TAMPEREEN
AMMATTIKORKEAKOULU

Reitittimen tunneliläpäisytesti

Iiro Hellman

Opinnäytetyö
1.10.2018
Tieto- ja viestintätekniikan koulutus
Tietoliikennetekniikka ja tietoverkot

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tieto- ja viestintätekniikan koulutus
Tietoliikennetekniikka ja tietoverkot

Iiro Hellman:
Reitittimen tunneliläpäisytesti

Opinnäytetyö 29 sivua, joista liitteitä 3 sivua
1.10.2018

Tässä opinnäytetyössä luotiin testausympäristö, jossa voidaan testata reitittimen tunnelien läpäisykykyä MPLS-verkossa. Työssä keskityttiin luomaan ratkaisu, jolla saadaan testaus tehtyä alusta loppuun. Tavoitteena oli toteuttaa työkalua varten serveri, reitityksen ja verkkorajapintojen konfiguraatio sekä mahdollisuus testauksen jatkokehitykseen.

Työssä luotiin verkkotopologia, jolla tämän tyyppinen tunnelien testaus onnistuu. Testattava laite voi sijaita missä tahansa, kunhan se on MPLS-saavutettavissa. Serveri toteutettiin yhdellä laitteella, ja tässä työssä esitellään sen toteutus, joka sisältää serverin verkkokonfiguraation ja työssä käytettyjen virtuaalikoneiden ratkaisut.

Testaus suoritetaan serveriltä käyttäen Iperf3-työkalua, jolla voidaan generoida dataa ja mitata siirtonopeutta. Iperf3:lla ajetaan testi, joka kulkee MPLS:n yli testattavalle reitittimelle tunnelien läpi, ja testaa näin tunnelien läpäisykykyä. Lopputuloksena on toimiva toteutus, jolla voidaan testata tunnelien läpäisykykyä, ja työkalu, jota voidaan hyväksi käyttää vian selvityksessä.

MPLS Multiprotocol Label Switching

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
ICT Engineering
Telecomminications and Networks

Iiro Hellman
Router GRE tunnel throughput test

Bachelor's thesis 29 pages, appendices 3 pages
October 2018

The purpose of this thesis was to create testing environment which can test router tunnel throughput in MPLS network. The thesis focused creating solution to make testing in end to end principle. Goal was to create necessary blocks for test. Including creating server, routing and interface configuration and possibility for further development.

In this thesis was created network topology on which this kind tunnel testing succeed. Device for testing can be located at anywhere once it is reached by MPLS. Server was accomplish on one device and in this thesis is told how. This include server's network configuration and conclusion of using virtual machines.

The testing is executed from server by using Iperf3 tool. This tool can generate data and measure bandwidth of the data steam. Executing test with Iperf3 over MPLS and to router with tunnels, test will measure throughput of the tunnels. Outcome of the thesis is working implementation and tool which can test tunnel throughput and tool which can be used for problem management.

MPLS Multiprotocol Label Switching

SISÄLLYS

LYHENTEET JA TERMIT

IP	Internet Protocol
	Internet-kerroksen protokolla
KVM	Kernel-based Virtual Machine
	Kerneliin rakennettu virtualisointi tuki
MPLS	Multiprotocol Label Switching
	Runkoverkon päälle rakennettu pakettien välitys menetelmä
point-to-point	Kuvaa yhteyttä kahden solmun tai päätepisteen välissä
TCP	Transmission Control Protoco
	TCP tietoliikenneprotokolla
UDP	User Datagram Protocol
	UDP tietoliikenneprotokolla
VLAN	Virtual LAN
	Virtuaalilähiverkko
VPN	Virtual private network
	Virtuaalinen erillisverkko
VRF	Virtual routing and forwarding
	Yksityinen reititysinstanssi

1 JOHDANTO

Tässä opinnäytetyössä tuotettiin verkonhallinnalle reitittimen tunnelin läpäisykykyä MPLS:n yli mittaava työkalu. Ratkaisu sisältää serverin, verkkotopologian ja eri laitteiden konfiguraation. Opinnäytetyössä käydään läpi eri komponentit, ja miten ja miksi näin on tehty.

Opinnäytetyön tavoitteet tulivat Telialta, jonne opinnäytetyön toteutus tehtiin. Tavoitteena toimivan testikokonaisuuden lisäksi on tuottaa mahdollisuus kerätä dataa eri laitteiden tunnelien läpäisykyvystä ja mahdollisuus käyttää työkalua vianselvitykseen. Osa näistä tavoitteista on opinnäytetyöntyön valmistumisen jälkeisiä vaiheita kuten datan kerääminen ja työkalun jatkokehitys.

Opinnäytetyössä on esitelty tyokalun kannalta olennaisimmat asiat ja todennettu sen toimivuus. Telian verkkoympäristö, missä työ on toteutettu, on laaja ja kompleksinen. Huomioon piti myös ottaa myös yritykseen liittyviä asioita, joita ei voitu tuoda tähän dokumenttiin. Tästä syystä kaikkia työhön liittyviä elementtejä ei ole voitu tässä opinnäytetyössä esitellä. Työssä olevat nimet ja osoitteet ovat korvattu todellisista yleiskielelliseksi viittauksiksi tai ovat keksittyjä.

2 TYÖN TAVOITTEET

Työn tärkein tavoite oli luoda toimiva testaustapa tunnelien läpäisykyvyn mittaamiseksi. Pyydettäessä voidaan testata yritysasiakkaiden yhteyttä, jotka voi olla vaikea pääsyisessä paikassa. Testin pitää olla helposti ajettavissa ja jatkokehitettävissä. Telia asetti työlle seuraavat vaatimukset:

- serverin pystyttämien
- konfiguraation luominen serverille
- rajapintojen, reitityksen ja tunnelirajapintojen konfiguraation luominen
- testin ajaminen yksinkertaisella tavalla
- ohjeistuksen luominen Telian sisäiseen käyttöön
- mahdollisuus jatkokehitykseen

Osaa näistä tavoitteista ei voida tässä dokumentissa käsitellä tai näyttää, koska ne ovat Telian sisäiseen käyttöön. Olennaiset asiat testin toimisen kannalta on esitelty tässä dokumentissa.

3 TEORIA

Tässä käydään lyhyesti läpi muutamia tekniikoita, mitä työssä käytetään. Koska tähän opinnäytetyöhön liittyy monia verkkoteknisiä asioita, käydään läpi vain yleisimmät termit ja niiden merkitys. Työssä esitellään ne tekniikat, jotka useimmin toistuvat tämän dokumentin sisällä.

3.1 MPLS

MPLS (Multiprotocol Label Switching) on tekniikka, jolla kuljetetaan dataa kuten IP-paketteja suoraa metroverkossa olevien solmujen kautta. Kuljettaminen toimii ilman solmujen välistä reititystä. Reititys tapahtuu ensimmäisellä laiteella, joka etsii lopullisen päämäärän, ja lähettää paketin MPLS yli suoraa päämäärän käyttäen MPLS-leimoja.

3.2 GRE tunneli

GRE (Generic Routing Encapsulation) on tunnelointi protokolla, jolla voidaan luoda virtuaalisia point-to-point -yhteyksiä IP-verkkojen ylitse. GRE-tunnelit ovat Cisco Systemsin kehittämiä, mutta lähes kaikki nykyaikaiset reitittimet tukevat sitä. Tässä työssä käytetään vain yleiskielellisesti tunneli-sanaa, jolla tarkoitetaan GRE-tunneleita.

3.3 TCP

TCP (Transmission Control Protocol) on tietoliikenneprotokolla, joka kuuluu toisen kuljetuskerroksen protokolliin. TCP muodostaa yhteyden kättelyn jälkeen laitteiden välille, ja muodostaa kaksisuuntaisen datan välitysväylän. TCP on yhteydellinen protolla ja takaa luotettavan tiedon siirron, mutta ei sovellu kovin hyvin esimerkiksi multimedian siirtoon. TCP ominaisuuksiin kuuluu myös se, että se tekee tarkistuksen ja paketti saapuu perille ehjänä tai pyytää uudelleenlähetystä.

3.4 UDP

UDP (User Datagram Protocol) kuten TCP kuuluu kuljetuskerroksen protokolliin. Toisin kuin TCP, se ei ole yhtä luotettava ja on yhteydetön protokolla. UDP itsessään ei käytä virheen korjausta, vaan tämä täytyy tehdä sovellustasolla.

3.5 VRF

VRF (virtual routing and forwarding) on teknologia, jolla voidaan tuoda useita reittitauluja yhden laitteelle. Jokainen reititysinstanssi on yksityinen, eli eri VRF:ssä olevat osoitteet ja reitit voi mennä päällekkäin. VRF sitoutuvat hyvin vahvasti yhteen MPLS kanssa.

3.6 Iperf

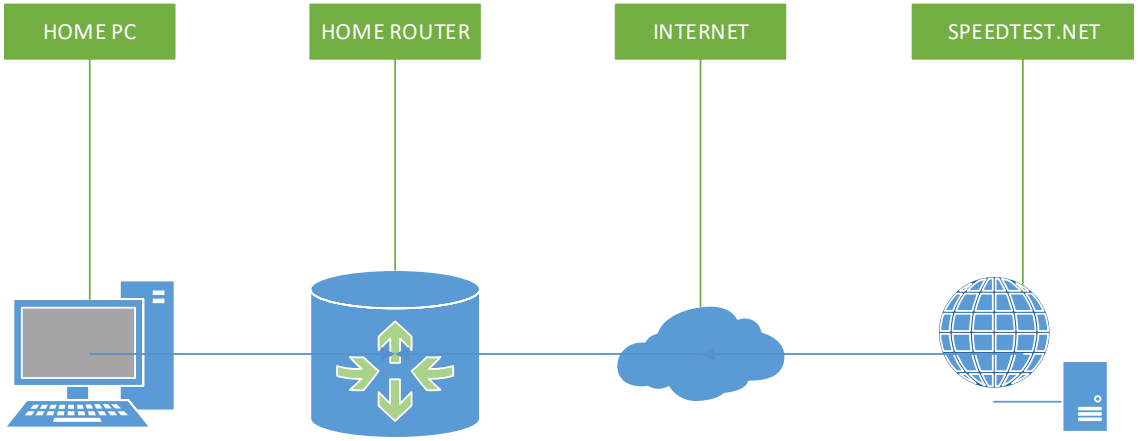
Iperf on verkon suorituskyvyn mittaamiseen kehitetty työkalu. Iperfista on kolme eri versiota Iperf1, Iperf2 ja Iperf3. Iperf pitää sisällään serveri- ja client-toiminnollisuudet, joilla ajetaan dataa joilla voidaan mitata kahden laitteen välistä liikenteen läpäisykykyä. Liikennettä voidaan ajaa yhteen tai molempiin suuntiin. Datavirta tuotetaan käyttäen joko TCP- tai UDP-protokollaa.

3.7 VPN

VPN (Virtual Private Network) on virtuaalinen erillisverkko, jossa kaksi tai useampi verkko voidaan yhdistää. Tässä työssä VPN tarkoittaa suljettua verkkoa, jossa liikennöidään vain VPN-verkon sisällä. VPN rinnastuu tässä opinnäytetyössä VRF:ään.

4 NOPEUSTESTIEN YLEINEN TOTEUTUS

Normaalissa verkon läpäisykyvyn testauksessa ajetaan liikennettä kahden päätepisteen välillä. Esimerkiksi käyttäjä voi testata internetinsä siirtonopeuden käyttäen verkosta löytyviä palveluita kuten internetistä löytyvä speedtest.net palvelua. Testi antaa viiveen, lataus- ja lähetysnopeuden serverin ja kotikoneen välillä. Tässä skenaariossa voidaan olettaa kapasiteetin internetin ja speedtest.net välisen verkkopalvelun olevan ääretön. Merkitseväksi kohdaksi jää kotireitittimen ja operaattorin kytkentäpaikan välinen yhteys, ja siinä käytetty teknologia, sekä operaattorin määrittelemä nopeus. Kuvassa 1 esitellään kuluttajan näkökulmasta internetyhteyden nopeuden testaamista.

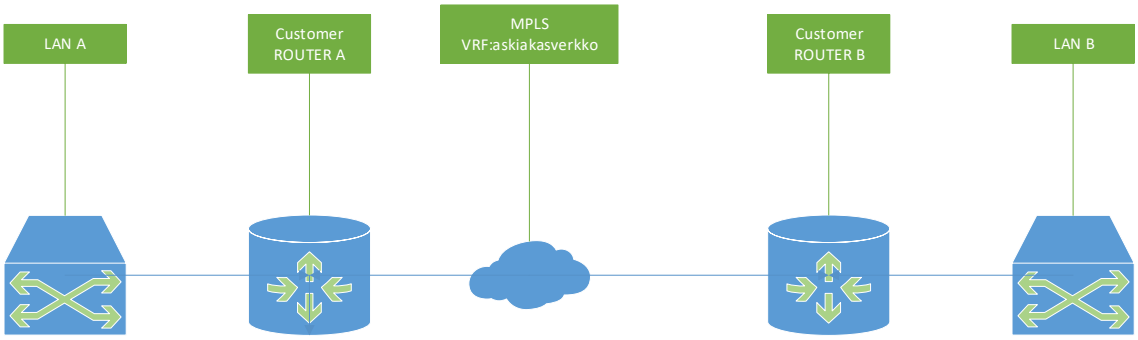


KUVA 1: Esimerkki kuluttajan suorittamasta nopeustestistä

4.1 Yritysverkot

Yrityksellä voi olla monta eri verkkoa ja liittymähäntiä saattaa olla muutamasta tuhansiin. Kyse voi olla yksittäisestä liittymästä, mikä ei ole yritysverkko vaan yhteydellä on rajoitettu pääsy internettiin. Tässä työssä käsitellään yrityksen verkkoja, jotka kulkevat, MPLS kautta yrityksen sisäisessä verkossa ja häntiä on useita. Osa verkoista voi olla täysin suljettuja ulkomaailmalta tai pääsy internettiin on yhden solmun kuten palomuurin kautta. Tässä opinnäytetyössä ei kuitenkaan ole tarkoitus ottaa kantaa internetin kautta kulkevan liikenteen kapasiteettiin, vaan yrityksenverkon häntäreitittimen läpäisykykyyn tunnelin ja MPLS läpi.

Yrityksen sisäverkkoa testatessa ajetaan liikennettä kahden päätepisteen välillä. Testaan kahden hännän välistä yhteyttä, hännät voivat olla eri paikkakuntien toimistolla tai konesalissa. Välillä voidaan ajaa liikennettä esimerkiksi käyttäen Iperf3:sta. Yrityksen sisäiset testit usein ajetaan kahden eri LAN-verkon välillä. Alla olevan esimerkkikuvan 2 mukaisesti voidaan ajaa LAN A ja B välillä liikennettä.



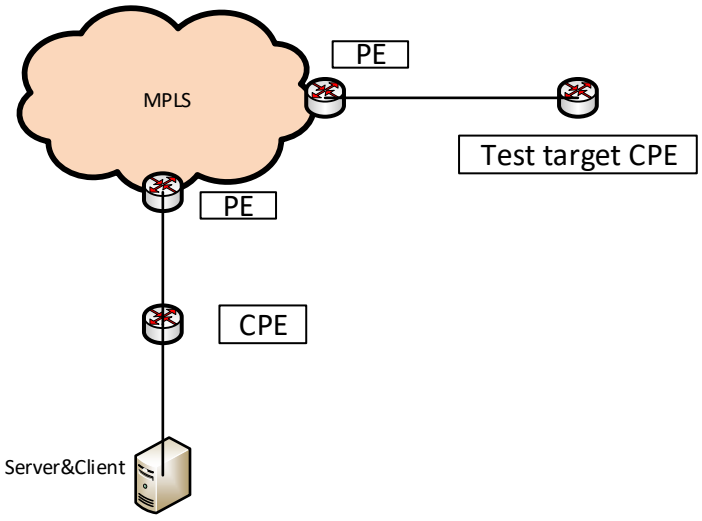
KUVA 2: Esimerkki yrityksen kahden eri toimipisteen välisestä yhteydestä

5 VERKON TOTEUTUS

Työssä piti kehittää tapa, jolla saadaan testattua tunnelien läpäisykykyä eri laitteille ja mahdollisuus testata asiakkaalle menevän yhteyden läpäisykykyä tunnelien avulla. Verkon toteutus on luotu siten, että nopeustestiä voidaan ajaa tunnelien läpi ja verkkojen sekaantumatta muihin asiakkaan MPLS verkkoihin. Testattava laite voi sijaita maantieteellisesti haastavassa paikassa tai paikassa mihin on vaikea pääsy.

5.1 Suunnittelu

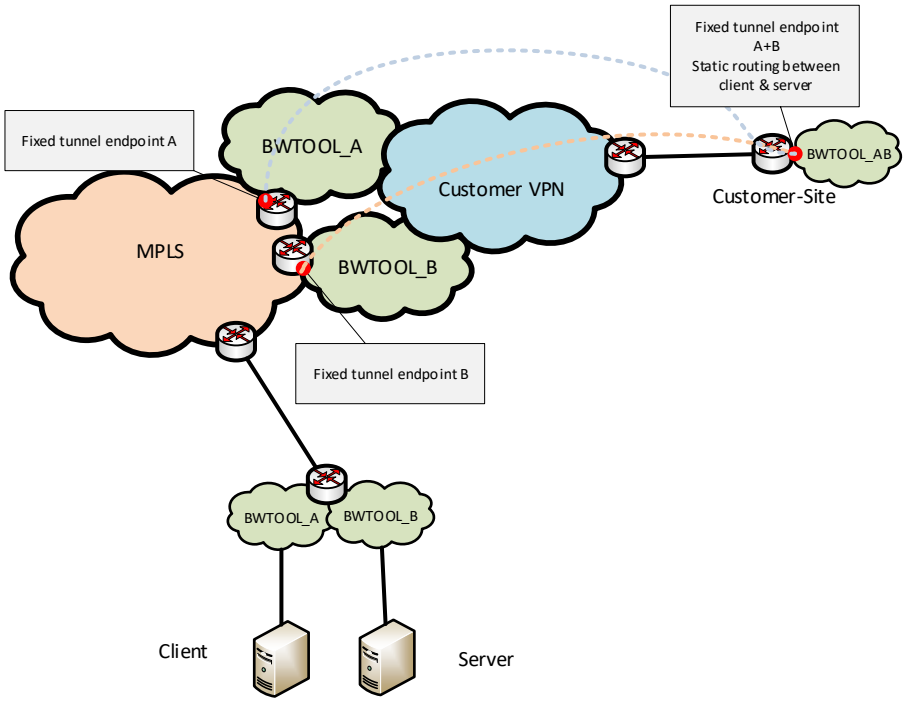
Suunnitteluvaiheessa piti miettiä, miten testiä voidaan ajaa annettujen speksien mukaisesti. Tässä vaiheessa kuvattiin toteutus yksinkertaisella kuvalla 3. Kuvassa 3 tulee esiin kaikki kohdat, jotka vaativat jonkin näköistä konfiguraatiota. Millaisella konfiguraatiolla ja verkkoteknisellä toteutuksella oli vielä auki. Toteutus lähti mallintumaan ja rakentumaan pala palalta, kun server- ja clienti -päädyn rakentaminen alkoi. Muutaman vuoden verkkoasiantuntijan kokemuksella ei varsinaista suunnittelua tarvinnut tämän jälkeen tehdä.



KUVA 3: Suunnittelu vaiheessa piirretty toimintakuvio

5.2 ”End to end” -verkkoratkaisu

Testin toteutus olisi helppoa, jos ajettaisiin liikennetestiä kahdelta eri toimipisteeltä ja välissä olisi tunnelit. Työhön piti kehittää verkkoratkaisu, jolla testi voidaan ajaa ilman erillistä asennusta, asiakkaan pyynnöstä. Tällaiseen toteutukseen piti kehittää uusi tapa ja tämä toteutus on esitettynä kuvassa 4.



KUVA 4: Toteutuksen verkkotopologia

Vaikka client ja serveri ovat fyysisesti samalla laitteella, verkkoteknisesti ne ovat erillään. Molemmat ovat omassa verkossaan eli VRF:issä BWTOOL_A ja BWTOOL_B. Verkot kulkevat serveriltä reitittimen kautta reunareitittimelle, jolloin niistä muodostuu erilliset suljetut virtuaaliset erillisverkot. Kun sama VRF tuodaan toiselle reunareitittimelle, tämä näkee reittitaulussaan esimerkiksi VRF BWTOOL_A -tapauksessa client-laitteen väliverkon ja client-laitteen LAN-verkon.

Testattavalta reitittimeltä on oltava joku olemassa oleva yhteys reunareitittimelle, sitä kuvaa kuvassa 4 Customer VPN. Tämän yhteyden päälle rakennetaan tunnelit BWTOOL_A ja BWTOOL_B -verkoista. Näitä tunneleita kuvassa 4 esittää katkoviivalla tehdyt viivat. Testattavalla reitittimellä nämä verkot yhdistetään samaan VRF:ään ja reititetään niin, että kulkeva liikenne voidaan kierrättää testattavan laitteen läpi. Näin saadaan kierrätettyä liikenne serveriltä clientille, niin että se kulkee tunnelien läpi testattavalle reitittimelle.

Kun oletetaan, että kapasiteetin on serveriltä ja clientiltä reunareitittimelle ääretön, jää rajoittavaksi osuudeksi vain yhteys testattavaan laitteeseen tunnelien läpi. Tällöin testistä saadaan tuloksena se, mitä tunneleista testattavalle reitittimelle menee läpi.

6 SERVERIN TOTEUTUS

Koko toteutus lähti liikkeelle serverin toteuttamisesta. Serveriä rakentaessa selvisi nopeasti, miten testikokonaisuus rakentuu serverin perään. Serverin rakentamiseen annettu laite ja vaatimus, että toteutus on avoimeen lähdekoodiin perustuva.

6.1 Laitteisto

Serverinä työssä pöytätietokone, jossa on 10 Gbit verkkokortti. Ajettavan testin kannalta mahdollisimman ison verkkokortin lisäksi ei ollut muita vaatimuksia. Testiä ajetaan Iperf3:lla, joka generoi TCP- tai UDP-liikennettä, ja tällaisen datan generoimiseen ei vaadita prosessorin tai muistin tehoa.

6.2 Käyttöjärjestelmä

Käytin työssä Linuxsia, Debian pohjaista jakeluversiota Ubuntu16.04, pääalustana. Tämän lisäksi käytössä oli virtuaalikoneita, joissa ajetaan Lubuntu 16.04. Molemmissa tapauksissa käytettiin 64-bittistä arkkitehtuuria.

Linux-pohjainen käyttöjärjestelmä oli soveltuvin työhön sen avoimen lähdekoodin ja muokattavuuden takia. Jakeluversio Ubuntu on hyvin tuettu, ja sillä on laaja käyttäjäyhteisö, joten sille on hyvin ohjeita moniin tarpeisiin.

6.3 Verkkoratkaisut

Yksi kriteeri testin ajamiseen on saada mahdollisimman paljon liikennettä lähtemään testiserveriltä, jotta testityökalulla voidaan testata mahdollisimman montaa kohdetta. Siksi vaaditaan rautatasolta verkkokortti, jolla on paljon suorituskykyä ja läpäisykykyä. Testiserveri on 10 Gb tiedonsiirtonopeuteen pystyvä verkkokortti, joka liitäntänä toimii RJ45.

Käyttöjärjestelmä Linux itsessään toimii verkonlaitteena juuri niin kuin se on konfiguroitu. Tästä kertoo paljon se, että useimmat verkonlaitteet nykyään perustuvat Unix-pohjaisiin ratkaisuihin. Linuxin voi konfiguroida olemaan palomuuri, reitityspiste tai se voi toimia kytkimenä. Linuxin verkko-ominaisuuksien tuki myös kehittyy jatkuvasti ja laajenee.

6.3.1 Suunnittelu ja haasteet

Verkkokonfiguraatio serverillä on haluttu toteuttaa mahdollisimman yksinkertaisesti. Vaihtoehtoisia toteutustapoja olisi muutamia, mutta tässä toteutuksessa käytetään kolmea VLAN:a ja väliverkkoja. Perusteluita käyttää VLAN:ja on muutamia: serverin erilliseen hallintaverkkoon vienti sekä serveri- ja client-verkkojen erottelu. Näin saadaan eroteltua kaikki omiin verkkoihinsa.

Ongelmaksi nousi, että kaikki rajapinnat nousevat serverillä samaan reittitauluun. Liikennettä on tarkoitus kierrättää serveriltä ulos ja takaisin, mutta verkkojen ollessa samassa reittitaulussa ne tuntevat toisensa ja liikenne kulkee käyttäen omaa reittitauluaan suoraan VLAN:sta toiseen. Sama ongelma olisi koitunut mitä tahansa rajapintaa käyttäen. Ongelma on ratkaistu käyttäen VRF:ejä. VRF tuki on tullut Linuxiin Kernel 4.3 lähtien, ja on suhteellisen uusi ominaisuus mutta toimiva. VRF:llä saadaan eroteltua VLAN:it eri reittitauluihin ja mahdollistettua testin suorittamien yhdellä laitteella.

VLAN-tuki ei ole suoraa sisällytetty Ubuntu 16.04 -vakiopaketteihin, joten tämä vaati oman paketti kokonaisuuden lataamista ja asentamista. Asentaminen Ubuntulla onnistuu helposti ja tapahtuu seuraavavilla syntakseilla. Tämän lisäksi VLAN-moduuli pitää lisätä /etc/modules, jotta konfiguraatio on toimiva uudelleenkäynnistyksen jälkeen.

```
sudo apt-get install vlan
sudo su -c 'echo "8021q" >> /etc/modules'
```

6.3.2 Konfiguraatio

Konfiguraation toteuttamiseen on kaksi tapaa. Antaa suorina komentoina terminaaliin käskyt tehdä asia esim. VLAN:in lisäys, tai tehdä konfiguraatio interface-tiedostoon. Kokeiluun ja testaamiseen terminaalilta käskytys toimii hyvin, mutta pysyvänä ratkaisuna se ei toimi. Tehdyt muutokset katoavat serverin uudelleenkäynnistyksen myötä.

Lopullinen ja pysyvä konfiguraatio tehdään */etc/network/interfaces* tiedostoon. Serverin käynnistyessä se lataa verkkokonfiguraation tämän tiedoston mukaan. Tämä vaati uuden syntaksin opettelua ja tutkimista, jotta konfiguraation pystyi toteuttamaan *interfaces* tiedostoa käyttäen.

```
#/etc/network/interfaces
auto lo
iface lo inet loopback

#10Gb Network Adapter
auto eth0 eth0.10 eth.20 eth.30

#VLAN 10 – MGMT
iface eth0.10 inet static
        address 192.168.10.2
        netmask 255.255.255.252
        vlan_raw_device eth0

#VLAN 20 – BWTOOL_A
iface eth0.20 inet static
        address 192.168.20.2
        netmask 255.255.255.252
        vlan_raw_device eth0

#VLAN 30 – BWTOOL_B
iface eth0.20 inet static
        address 192.168.30.2
        netmask 255.255.255.252
        vlan_raw_device eth0

#VRF binding
up ip link set dev eth0.20 master bwtool_a
up ip link set dev eth0.30 master bwtool_b

#Routes
up ip route add 0.0.0.0/0 via 192.168.10.1 dev eth0.10
up ip route add 0.0.0.0/0 via 192.168.10.1 table 1 dev eth0.20
up ip route add 0.0.0.0/0 via 192.168.10.1 table 2 dev eth0.30
```

Tässä esitellään esimerkki konfiguraatio serverin *interface*-tiedostoon. Serverin käynnistyessä se asettaa verkon asetukset kuten yllä on määrätty. Syntaksi up lisää reitit ja VRF:n käynnistyksen yhteydessä.

6.4 Virtuaalikoneet

VRF-tuen ollessa vielä Linuxille puutteellinen piti työssä käyttää virtuaalikoneita. Näin Iperf pystytään ajamaan VRF alla olevista VLAN:sta. Virtuaalikoneet ajetaan suoraa Linuxin ytimeen rakennettua KVM (Kernel-based Virtual Machine) -järjestelmää käyttäen. Vaihtoehtoisia virtualisointi platformeja on muutamia muun muassa VMware ja Virtualbox, mutta KVM soveltui käyttötarkoitukseen parhaiten. KVM hienous on, että sillä voidaan komentaa virtuaalikoneita suoraa terminaalin kautta. Samoin terminaalista hyppääminen virtuaalikoneelle onnistuu suoraan virtualisoidulla konsoli yhteydellä. Näin ollen graafista käyttöliittymää ei tarvita serverin tai virtuaalikoneen käyttöön.

6.4.1 Verkkoratkaisu

KVM voidaan tuoda verkko suoraan käyttäen passthrough-tekniikka. Tämä tarkoittaa sitä, että se lukee serverin interfacen suoraa omaksi verkkokortikseen. Näin ollen VRF oleva VLAN on virtuaalikoneen globaalilla tasolla, jolloin ipef3 ajaminen onnistuu.

Virtuaalikone 1 tuotiin VRF bwtool_a kuuluva VLAN 20 ja virtuaalikone 2 VRF bwtool_b VLAN 30. Molemmille virtuaalikoneille asetettiin VLAN:in mukainen IP-osoite. Nämä voitiin määrittää suoraa Lubuntun graafisessa käyttöliittymässä Network Managerin kautta taulukon 1 mukaan.

Virtuaalikone	IP-osoite	Maski	Yhdyskäytävä
1	192.168.20.2	255.255.255.252	192.168.20.1
2	192.168.30.2	255.255.255.252	192.168.30.1

TAULUKKO 1: Virtuaalikoneiden osoitteet

7 REITITIN JA TUNNELI KONFIGURAATIOT

Konfiguraatiot luotiin siten, että ne voidaan suoraa kopioida reunareitittimelle ja testaavalle reitittimelle. Koska käytössä on eri valmistajien laitteita, on konfiguraatiot luotu yleisemmin käytössä olevia laitteita ajatellen. Toimivaan kokonaisuuteen vaaditaan kaikkien pisteiden konfiguraatio, mikä on esitelty kuvassa 4. Jokaisella laitteella on tietynlainen pohja ja kustomoitu konfiguraatio ennestään, mihin tässä työssä ei oteta kantaa. Työlle jäi tehtäväksi rakentaa rajapintakonfiguraatio eli tunnelit ja tarvittavat reititykset.

Konfiguraation on yksinkertainen ja saadaan toimimaan vaihtamalla muutama muuttuja. Tunnelikonfiguraatio muuttujat on asetettava kohdelaitteen IP-osoitteiden mukaan. Kuten kuvassa 4 esitellään rakentuvat tunnelit olemassa olevan verkon päälle. Nämä osoitteet saadaan testattavan laitteen väliverkon osoitteista. Tämän lisäksi reunareitittimen puolella vaaditaan globaalin tason VRF.

7.1 Reunareitin konfiguraatio

Tässä esitellään reunareitittimen konfiguraatiot. Konfiguraatio on luotu Cisco Networks ja Nokia Networks valmistajien laiteille. Konfiguraatio pitää sisällään tunnelit ja tarvittavan reitityksen.

7.1.1 Nokia Networks

```
//////BWTOOL_A/////
 configure service vprn <Global vrf>
interface "BWTOOL_A_lo" create
address 192.168.50.29/30
sap tunnel-1.public:998 create
exit
exit

configure service vprn BWTOOL_A
auto-bind ldp
```

interface "BWTOOL_A" tunnel create
address 192.168.50.26/30
ip-mtu 1506
sap tunnel-1.private:998 create
ip-tunnel "BWTOOL_A" create
dest-ip 192.168.50.25
gre-header
source 192.168.50.30
remote-ip <wan ip cpe>
delivery-service <global VRRP service-id>
no shutdown
exit
exit
exit
exit

static-route 192.168.20.0/30 next-hop 192.168.50.25 cpe-check 192.168.50.25

//////BWTOOL_B/////

configure service vprn <Global vrf>
interface " BWTOOL_B_lo" create
address 192.168.50.37/30
sap tunnel-1.public:999 create
exit
exit

configure service vprn BWTOOL_B
auto-bind ldp
interface "BWTOOL_B" tunnel create
address 192.168.50.34/30
ip-mtu 1506
sap tunnel-1.private:999 create
ip-tunnel "BWTOOL_1" create
dest-ip 192.168.50.33

gre-header
source 192.168.50.38
remote-ip <wan ip cpe>
delivery-service <global VRRP service-id>
no shutdown
exit
exit
exit
exit

static-route 192.168.30.0/30 next-hop 192.168.50.33 cpe-check 192.168.50.33

7.1.2 Cisco Netwtoks

int lo BWTOOL_A_lo
ip vrf forwarding BWTOOL_A
ip address 192.168.50.29 255.255.255.252

interface Tunnel9999998
 description BWTOOL_A
 ip vrf forwarding BWTOOL_A
 ip address 192.168.50.26 255.255.255.252
 ip mtu 1500
 ip tcp adjust-mss 1380
 qos pre-classify
tunnel source vrf <source vrf>
 tunnel source <lo interface>
 tunnel destination 192.168.50.25

ip route vrf BWTOOL_A 192.168.20.0 255.255.255.252 192.168.50.25

int lo BWTOOL_B_lo
ip vrf forwarding BWTOOL_B
ip address 192.168.50.37 255.255.255.252

```
interface Tunnel9999999
description BWTOOL_B
ip vrf forwarding BWTOOL_B
ip address 192.168.50.33 255.255.255.252
ip mtu 1500
ip tcp adjust-mss 1380
qos pre-classify
tunnel source vrf <source vrf>
tunnel source < lo interface>
tunnel destination 192.168.50.33

ip route vrf BWTOOL_B 192.168.30.0 255.255.255.2552 192.168.50.33
```

7.2 Testikohde reitittimen konfiguraatio

Tässä esitellään reitittimen konfiguraatiot. Konfiguraatio on luotu Cisco ja Juniper valmistajien laiteille. Konfiguraatio pitää sisällään tunnelit ja tarvittavan reitityksen.

7.2.1 Cisco Networks

```
ip vrf bwtool_ab
rd 666:666

interface Tunnel9999998
description bwtool _tunnel_A
ip vrf forwarding bwtool_ab
ip address 192.168.50.25 255.255.255.252
ip mtu 1476
ip tcp adjust-mss 1380
qos pre-classify
tunnel source <wan interface>
tunnel destination 192.168.50.30
```

```
interface Tunnel9999999
description bwtool _tunnel_B
ip vrf forwarding bwtool_ab
ip address 192.168.50.33 255.255.255.252
ip mtu 1476
ip tcp adjust-mss 1380
qos pre-classify
tunnel source <wan interface>
tunnel destination 192.168.50.38

ip route 192.168.50.28 255.255.255.252 <WAN_PE>
ip route 192.168.50.36 255.255.255.252 <WAN_PE>
ip route vrf bwtool_ab 192.168.30.0 255.255.255.252 192.168.50.26
ip route vrf bwtool_ab 192.168.30.4 255.255.255.252 192.168.50.34
```

7.2.2 Juniper Networks

```
set interfaces gr-0/0/0 unit 998 description bwtool_ab_tunnel_A
set interfaces gr-0/0/0 unit 998 tunnel source <wan ip>
set interfaces gr-0/0/0 unit 998 tunnel destination 192.168.50.30
set interfaces gr-0/0/0 unit 998 family inet address 192.168.50.25/30
set interfaces gr-0/0/0 unit 998 copy-tos-to-outer-ip-header

set interfaces gr-0/0/0 unit 999 description bwtool_ab_tunnel_B
set interfaces gr-0/0/0 unit 999 tunnel source <wan ip>
set interfaces gr-0/0/0 unit 999 tunnel destination 192.168.50.38
set interfaces gr-0/0/0 unit 999 family inet address 192.168.50.33/30
set interfaces gr-0/0/0 unit 999 copy-tos-to-outer-ip-header

set routing-options static route 192.168.50.24/30 next-hop <WAN_PE>
set routing-options static route 192.168.50.24/30 no-readvertise

set routing-options static route 192.168.50.32/30 next-hop <WAN_PE>
```

set routing-options static route 192.168.50.32/30 no-readvertise

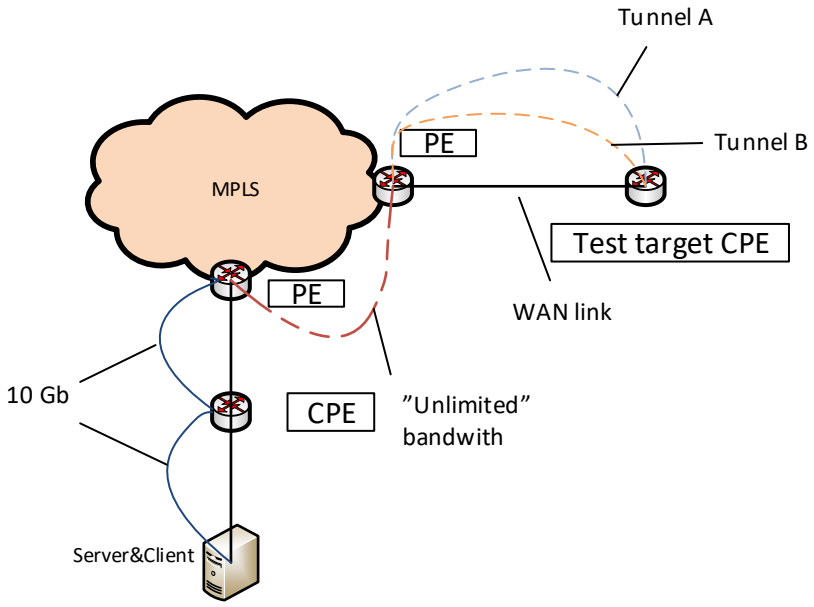
set routing-instances bwtool_ab instance-type virtual-router
set routing-instances bwtool_ab interface gr-0/0/0.998
set routing-instances bwtool_ab interface gr-0/0/0.999
set routing-instances bwtool_ab routing-options static route 192.168.30.0/30 next-hop 192.168.50.26
set routing-instances bwtool_ab routing-options static route 192.168.30.4/30 next-hop 192.168.50.34

8 LÄPÄISYKYVYN TESTAUS

Tunnelin läpäisykyvyn testaaminen voidaan suorittaa, kun luvun 7 konfiguraatiot on tehty. Testaus tapahtuu serverillä olevien virtuaalikoneita käyttäen. Kaikki testiin kuuluvat laitteet on etähallittavia ja fyysistä pääsyä ei mihinkään laitteeseen vaadita. Tämä tekee testin ajamisesta helpon ja mahdollistaa jatkokehityksen esimerkiksi automaation luomiseksi.

8.1 Toiminta

Testattavaan kohteeseen ajetaan Iperf3:lla TCP- tai UDP-pakettia serverin ja clientin välillä. Serverin ja clientin välinen yhteys reitittyy verkon lävitse, kuten luvussa 5 on kerrottu. Iperf3-serveri on aina päällä ja valmiudessa, kunnes client-puolisko käynnistetään. Kun client käynnistetään, se alkaa lähettämään dataa annettujen parametrien mukaan. Nämä Iperf3-parametrit on esiteltynä liitteessä 1. Mitattu tulos näyttää kuvan 5 mukaisten tunnelien A ja B läpäisykyvyn.



KUVA 5: Esitys vaikuttavista kaistojen nopeuksista ja testattavista tunneleista

8.2 Rajoittavuudet

Rajoittavia tekijöitä ei testin suorittamiseen ole työkalun puolelta. Serveri verkkokortti voisi olla yksi, mutta yli 10 Gb yhteydet ovat harvinaisia ja ne eivät ole tämän testin kohteita. Iperf3:sella on ohjelmalliset rajoitteet, mutta varsinaista teoreettista rajaa ei ole kerrottu. Verkossa käytyjen keskustelujen perusteella kapasiteetti riittää monen kertaisesti yli käytössä olevan 10 Gb:n. Kolmas on virtuaalikoneen ja käyttöjärjestelmän välinen looginen yhteys. Loogiselle yhteydelle ei teoreettista rajaa ole määritelty ja testeissä tämän ei ole huomattu vaikuttavan tulokseen

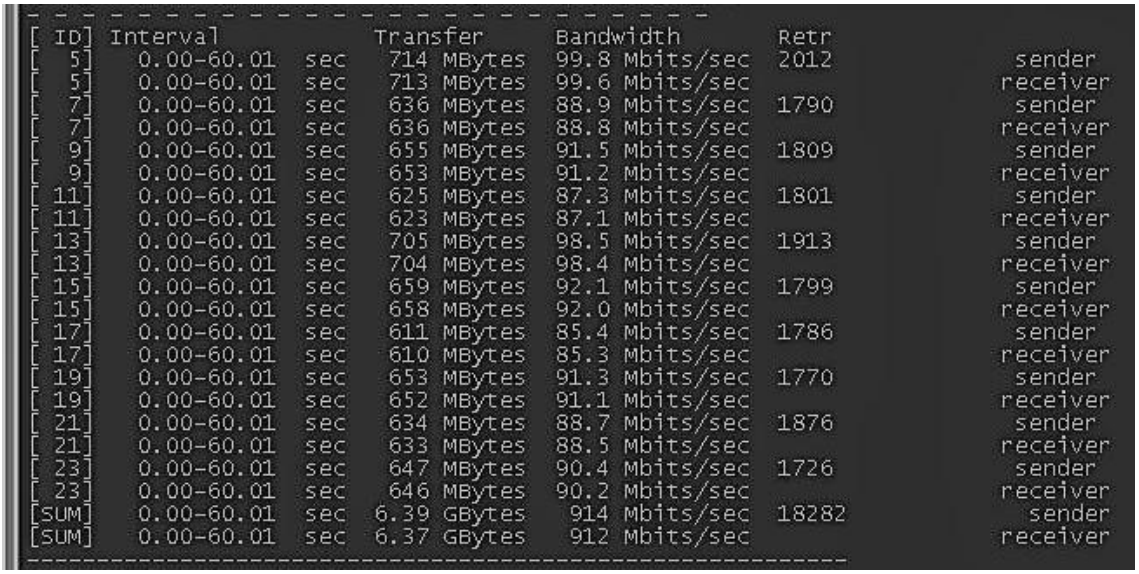
Ainoa rajoitteena CPE-laitteen tunnelin läpäisykyky, jota varten testi on kehitetty, ja tätä rajoittavuutta haluamme mitata. Kohdelaitteen WAN-linkki on tietenkin otettava huomioon, sillä tunnelista ei voi mennä läpi enempää mitä on WAN-linkkiin määritetty nopeus. Yksi huomioon otettava tekijä on myös, että WAN-linkin nopeuden pitää olla symmetrinen, jotta saadaan haluttu tulos. Mikäli linjassa on epäsymmetrinen nopeus, tulos tulee pienimmän nopeuden mukaan.

8.3 Testin ajaminen ja tuloksen tulkinta

Testin ajaminen vaatii, että tunneli- ja reititys konfiguraatiot on tehty. Tämän jälkeen mennään kiinni virtuualikoneeseen, joka toimii clienttinä. Serveri on aina päällä, ja siellä Iperf3 on komennettu käyntiin ”*Iperf3 –s*” komennolla. Clientti käynnistetään esimerkiksi komennolla ”*Iperf3 –c 192.168.20.2 -P 10 -t 60*”. Komennon parametrit ”-c *192.168.10.2* ” kertoo, että kyseessä on client ja serverin IP-osoitteen, ”-P 10” tarkoitta kymmentä rinnakkaista ajoa ja ”-t 60” testiä suoritetaan 60 sekuntia. Iperf3 lähettää oletuksena TCP-pakettia, ja se soveltuu läpäisykyvyn testaukseen hyvin. Lisäoptiona voidaan antaa ”-u” , jolloin Iperf3 lähettää UDP pakettia. Parametrin ”-P” on huomattu vaikuttavavan tulokseen muutamalla prosentilla, eli sitä kasvattamalla on todettu saatavan isoin tulos samaan testikohdetta testatessa.

Kuvassa 6 on esimerkki tulostuksesta, jonka Iperf3 antaa testin ajon jälkeen. Kyseisessä esimerkissä on ajettu Iperf3:sta yhtä reitittimen gigaista porttia vasten. Kyseinen ajo on ajettu yhden minuutin ajan ja kymmentä rinnakkaista ajoa käyttäen. iPef3 kertoo käytetyn

ajan, paljonko dataa on liikkunut per rinnakkaisen ajon ja tämän perusteella lasketun siirtonopeuden. Lopuksi se summaa nämä yhteen ja kertoo siirtonopeuden. Tämän tuloksella pystymme vertailemaan, tunnelien läpäisykykyä ja tuleeko WAN lävitse niin paljon kuin sille on määritelty. Retr kertoo kuinka monta TCP-pakettia on uudestaan lähetetty. Uudestaan lähetys tapahtuu, jos paketti on hukkunut tai korruptoitunut matkalla.



[ID]	Interval		Transfer	Bandwidth	Retr	
[5]	0.00-60.01	sec	714 MBytes	99.8 Mbits/sec	2012	sender
[5]	0.00-60.01	sec	713 MBytes	99.6 Mbits/sec		receiver
[7]	0.00-60.01	sec	636 MBytes	88.9 Mbits/sec	1790	sender
[7]	0.00-60.01	sec	636 MBytes	88.8 Mbits/sec		receiver
[9]	0.00-60.01	sec	655 MBytes	91.5 Mbits/sec	1809	sender
[9]	0.00-60.01	sec	653 MBytes	91.2 Mbits/sec		receiver
[11]	0.00-60.01	sec	625 MBytes	87.3 Mbits/sec	1801	sender
[11]	0.00-60.01	sec	623 MBytes	87.1 Mbits/sec		receiver
[13]	0.00-60.01	sec	705 MBytes	98.5 Mbits/sec	1913	sender
[13]	0.00-60.01	sec	704 MBytes	98.4 Mbits/sec		receiver
[15]	0.00-60.01	sec	659 MBytes	92.1 Mbits/sec	1799	sender
[15]	0.00-60.01	sec	658 MBytes	92.0 Mbits/sec		receiver
[17]	0.00-60.01	sec	611 MBytes	85.4 Mbits/sec	1786	sender
[17]	0.00-60.01	sec	610 MBytes	85.3 Mbits/sec		receiver
[19]	0.00-60.01	sec	653 MBytes	91.3 Mbits/sec	1770	sender
[19]	0.00-60.01	sec	652 MBytes	91.1 Mbits/sec		receiver
[21]	0.00-60.01	sec	634 MBytes	88.7 Mbits/sec	1876	sender
[21]	0.00-60.01	sec	633 MBytes	88.5 Mbits/sec		receiver
[23]	0.00-60.01	sec	647 MBytes	90.4 Mbits/sec	1726	sender
[23]	0.00-60.01	sec	646 MBytes	90.2 Mbits/sec		receiver
[SUM]	0.00-60.01	sec	6.39 GBytes	914 Mbits/sec	18282	sender
[SUM]	0.00-60.01	sec	6.37 GBytes	912 Mbits/sec		receiver

KUVA 6: Esimerkki kuva Iperf3 ulosannista

8.4 Tulokset tietokantaan

Tuloksista luodaan tietokanta Telian sisäiseen käyttöön. Työtä tehdessä testiserveri oli kiinni testilinjassa, jossa kapasiteetti ei riittänyt luomaan tarpeeksi laajaa tai merkityksellistä tietokantaa tuloksista. Tehdyillä testeillä todennettiin työn toimivuus. Serveri siirtyy tulevaisuudessa 10 Gb portin taakse.

9 POHDINTA

Tässä opinnäytetyössä luotiin toimiva konsepti tunnelinläpäisykyvyn testaamiseen. Samaa testiä voidaan myös hyödyntää haluttaessa WAN-linkin testaamiseen, joka on myös tärkeä osa tätä työtä. Tästä työstä tuli yksi työkalu lisää Telian verkonhallinnan käyttöön, ja se toimii suunnitellusti. Työkalussa on myös mahdollisuuksia jatkokehitykselle kuten automaatioon ja erilaisten testiskenaarioiden luontiin.

Mielestäni opinnäytetyö onnistui odotetulla tavalla. Tavoitteet täyttyivät odotetusti niiltä kohdin, mitä tähän mennessä voitiin suorittaa. Vaikka osaa tuloksista ei tässä dokumentissa käsitellä tai ei voitu vielä tähän dokumenttiin tuoda, täyttyvät nämä kohdat työkalun tuotantoon tuonnissa. Näihin kuuluu muun muassa tietokannan luominen eri laitteiden läpäisykyvyn testeistä.

Suurimmat haasteet ja ongelmat tulivat vastaan serverin luomisessa, ja tässä osuudessa vaadittiin paljon uuden opiskelua. Serverin alusta ei ollut paras mahdollinen, koska kyseessä on kaikki yhden laitteen taakse tehtynä toteutuksena. Mahdollisuuksien mukaan toteutus kannattaa siirtää eritettyyn virtuaalikoneympäristöön, jolloin monta turhaa palaa jää pois ja automatisointi helpottuu.

Verkkoteknisen osuuden luonti oli suunnittelun jälkeen helppo kokonaisuus toteuttaa. Tähän vaikutti kahden vuoden kokemus verkonhallinnan asiantuntijana työskentelystä, ja taitavien työtovereiden apu suunnitteluvaiheessa. Vastaavanlaista toteutusta ei välttämättä ole koskaan tehty tai vertailu kohdetta ei ainakaan löytynyt.

LÄHTEET

Xmodulo, Dan Nanni. How to use KVM from the command line on Debian or Ubuntu. Julkaistu 30.12.2015 Tulostettu 1.4.2018.
http://xmodulo.com/use-kvm-command-line-debian-ubuntu.html

Ubuntu community. Yhteisö ohjeita Ubuntuun. Tulostettu 1.4.2018.
https://help.ubuntu.com/community

Ubuntu wiki. KVM ohjeistu Ubuntulle. Tulostettu 1.4.2018.
https://help.ubuntu.com/community/KVM

Ubuntu wiki. VLAN ohjeistus Ubuntulle. Tulostettu 1.4.2018.
https://wiki.ubuntu.com/vlan

The Linux Kernel Archives. Virtual Routing and Forwarding (VRF). Tulostettu 1.4.2018.
https://www.kernel.org/doc/Documentation/networking/vrf.txt

Route Reflector Labs, Andrea Dainese. Working with VRF on Linux. Julkaistu 29.11.2016. Tulostettu 1.4.2018.
http://www.routereflector.com/2016/11/working-with-vrf-on-linux/

Liite 1, Iperf3 käyttäjän dokumentti. Tulostettu 1.5.2018.
https://Iperf.fr/Iperf-doc.php

LIITTEET

Liite 1. Iperf3 käyttäjän dokumentti.

GENERAL OPTIONS	
Command line option	**Description**
-p, --port ***n***	The server port for the server to listen on and the client to connect to. This should be the same in both client and server. Default is 5201.
--cport *n*	Option to specify the client-side port. (new in Iperf 3.1)

-f, --format ***[kmKM]***	A letter specifying the format to print bandwidth numbers in. Supported formats are
	`'k' = Kbits/sec  'K' = KBytes/sec`
	`'m' = Mbits/sec  'M' = MBytes/sec`
	The adaptive formats choose between kilo- and mega- as appropriate.

-i, --interval ***n***	Sets the interval time in seconds between periodic bandwidth, jitter, and loss reports. If non-zero, a report is made every ***interval*** seconds of the bandwidth since the last report. If zero, no periodic reports are printed. Default is zero.
-F, --file name	**client-side:** read from the file and write to the network, instead of using random data; **server-side:** read from the network and write to the file, instead of throwing the data away.
-A, --affinity ***n/n,m-F***	Set the CPU affinity, if possible (Linux and FreeBSD only). On both the client and server you can set the local affinity by using the n form of this argument (where n is a CPU number). In addition, on the client side you can override the server's affinity for just that one test, using the n,m form of argument. Note that when using this feature, a process will only be bound to a single CPU (as opposed to a set containing potentialy multiple CPUs).
-B, --bind ***host***	Bind to ***host***, one of this machine's addresses. For the client this sets the outbound interface. For a server this sets the incoming interface. This is only useful on multihomed hosts, which have multiple network interfaces.
-V, --verbose	give more detailed output

-J, --json	output in JSON format
--logfile file	send output to a log file. (new in Iperf 3.1)
--d, --debug	emit debugging output. Primarily (perhaps exclusively) of use to developers.
-v, --version	Show version information and quit.
-h, --help	Show a help synopsis and quit.

SERVER SPECIFIC OPTIONS	
Command line option	**Description**
-s, --server	Run Iperf in server mode. (This will only allow one Iperf connection at a time)
-D, --daemon	Run the server in background as a daemon.
-I, --pidfile***file***	write a file with the process ID, most useful when running as a daemon. (new in Iperf 3.1)

CLIENT SPECIFIC OPTIONS	
Command line option	**Description**
-c, --client ***host***	Run Iperf in client mode, connecting to an Iperf server running on ***host***.
--sctp	Use SCTP rather than TCP (Linux, FreeBSD and Solaris). (new in Iperf 3.1)
-u, --udp	Use UDP rather than TCP. See also the -b option.
-b, --bandwidth ***n[KM]***	Set target bandwidth to n bits/sec (default 1 Mbit/sec for UDP, unlimited for TCP). If there are multiple streams (-P flag), the bandwidth limit is applied separately to each stream. You can also add a '/' and a number to the bandwidth specifier. This is called "burst mode". It will send the given number of packets without pausing, even if that temporarily exceeds the specified bandwidth limit.
-t, --time ***n***	The time in seconds to transmit for. Iperf normally works by repeatedly sending an array of ***len*** bytes for ***time*** seconds. Default is 10 seconds. See also the -l, -k and -n options.

-n, --num ***n[KM]***	The number of buffers to transmit. Normally, Iperf sends for 10 seconds. The -n option overrides this and sends an array of ***len*** bytes ***num*** times, no matter how long that takes. See also the -l, -k and -t options.
-k, --blockcount ***n[KM]***	The number of blocks (packets) to transmit. (instead of -t or -n) See also the -t, -l and -n options.
-l, --length ***n[KM]***	The length of buffers to read or write. Iperf works by writing an array of ***len*** bytes a number of times. Default is 128 KB for TCP, 8 KB for UDP. See also the -n, -k and -t options.
-P, --parallel ***n***	The number of simultaneous connections to make to the server. Default is 1.
-R, --reverse	Run in reverse mode (server sends, client receives).
-w, --window ***n[KM]***	Sets the socket buffer sizes to the specified value. For TCP, this sets the TCP window size. (this gets sent to the server and used on that side too)
-M, --set-mss ***n***	Attempt to set the TCP maximum segment size (MSS). The MSS is usually the MTU - 40 bytes for the TCP/IP header. For ethernet, the MSS is 1460 bytes (1500 byte MTU).
-N, --no-delay	Set the TCP no delay option, disabling Nagle's algorithm. Normally this is only disabled for interactive applications like telnet.
-4, --version4	only use IPv4.
-6, --version4	only use IPv6.

-S, --tos ***n***	The type-of-service for outgoing packets. (Many routers ignore the TOS field.) You may specify the value in hex with a '0x' prefix, in octal with a '0' prefix, or in decimal. For example, '0x10' hex = '020' octal = '16' decimal. The TOS numbers specified in RFC 1349 are:
	`IPTOS_LOWDELAY minimize delay 0x10`
	`IPTOS_THROUGHPUT maximize throughput 0x08`
	`IPTOS_RELIABILITY maximize reliability 0x04`
	`IPTOS_LOWCOST minimize cost 0x02`

-L, --flowlabel ***n***	Set the IPv6 flow label (currently only supported on Linux).
-Z, --zerocopy	Use a "zero copy" method of sending data, such as sendfile(2), instead of the usual write(2). This uses much less CPU.
-O, --omit ***n***	Omit the first n seconds of the test, to skip past the TCP TCP slowstart period.
-T, --title *str*	Prefix every output line with this string.
-C, --linux-congestion ***algo***	Set the congestion control algorithm (Linux only for iPerf 3.0, Linux and FreeBSD for iPerf 3.1).