Jani Laitinen

Virtualisoidun tietoverkon toteuttaminen

Metropolia Ammattikorkeakoulu
Insinööri (AMK)
Tietotekniikka
Opinnäytetyö
26.4.2011

Metropolia

Tiivistelmä

Tekijä(t) Otsikko Sivumäärä Aika	Jani Laitinen Virtualisoidun tietoverkon toteuttaminen 47 sivua + 3 liitettä 26.4.2011
Tutkinto	insinööri (AMK)
Koulutusohjelma	tietotekniikka
Suuntautumisvaihtoehto	tietoverkot
Ohjaaja(t)	yliopettaja Matti Puska

Insinöörityössä oli tavoitteena luoda täysin virtualisoitu monipuolinen tietoverkko yksittäiselle fyysiselle palvelimelle käyttäen apuna uusimpia ilmaisohjelmia. Tätä tietoverkkoa voitaisiin myöhemmin hyödyntää Metropolia Ammattikorkeakoulun laboratorio- ja opetuskäytössä, koska virtualisoidulla ympäristöllä on tähän tarkoitukseen useita edullisia piirteitä.

Virtualisoitu verkkoympäristö pyrittiin suunnittelemaan siten, että se vastaisi suuren verkon toteutusta niin topologian, kuin reititysprotokollavalinnan suhteen. Verkkoon lisättiin myös erilaisia palvelimia, joiden tehtävä oli tarjota verkon käyttäjille tiedosto- ja videopalvelua, sekä muokata niiden läpi kulkevaa liikennettä.

Verkko saatiin sille asetettujen tavoitteiden mukaiseen toimintakuntoon, jonka jälkeen sille suoritettiin kevyitä testejä sen suorityskyvyn mittaamiseksi. Ne osoittivat, että verkko käyttäytyi suhteellisen hyvin sille odotetuin tavoin video- ja dataliikennettä testatessa, kun verkkoliikenteen laatua muutettiin keinotekoisesti.

Tämä projekti osoitti, että monipuolisen tietoverkon virtualisoiminen yhdelle fyysiselle palvelimelle pelkästään ilmaisohjelmia käyttämällä on täysin mahdollista. Sen myötä syntyneellä virtuaaliympäristöllä on useita potentiaalisia käyttö- ja kehitysmahdollisuuksia laboratorio- ja opetustarkoituksessa Metropolia Ammattikorkeakoulussa.

Avainsanat	palvelinvirtualisointi, työasemavirtualisointi, IP-verkko, reititys

Metropolia

Abstract

Author(s) Title Number of Pages Date	Jani Laitinen Creating a virtualized network 47 pages + 3 appendices 26 April 2011
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Data Networks
Instructor(s)	Matti Puska, Principal Lecturer

The aim of this thesis was to create a fully virtualized and complex network environment on a single physical server using only the newest free available software. Metropolia School of Applied Sciences could then use the network for laboratory and educational purposes afterwards because virtualized network has several advantageous properties from their point of view.

The virtualized network was designed to simulate a large public network and the topology and routing protocols were selected respectively. Additional servers were also added to the network to provide video and data traffic for the end user workstations and also to manipulate the passing network traffic.

The network was created successfully within the requirements set for it and it was tested afterwards to measure the performance. The tests proved that the network was functioning as expected when video and data traffic was manipulated artificially.

This project showed that creating a complex virtualized network on a single physical server using only free software is very much possible. The network environment created in the project has several applications within Metropolia School of Applied Sciences regarding laboratory and educational purposes.

Keywords	server virtualization, workstation virtualization, IP network, routing

Sisällys

1 Johdanto

Virtualisointi on ollut osa modernia käyttöjärjestelmää jo pidemmän aikaa. Sitä on tavallisemmin käytetty muun muassa muistinhallinnassa, levyjärjestelmien toteutuksessa ja prosessien resurssihallinnassa varsinkin uusimpien moniytimisten prosessorien yhteydessä. Laitteiston virtualisointi on tekniikka, joka luo uuden rajapinnan, niin sanotun virtuaalikoneen, fyysisen laitteiston ja virtuaalikoneelle asennetun käyttöjärjestelmän välille antaen uusia mahdollisuuksia käsitellä niin fyysisiä kuin loogisia resursseja. Laitteiston fyysinen konfiguraatio, maantieteellinen sijainti tai toteutus piilotetaan virtuaalikoneessa toimivalta ohjelmistolta, ja sille näkyvät laitteiston fyysiset resurssit voidaan määritellä tarpeen mukaisesti. Edellä mainitut ominaisuudet tuovat täysin uusia mahdollisuuksia toteuttaa palvelin- ja työasemaratkaisuja niin yrityksille kuin kuluttajillekin.

Virtualisoitu ympäristö soveltuu erityisen hyvin testaamiseen ja laboratoriopohjaiseen opetukseen, koska se tarjoaa turvallisen ja eristetyn systeemin, joka pystytään palauttamaan alkuperäiseen tilaansa nopeasti. Luotua ympäristöä pystytään muokkaamaan helposti ja nopeasti tarkoituksiin sopivaksi, koska uusia virtuaalikoneita pystytään luomaan fyysisen laitteiston muistin ja laskentatehon puitteissa niin monta, kuin tarve vaatii muuttamatta fyysistä konfiguraatiota. Virtualisoinnilla pystytään myös simuloimaan virtuaalikoneiden välisiä lähiverkkokytkentöjä, joten ympäristöstä saadaan todellisesti monipuolinen ja interaktiivinen. Kaikki nämä ominaisuudet ovatkin olleet avainasemassa työn suunnittelussa ja toteutuksessa.

Käytännön virtualisointiratkaisut voidaan jakaa karkeasti kahteen eri ryhmään, jotka ovat palvelinvirtualisointi ja työasemavirtualisointi. Palvelinvirtualisoinnilla saavutetaan useita etuja etenkin yritysmaailmassa, jossa sillä pystytään vähentämään laitteiston määrää ja niihin kuluvia kustannuksia, koska usein jokaiselle palvelinresurssille on pitänyt hankkia oma laitteisto. Tämä taas johtuu siitä, että eri sovellusten mahdollisten keskinäisten ongelmien välttämiseksi ne on jouduttu jakamaan omille palvelimilleen tehtävänsä perusteella. Tämänlainen käytäntö ei ole usein optimaalista, koska palvelimen käyttöaste saadaan harvoin pidettyä tarpeeksi korkealla johtuen juuri

”sovellus per palvelin”-ajattelumallista. Se onkin johtanut ajan myötä myös lisäksi kasvaneisiin jäähdytyskustannuksiin, vaikeampaan tukemiseen ja edelleen kustannustehokkuuden laskuun. Työasemakäytössä virtualisointi on mahdollistanut eri käyttöjärjestelmien samanaikaisen käytön ja siten lisännyt joustavuutta sekä mahdollisuuden käyttää aikaisemmin tietyllä käyttöjärjestelmällä toimimattomia ohjelmia virtualisoituna. Virtualisoinnilla pystytään myös toteuttamaan työasema-ympäristöjä, joissa käyttöjärjestelmä toimii täysin keskitetysti palvelimelta käsin terminaaliyhteyden välityksellä, jolloin loppukäyttäjälle ei enää tarvita omaa kokonaista työasemaa. [1, s. 3-6.]

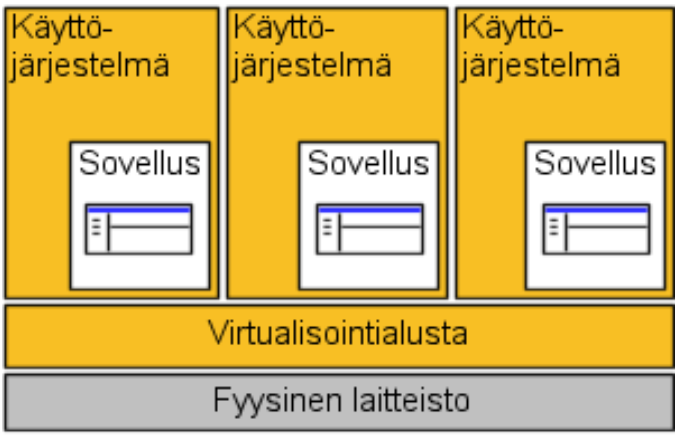
Tässä työssä tutkitaan täysin virtualisoidun ja monipuolisen tietoverkon toteutusta uusimmilla saatavilla olevilla ilmaisilla ohjelmilla. Työn perustana toimii virtualisoiduilla reitittimillä toteutettu tietoverkko, jonka tarkoitus on simuloida laajempaa tietoverkkoa loppukäyttäjineen ja verkkopalveluineen. Lisäksi verkkoliikennettä manipuloidaan keinotekoisesti, jotta vaikutelma suuresta verkosta tehostuisi. Työn tavoitteena on tutkia, miten tietoverkon virtualisointi onnistuu käytännössä ja tehdä pienimuotoisia testejä loppukäyttäjän näkökulmasta sekä reaaliaikaisella videoliikenteellä että tavanomaisella tiedonsiirrolla. Työn toisena tavoitteena on luoda Metropolia Ammattikorkeakoululle projektialusta, jota voidaan käyttää myöhempään kehitykseen, laboratoriotehtäviin tai insinöörityöprojekteihin. Virtualisoidulla ympäristöllä on tähän tavoitteeseen nähden kolme tärkeää ominaisuutta, jotka ovat lähes rajaton muokattavuus, mahdollisuus palauttaa ympäristö nopeasti ennalta määrättyyn tilaan sekä mahdollisuus hallita monimutkaistakin ympäristöä keskitetysti ja helposti.

Työn teoriaosuudessa tutustutaan laitteistopohjaisen virtualisoinnin perusteisiin ja tarkastellaan lähemmin saatavilla olevia palvelimen virtualisointiratkaisuja sekä käydään läpi projektissa luodun tietoverkon sisältämät elementit ja tekniikat. Projektiosuudessa käydään läpi verkon suunnittelu, virtuaalikoneiden sekä niille asennettujen ohjelmistojen toteutus ja lopuksi tehdään pienimuotoisia testejä verkon sisältämien palvelimien ja niiden loppukäyttäjien välillä. Työn lopussa käydään läpi tärkeimpiä sen dokumentointiin liittyviä asioita, ja pohditaan myöhempiä kehitysmahdollisuuksia Metropolia Ammattikorkeakoulussa.

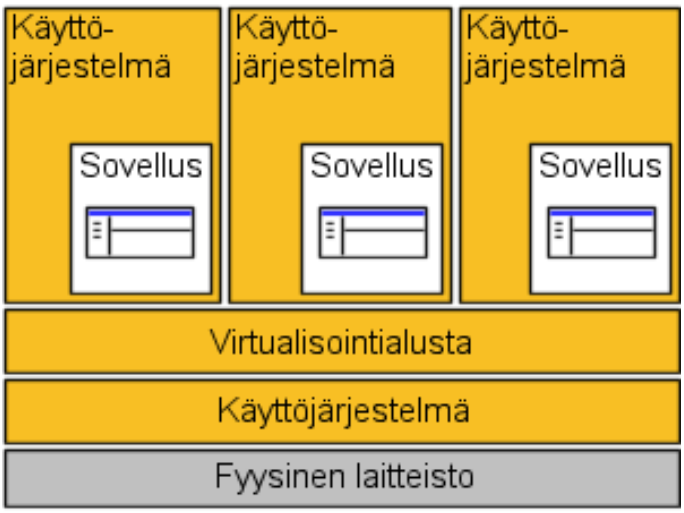
2 Laitteiston virtualisointi

2.1 Virtualisointialusta (Hypervisor)

Olennainen osa kaikkia virtualisointitekniikoita on virtuaalikoneiden valvonnasta ja ohjauksesta vastuussa oleva virtualisointialusta, jota myös kutsutaan Virtual Machine Motoriksi (VMM). Virtualisointialusta on ylimääräinen ohjelmakerros toteutuksesta riippuen, joka toimii suoraan joko laitteiston tai jo olemassa olevan käyttöjärjestelmän päällä. Virtualisointialustat voidaan myöhemmin jakaa tyyppeihin 1 ja 2, joista ensin mainitussa virtualisointialusta on asennettu suoraan käyttöjärjestelmäksi, josta se pystyy ohjaamaan laitteistoa suoraan itse ja valvomaan guest-käyttöjärjestelmiä. Tämän ansiosta tyypin 1 virtualisointialusta tarjoaa suuremman virtualisointitehokkuuden sekä paremman viansiedon ja turvallisuustason, ja on täten suositeltava vaihtoehto suuremmassa palvelinympäristössä (kuva 1). Tyypin 2 virtualisointialusta toimii jo valmiiksi asennetun käyttöjärjestelmän päällä, jolle se tarjoaa virtualisointipalveluja, kuten esimerkiksi muistinhallintaa ja I/O-laitetukea (kuva 2). Se ei ole niin tehokas ratkaisu kuin tyypin 1 virtualisointialusta, mutta se soveltuu pikemminkin ympäristöihin, joissa vaaditaan tavanomaisen käyttöjärjestelmän tarjoamaa laajempaa yhteensopivuutta eri I/O-laitteiden kanssa. [2, s. 7-9.]



Kuva 1.Tyypin 1 virtualisointialusta toimii suoraan laitteiston päällä.



Kuva 2. Tyypin 2 virtualisointialusta toimii käyttöjärjestelmän sisällä.

Virtualisointialusta käyttää muutamaa oleellista tekniikkaa virtuaalikoneiden kontrolloinnissa, joista ensimmäinen on resurssien jakaminen. Virtualisoidussa ympäristössä joko yksi tai useampi fyysinen resurssi muodostaa resurssiryhmän, jota virtualisointialusta käsittelee yhtenä isona objektina ja josta se jakaa resursseja virtuaalikoneille tarpeiden tai määritysten mukaisesti. Oletusarvoisesti resurssit jaetaan tasaisesti, jotta yksittäinen virtuaalikone ei pystyisi omimaan kaikkea itselleen ja hidastamaan koko järjestelmää dramaattisesti.

Toinen tekniikka, jonka tarkoitus on nimensä mukaisesti eristää virtuaalikoneet täydellisesti toisistaan, on nimeltään eristäminen. Usean virtuaalikoneen sisältävässä ympäristössä koneilla ei saa olla keskenään jaettuja resursseja ja operaatioita, jotta järjestelmä olisi mahdollisimman luotettava ja sen vakaus riippumaton yksittäisten virtuaalikoneiden virheellisestä toiminnasta.

Kolmas tekniikka, kapsulointi, sisällyttää virtuaalikoneet ja niiden sisällä toimivan guest-käyttöjärjestelmän konfiguraatioineen ja ohjelmineen tiedostoiksi, jotka on helppo siirtää ja laittaa toimintakuntoon toiselle alustalle. Suurten virtuaaliympäristöjen vikasietoisuus nojaa vahvasti tähän, koska onnettomuuden sattuessa virtuaalikone pitää olla siirrettävissä toiselle alustalle tietojen muuttumatta ja palvelu on saatava heti uudelleen toimintaan.

Neljäs tekniikka on nimeltään emulointi, ja sen tarkoitus on esittää täydellinen laiteympäristö virtuaalikoneiden guest-käyttöjärjestelmille, jotta ne eivät huomaisi millään tavalla toimivansa virtuaalikoneen sisällä. Emuloinnin avulla laitteiston kokoonpano esitetään guest-käyttöjärjestelmälle täysin geneerisinä erilaisten tarkkojen laitemallien sijaan, jotta yhteensopivuus olisi paras mahdollinen. [3, s. 20, taulukko 1.3.]

2.2 x86-arkkitehtuurin suojaustasot

Seuraavaksi tarkastellaan x86-arkkitehtuurin suojaustasoja, jotta voidaan ymmärtää virtualisointialustan toiminta paremmin. X86-prosessorit sisältävät useamman eri suojaustason, missä koodi suoritetaan (kuva 3). Ring 0 on suojaustaso, jossa käyttöjärjestelmän ydin perinteisesti suorittaa koodia ja pystyy ohjaamaan laitteistoa suoraan. Sen yläpuolella sijaitsevat tasot ovat suunniteltu muun muassa laitteistoajureille (Ring 1, Ring 2) ja käyttäjätason ohjelmille (Ring 3), ja niillä on aina hierarkkisesti vähemmän oikeuksia aina noustaessa ylemmälle tasolle. Jos uloimmalla tasolla sijaitsevat ohjelmat haluavat ohjata laitteistoa, joutuvat ne välittämään pyynnön sovellusliittymän (Application Program Interface) kautta käyttöjärjestelmän ytimelle.

Virtualisointialustan kannalta ongelmallinen tilanne perinteisessä arkkitehtuurissa oli, että käyttöjärjestelmien ydin on käytännössä aina toteutettu toimimaan vain Ring 0:lla, joten sen oli pakko yksinkertaisesti huijata virtuaalikoneessa toimivia käyttöjärjestelmiä luulemaan näin, koska virtualisointialusta varasi itse alimman tason [4, s. 21]. Tähän saatiin kuitenkin toimivampi ratkaisu AMD V- ja Intel VT-tekniikan omaavien prosessorien myötä, jolloin virtualisointialustalle luotiin täysin oma suojaustaso Ring -1; nyt virtuaalikoneessa oleva käyttöjärjestelmän ydin pystyy toimimaan todellisesti Ring 0:lla.

Kuva 3. AMD V- ja Intel VT -laajennuksen omaavien x86-prosessorien suojaustasot.

2.3 Laitteiston virtualisointitekniikat

Täysvirtualisointi

Täysvirtualisoinnin (eng. Full Virtualization) tarkoituksena on emuloida täydellistä laiteympäristöä guest-käyttöjärjestelmälle siten, että se olettaa toimivansa itse suoraan laitteiston yläpuolella. Laitteiston ja guest-käyttöjärjestelmän välissä toimiva virtualisointialusta käsittelee käyttöjärjestelmässä tapahtuvat operaatiot ja välittää täysin aidolta näyttävän laitteiston palautteen ohjelmille emulointia hyväksi käyttäen. Tämän tekniikan etu on sen yhteensopivuus käytännössä jokaisen käyttöjärjestelmän kanssa, joka voisi toimia myös itse fyysisessä laitteistossa ilman, että käyttöjärjestelmää tarvitsisi muokata millään lailla. Toinen etu on, että virtuaalikoneiden toiminta voidaan eristää toisistaan täydellisesti. Näin saavutetaan ympäristö, jossa sovellukset eivät häiritse toistensa toimintaa, ja siksi täysvirtualisointi soveltuu palvelinkäyttöön erityisen hyvin. Tätä tukee erityisesti virtuaalikoneiden helppo siirrettävyys vikatilanteissa, koska kohdelaitteiston ei tarvitse olla kuin arkkitehtuurin puolesta yhteensopiva vanhan kanssa. Tällä tavoin palvelu saadaan jatkumaan uudelleen mahdollisimman pian. Täysvirtualisoinnin huonona puolena voidaan pitää edellä olevista seikoista johtuvaa jatkuvaa emuloinnin tarvetta ja käskyjen välittämistä virtualisointialustan kautta, mikä laskee suorituskykyä. Toinen suorituskykyä laskeva seikka on se, että jokaisessa virtuaalikoneessa on oltava oma käyttöjärjestelmänsä. [4, s. 4.]

Käyttöjärjestelmätason virtualisointi

Käyttöjärjestelmätason virtualisoinnissa (eng. Paravirtualization) guest-käyttöjärjestelmä ja hypervisor kommunikoivat keskenään suorituskyvyn parantamiseksi. Tämä tekniikka vaatii varta vasten tehtyjä muutoksia guest-käyttöjärjestelmään, joka rajoittaa tuettuja vaihtoehtoja tuntuvasti. Käyttöjärjestelmätason virtualisoinnin etu täysvirtualisointiin verrattuna on, että virtualisointialustan ei tarvitse ylläpitää samanlaista laitteiston emulointia guest-käyttöjärjestelmäää varten, mikä taas nopeuttaa järjestelmän toimintaa. Suorituskyky voi kuitenkin vaihdella tuntuvasti eri tilanteista riippuen. Käyttöjärjestelmätason virtualisoinnin huonona puolena voidaan pitää ympäristön tukemisen ja ylläpidon huomattavampi haasteellisuus, koska guest-käyttöjärjestelmiin on jouduttu tekemään muutoksia aina käyttöjärjestelmän ytimestä saakka. Guest-käyttöjärjestelmien eristys on myös heikompi kuin täysvirtualisoinnissa, minkä ansiosta käyttöjärjestelmätason virtualisointi ei sovellu hyvin useiden käyttöjärjestelmien turvalliseen ja vakaaseen virtualisointiin niin hyvin. [4, s. 5.]

Laitteisto-ohjattu virtualisointi

Laitteisto-ohjattu virtualisointi (eng. Hardware-assisted Virtualization) on uusi virtualisointitekniikka, jossa käytetään hyväksi x86-arkkitehtuuriin tehtyjä laitteiston virtualisointilaajennuksia, joista esimerkkeinä on AMD:n kehittämä AMD Virtualization ja Intelin kehittämä Intel Virtualization Technology. Virtualisointiohjelmien lisääntyvä tuki laitteisto-ohjatulle virtualisoinnille antaa sille mahdollisuuden toimia aikaisemmin mainittujen virtualisointitekniikoiden kanssa samanaikaisesti, ja itse asiassa se tulee lopulta eliminoimaan tarpeen perinteiselle virtualisoinnille. Se tarjoaa erittäin hyvän suorituskyvyn prosessorin, muistin ja kiintovelyn käytössä, koska guest-käyttöjärjestelmä pääsee ohjaamaan laitteistoa suoraan, ja täten virtualisointialustan ei tarvitse turvautua hitaampaan laitteiston emulointiin. Laitteisto-ohjatun virtualisoinnin kanssa on otettava huomioon, että AMD:n ja Intelin ratkaisut eivät ole keskenään yhteensopivia, ja myös ohjelmistotuki molemmille on toistaiseksi rajallinen. [4, s. 6.]

2.4 Palvelinvirtualisointi

Palvelinvirtualisoinnissa käytetään tavallisimmin tyypin 1 virtualisointialustaa, koska ensisijaisena tavoitteena on yleensä luoda luotettava, kustannustehokas ja tietoturvallinen ympäristö. Virtualisoitu palvelinympäristö soveltuu myös hyvin ohjelmistokehitykseen ja testaukseen. Virtualisointi luo uuden ulottuvuuden luotettavuuden parantamiseksi, koska virtuaalikoneet ovat erittäin nopeasti siirrettävissä hajonneesta palvelimesta uuteen samassa tilassa, kuin missä ne olivat vian ilmetessä. Siirto voidaan myös automatisoida, jotta palvelun jatkuminen turvattaisiin vielä tehokkaammin. Virtualisoidussa ympäristössä fyysisten laitteiden ei enää tarvitse olla suoraan vastuussa pyörittämistään palveluista, vaan ne voivat vain olla osa isompaa kokonaisuutta, joten yksittäisen laitteiston rikkoutuminen ei keskeytä palvelua. Tämä tuo myös mahdollisuuden resurssien priorisointiin eri aikoina eri kohteisiin, esimerkiksi sähköpostijärjestelmälle toimistotunteina ja varmuuskopioinnille niiden ulkopuolella. Virtualisointiohjelmissa on myös omia automatisoituja rutiineja, jotka voivat tarvittaessa käynnistää uusia tai sammuttaa nykyisiä palvelimia riippuen palvelujen resurssitarpeesta. Palvelimien huoltaminen ja päivittäminen on myös entistä helpompaa, koska niiden sammuttaminen ei välttämättä katkaise palvelua.

Ilman virtualisointia toimivat palvelinympäristöt on pahimmillaan jouduttu hajauttamaan ”palvelu per palvelin”-periaatteella, joten niiden käyttöaste on ollut usein hyvin alhainen. Tämä on kuitenkin ollut välttämätöntä palveluiden toiminnan takaamiseksi, koska siten ehkäistään niiden keskinäiset konfliktit normaalissa käyttöjärjestelmässä. Virtuaalikoneet pystytään eristämään toisistaan täydellisesti, minkä ansiosta yksittäiselle fyysiselle palvelimelle voidaan keskittää laiteresurssien puitteissa jopa satoja virtuaalisia palvelimia. Näin pystytään varmistamaan, että laitteiston resurssienkäyttö pysyy optimaalisena koko ajan. Tällä saavutetaan kustannussäästöjä muun muassa vähentyneenä virrankulutuksen ja jäähdytyksen tarpeena, sekä fyysisen laitteiston, ohjelmistolisenssien ja ympäristön hallintaan tarvittavan työpanoksen määrässä. Eristetty ympäristö soveltuu lisäksi hyvin ohjelmistokehitykseen ja testaukseen, koska sitä varten luodut virtuaalikoneet eivät häiritse muiden samalla palvelimella sijaitsevien virtuaalikoneiden toimintoja millään tavalla.

Palvelinvirtualisoinnilla saavutetaan parempi tietoturva, koska pienempään määrään fyysisiä laitteita on vaikeampi tehdä esimerkiksi tietoturvahyökkäysiä, sillä niitä pystytään valvomaan tarkemmin ja haavoittuvuuksien korjaaminen on nopeampaa ja helpompaa. Virtualisoinnin takaisinpäin yhteensopivuuden ansiosta monen yrityksen vanhat ja usein toiminnaltaan kriittiset palvelimet voidaan keskittää modernimmalle laitteistolle. Ennen vaihtoehtona tähän oli vain uuden laitteistoalustan ja siihen yhteensopivan ohjelmiston hankkiminen. Joissain tapauksissa palvelinlaitteisto on niin vanhaa tekniikkaa, että korvaavaa laitteistoa ei olisi edes saatavilla vikatilanteessa.

Verkkoyhteyden virtualisointi onnistuu useilla eri tavoilla tavoitteesta riippuen. Vaikka virtuaalikoneet ovatkin eristettyjä ympäristöjä, niille voidaan luoda verkkoyhteys joko suoraan ulkoverkkoon sillatussa tilassa tai lähiverkkosegmenttien kautta samalla palvelimella sijaitseviin muihin virtuaalikoneisiin. Sillattuun tilaan määritelty virtuaalikoneen verkkokortti tarvitsee toimiakseen oman Internet Protocol (IP) -osoitteen, jonka se saa tavallisimmin Dynamic Host Configuration Protocol (DHCP) -palvelimen kautta verkosta. Lähiverkkosegmenteillä voidaan simuloida suoraa linkkiä virtuaalikoneiden välillä, jolloin oikealla IP-osoitteen määrittelyllä samassa segmentissä sijaitsevat verkkokortit pystyvät kommunikoimaan keskenään.

2.5 Työasemavirtualisointi

Työasemavirtualisoinnilla voidaan tarjota loppukäyttäjälle useita etuja toteutuksen mukaan, kuten joustavuutta ja paremman tietoturvan, tärkeiden tiedostojen paremman varmennuksen sekä ongelmattomamman työaseman käytön. Tyypin 2 virtualisointialustan omaavilla virtualisointiohjelmistoilla käyttäjän on mahdollista operoida useampia eri käyttöjärjestelmiä samanaikaisesti, kunhan vain itse virtualisointiohjelmisto on kaikissa tuettu. Tällä saavutetaan se, että käyttäjän ei tarvitse valita laitealustaa vain sille tarjolla olevien ohjelmistojen perusteella, vaan niiden käyttö pystytään toteuttamaan virtualisoimisella. Virtuaalikoneen eristävän luonteen takia se tarjoaa hyvän suojan arkaluontoisille dokumenteille, koska käyttäjä voi luoda sen vain tiettyä tarkoitusta varten haluamillaan rajoituksilla sen sijaan, että sillä hoidettaisiin myös esimerkiksi internet-selaus ja muu verkkokäyttö. Tyypin 2 -

pohjainen virtualisointiohjelmisto tarjoaa myös erityisen hyvän mahdollisuuden loppukäyttäjille suunnattujen ohjelmien testaukseen varsinkin haittaohjelmien parissa, koska niiden vaikutukset pystytään rajoittamaan virtuaalikoneen sisään. Koneen päästyä käyttökelvottomaan tilaan se voidaan aina palauttaa aiempaan ennalta määriteltyyn tilaan nopeasti.

Työasema on myös mahdollista virtualisoida suoraan palvelimelle sijaitsevaksi virtuaalikoneeksi, jonka käyttöön loppukäyttäjältä tarvitaan vain terminaaliyhteyteen soveltuva laitteisto. Tämä toteutus hyödyntää palvelinvirtualisoinnista tuttua tyypin 1 virtualisointialustaa. Kaikki loppukäyttäjän ohjelmistot ja tiedostot sijaitsevat palvelimella, joka hoitaa myös muut tietokoneen tehtävät. Tämänlainen ympäristö tuo lisää varmuutta tärkeiden tiedostojen säilymiseen, koska palvelimien sisältämät tiedostot ovat lähes poikkeuksetta kahdennettuja. Kuten palvelinvirtualisoinnin tapauksessa, myös tämän tyylinen työasemavirtualisointi lisää kustannustehokkuutta, koska erillisiä täysiverisiä työasemia ei enää tarvita. Uusien työasemien luonti ja ohjelmistopäivitysten keskitetty asentaminen helpottuu huomattavasti, kun itse koneet sijaitsevat jo valmiiksi jopa samassa fyysisessä laitteistossa, kuin missä itse päivityksistä vastaavat järjestelmät ovat.

Täysin ongelmaton tämänlainen keskitetty työasemaympäristö ei kuitenkaan ole. Käyttäjien vapaudet ja yksityisyyden taso laskevat ympäristön tietoturvapolitiikasta riippuen, koska itse virtuaalikone ei sijaitse enää heidän omalla itsenäisellä laitteistolla, vaan suoraan palvelimen tietovarastossa. Toimivan tietoverkon tärkeys korostuu myös entisestään, koska työasemien toimivuus on riippuvainen etäyhteyden toimivuudesta palvelimelle, jolloin vian sattuessa potentiaalisesti suuri määrä käyttäjiä estyy käyttämästä työasemaansa. Tämän seurauksena palvelimien ja niiden ylläpidon kustannukset tulevat nousemaan, koska järjestelmä on entistä monimutkaisempi siinä päässä. Näiden seikkojen takia työasemien keskitetty virtualisointi soveltuu parhaiten vain tietynlaisiin ympäristöihin, missä työasemien konfiguraatio on yhdenmukainen ja vaatimusta erikoisempien ohjelmien tai laitteiden tuelle ei ole.

3 Palvelimien virtualisointiohjelmistot

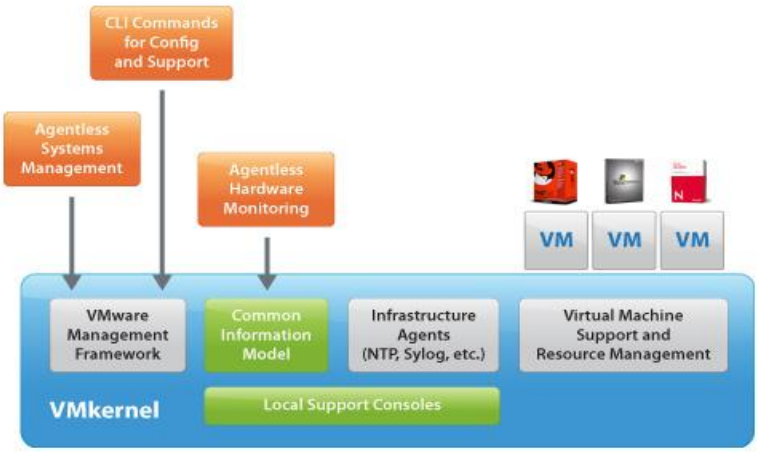
3.1 VMware ESXi

VMwaren kehittämä ESXi-virtualisointiohjelmisto on täysvirtualisointia hyödyntävä tyypin 1 virtualisointialusta, joka tukee myös AMD:n ja Intelin laitteistopohjaisen virtualisoinnin laajennuksia. VMware ESXi soveltuu hyvin tämänkaltaisen projektin alustaksi, koska se ei tarvitse erillistä käyttöjärjestelmää toimiakseen, ja sen perustoiminnot sisältävä lisenssi on ilmainen. Saatavilla on myös eritasoisia maksullisia lisenssejä, joiden tuomat lisäominaisuudet tulevat tarpeeseen erityisesti suurissa yritysympäristöissä, joissa tehokkuus, turvallisuus ja monipuolisempi hallittavuus ovat suuremmassa roolissa. Tässä projektissa käytettiin ESXi:n uusinta versiota 4.1, joka tukee ainoastaan 64-bittistä x86-prosessoriarkkitehtuuria. ESXi-asennuksen koko on vain 32 megatavua, minkä ansiosta se on kevyt ja sisältää vähemmän mahdollisia haavoittuvuuksia sisältäviä koodirivejä verrattuna muihin virtualisointiohjelmistoihin, ja sitä hallinnoidaan verkkoyhteyden kautta erilliselle työasemalle asennettavalla vSphere Clientillä. ESXi sisältää laajan tuen eri guest-käyttöjärjestelmille, minkä vuoksi se soveltuu hyvin monipuolisiin virtuaaliympäristöihin. ESXi:n laiteajurit ovat ennalta asennetut, jonka johdosta se on tarkka laitteiston yhteensopivuudesta.

VMware ESXi:n arkkitehtuurin käsittävät VMkernel-käyttöjärjestelmä ja sen päällä toimivat prosessit (kuva 4). VMkernel on VMwaren kehittämä POSIX:n kaltainen käyttöjärjestelmä ja sen tehtäviin kuuluu ohjata fyysistä laitteistoa ja jakaa resursseja ohjelmien kesken. Sen päällä toimivat seuraavat keskeisimmät prosessit:

- Direct Console User Interface (DCUI) on yksinkertainen käyttöliittymä, joka on tarkoitettu pääasiallisesti ESXi-palvelimen alkukonfiguraation tekemiseen.
- Virtual Machine Monitor (VMM) luo ympäristön virtuaalikoneille ja ohjaa niitä. Sitä avustaa tässä toinen prosessi nimeltä VMX.
- Useat erilaiset agent-prosessit, jotka mahdollistavat VMware Infrastructuren hallinnan ulkopuolisilta ohjelmilta käsin.

- Common Information Model (CIM) vastaavasti mahdollistaa laitetason hallinnan ulkopuolisilta ohjelmilta käsin. [5, s. 3.]



Kuva 4. VMware ESXi:n arkkitehtuuri [6.]

3.2 Microsoft Hyper-V

Microsoftin kehittämä Hyper-V on kilpaileva virtualisointiohjelmisto ja se on VMware ESXi:n tavoin myös saatavilla joko ilmaiseksi omana Hyper-V Server 2008 R2 -asennuksena tai maksulliseen Windows Server 2008:aan erillisesti asennettavana ohjelmana. Hyper-V vaatii myös toimiakseen 64-bittisen x86-prosessoriarkkitehtuurin ja tuen AMD:n ja Intelin laitteistopohjaiselle virtualisoinnille. Hyper-V Server 2008 R2 voidaan mieltää Type 1-hypervisoriksi, koska siinä Hyper-V on valmiiksi integroituna virtualisointitoiminnoille räätälöityyn Windows Server 2008 -ytimeen, josta kaikki ylimääräiset osat on karsittu pois laiteajureita lukuun ottamatta. Hyper-V Server 2008 R2 -asennuksen koko on noin 1,8 gigatavua. Hyper-V -palvelinta voidaan hallita Windows Server 2008 Management Consolen (MMC) tai Microsoft Powershell v2:n kautta.

VMware ESXi:in verrattuna Hyper-V:ssä on rajallinen tuki Microsoftin ulkopuolisille guest-käyttöjärjestelmille, minkä takia ESXi soveltui paremmin projektin tarpeisiin. Kuitenkin projekti olisi voitu toteuttaa Hyper-V:llä hyvin, jos Linuxin ja BSD:n sijaan guest-käyttöjärjestelmiksi olisi valittu Windows XP tai 7. Hyper-V Server 2008 R2 käyttää laiteajureinaan normaaleja Windows-ajureita, joita on helppo lisätä ja päivittää uusiin versioihin. Tämän ansiosta Hyper-V:llä saavutetaan laajempi laitetuki kuin ESXi-ympäristöllä.

4 Reititys

4.1.1 OSPF-reititys

Open Shortest Path First (OSPF) on reititysprotokolla, joka soveltuu reitittimien väliseen kommunikointiin autonomisen alueen (Autonomous System) sisällä. Esimerkkinä tämänlaisesta ympäristöstä ovat saman hallinnon alla toimivat asiakasverkot, kuten yritysten tai oppilaitosten sisäverkot. Tämän tyyppisiä vastaavia protokollia ovat esimerkiksi Routing Information Protocol (RIP) ja Ciscon kehittämä Interior Gateway Routing Protocol (IGRP), jotka kaikki ovat tyypiltään sisäisiä reititysprotokollia (Interior Gateway Protocol). OSPF:n etuja ovat tuki vaihtuvanmittaiselle aliverkkopeitteelle (Variable-lenght Subnet Mask), vähäinen verkkoliikenteen kuormitus ja nopea reagointi verkkotopologian muutoksiin sekä standardinmukaisuus. Kyseessä on myös avoin reititysprotokolla, joka ei ole minkään yksittäisen valmistajan omistuksessa.

OSPF:iä käytettäessä jokainen reititin rakentaa itselleen topologian koko verkosta muiden aktiivisten reitittimien lähettämistä linkkitiedoista, joiden perusteella ne koostavat itselleen reititystaulun ja laskevat jokaiselle reitille Cost-arvon Djikstra-algoritmilla. OSPF:n reitinvalinta kohteeseen toteutetaan laskemalla yhteen välillä olevien reittien Cost-arvot ja pienimmän lopputuloksen antanut yhdistelmä valitaan käytettäväksi reitiksi. Jos linkkien tilassa tapahtuu muutos, sen huomannut reititin lähettää linkkipäivityksen automaattisesti kaikille reitittimille saman OSPF-alueen sisällä, jotka taas laskevat sen perusteella itselleen uuden topologian ja reititystaulun. Etäisyysvektori (Distance-vector) -tyyppiseen reititysprotokollaan verrattuna tämä

toteutus kuormittaa verkkoa vähemmän, koska linkkitiedot välitetään muille reitittimille pienikokoisina paketteina kokonaisten reititystaulujen sijaan. Tämän tekniikan ansiosta OSPF:ä kutsutaan myös linkkitilalliseksi (Link-state) reititysprotokollaksi.

Autonomisen alueen sisälle on mahdollista luoda useampia OSPF-alueita (Area), joilla saadaan kontrolloitua paremmin reititystaulujen kokoa isoissa verkoissa, koska linkkien tilapäivitykset lähetetään aina vain oman alueen reitittimille. OSPF-alueiden konfigurointi tulisi aina aloittaa 0:sta, koska se on runkoverkon alueen numero. Muiden alueiden tulisi olla kytköksissä tähän alueeseen fyysisesti tai virtuaalilinkin kautta, koska verkon reitityspäivitykset muilta alueilta oletetaan kulkevan aina runkoverkon alueen kautta.

OSPF ylläpitää naapuruussuhteita saman alueen sisällä olevien reitittimien välillä, jos ovat fyysisesti kytkettynä toisiinsa. Jokainen OSPF-reititin lähettää säännöllisiä "hello"-päivityksiä erityiseen ryhmälähetysosoitteeseen, joka on OSPF:n tapauksessa 224.0.0.5, jonka kautta ne välitetään saman alueen muille reitittimille. Naapuruussuhde muodostetaan vasta, kun molemmat reitittimet näkevät itsensä toisen lähettämässä "hello"-päivityksessä, jotta kahdensuuntainen kommunikaatio voidaan taata. Kuvaukset OSPF:n lähettämistä tärkeimmistä viestityypeistä löytyvät taulukosta 1. OSPF-alueen sisällä voidaan määritellä yksi reititin Designated Routeriksi (DR) ja toinen Backup Designated Routeriksi (BDR), joiden tehtävä on olla keskuspiste muiden reitittimien tiedon vaihdolle. Näin vältetään turhaa verkkoliikennettä, jossa kaikki reitittimet kommunikoisivat vain toistensa kanssa ilman mitään koordinaatiota. [7.]

Taulukko 1. OSPF-protokollan viestityypit.[7.]

Viestityyppi	***Tehtävä***
Hello	Viesti, joka lähetetään alueen reitittimille, jotta ne huomaavat toisensa ja pystyvät muodostamaan naapuruussuhteen.
Database Description	Viesti, joka sisältää linkkitilallisen topologiakuvauksen OSPF-alueesta.
Link State Request	Viesti, joka sisältää pyynnön toiselle reitittimelle, jossa sen linkkitietokannasta pyydetään lähettämään tietty osa pyynnön lähettäneelle reitittimelle.
Link State Update	Viesti, joka lähetetään vastineena Link State Requestille, joka sisältää pyydetyn tiedon linkkitietokannasta.
Link State Acknowledgement	Viesti, joka lähetetään takaisin onnistuneesti saadusta Link State Updatesta.

4.1.2 BGP-reititys

Border Gateway Protocol (BGP) on reititysprotokolla, joka on vastuussa internetin sisältämien lukuisten autonomisten alueiden välisestä reitityksestä. Se on suunniteltu hallitsemaan suuria määriä reittejä kerralla, ja siksi sen pääasialliset käyttäjät ovat internetin yhteydentarjoajat. BGP on tyypiltään ulkoinen reititysprotokolla (Exterior Gateway Protocol), ja se suunniteltiin korvaamaan vanhempi EGP-protokolla, jotta internetistä tulisi aidosti keskittämätön systeemi. EGP:n tapauksessa kaiken liikenteen piti kulkea erillisen NSFNet-runkoverkon läpi, mikä ei ollut optimaalinen ratkaisu.

Uuden BGP4-version tärkeimpiä etuja ovat mahdollisuus luokattomaan internet-toimialueiden väliseen reititykseen ja reittien yhdistämiseen, jolla pystytään pienentämään reititystaulujen jo massiivista kokoa. BGP on yksinkertainen reititysprotokolla verrattuna esimerkiksi OSPF:iin, koska se ei tee monimutkaisia laskutoimituksia eri reittien paremmuuden päättelemiseksi. BGP:n reititystaulu on käytännössä vain lista aktiivisista verkoista, joihin liikennettä pystytään reitittämään. Reittimuutokset on tehtävä käsin, jotta verkossa toimiviin reitittimiin kohdistuva

kuorma voitaisiin pitää minimissä optimoimalla reititystaulut mahdollisimman kompakteiksi. Näin verkon topologiamuutokset eivät vaikuta reititystaulujen sisältöön, toisin kuin dynaamisen reititysprotokollan tapauksessa tapahtuu. BGP lähettää päivitysviesteissä naapurireitittimille vain oletetun kohdeverkon osoitteen, siihen liitetyn AS_Path-arvon ja tarvittaessa erilaisia lisämäärittelyjä (Taulukko 2).

Taulukko 2. BGP:n käyttämät lisämäärittelyt. [8.]

Määrittely	***Tehtävä***
Weight	Ciscon käyttämä arvo, joka on tarkoitettu tietyn reitin priorisoimiseen, jos samaan kohteeseen mainostetaan useampaa vaihtoehtoa.
Local Preference	Local Preference -arvolla voidaan suosia tiettyä reittiä, jos autonomisen alueen sisältä on useampi reitti ulos.
Multi-exit Discriminator	Multi-exit Discriminator -arvolla pystytään mainostamaan painotettua reittiä toisesta autonomisesta alueesta sisäänpäin kohdistuvalle liikenteelle.
Origin	Origin-arvo kertoo reitin alkuperän, joka voi olla IGP (AS:n sisäinen reitti), EGP (AS:n ulkopuolisen reitityksen kautta opittu) tai Incomplete (tuntematon).
AS_path	AS_path-arvo sisältää listan autonomisista alueista, joiden kautta kohteeseen pääsee.
Next-hop	Next-hop–arvo sisältää reittiä mainostavan reitittimen IP-osoitteen.
Community	Community-määrittelyllä voidaan ryhmittää kohdeverkkoja, joihin tiettyjä reitityspäätöksiä halutaan soveltaa. Nämä ryhmät luodaan Route Map-määrittelyjen avulla.

BGP:n naapuruussuhteet on konfiguroitava käsin reitittimien välille, minkä jälkeen ne alkavat kommunikoida toistensa kanssa TCP-pohjaisella tiedonsiirrolla portin 179 kautta. TCP-yhteydellä saavutetaan luotettava tiedonsiirtokanava ilman, että tarvitaan ajastetusti toistuvia päivitysviestejä naapurireitittimien välillä. Reitittimet pitävät TCP-

istunnon aktiivisena lähettämällä toisilleen 19 tavun pituisia keep-alive -paketteja oletuksena kerran 60 sekunnissa. BGP-yhteyden aktivoimiseksi naapurien on käytävä keskenään läpi prosessi nimeltä BGP Finite State Machine. Tämä prosessi sisältää useita välivaiheita, jolloin informaatiota vaihdetaan reitittimien välillä, ja keskeytyessään se pitää aloittaa alusta uudestaan. Kun prosessi saadaan vietyä läpi onnistuneesti, BGP-istunto muuttuu "established"-tilaan ja reitityspäivityksiä voidaan alkaa vaihtaa. Alussa reitittimet siirtävät toisilleen niiden koko reititystaulun, joka tämän hetken internetissä voi sisältää yli 300 000 reittiä, ja tämän jälkeen päivitykset tapahtuvat vain muutosten sattuessa, jolloin päivitysviesti sisältää tiedon vain muuttuneista reiteistä. Tällä tavoin naapurireitittimien välinen päivitysviesteihin kuluva tiedonsiirto pyritään minimoimaan. [8.]

4.2 Vyatta

Vyatta (http://www.vyatta.com) on amerikkalainen tietoliikennealan yritys, joka kehittää yritysmaailmaan soveltuvaa monipuolisilla verkko-ominaisuuksilla varustettua käyttöjärjestelmää. Pääasiallisesti virtuaalikoneeseen asennettavaksi tarkoitetun käyttöjärjestelmän päämäärä on korvata tavanomaiset verkkolaitteet, kuten reitittimet ja palomuurit, yhdellä monipuolisella verkkoratkaisulla. Vyatta toimittaa asiakkaille myös valmiita laitekokonaisuuksia, jos fyysistä laitteistoa ei haluta hankkia itse. Vyattan kehittämä käyttöjärjestelmä perustuu Debian Linuxiin ja siihen on liitetty monia avoimen lähdekoodin verkkosovelluksia, kuten esimerkiksi Quagga sekä OpenVPN. Mukana tulevia sovelluksia käytetään täydentämään Vyattan toimintoja, kuten esimerkiksi Quaggan tapauksessa hoitamaan laitteen reititysominaisuudet. Vyattan käyttöliittymänä toimii Xorphs (Xorp Shell), ja se on hyvin samankaltainen Juniper OS:n käyttöliittymän kanssa, mutta eroaa selvästi laajemmin tunnetusta Cisco IOS CLI:sta. Vaihtoehtona on myös aktivoida HTTP-pohjainen graafinen käyttöliittymä, jota voidaan käyttää verkkoselaimen kautta.

Vyattan tarjoaman verkkoratkaisun etu tavanomaisimpiin laitteistopohjaisiin ratkaisuihin verrattuna on se, että virtualisoinnin ansiosta se toimii käytännössä millä tahansa laitteistolla, kunhan itse virtualisointi on tuettuna. X86-pohjaisen laitteiston

kanssa suorituskyvyn skaalautuminen on käytännössä rajatonta, ja käyttöjärjestelmä sisältää tuen moniydinprosessoreille, jonka ansiosta suorituskyky on helppoa räätälöidä juuri omaan ympäristöön sopivaksi. Tämänlainen joustavuus ja x86-pohjaisen laitteiston edullinen hinta suorituskykyyn nähden antaa Vyattan tarjoamalle ratkaisulle suuremman potentiaalin kustannussäästöjen näkökulmasta verrattuna muihin valmistajiin.

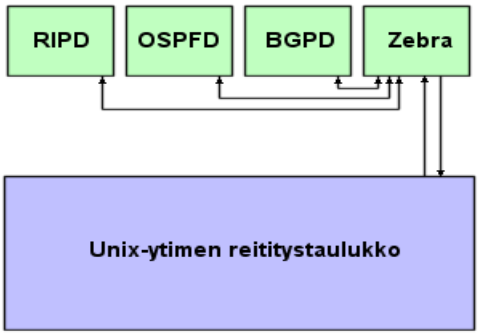
Vyattan tarjoamat keskeisimmät ominaisuudet ovat:

- Quagga-pohjainen reititys dynaamisilla reititysprotokollilla (BGP, OSPF, RIP) IPv4- ja IPv6-ympäristöissä
- Liitäntätuki 802.11-standardin mukaiselle langattomalle tukiasemalle, laajaverkkojen sarjaliikenneportille, sekä 10 Mbps–10 Gbps Ethernetille
- ennaltaehkäisevä haitallisen sisällön torjunta verkkosuodatuksen ja hyökkäyksenestojärjestelmän avulla
- Site-to-site IPsec VPN-tuki, sekä SSL-suojatut etäyhteydet loppukäyttäjille OpenVPN:n avulla
- verkkoliikenteen tunnistus ja priorisointi Quality of Service mekanismin avulla
- mekanismit palvelun jatkumisen takaamiseen rinnakkaisilla järjestelmillä
- etähallintamahdollisuus teksti- sekä selainpohjaisella käyttöliittymällä tarvittaessa SSHv2:n, RADIUS:n tai TACACS+:n kautta
- informaatiolokin ja diagnostiikan valvonta tavanomaisimmilla työkaluilla kuten SNMP, Netflow, Syslog ja Wireshark
- tilallinen IPv4/IPv6 palomuuri kriittisen tiedon suojaamiseen ja haittaohjelmien torjumiseen. [9, s. 3.]

4.3 Quagga

Quagga (http://www.quagga.net) on avoimeen lähdekoodiin perustuva ohjelmistopaketti, joka tarjoaa TCP/IP-pohjaisia reitityspalveluja RIP-, OSPF-, IS-IS- ja BGP-protokollilla sekä IPv4 että IPv6-ympäristöissä. Quagga perustuu aikaisempaan Zebra-reitittimeen, jonka kehitys lopetettiin vuonna 2003, ja se tukee tällä hetkellä GNU/Linuxia, BSD:tä ja Solarista. Quaggan käyttöliittymä on nimeltään Vty, ja se on Cisco IOS CLI:n kaltainen. Tässä työssä käytettiin vanhempaa Quaggan versiota, jossa tukea IS-IS:lle ei vielä ollut.

Quagga toimii itsenäisesti käyttöjärjestelmän päällä toisin kuin esimerkiksi Vyatta, ja se eroaa perinteisistä reititysohjelmistoista siten, että siinä reitityksestä ja reitytystaulukosta vastaavat ohjelman osat on jaettu omiin taustaprosesseihinsa yhden ohjelmaprosessin sijaan (kuva 5). Nämä taustaprosessit ovat nimeltään RIPD (RIP-reititys), OSPFD (OSPFv2-reititys), BGPD (BGP4-reititys) sekä Zebra, joka vastaa Unix-ytimen reititystaulukkojen muutoksesta ja reittien jakamisesta muille taustaprosesseille. Toimintaperiaatteen ansiosta osa taustaprosesseista voidaan jättää aktivoimatta, jos niiden tarjoamaa reitityspalvelua ei tarvita, ja siten järjestelmästä saadaan modulaarisempi ja helpommin ylläpidettävä. Quaggan modulaarisuudesta johtuen jokaisella taustaprosessilla on oma konfiguraatiotiedostonsa ja terminaali-ikkunansa, joiden yksittäinen hallinta on työlästä. Tätä varten Quaggaan on luotu yhdistetty käyttöliittymä Vtysh, joka ottaa yhteyden jokaiseen taustaprosessiin Unix Domain Socketin avulla ja liittää ne yhdeksi terminaali-ikkunaksi. [10.]



Kuva 5. Quaggan järjestelmäarkkitehtuuri.

5 Verkon sisältämät palvelimet

Dummynet-palvelin

Dummynet (http://info.iet.unipi.it/~luigi/dummynet/) on verkkoliikenteen muokkaamiseen tarkoitettu ohjelma, jonka nykyinen versio tukee kaikkia suosituimpia käyttöjärjestelmiä. Dummynet-palvelin asennetaan tavallisimmin läpinäkyväksi verkkosillaksi, jonka läpi manipuloitava liikenne ohjataan. Työssä käytetyssä BSD-pohjaisessa käyttöjärjestelmässä Dummynet valikoi TCP/IP-protokollapinon läpi meneviä paketteja käyttöjärjestelmän oman IPFW-nimisen palomuurin sääntöjen perusteella ja pakottaa ne menemään ohjelman luomien objektien läpi, joita kutsutaan myös Dummynet-putkiksi ja -jonoiksi. Dummynet-putki on kanava, jolle on määritetty tietyt staattiset arvot, kuten sen läpi menevä maksimikaistanleveys, lisätty latenssi, prosentiaalinen pakettihäviö ja signaalin monitieilmiön häiriötä simuloiva muuttuja. Dummynet-jono on objekti, johon voidaan liittää halutun tyyppiset paketit, ja ne voidaan priorisoida painotusluvun perusteella saamaan enemmän kaistanleveyttä Dummynet-putkelta, jonka läpi ne kulkevat [11]. Nämä ominaisuudet tekevät Dummynetistä mainion ohjelman muun muassa verkkojen testausta varten, ja samasta syystä se myös valittiin tähän projektiin.

VLC-videopalvelin

VLC Media Player (http://www.videolan.org/vlc/) on avoimen lähdekoodin mediasoitin, joka tukee kaikkia suosituimpia käyttöjärjestelmiä. Sen kehittivät alun perin opiskelijat École Centrale Parisista, ja nykyään kehitystä valvoo voittoa tavoittelematon yritys nimeltä VideoLan. VLC Media Player on erittäin monipuolinen soitin, ja sillä on laaja tuki eri tiedostoformaateille. Videon toiston lisäksi VLC Media Player tukee myös videon lähettämistä verkon yli joko yksittäisenä tai ryhmälähetyksenä RTP/UDP-protokollalla. [12.]

VLC Media Player haluttiin ottaa mukaan tähän projektiin, koska siten projektin verkkoon saadaan mukaan liikennettä, jonka laadun pystyy toteamaan paljaalla silmällä ilman ylimääräisiä valvontaohjelmia. Sen luoma videoliikenne on häiriöherkkää ja reaaliaikaista, joten pienetkin verkkoliikenteen laadun heikkenemiset voivat aiheuttaa selvän muutoksen loppukäyttäjille näkyvässä kuvassa ja äänessä. Verkkoon on tarkoitus toteuttaa videopalvelimena toimiva työasema, jolta pystytään katsomaan reaaliaikaista kuvaa ja ääntä useammalta muulta verkon muissa osissa sijaitsevilta työasemilta. Normaali TCP-pohjainen tiedonsiirto ei ole erityisen altis virheille, koska protokolla sisältää itsessään pakettien oikeellisuuden varmistuksen ja verkkoliikenteen huonon laadun aiheuttamat vaikutukset on vaikea todeta ilman valvontaohjelmistoja muuten kuin pidentyneenä tiedonsiirtoon kuluvana aikana.

FTP-tiedostopalvelin

Videopalvelimen lisäksi verkkoon haluttiin lisätä FTP-palvelin, jolla pystytään generoimaan tavallisempaa TCP-pohjaista liikennettä. Tällä tavoin verkkoliikenne saadaan haluttaessa tasaisen kuormittavaksi ja siihen saadaan sisällytettyä useampia erityyppisiä paketteja mahdollista myöhempää Quality of Service palvelun hyödyntämistä varten. FTP-palvelinsovellukseksi valittiin avoimeen lähdekoodiin perustuva Proftpd (http://www.proftpd.org/), joka tukee tällä hetkellä Linux- ja Unix-pohjaisia käyttöjärjestelmiä.

Proftpd:n keskeisimpiä ominaisuuksia ovat:

- graafinen käyttöliittymä ladattavan Gadmin-lisämoduulin kautta
- yhteyden salaus SSH over FTP:llä
- modulaarinen rakenne, joka mahdollistaa vaihtoehtoisten ominaisuuksien lisäämisen tarpeen mukaan
- IPv6-tuki.

6 Virtualisoidun tietoverkon toteutus

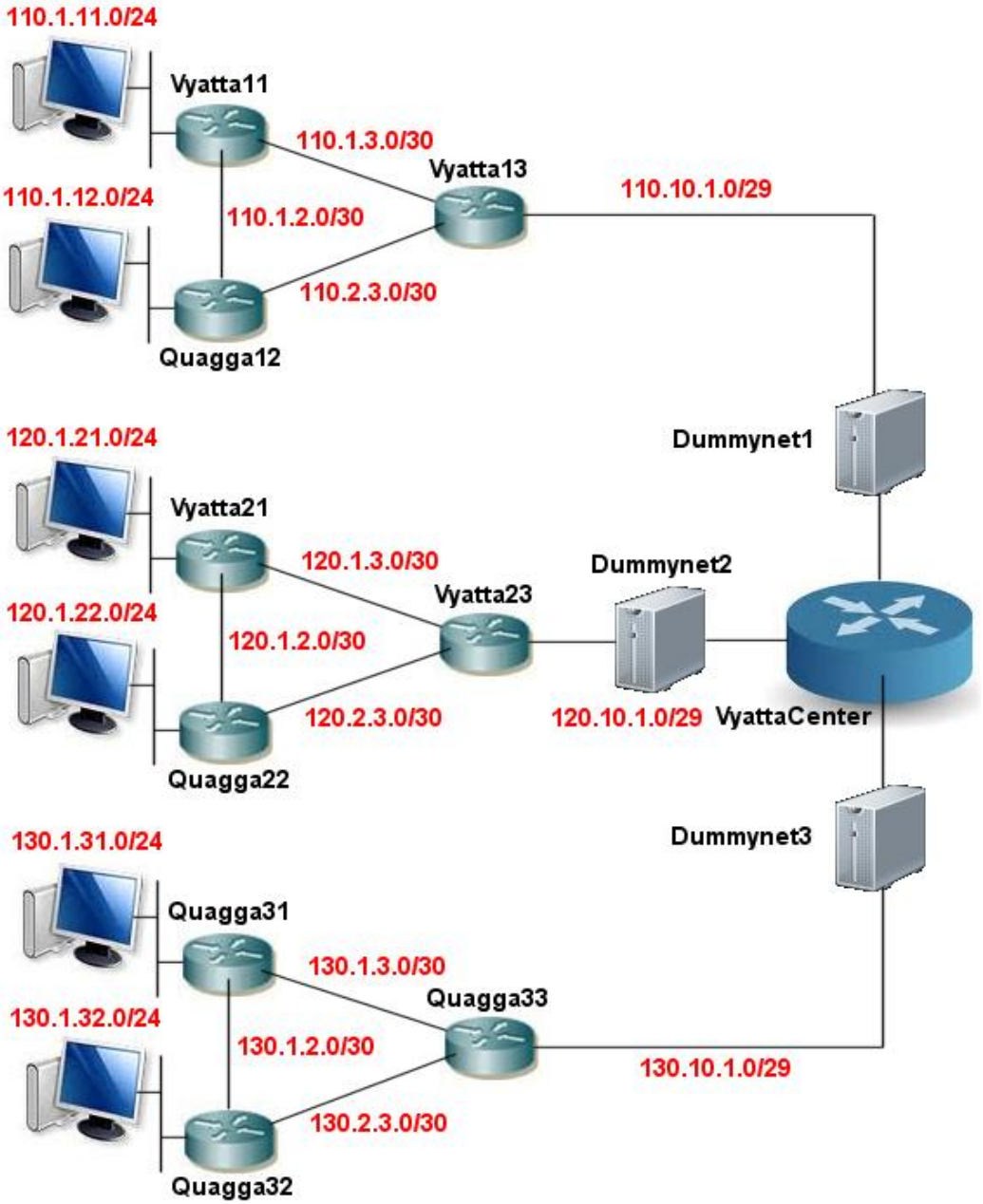
6.1 Verkon suunnittelu

Verkon suunnittelussa pyrittiin kiinnittämään huomiota siihen, että sen toteutus vastaisi reaalimaailman vaatimukset täyttävää tietoverkkoa virtualisoidun ympäristön tarjoamien mahdollisuuksien puitteissa. IP-osoitteisto on jaettu siten, että reitittimien väliset verkot käyttävät pieniä 4 bitin verkkoja, ja loppukäyttäjille tarkoitetut verkot ovat kooltaan 24-bittisiä. Tällä tavoin vapaat osoitteet kohdennetaan tehokkaasti vain sinne, missä niille on käyttöä. Loppukäyttäjille tarkoitettuja verkkoja on luotu yhteensä kuusi kappaletta, jotka on sijoitettu ympäristön sisäreitittimien taakse. Verkon IP-osoitteet on määritelty sillä oletuksella, että ympäristö tulee olemaan suljettu kokonaisuus ilman pääsyä julkiseen verkkoon. Siksi osoitteisto on pyritty luomaan enemmänkin selkeäksi ja helposti hahmotettavaksi kuin mahdollisimman tehokkaaksi reaalimaailman kannalta.

Verkon topologia

Työn tavoitteena on luoda verkkoympäristö, jolla pystytään simuloimaan laajempaa kokonaisuutta, jossa myös maantieteellisten erojen ja huonolaatuisten verkkoyhteyksien tuomat vaikutukset näkyisivät. Se koostuu kolmesta eri sisäverkosta, jotka ovat kytketty toisiinsa runkoverkossa toimivan keskusreitittimen kautta (kuva 6). Jokainen näistä sisäverkoista sisältää kolme toisiinsa kytkettyä reititintä, ja ne voidaan mieltää toisistaan erillisiksi lähiverkoiksi. Jokainen ulko- ja sisäverkon välinen linkkiyhteys kulkee verkkosiltana toimivien Dummynet-palvelinten kautta, jotta niiden läpi kulkevaa verkkoliikennettä voidaan manipuloida halutulla tavalla. Verkon loppukäyttäjinä toimivat Debian-työasemat, jotka on sijoitettu sisäreitittimille määriteltyihin käyttäjäverkkoihin. Nämä valinnat tehtiin verkon käyttötarkoitusta silmällä pitäen, jotta topologiassa olisi useita eri osia, joita voidaan käyttää joko yksinään tai toisiaan yhdistellen. Näin voidaan välttää tilannetta, jossa itse topologiaa

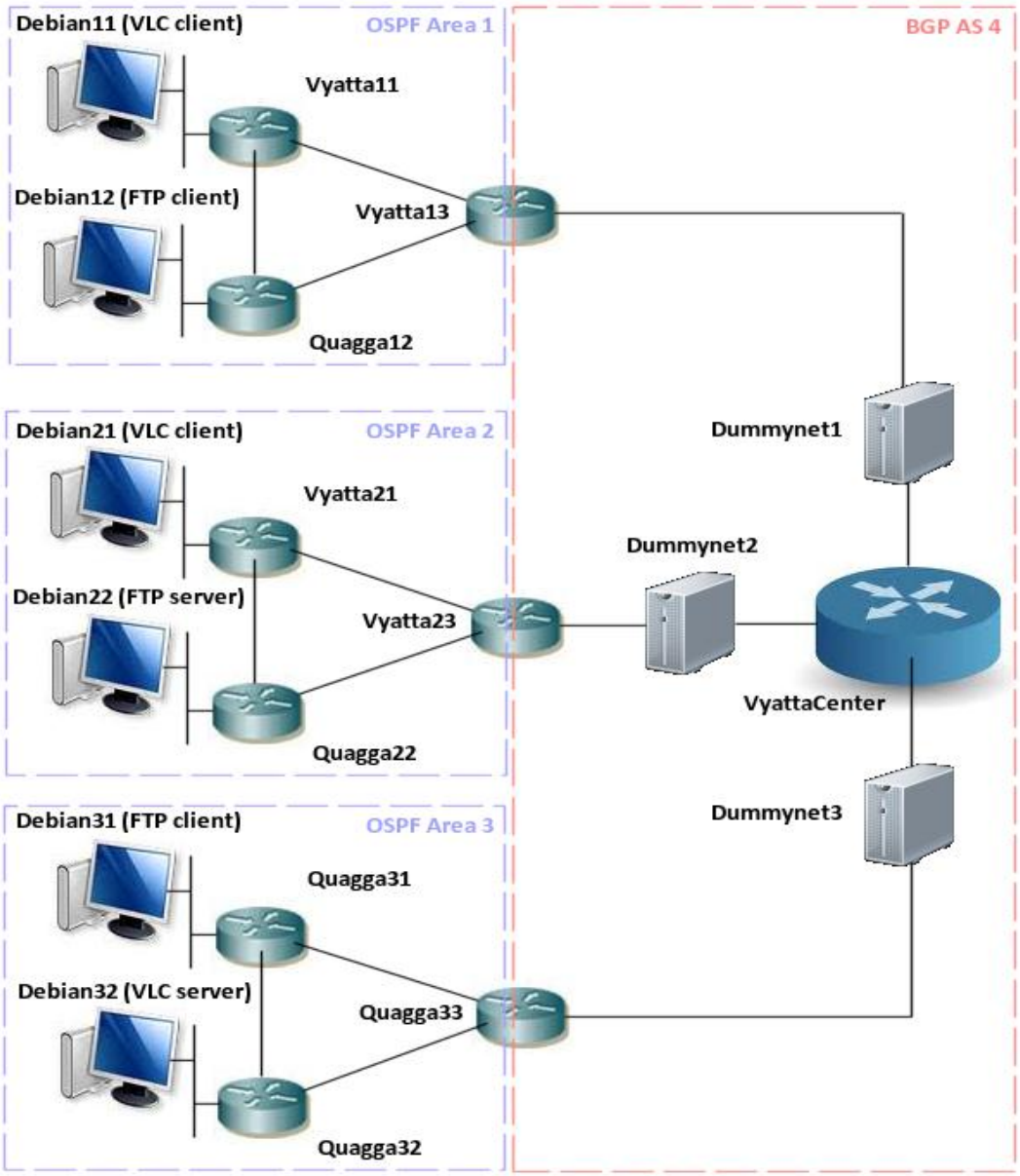
joudutaan alkaa muutella reitityskonfiguraatioiden sijaan, koska se on huomattavasti työläämpää.



Kuva 6. Verkon topologia.

Reititysalueiden määrittely

Verkon reititysprotokollat valittiin työlle määriteltyjen tavoitteiden perusteella, jotta niiden mukaisesti rakennetun ympäristön toiminta voidaan todentaa. Verkon reititysalueet on määritelty siten, että toteutus vastaisi mahdollisimman hyvin reaalimaailman suurta verkkoa eli internetiä. OSPF-protokolla soveltuu hyvin sisäverkkojen reititysprotokollaksi, koska se tukee vaihtuvanmittaista aliverkon peitettä ja alkukonfiguraation jälkeen sen hallittavuus on yksinkertaista. Näiden ominaisuuksien ansiosta OSPF valittiin hoitamaan sisäverkkojen reititys kolmessa eri OSPF-alueessa. Sisäverkkojen ulkopuolisen runkoverkon reititysprotokollaksi valittiin BGP, koska tähän tarkoitukseen tulevan reititysprotokollan tulee pystyä hallitsemaan suurta määrää reittejä kerralla tehokkaalla tavalla. Kahden reititysprotokollan käyttö lisää verkon monipuolisuutta, joka on työn yksi tavoite. Normaalista poiketen verkko ei sisällä OSPF Area 0:aa, koska BGP-alue hoitaa runkoverkon tehtävän kuvan 7 mukaisesti.



Kuva 7. Reititysalueiden määrittely.

6.2 VMware ESXi:n asennus ja konfigurointi

Palvelinohjelmiston asennus

VMware ESXi 4.1 asennettiin Dell PowerEdge 2850 -mallin palvelimelle, joka sisältää 2,8 Gigahertsin Intel Xeon -suorittimen ja 8 gigatavua muistia. Asennus sujuu vaivattomasti käynnistämällä palvelin esimerkiksi CD-levyltä, jolle VMwaren internet-sivuilta saatava levykuva on poltettu. Asennusohjelma on käytännössä automaattinen, ja sen valmistuttua palvelin on uudelleenkäynnistyksen jälkeen valmis konfiguroitavaksi. Palvelimelle tulee tämän jälkeen syöttää haluttu root-käyttäjätunnus, IP-osoite, aliverkon peite ja yhdyskäytävä, jotka määritellään ulkoisen verkon ehdoilla. Palvelimen IP-osoite toimii samalla virtuaalikoneiden yhdyskäytävänä, jos niille halutaan luoda yhteys virtuaaliympäristön ulkopuoliseen verkkoon.

Palvelinta kontrolloidaan tästä edespäin VMware vSphere Clientin kautta verkkoyhteyden välityksellä ottamalla yhteys sen IP-osoitteeseen ja kirjautumalla palvelimelle sille määritellyllä root-tunnuksella (kuva 8). VSphere Client on ESXi-palvelimen konfiguroimiseen tarkoitettu graafinen työkalu, joka on ladattavissa ilmaiseksi VMwaren verkkosivuilta.

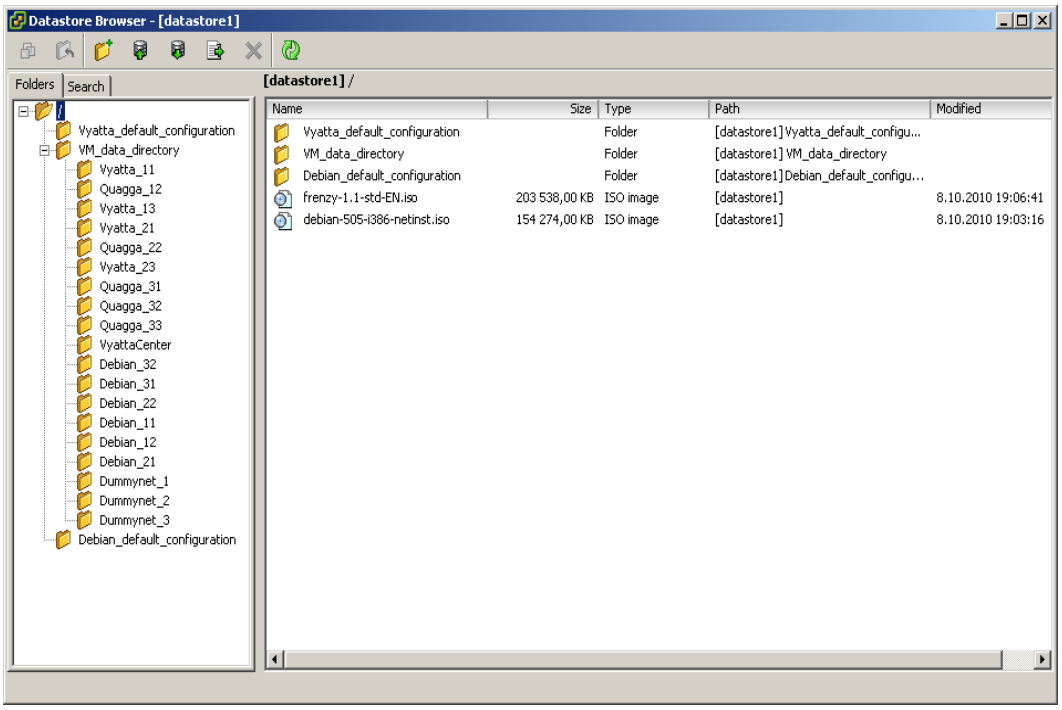
Kuva 8. Kirjautuminen VMware vSphere Clientiin.

Virtuaalisten verkkosegmenttien määrittely

Seuraavaksi palvelimelle tulee määritellä virtuaaliset verkkosegmentit, joiden tarkoitus on simuloida normaaleja verkkokytkentöjä. Jokaiselle ympäristön aliverkolle tulee luoda oma virtuaalinen verkkosegmentti, jotta verkkoliikenne käyttäytyisi vastaavan reaalimaailman toteutuksen tavoin. Jos jokainen virtuaalikone on määritelty samaan virtuaaliseen verkkosegmenttiin, verkkoliikenne ei noudata topologian mukaista reittiä, vaan se hyppää suoraan kohdeosoitteeseen. Jokaiselle virtuaaliselle verkkosegmentille pitää myös määritellä yksilöllinen virtuaalisen lähiverkon tunniste (VLAN ID), koska ESXi-palvelin käyttäytyy virtuaalikoneiden näkökulmasta verkkokytkimenä. Näin saavutetaan todellinen aliverkkojen eristys toisistaan, kuten reititettävässä verkossa tuleekin olla. Lisäksi Dummynet-palvelimiin kytketyissä verkkosegmenteissä tulee olla valikoimaton tila (Promiscuous Mode) aktiivisena, koska muuten paketit eivät välity oikein verkkosillan läpi virtuaalisessa ympäristössä. Dummynet-palvelimessa verkkoliikenteen välittämisen hoitavat tavanomaiset verkkokortit, jotka hylkäävät oletuksena jokaisen paketin, jonka päämääränä on jokin muu osoite kuin niiden oma. Promiscuous Mode muuttaa virtuaalisen verkkosegmentin verkkoliikennettä siten, että Dummynet-palvelin sallii myös muihin verkkoihin tai koneisiin kulkevan verkkoliikenteen mennä sen läpi. [13.]

Virtuaalikoneiden luominen

Ennen virtuaalikoneiden luomista on palvelimen levytilaan siirrettävä tarvittavat tiedostot eli tässä tapauksessa käyttöjärjestelmien levykuvat ja Vyatta-reitittimen virtuaalikoneen tiedostot. Siirto onnistuu Vsphere Client Datastore Browserista, jonka kautta työasemalla sijaitsevat tiedostot siirretään Upload-toiminnon avulla palvelimen levytilaan (kuva 9).



Kuva 9. Datastore Browseriin luotu hakemistorakenne.

Projektin sisältämien virtuaalikoneiden luonnissa kiinnitettiin huomiota erityisesti niille määriteltyihin resursseihin, kuten muistin ja levytilan määrään. Tämä johtui siitä, että suuri määrä virtuaalikoneita voi helposti viedä palvelimen koko resurssit, jos niiden asetuksia ei suhteuteta resurssitarpeen mukaan. Päämääräksi asetettiin minimoida virtuaali-koneille annetut resurssit kuitenkin siten, että ne pystyvät toimimaan vaaditulla tehokkuudella. Lisäksi reitityksestä ja Dummynet-palvelimista vastaavien virtuaali-koneiden verkkokorttien määrää kasvatettiin, jotta verkkotopologian mukaiset kytkennät saadaan toteutettua. Jotta verkko toimisi suunnitellun topologian mukaisesti, pitää jokaisen virtuaalikoneen verkkokortti vielä liittää sille kuuluviin virtuaalisiin verkkosegmentteihin, joita käsiteltiin aiemmassa luvussa.

Debian Linux ja Dummynet-ohjelmiston valmiiksi sisältävä FrenzyBSD-käyttöjärjestelmä ei ole valmiiksi virtuaalikoneen muodossa, vaan ne pitää asentaa erikseen aiemmin luotuihin virtuaalikoneisiin. Asennus sujuu kuten normaalilla tietokoneella, eli tässä tapauksessa määrittelemällä virtuaalikone käynnistymään sen omalta CD-asemalta, jonka mediaksi käyttöjärjestelmän levykuva on määritelty. Ajan säästämiseksi on viisainta asentaa ensin yksi Vyatta-, Debian- ja FrenzyBSD-virtuaalikone valmiiksi

sopivilla perusasetuksilla ja sen jälkeen tehdä siitä Datastore Browserin kautta tarvittava määrä kopioita, jotka voidaan myöhemmin yksilöidä niiden roolin mukaiseksi. Virtuaalikoneiden sisältämän ohjelmiston konfigurointi käydään läpi yksityiskohtaisemmin työn myöhemmissä luvuissa.

6.3 Virtualisoidun ohjelmiston toteutus

OSPF-reititys

OSPF valittiin sisäverkkojen reititysprotokollaksi, koska se täyttää projektin asettamat vaatimukset hyvin. Tärkeimmät niistä ovat automaattinen muutoksiin reagointi, joka vähentää uudelleenkonfiguroinnin tarvetta tehtäessä muutoksia verkkoon, ja tuki vaihtuvanmittaiselle aliverkon peitteelle, jonka ansiosta aliverkkojen koko voidaan määrittää tarkasti niiden roolin mukaiseksi. Vaihtoehtoja OSPF:lle ei juuri ollut, koska projektissa käytetty Quagga-reitittimen versio ei tue muita samantyylisiä reititysprotokollia, kuten esimerkiksi IS-IS:ää. Sisäverkkojen kytkentä suunniteltiin OSPF:ää silmällä pitäen, jotta verkossa olisi useampi kuin vain yksi reitti loppukäyttäjille saakka. Tämä pakottaa reititysprotokollan tekemään optimaalisen reittivalinnan ja myös tuo mahdollisuuksia toteuttaa erilaisia testejä ympäristössä, kun esimerkiksi yksi reitti katkaistaan.

OSPF-alueiden konfiguraatio toteutettiin siten, että kaikki kolme reititintä mainostavat OSPF:lle niihin kytkettyjä verkkoja omien alueidensa sisällä. Poikkeuksena tähän on rajareititin, joka on yhteydessä myös BGP-alueeseen. OSPF-alueet alkavat poikkeuksellisesti numerosta 1, koska BGP AS 4 toimii ympäristön runkoverkkona OSPF-alueen 0 sijaan. Tämä reititin levittää sisäverkkojen OSPF-reititystaulun BGP-protokollalle, jotta tieto yhden OSPF-alueen muutoksista päivittyy myös toisen alueen reitittimille. Levitys tehdään, koska ympäristö on tarkoitettu testi- ja tutkimuskäyttöön, jolloin siihen oletetaan tehtävän muutoksia usein. OSPF-alueiden sisimmille reitittimille lisättiin myös staattinen reitti ohjaamaan sisäverkon ulkopuolelle lähtevä liikenne

rajareitittimelle, koska niillä ei ole tietoa oman OSPF-alueensa ulkopuolisista verkoista johtuen BGP-protokollan konfiguraatiosta. Rajareititin taas näkee koko ulkoverkon ja osaa siten ohjata liikenteen eteenpäin oikealle reitittimelle. Seuraavassa OSPF:n esimerkkimäärittely Vyatta11:n näkökulmasta:

vyatta@Vyatta11# set protocols ospf area 1 network 110.1.11.0/24

BGP-reititys

BGP-protokolla valittiin runkoverkon reititysprotokollaksi, jotta ympäristö saataisiin monipuolisemmaksi. BGP hoitaa runkoverkon reitityksen Internetissä, joten sen valitseminen myös projektin ympäristön runkoverkon reititysprotokollaksi tuntui luonnolliselta. Sisäverkkojen rajareitittimien verkkoliitännöistä kaksi kuuluu OSPF-alueeseen, joten jäljelle jäävän liitännän tehtävä on toimia BGP-naapurina VyattaCenter-keskusreitittimelle. Rajareititin siis muuntaa OSPF-päivitykset BGP:n formaattiin ja lähettää ne edelleen keskusreitittimelle. Muuntoa ei kuitenkaan tehdä sisäverkon suuntaan BGP:stä OSPF:ksi johtuen siitä, että OSPF on sisäinen reititysprotokolla eikä sitä ole tarkoitettu käsittelemään suuria reititystauluja.

Rajareitittimien ulkoverkon puoleiset verkkoliitännät määriteltiin BGP-naapureiksi AS-tunnuksilla 1, 2 ja 3, ja ne muodostavat naapuruussuhteen keskusreitittimen kanssa, jonka AS-tunnus on 4. Tunnukset voitaisiin valita mielivaltaisesti väliltä 1–65536, mutta edellä mainitut arvot valittiin projektiin selvyyden vuoksi. Projektissa BGP-naapuruussuhteen muodostumisen ehdoksi määriteltiin erillinen salasana, joka pitää olla määritelty molemmille reitittimille. Tämä on BGP:n tietoturvaominaisuus, jotta naapurin oikeellisuus voitaisiin varmistaa. Tästä ei toistaiseksi ole varsinaista hyötyä, koska projektiympäristö on suljettu verkko, mutta tulevaisuudessa sitä voidaan hyvinkin muuttaa siten, että virtualisoidusta verkosta on pääsy julkiseen verkkoon. BGP-prosessi luo määritellystä salasanasta MD-5 -hash-merkkijonon, joka lähetetään toiselle reitittimelle, ja se vertaa sitä itse luomaansa vastaavaan merkkijonoon. Jos molemmat täsmäävät, voidaan naapuruussuhde muodostaa. Jotta OSPF-reitit välittyisivät BGP-protokollalle, pitää rajareitittimien BGP-määrityksiin lisää komento

"redistribute ospf" ja *"redistribute connected"*. Teoriassa pelkän OSPF-levityksen pitäisi riittää, mutta käytännössä yhteys lähti toimimaan vasta kun molemmat komennot lisättiin. Näin OSPF-reititystaulun sisältö välittyy runkoverkon keskusreitittimelle ja sitä kautta toisiin sisäverkkoihin. Seuraavassa on BGP:n esimerkkimäärittely Vyatta13:n näkökulmasta:

vyatta@Vyatta13# set protocols bgp 1 neighbor 110.10.1.4 remote-as 4

vyatta@Vyatta13# set protocols bgp 1 neighbor 110.10.1.4 password vyatta

Keskusreitittimelle luotiin lisäksi useita ylimääräisiä loopback-liitäntöjä, joita mainostettiin BGP:lle, jotta reititystaulun kokoa saataisiin kasvatettua ja rajareitittimille välittyisi tieto kuvitellusta ulkoverkosta realistisemmin. Vyattan ja Quaggan esimerkkikonfiguraatiot ovat liitteissä 1 ja 2.

Debian-työasema

Debian-työasemat toimivat työssä pohjana Quagga-reitittimille, VLC- sekä FTP-palvelimille ja loppukäyttäjien asiakasohjelmille. Debian Linux 5 valittiin näiden virtuaalikoneiden käyttöjärjestelmäksi, koska sillä on pitkäaikainen kehitys takanaan, ja sen ansiosta siitä on useita vakaita julkaisuja. Projektissa käytettiin Debian Linuxin Net Installer -asennuspakettia, jotta tarvittava levytila saataisiin minimoitua. Debian 5 on suoraan VMware ESXi:n tuettujen guest-käyttöjärjestelmien listalla, jonka ansiosta sen toimivuus virtuaalikoneessa voitiin olettaa hyväksi. Debian Linuxin edut projektin kannalta ovat helppo asennusprosessi ja kätevä ohjelmistonhallinnan työkalu nimeltä Aptitude, jolla ohjelmien asennus onnistuu internet-yhteyden välityksellä ilman tarvetta syvällisemmälle Linux-osaamiselle.

Sen sijaan, että jokainen Debian-työasema asennettaisiin erikseen alusta asti, ESXi-palvelimelle luotiin yksi virtuaalikone, joka sisältää Debian-asennuksen kaikilla projektiin tarvittavilla ohjelmilla, ja josta on helppo tehdä kopioita myöhemmin

tarvittavia työasemia varten. Ohjelmien asennusta varten virtuaalikone piti liittää väliaikaisesti julkiseen verkkoon antamalla sille IP-osoite samasta aliverkosta ESXi-palvelimen kanssa ja määrittelemällä verkon Domain Name System (DNS) -palvelin. Sille myös määriteltiin ylimääräisiä Debian-ohjelmavarastoja Aptituden konfiguraatiotiedoston kautta, jotta sen tarjoama ohjelmistovalikoima olisi laajempi. Käyttöjärjestelmään asennettiin ensin graafinen käyttöliittymä Xfce4, jotta sen käyttö helpottuisi. Tämän jälkeen asennettiin muut projektissa tarvitut ohjelmat:

- Quagga
- VLC Player
- Proftpd + Gadmin-Proftpd
- Filezilla.

Seuraavassa on Aptituden asennuskomento Quaggan tapauksessa:

debian@Quagga12:/# aptitude install quagga

Quagga-reitittin

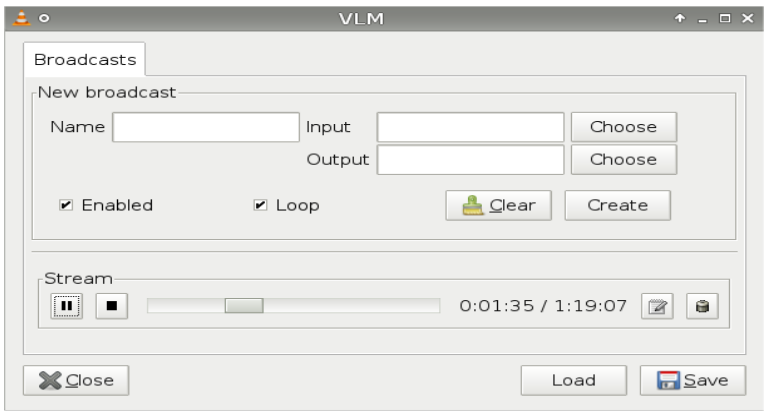
Tässä osiossa käydään läpi Quagga-reitittimen toimintakuntoon saattaminen. Ensimmäisenä Debian-käyttöjärjestelmästä tulee aktivoida IP Forwarding, jotta se pystyy välittämään verkkoliikenteen verkkokortilta toiselle. Käyttöjärjestelmän IP-osoitteiden määritys ei ole välttämätöntä tässä tapauksessa, koska Quagga tekee sen oman verkkoliitäntäkonfiguraationsa perusteella. Jotta Quaggan taustaprosessit voitaisiin käynnistää, niiden konfiguraatiotiedostot on luotava ensin hakemistoon */etc/quagga.* Alkukonfiguraation tulee sisältää vain reitittimen nimen ja kirjautumiseen vaaditun salasanan. Kun tarpeelliset Quagga-taustaprosessien konfiguraatiotiedostot on luotu, käynnistetään niitä hallitsevat prosessit lisäkomennolla &, jotta Debian käynnistää ne taustalle. Seuraavassa on esimerkkinä Zebran käynnistys:

debian@Quagga12:/# zebra &

Kun kaikki prosessit on saatu käyntiin, voidaan Quaggan käyttöliittymään ottaa yhteys komennolla *vtysh*, jonka jälkeen reitittimelle voidaan alkaa syöttämään käskyjä. [14.]

VLC-videopalvelin

VLC-videopalvelin asetettiin toimimaan Debian32-työasemalla, jossa se toistaa DVD-tasoista videota aloittaen sen aina alusta uudelleen automaattisesti. Näin videon toistoa ei tarvitse aloittaa aina uudelleen, vaan yhteys voidaan muodostaa asiakasohjelmiston päästä aina tarpeen mukaan. Palvelimen mediaksi valittiin teostovapaa DVD-tasoinen video, jotta videoliikenteellä olisi korkea vaatimustaso esimerkiksi bittinopeuden suhteen. Projektissa käytetyn videon formaatti on Ogg Video (.ogv), jonka takia videolähetyksen kapsulointityypiksi valittiin myös Ogg-formaatti. Videolähetyksen toimitus taas tapahtuu HTTP-protokollalla, koska projektin ympäristössä se on ainoa VLC Playerin vaihtoehto välittää videota useaan koneeseen samanaikaisesti johtuen siitä, että Quagga ei toistaiseksi tue RTP-ryhmälähetystä. HTTP:n ja RTP:n keskeisin ero on projektin kannalta tapa, jolla ne välittävät verkkoliikenteen kohteeseensa. HTTP käyttää varmennettua TCP-tiedonsiirtoa, joka pysäyttää videon toiston ja tukeutuu datan puskurointiin, jos verkkoliikenne on huonolaatuinen. RTP:tä käytettäessä kuva päivittyy tasaisesti, jolloin videon toisto ei pysähtele, mutta sen laadun heikkeneminen huomataan välittömästi virheinä kuvassa. [15.] Videolähetyksien kontrollointi VLC Playerissä tapahtuu VLM Control -valikon kautta (kuva 10).



Kuva 10. VLM Control-valikko.

Videolähetystä voidaan katsoa muilta työasemilta käsin ottamalla yhteys VLC-palvelimen IP-osoitteeseen VLC Playerin Network Streaming valikon kautta. Asiakasohjelmiston videotoistoasetuksista ulostuloformaatti muutettiin Linuxin omaan X11 Output Moduleen, jotta video toistuisi oikein.

FTP-tiedostopalvelin

FTP-palvelinohjelmistoksi valittiin ProFTP-Gadmin sen selkeän graafisen käyttöliittymän ja helpon käytettävyyden vuoksi, ja se päätettiin sijoittaa Debian22-asemalle. Palvelimelle tehtiin yksinkertainen peruskonfiguraatio, jossa sille luotiin käyttäjätunnus "*debian",* ja määritettiin palvelimen IP-osoite, joka on tässä tapauksessa sama kuin käyttöjärjestelmän verkkoliitännän osoite. Idle-timeout -laskurien arvo asetettiin nollaan, jotta palvelin ei katkaise yhteyttä asiakasohjelmistoon automaattisesti tietyn ajan päästä, jos käyttäjä ei ole aktiivinen. Debian-käyttöjärjestelmän hosts-tiedostoa piti muuttaa siten, että käyttöjärjestelmän nimi on määritetty oikein localhost-osoitteeseen 127.0.0.1. Ilman tätä määritystä FTP-palvelinta ei saa aktivoitua. Työasemien FTP-asiakasohjelmistoksi valittiin helppokäyttöinen Filezilla.

Dummynet-palvelin

Virtualisoidun toteutuksen takia viive tai muu häiriö on verkossa käytännössä olematonta, koska liikenne kulkee virtuaalisesti saman palvelimen sisällä eikä fyysisen tai langattoman median kautta. Yhdeksi projektin keskeisimmäksi tavoitteeksi asetettiin juuri mahdollisuus simuloida muun muassa maantieteellistä etäisyyttä ja muuten huonolaatuista verkkoyhteyttä. Dummynet-palvelimet soveltuvat tähän tarkoitukseen erittäin hyvin, ja niiden ansiosta verkkoympäristöä voidaan käyttää testialustana mittaamaan huonolaatuisen verkon aiheuttamaa vaikutusta verkkoliikenteeseen.

Frenzy BSD valittiin Dummynet-palvelimien käyttöjärjestelmäksi, koska tarvittava ohjelmisto on jo valmiiksi asennettuna siihen. Käyttöjärjestelmälle tulee määritellä verkkoliitäntöjen osoitteet ja luoda verkkosilta niiden välille, sekä aktivoida IP Forwarding, jotta verkkoliikenne pystytään ohjaamaan palvelimen läpi. Jokaisen Dummynet-palvelimen verkkoliitännöille tulee myös luoda oma yksilöllisellä VLAN-tunnuksella varustettu virtuaalinen verkkosegmentti ESXi-palvelimelta käsin, jotta verkkosilta toimisi kuten fyysisessä verkossa. Kuitenkin nämä erilliset segmentit sijaitsevat samassa aliverkossa, koska Ethernet-sillalla ei ole reititysominaisuuksia. Tällä konfiguraatiolla varmistetaan, että verkkoliikenne menee aina Dummynet-palvelimen läpi kummankin verkkokortin kautta, eikä pääse oikaisemaan Ethernet-sillan ohi. Seuraavassa esimerkissä Dummynet1:lle määritetään 10 Mbps:n bittinopeus, 1 %:n pakettihävikki ja 25 ms:n viive:

Frenzy:~# ipfw pipe 1 config bw 10000Kbit/s plr 0.01 delay 25ms

6.4 Video- ja dataliikenteen testaus

Seuraavaksi verkkoympäristölle tehtiin pienimuotoisia testejä. Aloitetaan mittaamalla verkkoyhteyden viive kahden pisteen välillä, mikä onnistuu Ping-komennolla (kuva 11).

```
Debian12:/home/debian# ping 120.1.22.2
PING 120.1.22.2 (120.1.22.2) 56(84) bytes of data.
64 bytes from 120.1.22.2: icmp_seq=1 ttl=59 time=167 ms
64 bytes from 120.1.22.2: icmp_seq=2 ttl=59 time=156 ms
64 bytes from 120.1.22.2: icmp_seq=3 ttl=59 time=155 ms
64 bytes from 120.1.22.2: icmp_seq=4 ttl=59 time=156 ms
64 bytes from 120.1.22.2: icmp_seq=5 ttl=59 time=157 ms
^C
--- 120.1.22.2 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4013ms
rtt min/avg/max/mdev = 155.076/158.569/167.740/4.639 ms
Debian12:/home/debian#
```

Kuva 11. Verkkoliikenteen viiveen mittaus.

Tuloksista nähdään, miten Dummynet-palvelimet vaikuttavat verkkoliikenteen viiveeseen. Ilman keinotekoisesti lisättyä viivettä virtualisoidun verkon tuottama viive on käytännössä olematon, koska verkkoliikenne ei kulje erillisessä mediassa lainkaan. Tässä testissä Dummynet-konfiguraatio ei sisältänyt erillistä pakettihäviötä.

Kokeillaan tämän jälkeen VLC Playerin kautta videon toistoa Debian32:lla sijaitsevalta videopalvelimelta. Asetetaan ensimmäistä testiä varten Dummynet-palvelimille seuraavat arvot:

- bittinopeus: 10 Mbps
- viive: 25 ms.

Video toistuu tasaisesti eikä kuvassa ei ole häiriötä, kuten voidaan nähdä kuvasta 12.

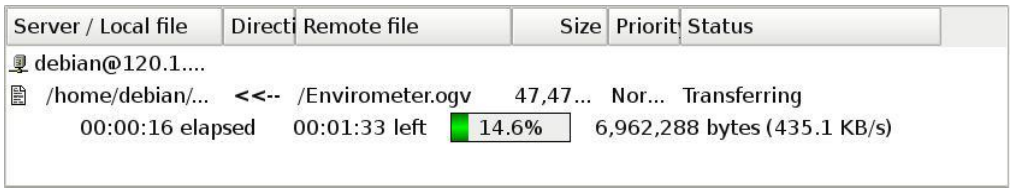
Kuva 12. Videoliikenteen testaus ilman pakettihäviötä.

Seuraavaksi lisätään aiempaan konfiguraatioon kolmanneksi parametriksi pakettihäviö arvolla ”plr 0.03”, jolloin Dummynet-palvelin hylkää sen läpi kulkevat paketit 3 %:n todennäköisyydellä:

Kuva 13. Videoliikenteen testaus 3 % pakettihäviöllä.

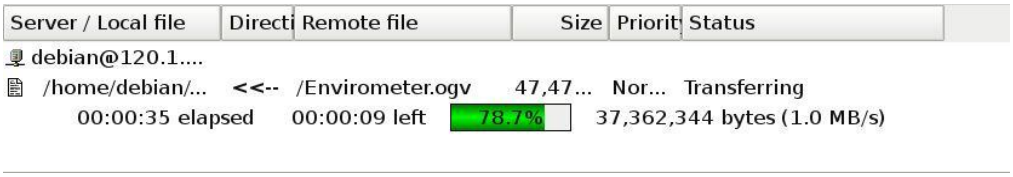
Kuvasta 13 voidaan todeta, että 3 % pakettihäviö vaikuttaa kuvanlaatuun selvästi. Toinen selkeä vaikutus on videon pysähtely, joka johtuu siitä, että videoliikennettä joudutaan puskuroimaan paljon, koska ohjauspaketit eivät aina saavu perille asti.

Lopuksi testataan vielä verkkoliikenteen muutosten vaikutusta FTP-tiedonsiirtoon. Säilytetään aiemmat Dummynet-asetukset samana, paitsi että muutetaan viive 50 millisekuntiin (kuva 14):



Kuva 14. Tiedonsiirto viiveellä 50 ms.

Sama testi suoritettiin tämän jälkeen ilman verkkoliikenteen viivettä (kuva 15):



Kuva 15. Tiedonsiirto ilman viivettä.

Kuten kuvista näkyy, verkkoliikenteen viive vaikuttaa olennaisesti tiedonsiirron nopeuteen. Tämä johtuu siitä, että TCP-pohjaisessa tiedonsiirrossa lähetetään runsaasti ylimääräisiä paketteja lähettäjän ja vastaanottajan välillä, jotta tiedoston virheetön siirtyminen voidaan varmistaa. [16, s. 72.] Verkon viive hidastaa tätä kommunikaatiota, mikä taas vaikuttaa itse tiedonsiirtoon ratkaisevasti.

ESXi-palvelimen resurssien käyttöä valvottiin Vsphere Clientin kautta testien yhteydessä, jotta niiden vaikutukset palvelimen resurssitarpeelle saataisiin selville. Kategorioiksi valittiin prosessorin, muistin ja verkkokortin käyttöasteet, koska nämä ovat rajoittavia tekijöitä virtualisoidun ympäristön kasvattamista silmällä pitäen. Erityisesti muistin käyttö on olennaista, koska se määrittää suoraan aktiivisena olevien virtuaalikoneiden maksimimäärän. Kuvaajista selvisi, että palvelin suoriutuu hyvin projektin ympäristön rasituksesta, kun video- ja dataliikennettä generoidaan verkon läpi. Muistin käyttö pysyi tasaisesti 30 %:ssa, mikä antaa mahdollisuuden tuntuvalle ympäristön laajentamiselle. Eri resurssien käyttöastetta kuvaavat kuvaajat ovat liitteessä 3.

7 Dokumentointi

Tämän insinöörityön toteutusvaiheessa syntyi suuri määrä dokumentaatiota aina verkon topologiakuvauksesta reitittimien ja käyttöjärjestelmien konfiguraatio-listaukseen. Itse työssä kuitenkin pyrittiin kertomaan jokaisesta osa-alueesta ja niissä tehdyistä päätöksistä yleisemmällä tasolla tiettyjen konfiguraatioesimerkkien kanssa, koska täydet konfiguraatiolistaukset itsessään eivät edistä työn luettavuutta. Tarkemmat konfiguraatioesimerkit sijoitettiin sen sijaan työn liitteisiin 1-2.

Toteutetun verkon jatkokehitystä ja ylläpitoa varten siitä toimitettiin erillinen dokumentti Metropolia Ammattikorkeakoululle, joka sisältää verkon topologia-kuvauksen, verkkoliitäntöjen IP-osoitteet ja niiden virtuaalisen verkkosegmentin sekä palvelimien ja käyttöjärjestelmien käyttäjätunnukset. Dokumentin päämäärä oli antaa tarpeellinen määrä tietoa verkosta, jotta sitä hallitseva henkilö saa hyvän yleiskuvan verkkoympäristöstä ja pystyy aloittamaan sen hallinnan.

8 Johtopäätökset ja kehitysmahdollisuudet

Tämän työn tarkoituksena oli tutkia mahdollisuutta virtualisoida ilmaisohjelmilla toteutettu monimutkainen tietoverkko toimimaan yksittäisen fyysisen palvelimen sisällä, mitä voitaisiin myöhemmin hyödyntää Metropolia Ammattikorkeakoulun laboratorio- ja opetuskäytössä. Virtualisoinnin avulla fyysiset laitteet voidaan muuttaa ohjelmallisiksi virtuaalikoneiksi, joita voidaan luoda palvelimelle tarpeen mukaan sen fyysisten resurssien rajoissa. Tämä ominaisuus antaa ympäristölle monipuolisen muokattavuuden, koska vastaavan fyysisen ympäristön toteutus ei välttämättä olisi resurssien puitteissa mahdollista. Muita työn kannalta olennaisia ominaisuuksia virtualisoidussa ympäristössä ovat virtuaalikoneiden helppo siirrettävyys ja kopiointi, mahdollisuus palauttaa virtuaalikone aikaisemmin talletettuun tilaan ja niiden helppo keskitetty hallinta.

Työn varsinainen haaste oli luoda monipuolinen verkkoympäristö, jossa jokainen osa pystytään toteuttamaan ilmaisohjelmistoilla. Verkkoon valitut reitittimet olivat käytännössä ainoat varteenotettavat vaihtoehdot tällä hetkellä, mutta ne toisaalta sopivat työn tavoitteisiin erittäin hyvin ominaisuuksiensa ja eroavaisuuksiensa puolesta. Niiden avulla verkkoon pystyttiin luomaan kahden eri reititysprotokollan avulla toimiva tietoverkko, jolla virtuaalikoneet kytkettiin kiinni toisiinsa. Muissa ohjelmistovalinnoissa turvauduttiin varmatoimisiin ja yksinkertaisiin ratkaisuihin, jotta verkolle asetetut tavoitteet saavutettaisiin. Yhteen fyysiseen palvelimeen virtualisoidun verkon luonteen takia siinä kulkevaa liikennettä oli olennaista pystyä manipuloimaan keinotekoisesti, jotta verkko saataisiin käyttäytymään vastaavan reaalimaailman toteutuksen tavoin. Tämäkin tavoite saavutettiin alkuperäisten kriteerien puitteissa.

Työn lopputuloksena saavutettu verkko vastasi hyvin sille asetettuja tavoitteita ja näytti, että virtualisointi soveltuu monimutkaisen tietoverkon toteutukseen. Työn lopuksi video- ja dataliikenteellä tehdyt testit osoittivat, että verkkoliikenteen laatuun voidaan vaikuttaa tilanteesta riippuen olennaisesti verkkoon lisätyllä manipulointi-mekanismilla, joka on olennainen ominaisuus verkon käyttötarkoitusta ajatellen. Henkilökohtaisella tasolla työ lisäsi tuntemustani virtualisoinnista ja tietoverkkojen toteutuksesta, mutta ennen kaikkea Linux-osaamiseni karttui tuntuvasti.

Työssä toteutetulla tietoverkolla on lukuisia kehitysmahdollisuuksia laboratorio- ja opetuskäytössä. Muokattavuuden ansiosta sillä pystytään toteuttamaan opetuskäyttöä varten konfiguraatioita, joihin ei syystä tai toisesta ole tällä hetkellä saatavilla fyysistä laitteistoa. Virtualisoitu verkko voitaisiin myös liittää osaksi Metropolia Ammattikorkeakoulun sisäverkkoa, joka mahdollistaisi virtuaalikoneiden pääsyn julkiseen verkkoon. Se tarjoaa myös hyvän pohjan tuleville insinööritöille, koska verkkoa pystytään kehittämään vapaasti palvelimen resurssien puitteissa. Kehitysideoita voisi olla esimerkiksi verkon tietoturvan kehittäminen ja testaaminen hyökkäysohjelmistoilla tai uusien palveluiden lisääminen verkkoon.

Lähteet

1. The Business Value Of Virtualization. (WWW-dokumentti.) WMware, 2009. <http://www.vmware.com/files/pdf/solutions/Business-Value-Virtualization.pdf>.

2. Golden, Bernard. Virtualization For Dummies 2nd Edition. (WWW-dokumentti.) Wiley Publishing Inc, 2009. <https://dct.sun.com/dct/forms/reg_us_0506_789_0.jsp>.

3. Dittner, Rogier & Rule, David. Best Damn Server Virtualization Book Period. Syngress Publishing Inc, 2007.

4. WMware White Paper: Understanding Full Virtualization, Paravirtualization And Hardware Assist. (WWW-dokumentti.) WMware, 2007. <http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf>.

5. The Architecture of VMware ESXi. (WWW-dokumentti.) WMware, 2008. <http://www.vmware.com/files/pdf/vmware_esxi_architecture_wp.pdf>

6. WMware ESXi and ESX Comparison. (WWW-dokumentti.) WMware. <http://www.vmware.com/products/vsphere/esxi-and-esx/compare.html>. Luettu 15.1.2011.

7. OSPF Design Guide. (WWW-dokumentti.) Cisco Systems. <http://www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a0080094e9e.shtml>. Luettu 21.1.2011.

8. Border Gateway Protocol. (WWW-dokumentti.) Cisco Systems Docwiki. <http://docwiki.cisco.com/w/index.php?title=Border_Gateway_Protocol&oldid=23504>. Päivitetty 17.12.2009. Luettu 18.1.2011.

9. Vyatta Virtual Datasheet. (WWW-dokumentti.) Vyatta. <http://www.vyatta.com/sites/vyatta.com/files/pdfs/vyatta_virtual_datasheet.pdf>

10. Ishiguro, Kunihiro. Quagga Routing Suite Documentation. (WWW-dokumentti.). <http://www.quagga.net/docs/docs-info.php#SEC1>. Luettu 27.12.2010.

11. Rizzo, Luigi. Dummynet Documentation. (WWW-dokumentti.). <http://info.iet.unipi.it/~luigi/dummynet/>. Luettu 10.11.2010.

12. VLC Player Features. (WWW-dokumentti.) VideoLan. <http://www.videolan.org/vlc/features.html>. Luettu 5.4.2011.

13. Promiscuous Mode. (WWW-dokumentti.). <http://en.wikipedia.org/w/index.php?title=Promiscuous_mode&oldid=417254822>. Päivitetty 5.3.2011. Luettu 5.4.2011.

14. Ishiguro, Kunihiro. Quagga Routing Suite Documentation. (WWW-dokumentti.). <http://www.quagga.net/docs/docs-info.php#SEC123>. Luettu 5.4.2011.

15. Denis-Courmont, Rémi. Video On Demand: RTSP Vs HTTP. (WWW-dokumentti.). <http://www.remlab.net/op/vod.shtml>. Päivitetty 8.7.2009. Luettu 5.4.2011.

16. Blank, Andrew G. TCP/IP Jumpstart. Sybex Inc, 2000.

Liite 1: Vyatta-reitittimen esimerkkikonfiguraatio

```
interfaces {
    ethernet eth0 {
        address 110.1.3.2/30
        duplex auto
        hw-id 00:0c:29:58:13:cc
        speed auto
    }
    ethernet eth1 {
        address 110.2.3.2/30
        duplex auto
        hw-id 00:0c:29:58:13:d6
        speed auto
    }
    ethernet eth2 {
        address 110.10.1.1/29
        duplex auto
        hw-id 00:0c:29:58:13:e0
        speed auto
    }
    loopback lo {
    }
}
protocols {
    bgp 1 {
        neighbor 110.10.1.4 {
            password vyatta
            remote-as 4
        }
        redistribute {
            connected {
            }
            ospf {
            }
        }
    }
    ospf {
        area 1 {
            network 110.1.3.0/30
            network 110.2.3.0/30
         }
    }
}
service {
    https
}
system {
    host-name Vyatta13
```

```
    login {
        user root {
            authentication {
                encrypted-password $1$$Ht7gBYnxI1xCdO/JOnodh.
            }
            level admin
        }
        user vyatta {
            authentication {
                encrypted-password $1$$Ht7gBYnxI1xCdO/JOnodh.
            }
            level admin
        }
    }
    ntp-server 69.59.150.135
    package {
        auto-sync 1
        repository community {
            components main
            distribution stable
            password ""
            url http://packages.vyatta.com/vyatta
            username ""
        }
    }
    time-zone GMT
}
```

Liite 2: Quagga-reitittimen esimerkkikonfiguraatio

```
! Zebra configuration saved from vty
!   2010/08/15 14:31:49
!
hostname Quagga33
password quagga
!
interface eth0
 ip address 130.1.3.2/30
 ipv6 nd suppress-ra
!
interface eth1
 ip address 130.2.3.2/30
 ipv6 nd suppress-ra
!
interface eth2
 ip address 130.10.1.1/29
 ipv6 nd suppress-ra
!
interface lo
!
ip forwarding
!
router ospf
 network 130.1.3.0/30 area 0.0.0.3
 network 130.2.3.0/30 area 0.0.0.3
!
router bgp 3
 bgp router-id 130.10.1.1
 network 130.10.1.0/29
 redistribute connected
 redistribute ospf
 neighbor 130.10.1.4 remote-as 4
 neighbor 130.10.1.4 password vyatta
!
!
line vty
!
```

Liite 3: ESXi-palvelimen resurssikäytön kuvaajat