie asettaa lukon lukituksi, muut säikeet eivät voi lukita kyseistä lukkoa ja jäävät tarvittaessa odottamaan, kunnes lukon lukinnut säie avaa sen [23]. Mittausoliot tallentavat toistuvien mittauksien tulokset jokaisen kertamittauksen jälkeen mittaustyypille tarkoitettuun tulostiedostoon toistuvan mittauksen tulokset tallentavan funktion avulla. Ennen tulostiedoston lukemista tai siihen kirjoittamista funktio lukitsee mittaustyypin lukon, jotta muut mahdolliset samanaikaiset mittaussäikeet eivät pysty lukemaan tai kirjoittamaan samaan aikaan. Tällä toimenpiteellä estetään tulostiedoston korruptoituminen. Lukot on toteutettu Pythonin threading-kirjaston Lock-aliluokan avulla.

Funktio lukee tulostiedostoa ennen tallentamiseen liittyviä toimenpiteitä. Jos tiedosto on täysin tyhjä, funktio luo mittaukset sisältävän taulukon ja luomaansa taulukkoon mittaukselle uuden objektin, johon funktio tallentaa mittauksen tunnisteen ja tulostaulukkoon mittauksen tuloksen omaan objektiinsa. Kuvassa 10 havainnollistetaan tulostiedoston rakennetta, ”id” tarkoittaa mittauksen tunnistetta ja ”results” tulostaulukkoa. Kertamittauksista tallennetaan mittauksen suoritusajankohta ja mittaustyypistä riippuvat tulokset. Esimerkiksi kuvassa 10 toistuva mittaus tunnisteella ”abcdef0123456789” on suorittanut kaksi kertamittausta tunnin aikavälillä 1. helmikuuta 2020.



```
{
  {
    "id": "abcdef0123456789".
    "results": [
      {
        "date": "2020-02-01 12:00:00",
        "jitter": "0.015",
        "packet_loss_percent": "0"
      },
      {
        "date": "2020-02-01 13:00:00",
        "jitter": "0.016",
        "packet_loss_percent": "0"
      }
    ]
  },
  {
    "id": "9876543210fedbca".
    "results": [
      {
        "date": "2020-02-02 06:00:00",
        "jitter": "0.022",
        "packet_loss_percent": "0"
      }
    ]
  }
}
```


Kuva 10. Esimerkki tulostiedoston rakenteesta

Jos tiedosto ei ole tyhjä, funktio tarkistaa erillisen funktion avulla tallennettujen toistuvien mittausten ja niiden mittaustulosten lukumäärän. Jos lukumäärät ylittävät niille määritetyt enimmäismäärät, niin kyseinen erillinen funktio poistaa mittauksia tai tuloksia vanhimmasta päästä aloittaen. Tällä toimenpiteellä estetään laitteen tallennustilan liiallinen täyttyminen. Tarkistuksen jälkeen alkuperäinen tallennusfunktio etsii mittausobjektitaulukosta mittauksen tunnisteen perusteella oikean mittausobjektin ja tallentaa sen tulostaulukkoon kertamittauksen tulokset. Jos mittausobjektia ei löydy, niin mittauksen tuloksia tallennetaan ensimmäisen kerran, joten funktio luo uuden mittausobjektin ja tallentaa mittauksen tiedot ja tulokset siihen. Tiedoston lukemiseen ja siihen kirjoittamiseen käytetään Pythonin json-kirjaston load- ja dump-funktioita.

4.3.2 Mittaustulosten poistaminen

Mittaustulosten poistamispyynnöt tulevat socket-yhteyden kautta. Mittaamisjärjestelmän socket-palvelin välittää pyynnöt eteenpäin mittaustulosten poistamisesta vastaavalle funktiolle. Pyynnön mukana tulevien parametrien perusteella funktio poistaa joko yhden mittauksen tulokset, mittaustyypin tulokset tai kaikki tulokset.

Funktio lukee parametreissa määritetyn mittaustyypin tulostiedostoa, tai poistettaessa kaikkia tuloksia molempia tulostiedostoja vuorotellen. Jos vaatimuksena on poistaa yhden mittauksen tulokset, niin funktio palauttaa virheilmoituksen, jos tiedosto on tyhjä, sen sisältämä taulukko on tyhjä, tai taulukosta ei löydy annetulla mittauksen tunnisteella mittausobjektia.

Jos tiedostosta löytyy tuloksia, funktio käsittelee joko kaikki mittaukset ja niiden tulokset, tai yhden mittauksen tuloksia poistettaessa vain kyseisen mittauksen ja sen tulokset. Funktio tarkistaa konfiguraatiotiedostosta mittauksen aktiivisuuden. Jos mittaus on määritetty aktiiviseksi eli se on edelleen käynnissä, niin viimeisin mittaustulos jätetään poistamatta, koska järjestelmä määrittää seuraavan kertamittauksen suoritusajan sen perusteella. Näin mittauksien aikaväli säilyy määritetyn mukaisesti tasaisena.

Kun haluttujen mittaustulosten poistaminen on suoritettu, funktio lukitsee tulostiedostolle määritetyn lukon, tallentaa käsitellyt tulokset tiedostoon ja avaa lukon. Lopuksi funktio palauttaa tiedon poistamisen onnistumisesta JSON-muodossa socket-palvelimelle, joka puolestaan lähettää tiedon takaisin sockettiin.

5 Muu mittaamisjärjestelmän toimintaan liittyvä toteutus

Mittaamisjärjestelmän ohjaaminen toteutettiin tehtäväksi REST-rajapinnan kautta, jonka kautta voi ohjata ja konfiguroida myös muita kohdealustan palveluja ja ominaisuuksia. Kohdealustassa toimii REST-palvelin, joka vastaanottaa käyttäjän lähettämät HTTP-pyynnöt, käsittelee ja jäsentelee ne ohjelmistokomponentille, joka toteuttaa pyynnön. Ohjelmistokomponentti palauttaa vastauksen takaisin REST-palvelimelle, joka lähettää sen takaisin käyttäjälle jälleen HTTP-muotoisena.

REST (Representational State Transfer) on arkkitehtuurimalli rajapintojen toteuttamiseen hajautettujen järjestelmien välille. REST-mallin periaatteisiin kuuluvat esimerkiksi palvelin-asiakas-malli, jossa palvelin- ja asiakasohjelmistot ovat täysin itsenäisiä, ja tilattomuus, jolloin aiemmista pyynnöistä ei jää mitään historiatietoa palvelimelle, vaan jokainen pyyntö käsitellään täysin uutena pyyntönä. REST-mallissa dataa ja toiminnallisuutta kutsutaan resursseiksi, joita käsitellään osoitepolkujen kautta. [24.]

5.1 Mittaamisjärjestelmän käyttäminen REST-palvelimen kautta

HTTP-pyyntö tulee lähettää REST-palvelimella halutusta toimenpiteestä riippuen oikeaan osoitepolkuun. Kohdealustassa esimerkiksi laitteen konfigurointi, operaatiot ja tilakyselyt on sijoitettu omiin polkuihinsa. Kohdealustaa voi kontrolloida näiden polkujen kautta neljällä eri HTTP-komennolla:

- POST
- GET
- PUT
- DELETE

Konfiguraatiopolkuun voi käyttää kaikkia neljää komentoa, kun taas operaatioihin käytetään pelkkää POST-komentoa ja tilakyselyihin pelkkää GET-komentoa.

5.1.1 POST

POST-komennolla pyydetään palvelinta hyväksymään pyynnön mukana tullut data osaksi osoitepolussa määriteltyä kohdetta [25]. Komennon mukana on siis oltava aina jotain dataa. Esimerkiksi uuden toistuvan mittauksen voi luoda lähettämällä POST-komennolla mittauksen parametrit kuvan 11 mukaiseen tapaan objektina konfiguraatiopolun alipolkuun ”/performance_measurement/recurring_measurements”, jossa sijaitsee mittaamisjärjestelmän toistuvien mittauksien taulukko. Tällöin konfiguraationhallinta päivittää mittaamisjärjestelmän konfiguraatiotiedoston uudella mittausobjektilla ja luo samalla mittaukselle oman yksilöllisen tunnisteen. Mittaamisjärjestelmä aloittaa mittaukseen liittyvät toimenpiteet luettuaan päivitettyä tiedostoa. Vastauksena komentoon käyttäjä saa toimenpiteen onnistuessa OK-koodin ja lisäämänsä mittauksen tiedot, mukaan lukien automaattisesti luodun tunnisteen.


```
{
    "meas_type": "rtt_and_ploss",
    "active": true,
    "src_ip": "21.0.0.1",
    "dst_ip": "21.0.0.2",
    "packet_size": 100,
    "count": 5,
    "packet_interval": 1,
    "tos": 8,
    "timeout": 30,
    "dont_fragment": false,
    "interval": 7200,
    "start_time": "2020-03-01T12:00:00",
    "end_time": "2020-03-02T00:00:00"
}
```


Kuva 11. Toistuvan mittauksen objekti parametreineen

Kertamittauksen luominen toimii samalla periaatteella, mutta POST-komento lähetetään puolestaan operaatiopolun kertamittaus-alipolkuun ”/adhoc_performance_measurement”. Tällöin REST-palvelin lähettää mittausobjektin parametreineen mittaamisjärjestelmään socket-yhteyden kautta. Vastauksena komentoon käyttäjä saa toimenpiteen onnistuessa OK-koodin lisäksi mittauksen tulokset.

Myös mittaustulosten poistaminen tapahtuu POST-komennolla. Tulosten poistamista koskevat parametrit lähetetään operaatiopolun alipolkuun ”/clear_measurement_results”. Tällöinkin yhteydenpito mittaamisjärjestelmään tapahtuu socket-yhteyden kautta ja vastauksena komentoon käyttäjä saa toimenpiteen onnistuessa OK-koodin.

5.1.2 GET

GET-komennolla pyydetään palvelinta lähettämään osoitepolussa olevat tiedot [25]. Käyttäjä ei koskaan lähetä GET-komennon mukana dataa. Esimerkiksi lähettämällä GET-komennon toistuvien mittauksien polkuun saa vastauksena taulukkoon konfiguroidut mittausobjektit.

Mittaustuloksia voi myös tarkastella GET-komennon avulla. Komento lähetetään tilakyselypolun alipolkuun, jonka muoto riippuu siitä, mitä tuloksia halutaan tarkastella. Mahdollisia alipolkuja ovat esimerkiksi:

- ”/performance_measurement”, kaikki tulokset
- ”/performance_measurement/rtt_and_ploss”, kaikki tulokset, joiden mittaustyyppi on latenssi ja pakettihäviö
- ”/performance_measurement/jitter_and_ploss/abc123def4567890”, tulokset mittauksesta, jonka tunniste on abc123def4567890 ja mittaustyyppi jitter ja pakettihäviö

5.1.3 PUT

PUT-komennolla pyydetään palvelinta tallentamaan pyynnön mukana tullut data osoitepolussa olevaan kohteeseen ja korvaamaan mahdollinen kohteessa jo oleva data [25]. Komennon suurin ero POST-komentoon on juuri aiemman sisällön poistaminen uuden tieltä. Esimerkiksi jos toistuva mittaus luodaan lähettämällä uusi mittausobjekti PUT-komennolla POST-komennon sijaan, niin aiemmat taulukossa mahdollisesti olleet mittausobjektit katoavat ja jäljelle jää vain uusi mittausobjekti.

5.1.4 DELETE

DELETE-komennolla pyydetään palvelinta poistamaan osoitepolussa oleva kohde [25]. GET-komennon mukaisesti myöskään tämän komennon mukana käyttäjä ei lähetä dataa. Esimerkiksi aiemmin konfiguroidun toistuvan mittauksen tunnisteella ”1234abcd5678efgh” poistaminen tapahtuu lähettämällä DELETE-komennon toistuvien mittauksien polkuun

”/performance_measurement/recurring_measurements/1234abcd5678efgh”, jolloin vastauksena komentoon käyttäjä saa OK-koodin ja ”recurring_measurements”-taulukon nykyisen sisällön.

5.2 JSON Schema

Kun käyttäjä yrittää luoda uuden mittauksen REST-palvelimen kautta, tarkastetaan käyttäjän tekemän mittausobjektin oikeellisuus ennen sen lähettämistä eteenpäin mittaamisjärjestelmälle. Tällä toimenpiteellä ennaltaehkäistään mahdolliset käyttäjän tekemät virheet esimerkiksi mittausparametreissa ja estetään myös mittaamisjärjestelmän virheellinen toiminta, kun sille syötettävät tiedot noudattavat aina samaa rakennetta. Mittausobjektia verrataan mittauksen toistuvuudesta riippuen joko kertaluontoisen tai toistuvan mittauksen JSON Schemaan. JSON Schema toimii eräänlaisena muottina JSON-tiedostolle. Se mahdollistaa tiedoston rakenteen ja sisällön tarkemman esittelyn ja validoimisen [26]. Käyttäjä kykenee scheman perusteella rakentamaan järjestelmän vaatiman JSON-muotoisen mittausobjektin, kun taas laite pystyy sen avulla validoimaan muun muassa objektin rakenteen, parametrien tietomuodon ja sisällön.

Mittausparametrit on esitelty ja määritelty tarkasti mittausjärjestelmän JSON Schemoissa. Parametreille on aina määritelty vähintään kuvaus ja tietomuoto; esimerkiksi kuvan 12 tapauksessa tietomuotona on ”integer”, joka tarkoittaa kokonaislukua. Tietomuodosta riippuen parametreilla on myös määritelty tarkentavia ominaisuuksia, kuten kuvan 12 tapauksessa lähetettävien pakettien määrän vähimmäis- ja enimmäisarvot. Jos parametri on tietomuodoltaan ”string” eli merkkijono, niin tarkentavina ominaisuuksina voi määrittää esimerkiksi formaatin, kuten IPv4-osoite tai päivämäärä.

```
"count": {
  "description": "Number of packets sent per one measurement cycle",
  "type": "integer",
  "minimum": 1,
  "maximum": 100
}
```

Kuva 12. Lähetettävien pakettien määrä -parametrin esittely ja määrittely JSON Schemassa

Mittaustyypeille on annettu tarkat ”enum”-arvot, jolloin mittaustyypiksi ei voi asettaa muita arvoja kuin kuvan 13 tapaisesti ”enum”-kentässä määritellyt arvot. Schemoissa on lisäksi ”required”-taulukot, joissa määritellään pakolliset parametrit. Schemaa vasten validoiminen

palauttaa virheilmoituksen, jos jokin pakollinen parametri puuttuu. Pakollisista parametreista on kerrottu tämän opinnäytetyön kohdassa 3.3. Muiden kuin schemoissa esiteltyjen parametrien lisääminen palauttaa myös virheilmoituksen, koska ylimääräisiä parametreja ei ole sallittu.

```
"meas_type": {
  "description": "Type of measurement to be run",
  "description": "Valid values are: rtt_and_ploss, tracepath, jitter_and_ploss, throughput",
  "type": "string",
  "enum": ["rtt_and_ploss", "tracepath", "jitter_and_ploss", "throughput"]
}
```

Kuva 13. Mittaustyypin esittely ja määrittely JSON Schemassa

6 Yhteenveto

Opinnäytetyön aiheena oli suunnitella ja toteuttaa IP-verkon yhteyksien suorituskykyä mittaava järjestelmä. Järjestelmän suorittamat mittaukset eivät saisi häiritä verkon normaalia toimintaa. Järjestelmä sulautui osaksi taktisen runkoverkon olemassa olevia ohjelmistokomponentteja.

Mittaamisjärjestelmän toteutus onnistui suurimmilta osin toimeksiantajan vaatimusten mukaisesti. Yksi alkuperäisten vaatimusten mukaisista mittaustyypeistä jouduttiin jättämään suunnitteluvaiheessa pois aikataulun rajallisuuden vuoksi. Muut mittaustyypit ja niihin vaaditut parametrit ja ominaisuudet saatiin toteutettua vaatimusten mukaisesti.

Mittaamisjärjestelmä mahdollistaa välittömästi suoritettavat kertaluontoiset mittaukset sekä ajastettavat toistuvat mittaukset. Kertaluontoisten mittausten tulokset palautetaan käyttäjälle suoraan mittauksen jälkeen. Toistuvien mittauksien tulokset tallennetaan, jolloin käyttäjä voi lukea tuloksia koska tahansa haluamanaan ajankohtana. Toistuvien mittauksien tulosten poistaminen on myös mahdollista, ja ne myös poistuvat automaattisesti, kun tulosten määrä ylittää sille annetun enimmäisrajan.

Toteutus tehtiin suurimmalta osin käyttäen Python-ohjelmointikieltä, jonka käyttämisestä ei ennen työn suorittamista ollut lainkaan kokemusta. Python kuitenkin osoittautui varsin käteväksi ja helposti omaksuttavaksi kieleksi. Koska mittaamisjärjestelmä tuli osaksi valmiita ohjelmistokomponentteja, tuli järjestelmää toteutettaessa ymmärtää myös sitä ympäröivän ohjelmiston toimintaa. Monta vuotta kehitetyn järjestelmän olennaisten osien sisäistäminen mittaamisjärjestelmän toiminnan kannalta osoittautuikin odotetun haastavaksi mutta lopulta palkitsevaksi.

Mittaamisjärjestelmää voisi kehittää jatkossa älykkäämmäksi. Etenkin toistuvien mittauksien aikana järjestelmä voisi havaita mittauksen kohteena olevan yhteyden laadun heikkenemisen vertaamalla tuloksia toisiinsa. Myös kertamittauksen tuloksista voisi tallentaa esimerkiksi keskiarvoa, jolloin käyttäjä saisi välittömästi tiedon, jos saatu mittaustulos on keskimääräistä huonompi tai parempi. Tällä hetkellä järjestelmä vain suorittaa mittaukset ja tallentaa tai palauttaa tulokset, eikä ota itse niihin mitään kantaa.

Lähteet

1 Tietoa yhtiöstä – Bittium, haettu 27.1.2020, osoitteesta: https://www.bittium.com/bittium-lyhyesti/tietoa-ja-taloudellisia-lukuja/tietoa-yhtiosta

2 2018 – Bittium Oyj:n taktinen runkoverkko osaksi Itävallan puolustusvoimien taktisen tiedonsiirron järjestelmää – Bittium, haettu 4.2.2020, osoitteesta: https://www.bittium.com/index.php?id=2003&locate=PRM%2F2018%2F3144959

3 What is the Internet Protocol? | Cloudflare, haettu 5.3.2020, osoitteesta: https://www.cloudflare.com/learning/ddos/glossary/internet-protocol/

4 RFC 791 – Internet Protocol, haettu 5.3.2020, osoitteesta: https://tools.ietf.org/html/rfc791

5 RFC 1180 – TCP/IP tutorial, haettu 12.3.2020, osoitteesta: https://tools.ietf.org/html/rfc1180

6 RFC 1122 – Requirements for Internet Hosts – Communication Layers, haettu 12.3.2020, osoitteesta: https://tools.ietf.org/html/rfc1122

7 Link Layer – TCP/IP Model, haettu 16.3.2020, osoitteesta: http://www.codesandtutorials.com/networking/tcpipmodel/link_layer.php

8 1.6 Encapsulation | TCP/IP Illustrated, Vol. 1: The Protocols (Addison-Wesley Professional Computing Series), haettu 1.4.2020, osoitteesta: https://flylib.com/books/en/3.223.1.18/1/

9 What Causes Packet Loss? – Packet Loss Test, haettu 3.4.2020, osoitteesta: https://packetlosstest.com/causes

10 Jitter – Packet Loss Test, haettu 3.4.2020, osoitteesta: https://packetlosstest.com/jitter

11 Bandwidth vs. Throughput | Study.com, haettu 7.4.2020, osoitteesta: https://study.com/academy/lesson/bandwidth-vs-throughput.html

12 iPerf – iPerf3 and iPerf2 user documentation, haettu 4.2.2020, osoitteesta: https://iperf.fr/iperf-doc.php

13 ping(8) – Linux man page, haettu 5.2.2020, osoitteesta : https://linux.die.net/man/8/ping

14 RFC 792 – Internet Control Message Protocol, haettu 5.2.2020, osoitteesta: https://tools.ietf.org/html/rfc792

15 traceroute(8) – Linux man page, haettu 5.2.2020, osoitteesta : https://linux.die.net/man/8/traceroute

16 iPerf – The TCP, UDP and STCP network bandwidth measurement tool, haettu 6.2.2020, osoitteesta: https://iperf.fr/

17 iperf3 FAQ – iperf3 3.7 documentation, haettu 6.2.2020, osoitteesta: https://software.es.net/iperf/faq.html

18 daemonize(1): run program as Unix daemon – Linux man page, haettu 12.2.2020, osoitteesta: https://linux.die.net/man/1/daemonize

19 unix(7) – Linux manual page, haettu 24.3.2020, osoitteesta: http://man7.org/linux/man-pages/man7/unix.7.html

20 pthreads(7) – Linux manual page, haettu 24.3.2020, osoitteesta: http://man7.org/linux/man-pages/man7/pthreads.7.html

21 9. Classes – Python 3.8.2rc1 documentation, haettu 12.2.2020, osoitteesta: https://docs.python.org/3/tutorial/classes.html

22 queue – A synchronized queue class – Python 3.8.2rc1 documentation, haettu 18.2.2020, osoitteesta: https://docs.python.org/3/library/queue.html

23 16.2 threading – Higher-level threading interface – Python 2.7.17 documentation, haettu 20.2.2020, osoitteesta: https://docs.python.org/2/library/threading.html#lock-objects

24 What is REST – Learn to create timeless REST APIs, haettu 24.3.2020, osoitteesta: https://restfulapi.net/

25 HTTP/1.1: Method Definitions, haettu 3.3.2020, osoitteesta: https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html

26 JSON Schema | The home of JSON Schema, haettu 4.3.2020, osoitteesta: https://json-schema.org/