TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

RONI IJÄS

SOFTWARE-DEFINED NETWORKING KONESALIVERKKOJEN VIRTUALISOINNISSA

Kandidaatintyö

Tarkastaja: Taneli Riihonen
22.12.2017

TIIVISTELMÄ

Software-Defined Networking konesaliverkkojen virtualisoinnissa

TOIMIKUNTA: Tieto- ja sähkötekniikan tiedekunta
Tampereen teknillinen yliopisto
Kandidaatintyö, 32 sivua
Joulukuu 2017
Tietotekniikan kandidaatin tutkinto-ohjelma
Pääaine: Tietoliikennetekniikka
Tarkastaja: Taneli Riihonen

Avainsanat: Software-Defined Networking, SDN, verkko, konesali, virtualisointi

Työn aiheena on tutustua SDN-arkkitehtuuriin ja selvittää, miten se soveltuu osaksi konesaliverkkojen virtualisointia. Työssä käydään läpi verkkoarkkitehtuurien historiaa, sekä tarpeita, joista SDN-arkkitehtuuri on syntynyt. Työssä käydään läpi muutamia avoimen ja suljetun lähdekoodin SDN-sovelluksia, sekä esitellään verkkojen virtualisoinnin perusteet. Tarkempaan käsittelyyn valitaan avoimen ja suljetun lähdekoodin sovellukset, jotka yhdistävät sekä SDN-arkkitehtuurin, että verkkojen virtualisoinnin. Tutkimustulokseksi saadaan, että SDN-arkkitehtuurista on konkreettista hyötyä konesaliverkkojen virtualisoinnista. Saavutettava hyöty riippuu kuitenkin lähtökohdista. Olemassa olevassa konesalissa SDN-sovellus on riippuvainen kytkinlaitteistosta. Uudessa konesalissa SDN-sovellus voidaan valita vapaammin.

ABSTRACT

Software-Defined Networking in Datacenter Network Virtualization

Keywords: Software-Defined Networking, SDN, network, datacenter, virtualization

The purpose of this thesis is to find out, how SDN architecture can be used as a part of datacenter network virtualization. The report starts with a segment of computer network history and discusses the reasons why SDN architecture is needed. The study is then continued with an overview of few open and closed source SDN applications and the basics of network virtualization. The following section will evaluate in more detail one open source application and one commercial application that utilize both SDN architecture and network virtualization techniques. The thesis concludes, that SDN architecture benefits datacenter network virtualization. The gained benefits depend on datacenter conditions. When the SDN application is used in an existing datacenter, compatibility greatly depends on the used hardware switches. While SDN application deployment in a new datacenter can be done more freely.

SISÄLLYSLUETTELO

KUVALUETTELO

LYHENTEET JA MERKINNÄT

API	Application Programming Interface, ohjelmointirajapinta
ASIC	Application Specific Integrated Circuit, sovelluskohtainen mikropiiri
ATM	Asynchronous Transfer Mode, tahdistamaton siirtotapa
DPDK	Data Plane Development Kit, pakettimuotoisen datan prosessointiin suunniteltu kehitystyökalu
ETSI	European Telecommunications Standards Institute, standardointiorganisaatio
GSMP	General Switch Management Protocol, välityslaitteiston hallintaprotokolla
HTTP	Hypertext Transfer Protocol, linkitetyn tekstin siirtoon käytetty protokolla
IEEE	Institute of Electrical and Electronics Engineers, kansainvälinen tekniikan alan järjestö
IETF	Internet Engineering Task Force, internet-protokollien standardointiorganisaatio
IoT	Internet of Things, Esineiden internet
IP	Internet Protocol, internetin siirtoprotokolla
MAC	Medium Access Control, verkon varaamisen ja liikennöinnin osajärjestelmä
MANO	Management and Orchestration, verkkoresurssien hallinnan kehysympäristö
MIB	Management Information Base, tietokantajärjestelmä
NFV	Network Functions Virtualization, verkkotoimintojen virtualisointi
OF-CONFIG	OpenFlow Management and Configuration Protocol, OpenFlow laitteiden hallinnan mahdollistava protokolla
OF-DPA	OpenFlow Data Plane Abstraction, mahdollistaa OpenFlow-protokollan käytön Broadcom:n sovelluskohtaista piirisarjaa käyttävissä laitteissa
ONF	Open Networking Foundation, säätiö
OVS	Open vSwitch, avoimen lähdekoodin virtuaalikytkin
OVSDB	Open vSwitch Data Base management, hallintaprotokolla
PCI DSS	Payment Card Industry Data Security Standard, standardi korttimaksujen turvallisuuden teknisiin vaatimuksiin
QoS	Quality of Service, tietoliikenteen luokittelu ja priorisointi
REST	Representational State Transfer, HTTP-protokollaan perustuva arkkitehtuurimalli
RFC	Request for Comments, IETF:n julkaisema internetiä koskeva standardi
SMI	Structure of Management Information, SNMP kehysympäristö
SNMP	Simple Network Management Protocol, verkonhallintaprotokolla
SDN	Software-Defined Network, ohjelmallisesti määritetty verkko
VLAN	Virtual Local Area Network, virtuaalilähiverkko
VXLAN	Virtual eXtensible Local Area Network, virtuaalilähiverkon laajennos

1. JOHDANTO

Tämän työn tarkoitus on luoda katsaus Software-Defined Networking -arkkitehtuuriin, eli ohjelmallisesti määriteltyyn verkkoon ja tarkastella sen pohjalta tapoja hyödyntää sitä osana konesaliverkkojen virtualisointia. Vuosien 2012–2014 aikana modernin konesalin koko kaksinkertaistui [1]. Tämän trendin jatkuessa verkkoinfrastruktuurin ketterä hallinta on noussut keskeiseen asemaan. Perinteisesti konesalin verkkoinfrastruktuuri rakennetaan kerran, jonka jälkeen se pysyy muuttumattomana, mutta kasvaa kooltaan suuremmaksi. Esineiden internet, eli IoT (Internet of Things), ja pilvipalvelut ovat kuitenkin luoneet sellaisia vaatimuksia, jotka pakottavat myös konesaleja uudistumaan.

Perinteinen verkkoarkkitehtuuri perustuu verkkolaitteiden väliseen tiedonsiirtoon. Näistä laitteista käytetään yleisnimitystä kytkin. Kytkimen tehtävä on vastaanottaa ja reitittää eteenpäin saapuva pakettimuotoinen tieto. Laitteiden ja ohjelmistojen kehitys on johtanut tilanteeseen, jossa verkkolaitteissa ajettavien reititysohjelmistojen käyttämä yleisrasite muodostaa noin 30% käytössä olevasta laskentatehosta [2]. Software-Defined Networking [3], eli ohjelmallisesti määritelty verkko on kehittyvä arkkitehtuuri, jonka kantava ajatus on erottaa verkon hallinta- ja reititystaso toisistaan. Tasojen erotteleminen mahdollistaa verkkolaitteiden käytössä olevan kapasiteetin kasvun sekä hallinnan keskittämisen. Keskitetyn hallinnan ansiosta verkkoa voidaan helposti segmentoida, eli jakaa osiin [4] ja sinne voidaan tuoda ohjelmallisia lisäominaisuuksia, kuten liikenteen luokittelua ja priorisointia (Quality of Service, QoS) [5]. Virtualisointi tietotekniikassa tarkoittaa fyysisten laitteiden jakamista loogisiksi resursseiksi. Virtualisoinnin avulla verkon fyysiset komponentit, kuten kytkimet, voidaan muuttaa ohjelmallisiksi komponenteiksi [6].

Tämän työn tutkimusosassa tarkastellaan ensin yleisesti SDN-sovelluksia ja sitten verkkojen virtualisointia. Tämän jälkeen tarkastellaan kahta sovellusta, joiden avulla SDN-arkkitehtuuria voidaan hyödyntää osana konesaliverkkojen virtualisointia. Työn pohjalta päädytään lopputulokseen, jossa SDN-tekniikoista saatava hyöty konesaliverkkojen virtualisoinnissa on konkreettinen, mutta riippuvainen konesalin ennalta asetetuista lähtökohdista. Olemassa olevaan konesaliin soveltuvat parhaiten SDN-sovellukset, jotka ovat yhteensopivia käytössä olevien kytkinlaitteiden kanssa. Uudessa konesalissa voidaan ensin valita haluttuja ominaisuuksia tukeva SDN-sovellus, joka määrittelee minkälaisia kytkinlaitteita voidaan käyttää.

Toinen luku alkaa katsauksella verkkoarkkitehtuurien historiaan ja kehitykseen. Luvussa käydään läpi verkkojen toimintaperiaate ja lopuksi tarkastellaan perinteisten verkkoarkkitehtuurien hyötyä, rajoituksia ja haasteita. Kolmannessa luvussa tutustutaan esimerkkien avulla uuden sukupolven verkkoarkkitehtuurien kehitykseen. Neljännessä luvussa

perehdytään SDN-arkkitehtuuriin, sen hyötyihin ja erilaisiin toteutuksiin syvemmin. Viidennessä luvussa käsitellään konesaliverkkojen virtualisoinnin perusteita ja haasteita. Samassa luvussa otetaan tarkempaan käsittelyyn kaksi sovellusta, jotka yhdistävät sekä SDN-arkkitehtuurin, että verkkojen virtualisoinnin. Lopuksi luvussa esitetään kaksi tapaa, miten käsiteltyjä sovelluksia voidaan hyödyntää konesaliverkkojen virtualisoinnissa. Yhteenvedossa pohditaan esitettyjen tapojen pohjalta, miten SDN-arkkitehtuuri soveltuu konesaliverkkojen virtualisointiin.

2. PERINTEINEN VERKKOARKKITEHTUURI

Ensimmäiset kytkentään perustuvat verkot olivat puhelinverkkoja. Kytkentöjä hallittiin operaattorikeskuksista, tällaisista verkoista käytetään nimitystä yhteydellinen verkko (Connection-Oriented Network). 1960-luvulla Rand Corporationin tutkija Paul Baran [7] väitti, että sotatilanteessa keskitetty isku operaattorikeskukseen katkaisisi yhteydet laajalta alueelta. Hän ehdotti ratkaisuksi siirtää puhetta puhelinverkkoja pitkin paketteina, jotka osaisivat liikkua kytkinverkossa autonomisesti. [8] Kytkinverkon osan vaurioituessa, paketti osaa löytää vaihtoehtoista reittiä perille.

Yhteydettömässä verkossa (Connectionless Network) [9] jokainen paketti siirretään riippumattomana toisistaan. Jokainen siirrettävä paketti reititetään sen sisältämän ylätunnuksen (Header) mukaan. Yhteydettömässä verkossa paketin kulkema reitti voi vaihdella paketista toiseen. Saapumisjärjestyksellä ei ole väliä, paketit numeroidaan lähetysjärjestyksessä, minkä mukaan vastaanottaja voi koota ne oikeassa järjestyksessä [10]. Ensimmäinen tietoverkko, ARPANET toimi juuri edellä kuvatulla tavalla. Myös vaihtoehtoisia yhteydellisiä verkkoja (Connection-Oriented Network) [9] kehitettiin, kuten esimerkiksi ATM (Asynchronous Transfer Mode), jossa tiedon lähettäjän ja vastaanottajan väliin muodostetaan ensin kiinteä reitti. [11] Internetin nopea kehitys 1990-luvulla vakiinnutti yhteydettömät verkot tietokoneiden väliseen kommunikointiin.

Tässä luvussa perehdytään ensimmäiseksi perinteisten tietoverkkojen toimintaperiaatteeseen. Luvussa käydään myös läpi, mitä termillä kytkin tarkoitetaan tässä työssä. Lopuksi tarkastellaan, minkälaisia rajoituksia ja haasteita perinteinen verkkoarkkitehtuuri asettaa.

2.1 Verkkojen toimintaperiaatteet

Pakettimuotoinen tiedonsiirto perustuu pakettien lajitteluun eli reitittämiseen. Pakettien reitittämisestä vastaavia laitteita kutsutaan yleisnimellä kytkin. Kytkin koostuu fyysisistä porteista, joilla yhdistetään verkon osat. Varhaiset kytkimet käyttivät pakettien reitittämiseen ohjelmistopohjaisia ratkaisuja. Ensimmäiset kytkimet olivatkin vain Unix-työasemia, joissa ajettiin ohjelmistoa, joka tutki saapuvan paketin ylätunnuksen, haki reititystaulusta (Routing Table) paketin kohteen ja reititti paketin sen perusteella seuraavaan paikkaan. [12] Kehitys optisen johtimen ja parikaapelin saralla joudutti myös kytkinten kehitystä. Lopulta fyysisen siirtomedian kehitys johti siihen, että pakettien lajitteluun vaadittu prosessointiaika ylitti ohjelmistopohjaisen ratkaisun kapasiteetin [14].

Muisti- ja puolijohdetekniikan kehittyminen 90-luvun puolivälissä mahdollisti sovelluskohtaisten mikropiirien (ASIC) valmistuksen, mikä puolestaan mahdollisti täysin laitteistopohjaisten kytkinten valmistuksen. Open Systems Interconnection Reference Model eli

OSI-malli, joka on esitetty kuvassa 1, on seitsemän portainen pakettivälitteisen tietoliikenteen käsitemalli. Kytkimet jaetaan OSI-mallin mukaan luokkiin sen perusteella, minkä kerroksen informaatiota voidaan käyttää hyödyksi pakettien reitityksessä. Layer 1 -keskittimet (Hub) toimivat alimmalla kerroksella ja lähettävät laitteeseen sisään tulevan liikenteen kaikkiin muihin portteihin, kuin mistä se on saapunut [11]. Layer 2 -kytkimet käyttävät pakettien reititykseen siirtokerroksessa välitettävää Medium Access Control, eli MAC-tietoa [11]. Jokaisella Layer 2 -tasolla toimivalla laitteella on yksilöity MAC-osoite. Kytkin säilyttää reititystaulussa lähdeosoitetta ja tietoa siitä, mistä fyysisestä portista paketti on tullut sisään. Kun kytkimeen saapuu paketti, jonka kohdeosoite löytyy reititystaulusta, paketti ohjataan sitä vastaavaan porttiin. Jos saapuvan paketin kohdeosoitetta ei tunnisteta, paketti ohjataan kaikkiin kytkimen portteihin.

Kuva 1. *Pakettivälitteisen tietoliikenteen käsitemalli*

Layer 3 -kytkimessä paketti reititetään perille Internet-protokollan eli IP:n avulla [11]. IP-osoite on numerosarja, joka yksilöi IP-verkkoon kytketyt päätelaitteet. Alun perin termillä kytkin on tarkoitettu sovelluskohtaisiin mikropiireihin perustuvaa laitetta, kun taas reitittimellä on tarkoitettu mikroprosessoriin perustuvaa laitetta, jossa paketteja lajitellaan ohjelmallisesti. Nykyään myös sovelluskohtaisilla mikropiireillä on mahdollista ajaa kehittyneitä reititysalgoritmejä. Puhekielessä kytkimen ja reitittimen ero on hämärtynyt. Tässä työssä käytetään termiä kytkin kuvaamaan laitetta, joka pystyy reitittämään liikennettä verkkokerroksessa sekä sitä ylemmillä kerroksilla.

2.2 Hyödyt

Tutkimuksen [12] mukaan verkkoprotokollien suunnittelusta on kirjoitettu tieteellisesti hyvin vähän. Verkkoprotokollat ovat monimutkaisia, joten pakettivälitteisen tietoliikenteen käsitemallin avulla niitä on voitu jakaa helpommin pienemmiksi osiksi. Jokainen ylemmän tason osa perustuu alemman tason luomaan pohjaan. Perinteisen verkkoarkkitehtuurin edut muodostuvatkin yksinkertaisista protokollista, jotka kuormittavat verkkoa ja sen laitteita mahdollisimman vähän. Lähdemateriaalin vähyyden vuoksi tässä työssä ei käsitellä perinteisen verkkoarkkitehtuurin hyötyjä enempää.

2.3 Haasteet

Vaikka pakettien reititys on nykyään täysin laitteistopohjaista, todellisuudessa pelkkä reititystauluun perustuva reititys ei riitä, vaan osaa paketeista pitää myös käsitellä. Reititystaulusta löytyvän tiedon lisäksi osa saapuvista paketeista sisältää osoitteen, jota kytkin ei tunne. Koska teknisesti ei ole mielekästä reitittää tällaista pakettia kaikkiin kytkimen tuntemiin osoitteisiin, on kytkimen opeteltava, mitä tällaisille paketeille tehdään. Tämän seurauksena kytkimiin on kehitetty ohjelmallisia sääntöjä, näitä sääntöjä kutsutaan reititysprotokolliksi.

Tilanteessa, jossa verkko on ruuhkautunut, linkki katkennut tai kohdeosoitteeseen ei saada yhteyttä, kytkimen pitää tietää asia etukäteen, jotta vaihtoehtoinen reitti voidaan muodostaa. Tämä tarkoittaa, että tarvitaan tapa, jolla voidaan välittää tietoa verkossa olevassa ruuhkasta tai viasta. Reititysprotokollan kautta saatua tietoa käytetään reititystaulun ylläpitämiseen etuajassa. Jos uutta tietoa verkon tilasta saadaan, käytetään reititysalgoritmiä päättelemään, mikä on paras tapa reitittää paketti uudelleen, minkä jälkeen uusi tieto päivitetään kytkimen reititystauluun. Päivitetyn reititystaulun pohjalta luodaan välitystaulu (Forwarding Table), jossa määritellään kytkimen portit, joihin paketit välitetään. [16] Älykkäiden reititysprotokollien kehitys on johtanut siihen, että tutkimuksen [2] mukaan vuonna 2011 kytkimet käyttivät 30 % laskentatehostaan verkon topologiatiedon välittämiseen ja ylläpitämiseen.

3. UUDEN SUKUPOLVEN VERKKOARKKITEHTUURIT

Internet Engineering Task Force, eli IETF-organisaatio, kutsuu julkaisemiaan internetiä koskevia standardeja RFC:ksi (Request for Comments). Ensimmäinen IPv4-yhteensopivan kytkimen vaatimuksia käsittelevä RFC 1716 julkaistiin vuonna 1994 [17]. Ollakseen yhteensopiva IP-protokollan kanssa nykyaikaisen kytkimen on tuettava edelleen myös vanhempia standardoituja reititysprotokollia.

Vanhimmat reititysprotokollat on kehitetty internetin alkuaikoina, jolloin muutokset verkon topologiassa olivat yleisiä. Verkkolaitteistojen kehitys on johtanut tilanteeseen, jossa verkossa tapahtuu yhä harvemmin odottamattomia muutoksia. Varsinkin konesaleista on kehittynyt sellainen ympäristö, missä odottamattomia keskeytyksiä eli häiriöaikaa ei käytännössä ole. Laiterikon sattuessa pysyy konesalin infrastruktuuri muuttumattomana, sillä fyysiset tai virtuaaliset palvelimet eivät siirry verkkojen välillä ilman, että niin on erikseen määritelty. Vaikka verkon fyysinen topologia pysyykin muuttumattomana, verkkoon tuodaan tai sieltä poistetaan jatkuvasti palvelimia ja virtuaalisia verkkoja. Koska reititysprotokollien tarkoitus on viestiä verkon muutoksista, myös palvelimen tahallinen lisääminen tai poistaminen aiheuttaa verkkoon liikehdintää, jonka välittyminen kytkinten reititystauluihin vie aikaa. [18] Erottamalla kytkimen hallintakerros (Control Plane) ja tiedonvälityskerros (Data Plane) toisistaan, voidaan kytkinten toimintojen päällekkäisyydestä johtuvaa yleisrasitetta (Overhead) vähentää. Kytkimen hallintakerros välittää kytkinten välillä tietoa verkon tilasta. Tiedonvälityskerros vastaa pakettien reitityksestä.

Tässä luvussa tutustutaan uuden sukupolven verkkoarkkitehtuurien perusperiaatteisiin. Ensimmäisenä käsitellään Open Signaling -työryhmää, joka perustettiin keräämään yhteen uusien verkkoratkaisujen ympärillä työskenteleviä ihmisiä ja antamaan vertaisarviointeja hankkeiden teknisistä ratkaisuista. Työryhmän ideoista on syntynyt useampia konkreettisia yrityksiä tuoda verkkoihin ohjelmallisia lisätoimintoja. Seuraavaksi työssä käsitellään NETCONF-protokollaa, joka keskittyi paikkaamaan vanhojen menetelmien puutteita tarjotakseen verkkolaitteiden hallintaan parempia työkaluja. Lopuksi luvussa tarkastellaan OpenFlow-protokollaa, joka on yksi ensimmäisistä SDN-standardeista.

3.1 Open Signaling

Vuodesta 1995 alkaen järjestettiin sarja työryhmiä Open Signaling (OPENSIG) -nimikkeen alla. Työryhmien tarkoituksena oli tehdä ATM, IP ja mobiili -verkoista avoimempia, laajennettavia sekä ohjelmoitavia. Päämääränään työryhmillä oli kehittää avoin, ohjelmallisesti ohjelmoitava verkko erottamalla verkkolaitteisto ja sitä ohjaava ohjelmisto.

Ohjelmoitavuudelle oli kysyntää, näin myös palveluntarjoajille saataisiin pääsy verkkolaitteistoihin, jotka normaalisti olivat suljettuja järjestelmiä. [19]

Ohjelmoitavien verkkojen historian kannalta merkittävimpiä työryhmän aikaansaannoksia lienee IEEE (Institute of Electrical and Electronics Engineers) P1520 -standardointiprojekti. Projekti visioi tulevaisuuden tietoverkon suureksi tietokoneeksi, joka tarjoaisi ääni, data ja video -palveluita globaalisti. Vanhan mallin mukaan verkkojen äly sijaitsi niiden välisessä signaloinnissa, joka oli rakennettu standardoitujen reititysprotokollien varaan. Älykkäämpien reititysprotokollien kehityksestä oli muodostunut vaativa ja työläs prosessi, sillä modernin tietoverkon standardien huomioonottaminen edellytti yhteisymmärrykseen pääsyä myös eri laitteistovalmistajilta. OPENSIG-työryhmän esittelemä vaihtoehtoinen tapa tarjoaisi verkkolaitteistoihin ohjelmointirajapinnan eli API:n (Application Programming Interface), joka mahdollistaisi pääsyn niiden alemman tason toimintoihin. Ylemmällä tasolla rajapinta mahdollistaisi älykkään hallinnan ohjelmoitavuuden kautta. Vanhan mallin tarkoin määritellyt reititysprotokollat korvattaisiin siis uusilla ohjelmallisilla ja laajennettavilla reititysprotokollilla. Yhteensopivuus vanhojen protokollien kanssa mahdollistettaisiin yhdyskäytävillä. [20]

Merkittäväksi voidaan laskea myös IETF:n perustama GSMP (General Switch Management Protocol) -työryhmä, jonka perusta luotiin varhaisissa OPENSIG-tapaamisissa [19]. GSMP-työryhmä määritteli saman nimisen yleisprotokollan leimakytkinreitittimien (Label Switch Router) hallintaan [21], joita käytettiin yleisesti ATM-verkoissa. Viimeisin versio GSMP-protokollasta teki määritelmään lisäyksiä myös IP-verkkojen osalta, jolloin protokollaa voitaisiin käyttää nimensä mukaisesti yleisesti [22].

3.2 NETCONF

Vuonna 2006 IETF ehdotti NETCONF:ia [23] verkkolaitteiden asetusten hallintaprotokollaksi. Hallinta perustui laitteissa olevaan ohjelmistorajapintaan, jonka kautta voitiin sekä lähettää, että vastaanottaa asetusmuutoksia. Edelleen laajassa käytössä oleva hallintaprotokolla SNMP (Simple Network Management Protocol) [24] oli kehitetty samaiseen tarkoitukseen jo 80-luvun lopulla. Se käyttää SMI (Structure of Management Information) -määritelmää muodostamaan MIB (Management Information Base) -tietokannan. Se sisältää tiedon niistä hallittavista toiminnoista, joita verkkolaitteessa voidaan ohjata. Verkonhallinta-protokollalla voidaan muuttaa MIB-tietokannan muuttujia, jotka puolestaan ohjailevat laitteen sisäisiä toimintoja. Myöhemmin huomattiin, että SNMP:n käyttö on yleisesti vakiintunut verkkolaitteiden tilan seurantaan sopivaksi työkaluksi, eikä niinkään asetusten muutosten hallintaan. Verkon hallintaan käytetyn SNMP-protokollan ensimmäisessä versiossa oli kuitenkin useita puutteita, kuten vahvan salauksen puuttuminen, joka sittemmin on lisätty SNMPv3-versiossa. [25][26]

NETCONF:ia pidettiin julkaisunsa aikana uutena tapana hallita verkkolaitteita, joka korjaisi SNMP protokollan puutteita. Vaikka tavoite ketterämmästä laitehallinnasta saavutettiin, NETCONF ei kuitenkaan erottele verkon hallintakerrosta ja tiedonsiirtokerrosta. Ratkaisua ei voidakaan pitää kokonaan ohjelmallisena, sillä uuden ominaisuuden lisääminen vaati muutoksia sekä laitteistoon, että hallintasovellukseen. NETCONF:ia pidetäänkin enemmän automaattisen käyttöönoton työkaluna, eikä niinkään laitteistojen kokonaisvaltaiseen hallintaan soveltuvana ratkaisuna. Sekä SNMP, että NETCONF ovat edelleen hyödyllisiä työkaluja, jotka lisäävät verkkojen ohjelmallisuutta. NETCONF työryhmä on edelleen aktiivinen ja viimeisin ehdotettu standardi on vuodelta 2011. [26]

3.3 OpenFlow

Vuonna 2008 julkaistussa raportissa [27] ryhmä Stanfordin tutkijoita ehdotti OpenFlow:ta ratkaisuksi, jonka avulla kampusverkkojen sisällä voitaisiin testata kokeellisia protokollia. Määritelmän mukaan ratkaisun pitäisi soveltua sekä pieneen että suureen ympäristöön, tukea monenlaista tutkimusmateriaalia, eristää verkon kokeellinen puoli ja soveltua käytettäväksi myös laitevalmistajien suljetuissa järjestelmissä. Näin syntyi idea OpenFlow-kytkimestä. Tutkijat huomasivat, että vaikka valmistajakohtaisissa toteutuksissa oli eroja, perustuvat ylemmän tason tietoa käyttävät reititystaulut eli vuotaulut (Flow Table) samoihin perustoimintoihin. OpenFlow-protokolla määrittelee näille perustoiminnoille avoimen ohjelmoitavan rajapinnan, jota jokainen laitevalmistaja voi hyödyntää. Verkon ylläpitäjä voi lajitella liikennettä kytkimissä sen mukaan onko se kokeellista vai tuotantoon kuuluvaa. Tällä tavoin uutta kokeellista liikenteen reititystä voidaan kokeilla ilman että se vaikuttaa verkon luotettavuuteen. [27]

OpenFlow-kytkin koostuu kolmesta osasta; vuotaulusta, salatusta kommunikointikanavasta sekä OpenFlow-protokollasta. Vuotaulun tehtävä on kertoa kytkimelle minkälaisilla kulloinkin voimassa olevilla säännöillä liikennettä kuuluu reitittää. Salattu kommunikointikanava yhdistää kytkimen ja hallintakonsolin, jonka kautta OpenFlow-protokolla mahdollistaa standardoidun tavan hallita kytkintä. OpenFlow-standardi jakaa kytkimet kahteen eri luokkaan. Puhtaasti OpenFlow-perusteiset kytkimet eivät tue perinteistä Layer 2 tai 3 -reititystapaa, kun taas OpenFlow-tuetuissa kytkimissä OpenFlow on lisättynä protokollana muiden joukossa. Jälkimmäisten kytkinten etuna on mahdollisuus erotella osa liikenteestä kokeelliseksi ja osa tavalliseksi. Tavallinen liikenne reititetään perinteisellä Layer 2 ja 3 -tasolla, kun taas kokeellinen liikenne reititetään OpenFlow-vuotaulun mukaan. [27]

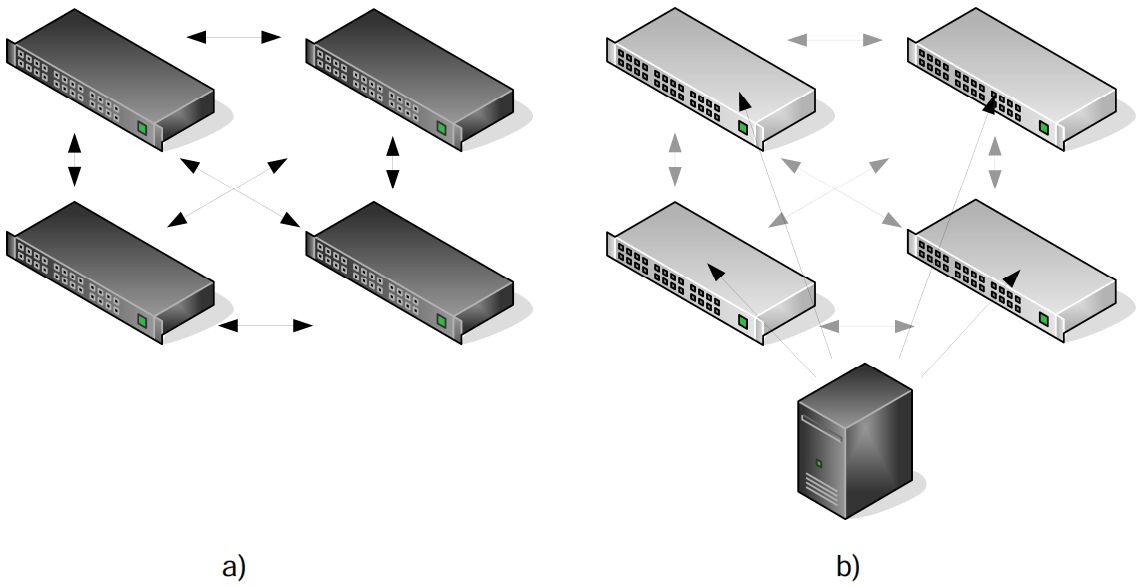
OpenFlow:n alkuperäinen tarkoitus oli tarjota tutkijoille alusta tehdä tutkimusta verkkojen parissa. Laitteistovalmistajat ovat kuitenkin omaksuneet OpenFlow-protokollan myös keinoksi pienentää verkkolaitteiden monimutkaisuudesta aiheutuvia kustannuksia. Kaupallisia OpenFlow-yhteensopivia laitteita löytyykin nykyisin usealta valmistajalta. Open Networking Foundation:n (ONF) perusti 2011 Deutche Telekom, Facebook, Google,

Microsoft, Veizon ja Yahoo tukeakseen SDN-verkkojen ja OpenFlow-protokollan kehitystä kaupallisesti. OpenFlow-yhteensopivan kytkimen pitää tukea standardin määrittelemiä ominaisuuksia. OpenFlow-kytkimen tekninen toteutus on kuitenkin laitteistovalmistajan vastuulla. Kahden eri valmistajan OpenFlow-kytkimet eivät välttämättä ole keskenään yhteensopivia. OpenFlow-protokollan versio 1.0.0 [28] julkaistiin 2011. Nopea kehitys ja kaupallisten tahojen tuki on johtanut tilanteeseen, jossa ensimmäisenä markkinoille tulleet laitteet vaativat ohjelmistopäivityksiä tukeakseen protokollan uusia versioita. [29]

OpenFlow:n muodostuttua suosituimmaksi SDN-toteutukseksi on syntynyt käsitys, että näillä tarkoitettaisiin keskenään samaa asiaa. Yleisemmin SDN tarkoittaa kuitenkin ratkaisua, jossa hallintakerros (Control Plane) on eroteltu tiedonsiirtokerroksesta (Data Plane), kun taas OpenFlow määrittelee standardin, miten kytkimen vuotaulua käsitellään. SDN muodostaa käyttäjälle yleiskäsitteen verkon tilasta, kun taas OpenFlow yleistää yksittäiset verkon laitteet.

4. SOFTWARE-DEFINED NETWORKING

Software-Defined Networking, tai lyhemmin SDN on yleistynyt käyttöön vuonna 2009, jolloin sillä oli vielä melko tarkkaan määritelty merkitys [30]. Termin merkitys on kuitenkin muuttunut, sillä nykyään se yhdistetään yleisesti uuden sukupolven verkkoihin. Ensimmäisenä SDN-toteutuksena voitaneen pitää Sun:n ja AT&T Labs:n GeoPlex:ia vuodelta 1995, jonka tarkoitus oli tuottaa verkkoon standardoituja ohjelmallisia ominaisuuksia [31].



Kuva 2. *Reititys- ja hallintakerroksen erottelu*

Nykyään SDN:n mielletään tarkoittavan yleisesti mitä tahansa ratkaisua, jossa verkon hallinta on keskitettyä, mikä tarjoaa ohjelmistopohjaisen rajapinnan verkon asetusten ja hallinnan automatisointiin. Kuvassa 2 on havainnollistettu kohdassa a) perinteinen kytkinverkko ja kohdassa b) hallintakerroksen äly on keskitetty erilliseen hallintakonsoliin.

Tässä luvussa perehdytään ensin SDN-arkkitehtuurin hyötyihin käymällä läpi tapoja, joilla SDN-arkkitehtuuri vastaa perinteisen arkkitehtuurin haasteisiin. Sen jälkeen pohditaan, miksi SDN-sovellukset ovat jakautuneet sekä avoimen, että suljetun lähdekoodin sovelluksiin. Lopuksi käydään syvemmin läpi muutamia erilaisia SDN-sovelluksia vertailemalla niiden ominaisuuksia taulukossa 1.

4.1 Hyödyt

Göransson et al. käsittelevät teoksessaan [18] tapoja, joilla SDN vastaa konesalien kehittyviin tarpeisiin. Ensimmäiseksi mainitaan automatisointi. Automatisoinnin avulla verk-

koja voidaan lisätä ja poistaa silloin kun tarve vaatii. Heidän mukaansa tällaista erikoispiirrettä voidaan kutsua verkon ketteryydeksi. Ketteryyden lisäksi SDN tarjoaa verkkoihin myös parempaa skaalautuvuutta, eli muuntautumiskykyä. Rajoitukset reititystaulujen koossa ja virtuaalisten verkkojen lukumäärässä rajoittavat perinteisen konesalin skaalautumista. Software-Defined Networking ratkaisee skaalautumisen rajoitukset ylemmän kerroksen reititystietoon perustuvilla uusilla tavoilla. Skaalautumisen lisäksi Göransson et al. tuo esille myös rinnakkaisen reitityksen (Multipathing) tuomat edut. Rinnakkaisen reitityksen avulla verkko voi käyttää resurssejaan hyödyksi parhaimmalla mahdollisella tavalla. Luopumalla vanhoista reititysprotokollista voidaan paketti reitittää perille virheen sattuessa myös vaihtoehtoista reittiä. Vaihtoehtoiset reitit eivät ole perinteisissä verkoissa mahdollisia, sillä vaikka fyysinen kytkentä olisikin olemassa, ei esimerkiksi lyhimpään reittiin perustuva reititysprotokolla osaa reitittää pakettia muulla tavalla.

Verkkolaitteiden kehitys on johtanut tilaan, jossa laitteet ovat tarpeeksi tehokkaita tarjoamaan palvelua samanaikaisesti useille asiakkaille. Viimeisenä Göransson et al. mainitseekin itse verkkojen virtualisoinnin. Virtuaaliset asiakasverkot muodostuvat fyysisten laitteiden päälle häivyttäen perinteisen mallin, jossa verkko on joukko erikseen hallittavia laitteita. Virtuaalista verkkoa hallitaan kokonaisuutena, jossa verkon ylläpitäjä voi luoda, poistaa ja laajentaa verkkoa tarpeen mukaan ilman pääsyä fyysiselle laitteistotasolle. [18]

4.2 SDN-sovellukset

SDN-sovellusten tilaa käsittelevässä tutkimuksessa [32] on todettu olemassa olevien toteutusten jakautuvan avoimen lähdekoodin ratkaisuihin ja suljetun lähdekoodin ratkaisuihin. Selkeän kahtiajaon taustalla lieneekin laitteistovalmistajien haluttomuus päästää käyttäjää käsiksi alemman tason rajapintoihin, kun taas avoimen lähdekoodin varaan rakentuvat tuotteet eivät korkean pelkistystasonsa ansiosta tällaista mahdollisuutta edes tarjoa. Tutkimuksen mukaan kehittyvä trendi on antaa laitevalmistajien tehdä laitteistoistaan SDN-yhteensopivia käyttäen suljetun lähdekoodin ratkaisuja. Tutkimuksen mukaan asiakas on kiinnostunut enemmän verkon ohjelmoitavuuden mahdollistamisesta, ei niinkään sen teknisestä toteutuksesta.

SDN-sovellusten toteutustavasta huolimatta voitaneen todeta, että suurin osa sovelluksista on OpenFlow-yhteensopivia. Tutkimuksen [32] mukaan osa laitteistovalmistajista on kuitenkin sitä mieltä, että SDN:n hyödyt voidaan saavuttaa myös ilman OpenFlow-protokollaa. Tutkimuksen mukaan laitteistovalmistajien omat verkkoratkaisut tarjoavat jo nykyisellään SDN-arkkitehtuuria noudattavia periaatteita. Esimerkki tällaisesta on Cisco NX-OS -alusta [33], joka on modulaarinen ja ohjelmoitava. Tutkimuksen [32] mukaan tällaiset ratkaisut ovat houkuttelevia tapauksissa, joissa vanhoja verkkolaitteita ei haluta korvata puhtaasti OpenFlow-perustaisilla laitteilla.

Taulukko 1. *Yhteenveto SDN-sovelluksista*

Sovellus	Lähdekoodi	OpenFlow-tuki	API	Käyttöliittymä
NOX	Avoin	1.0	C++	Merkkipohjainen
Floodlight	Avoin	1.4	Java/REST	Graafinen
OpenStack	Avoin	1.3	Python/REST	Graafinen
OpenDaylight	Avoin	1.3	Java/REST	Graafinen
ONOS	Avoin	1.5	Java/REST	Graafinen
VMware NSX	Suljettu	-	REST	Graafinen
Cisco ACI	Suljettu	-	Python/REST	Graafinen

Taulukossa 1 on kerätty yhteen syvemmän tarkastelun kohteeksi valitut SDN-sovellukset. Taulukosta ilmenee, onko sovellus avointa vai suljettua lähdekoodia, mitä OpenFlow-versiota sovellus tukee, minkälaisen ohjelmointirajapinnan eli API:n se tarjoaa, sekä onko sovellukseen olemassa myös graafinen käyttöliittymä. Representational State Transfer eli REST [48] on HTTP-protokollaan (Hypertext Transfer Protocol), eli hypertekstin siirtoprotokollaan perustuva arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseen.

4.2.1 Avoimen lähdekoodin sovellukset

Ensimmäisenä avoimen lähdekoodin OpenFlow-sovelluksena voitaneen pitää vuonna 2008 SIGCOMM-konferenssissa esiteltyä NOX:ia [34]. Se on OpenFlow 1.0 -yhteensopiva ja tarjoaa kehittäjille C++-pohjaisen ohjelmoitavan rajapinnan. NOX vaatii alustakseen Linux-käyttöjärjestelmän. Vaikka toteutuksen taustalla oli osaksi kaupallinen taho Nicira, annettiin NOX:n lähdekoodi vapaaseen käyttöön. Se on myöhemmin toiminut pohjana usealle avoimen lähdekoodin SDN-toteutukselle. [35]

Avoimeen lähdekoodiin perustuva, vuonna 2011 julkaistu Floodlight [36] on OpenFlow 1.0–1.4 -yhteensopiva SDN-hallintakonsoli. Floodlight on Java-pohjainen ja vaati alustakseen käyttöjärjestelmän sekä Java-ympäristön. Floodlight tukee muutamia eri valmistajien kytkimiä, kuten Extreme, HP ja Juniper [37].

OpenStack on avoimen lähdekoodin pilvikäyttöjärjestelmä [38]. OpenStack on modulaarinen, eli ohjelmallisesti laajennettava ja tarjoaa Neutron-verkkomoduulin lisäksi myös muita moduuleja eli laajennoksia joiden avulla voidaan hallita eri konesaliresursseja. Neutron on Python-pohjainen ja perustuu Ryu SDN -kontrollerin lähdekoodiin [39]. Neutron:n ominaisuuksia, kuten esimerkiksi laitetukea laajennetaan liitännäisillä. Liitännäisiä on saatavilla esimerkiksi Cisco:n, IBM:n ja Juniperin kytkimille [40].

Vuonna 2014 julkaistu OpenDaylight [41] on Foodlight:n tapaan avoimeen lähdekoodiin perustuva Java-pohjainen SDN-hallintakonsoli. OpenDaylight vaatii alustakseen käyttöjärjestelmän. OpenDaylight on modulaarinen ja tukee OpenFlow 1.0–1.3 -protokollien lisäksi myös vanhoja, sekä useampia uuden sukupolven verkkoprotokollia kuten esimerkiksi NETCONF:ia. OpenDaylight tukee useimpia OpenFlow-yhteensopivia kytkimiä.

Open Networking Operating System eli ONOS [42] on vuonna 2014 SIGCOMM konferenssissa esitelty ON.Lab:n (Open Networking Lab) modulaarinen verkkokäyttöjärjestelmä, jonka suunnittelun lähtökohtina ovat olleet korkea suorituskyky, verkon viiveiden minimoiminen, skaalautuminen sekä korkea tavoitettavuus. ONOS on Java-pohjainen, mutta toisin kuin esimerkiksi Floodlight, ONOS-palvelimet voivat toimia keskenään klusterina. Klusteri on useamman laitteen verkotettu malli, jossa siihen kytketyt laitteet jakavat muiden laitteiden kesken resursseja. Klusteroitu ONOS-verkko takaa myös verkon korkean tavoitettavuuden. Tilanteessa, jossa yksi ONOS-hallintakonsoli vikaantuu, siirtyy hallinta automaattisesti toiselle konsolille [43]. Vuonna 2016 ON.Lab sulautui osaksi ONF-organisaatiota yhdistääkseen OpenFlow-standardin ja avointen ohjelmistojen kehityksen [44].

4.2.2 Kaupalliset sovellukset

Ensimmäinen kaupallinen SDN-ratkaisu oli vuonna 2012 julkaistu Nicira NVP (Network Virtualization Platform) [45]. Se on ohjelmistopohjainen ratkaisu, joka kykenee luomaan hajautetun virtuaalisen verkkoinfrastruktuurin jo olemassa olevan verkon päälle. Järjestelmä on suunniteltu ratkaisemaan perinteisen verkon ongelmat, soveltamalla niihin palvelinvirtualisoinnin periaatteita. Myöhemmin samana vuonna VMware osti Niciran [46] ja NVP:stä jalostui VMware NSX. OpenFlow-protokollan sijaan NVP ja NSX käyttävät omaa suljetun lähdekoodin protokollaa [47].

Cisco ACI (Application Centric Infrastructure) [48] on vuonna 2014 julkaistu järjestelmä Cisco:n Nexus-sarjan verkkolaitteille. Cisco ACI -järjestelmä koostuu hallintakonsolista, sekä ACI-yhteensopivista fyysisistä tai virtuaalisista verkkolaitteista. Cisco ACI ei tue OpenFlow-protokollaa, mutta täyttää muilta osin SDN-arkkitehtuurin määritelmän. Cisco ACI on kuitenkin esimerkiksi OpenStack Neutron -yhteensopiva, joka mahdollistaa ACI-verkkolaitteiden hallinnan OpenStack-hallintakonsolin kautta.

5. VIRTUAALISET VERKOT

Virtualisointi tietotekniikassa tarkoittaa fyysisten laitteiden jakamista loogisiksi resursseiksi. Verkkojen virtualisointi tarkoittaa ohjelmallista verkkopalvelua, joka on eriytetty sitä tarjoavasta fyysisestä ratkaisusta. Virtuaalinen ratkaisu tarjoaa samoja ominaisuuksia kuin fyysinen ratkaisu.

Yksi ensimmäisiä kaupallisia verkkojen virtualisointiratkaisua tarjoavia tahoja Nicira, nykyään VMware [50], kehitti OpenFlow-protokollan ja NOX-kontrollerin pohjalta oman suljetun lähdekoodin järjestelmän. Tähän asti SDN käsitteenä oli tarkoittanut vain hallintakerroksen erottamista tiedonsiirtokerroksesta. Verkkojen ja varsinkin konesalien kannalta on tärkeää, että myös verkot voitaisiin irrottaa fyysisten laitteista rajoitteista. Nicira määrittelikin seitsemän teesiä [51], joiden avulla verkot voitaisiin virtualisoida.

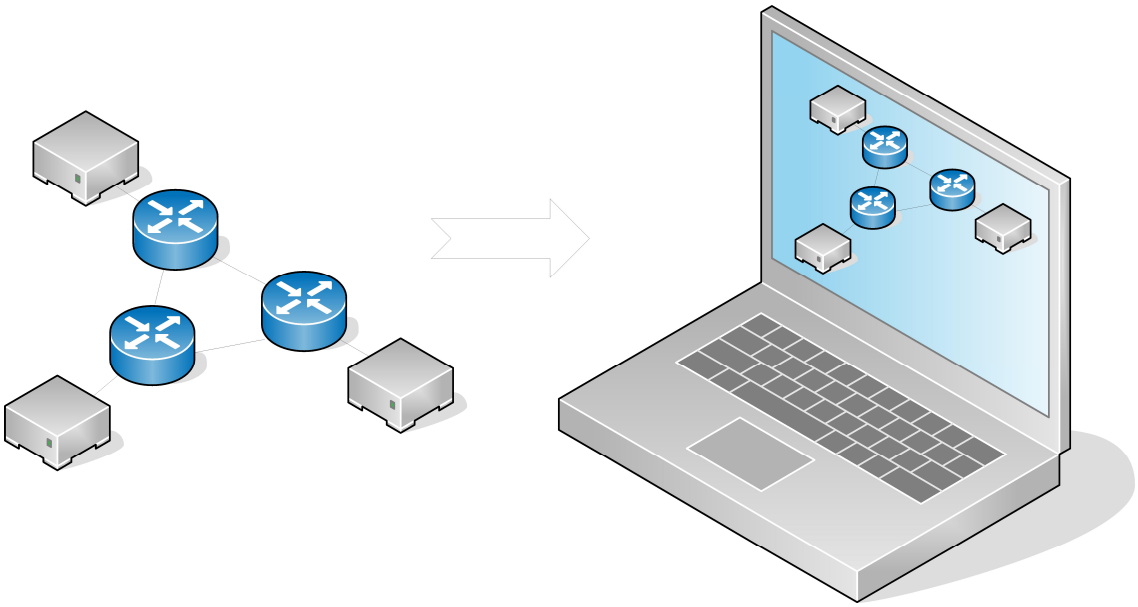
1. **Laitteistoriippumaton.** Verkkovirtualisointialustan pitää olla laitteistoriippumaton. Laitteistoriippumattomuus tarkoittaa, että verkon fyysiseen toteutukseen voidaan käyttää minkä tahansa laitevalmistajan tuotteita.
2. **Fyysisen verkkokerroksen tarkka virtualisointi.** Virtualisoidun verkkokerroksen pitää toteuttaa modernin fyysisen Layer 2 ja 3 -verkon standardit tarkasti. Fyysisen kerroksen rajoitukset, kuten päällekkäiset MAC ja IP -osoitteet eivät kuitenkaan saa olla esteenä virtualisoiduissa verkoissa.
3. **Noudattaa virtuaalikoneen periaatteita.** Virtuaalikoneen tärkeimpiä ominaisuuksia on mahdollisuus siirtää, keskeyttää, jatkaa, varmuuskopioida ja palauttaa virtuaalikone milloin tahansa. Verkkovirtualisointialustan pitäisi sisältää nämä ominaisuudet.
4. **Yhteensopivuus eri hypervisorien (hypervizor) kanssa.** Verkkovirtualisointialustan pitää toimia olemassa olevien hypervisoreiden, eli virtualisoinnin mahdollistavien ohjelmistojen kanssa.
5. **Turvallisesti multitenantti.** Multitenantin palvelun suorittamiseen vaadittavat laitteistoresurssit on jaettu kaikkien käyttäjien kesken. Virtualisoidut verkot ovat keskenään eristettyjä ja noudattavat esimerkiksi finanssialan PCI DSS -standardia (Payment Card Industry Data Security Standard). Virtuaalisen tason pitää olla täysin eristetty, sieltä ei saa olla pääsyä toiselle virtuaaliselle tasolle, eikä fyysiselle tasolle.
6. **Skaalautuva.** Virtualisoinnin pitää skaalautua konesalin vaatimuksiin.
7. **Ohjelmoitava provisiointi ja hallinta.** Provisiointi tarkoittaa laiteresursseista ja ohjelmistoista muodostettavien palveluiden valmistelua ennalta. Perinteisesti verkkoa hallitaan laite kerrallaan. Vaikka hallintaa voidaan nopeuttaa valmiilla kokoonpanoilla, on nykyinen lähestymistapa hidas ja altis näppäilyvirheille. Keskitetyn hallinnan ansiosta verkon kokonaiskuva säilyy, mikä puolestaan yksinkertaistaa uusien komponenttien käyttöönottoa. [51]

Tässä luvussa käsitellään ensin konesaliverkkojen virtualisoinnin periaatteita, jonka jälkeen tarkastellaan haasteita, joita verkkojen virtualisointi SDN-tekniikoilla asettaa. Lu-

vussa otetaan tarkempaan käsittelyyn kaksi sovellusta, jotka yhdistävät sekä SDN-arkkitehtuurin, että verkkojen virtualisoinnin. Lopuksi esitetään kaksi tapaa, miten näitä sovelluksia voidaan hyödyntää konesaliverkkojen virtualisoinnissa.

5.1 Toimintaperiaate

Ensimmäinen askel verkkojen virtualisoinnissa oli Virtual Local Area Network, eli VLAN [52]. Sen avulla useita virtuaalisia verkkoja voitiin perustaa samaan fyysiseen verkkoon, niiden ollessa kuitenkin täysin eristettyjä toisistaan. Tekniikan ongelmaksi on muodostunut kuitenkin sen skaalautumattomuus. Standardi 802.1Q [52] määrittelee Ethernet-kehyksessä käytetyn VLAN-tunnisteen pituuden 12 bittiin. Ethernet-kehys on tiedonsiirron perusyksikkö, joka sisältää esimerkiksi tietoa siitä, mihin kohdeosoitteeseen paketti on matkalla. VLAN-tunnisteen pituus rajaa virtuaalisten verkkojen määrän 4094:n. Määrä ei riitä suurille internet-palveluntarjoajille, jotka palvelevat massiivista määrää asiakkaita samassa verkossa [53].



Kuva 3. *Fyysisen verkon virtualisointi*

Software-Defined Networking -arkkitehtuurin kehitys ja testaaminen on luonut tarpeen suurille testiympäristöille. Pääsy tällaisiin testiympäristöihin on usein haastavaa tai kallista. Tutkimuksessa [54] kuvaillaan ympäristö, joka käyttää kokonaisten virtuaalikoneiden sijasta kevyttä virtualisointia Linux-ytimen prosessien ja verkkosolmujen monistamiseen. Mininet [55] on verkko-emulaattori, joka voi emuloida eli jäljitellä verkkopäätteitä, kytkimiä, reitittimiä ja linkkejä Linux-ytimen päällä. Mininet käyttää virtualisointia luodakseen fyysisestä verkosta ohjelmallisen jäljitelmän, kuten kuvassa 3 on havainnollistettu. Mininetin luomat virtuaaliset komponentit käyttäytyvät kuin fyysiset verkkolaitteet. Open vSwitch [56] on ohjelmallisesti toteutettu virtuaalinen verkkokytkin, jota voi-

daan käyttää virtuaalisessa Mininet-verkossa. Open vSwitch tukee useimpia standardoituja reititysprotokollia, kuten OpenFlow ja on hallittavissa useimmilla SDN-hallintakonsoleilla.

Verkkojen lisäksi myös SDN-hallintakonsolin ja kytkimien välinen yhteys on mahdollista virtualisoida. Hallintakonsolin ja kytkimien välille lisätään uusi välikerros, joka jakaa kytkimien resurssit ja liikenteen usean hallintakonsolin kesken. Esimerkki tällaisesta sovelluksesta on FlowVisor, joka jakaa verkon fyysiset resurssit Layer 1–4 -tason liikenteen perusteella eristetyiksi osiksi [57].

5.2 Haasteet

OpenFlow-protokolla määrittelee pakettien välittämiseen tiedonsiirtokanavan (OpenFlow Pipeline). Tiedonsiirtokanava voi sisältää OpenFlow-versiosta riippuen yhden tai useamman vuotaulun. Vuotaulu sisältää tietoa siitä, minkälaisia toimenpiteitä (Action) saapuville paketeille tehdään. Kytkimen ei kuitenkaan tarvitse osata suorittaa kaikkia näitä toimenpiteitä, sillä OpenFlow-standardi määrittelee sekä pakollisia, että valinnaisesti toteutettavia toimenpiteitä. Näiden valinnaisten toimintojen toteuttaminen on usein laitteistovalmistajan vastuulla. Tutkimuksen [58] mukaan OpenFlow-protokollan nopea kehitys onkin johtanut tilanteeseen, missä osa laitteistovalmistajista on epäonnistunut kehittämään laiteohjelmistojaan OpenFlow-standardin kehityksen tahdissa. Avoimen lähdekoodin ohjelmistopohjaisiin kytkimiin tuen lisääminen on verraten helpompaa. Tutkimuksen [59] mukaan muita tunnistettavia haasteita SDN-verkoissa ovat; ylemmän tason sääntöjen kääntäminen alemman tason laiteasetuksiksi, autonominen ja verkon sisäinen hallinta, joustava hallinta rajapintojen läpi, älykäs verkkojen suunnittelu sekä olosuhteiden muutoksiin mukautuva hallinta.

Koska virtuaaliset kytkimet ovat fyysisen verkon jatke, on keskitetyn hallinnan lisäksi tärkeää verkon topologian eli verkon kohteiden visuaalinen mallintaminen suhteessa toisiinsa. Verkkotopologian visualisointi löytyy sisäänrakennettuna esimerkiksi OpenDaylight [59] ja ONOS [60] -kontrollereista. VMware NSX:n käyttämä Open vSwitch -virtuaalikytkin tarjoaa sen sijasta SNMP-rajapinnan, jonka avulla verkon topologian visualisointi voidaan toteuttaa kolmannen osapuolen sovelluksella. Kaupallista ratkaisua edustaa HyperGlance [61], joka voidaan liittää esimerkiksi osaksi OpenStack-ympäristöä. HyperGlance käyttää verkon topologian mallintamiseen SNMP-protokollaa. Keskitetty hallinta on kuitenkin tehokkaampaa silloin kun verkon visualisointi ja hallinta ovat samassa konsolissa.

Ylemmän tason sääntöjen kääntämisestä laiteasetuksiksi voitaneen ottaa konkreettinen esimerkki keskitetyn hallinnan toteuttamistavoista. SDN-arkkitehtuurin periaatteiden mukaisesti hallinnan pitää olla keskitettyä. Keskitetty hallinta ei kuitenkaan mahdollista verkkojen skaalautumista, joten OpenFlow-protokollaa on pitänyt päivittää lisäämällä mahdollisuus keskitetylle, mutta fyysisesti hajautetulle hallinnalle [63]. Tuki useammille

hallintakonsoleille lisättiin OpenFlow-versiosta 1.2 lähtien [58]. Suuri osa fyysisistä kytkinlaitteista tukee kuitenkin edelleen vain OpenFlow 1.0 -protokollaa [29].

Verkkolaitteiden laiteasetusten hallinta OpenFlow-kytkinten osalta on haastavaa. Laitteita hallitaan joko laitevalmistajien omilla työkaluilla, tai jollain yleisellä hallintaprotokollalla, kuten NETCONF. Tämä tarkoittaa, että ennen kuin niin sanottu puhdas OpenFlow-kytkin voidaan tuoda verkkoon, on sen ensiasetusten hallintaan oltava tapa. Open Networking Foundation on ratkaissut tämän kehittämällä oman hallintaprotokollan OF-CONFIG (OpenFlow Management and Configuration Protocol) [64]. OF-CONFIG -protokollan käyttöönottoa hidastaa kuitenkin Open vSwitch -projekti, joka käyttää yleisempää OVSDB (Open vSwitch Data Base management) -protokollaa. OVSDB on edelleen joiltain osin epäyhteensopiva OF-CONFIG -protokollan kanssa. [65]

5.3 Sovellukset

SDN-arkkitehtuurin kehitys on tapahtunut osanen kerrallaan. Protokollat, kuten OpenFlow, OF-CONFIG ja OVSBD on luotu ratkaisemaan SDN-arkkitehtuurin kehityksessä esiin nousseita puutteita. SDN-kontrollerit, kuten esimerkiksi OpenStack ja ONOS ovat luotu hyödyntämään kehitettyjä protokollia, auttamalla hallitsemaan sekä fyysisiä, että virtuaalisia verkkolaitteita. Nämä kaikki osaset nivoutuvat yhteen kokonaisuudeksi, jonka tarkoituksena on hallita verkkoa kokonaisuutena mahdollisimman tehokkaasti. Verkkojen virtualisointi ei ole kuitenkaan riippuvainen SDN-arkkitehtuurista. Tästä syystä onkin ollut tarpeellista perustaa European Telecommunications Standards Institute, eli ETSI:n alla toimiva NFV-työryhmä (Network Functions Virtualisation) [67], jonka tarkoituksena on standardoida verkkojen ja sen toiminnallisuuksien virtualisointiin liittyviä teknologioita.

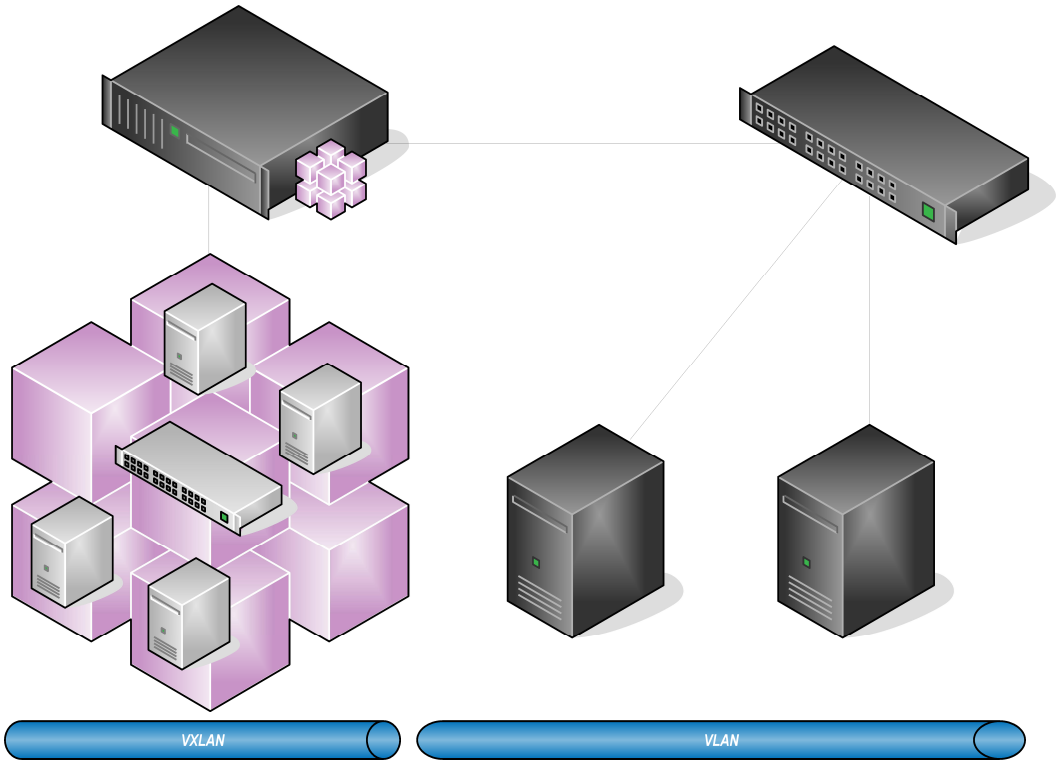
SDN ja NFV -arkkitehtuurit eivät ole riippuvaisia toisistaan, mutta SDN-arkkitehtuurin käyttäminen myös virtuaalisissa verkoissa on hyödyllistä. Monet olemassa olevista NFV-sovelluksista käyttävät tai laajentavat jo olemassa olevia OpenFlow-sovelluksia, kuten esimerkiksi FlowVisor. Arkkitehtuurit eroavat kuitenkin toisistaan joiltakin osa-alueilta. Tutkimuksen [63] mukaan todellisen NFV-arkkitehtuurin toteutuminen esimerkiksi OpenFlow-protokollassa vaatii muutoksia sekä OpenFlow-protokollaan, että sitä hyödyntäviin SDN-kontrollereihin. OpenFlow toimii tällä hetkellä vain L2–L4 -tasoilla, eikä sisällä tukea ylemmän L5–L7 -tason protokollille. NFV-arkkitehtuuri määrittelee toimintoja myös ylemmillä L5–L7 -tasoilla [63]. Tutkimuksen [68] mukaan NFV-arkkitehtuuri tarvitsee SDN-arkkitehtuurin tapaan myös keskitetyn hallinnan. Tällainen kehityshanke on ETSI:n MANO (Management and Orchestration) [69], jonka tarkoituksena on standardoida NFV-hallinta. SDN ja NFV -sovellusten runsaan määrään johdosta on noussut tarve kehittää kokonaisia järjestelmiä, jotka kykenevät hallitsemaan ja seuraamaan sekä SDN-, että NVF -verkkoja. Tällaisia kokonaisia järjestelmiä ovat jo aikaisemmin mainitut ONOS ja VMware NSX. Taulukossa 2 on vertailtu ONOS ja NSX järjestelmiä. Taulukosta myös huomataan, että järjestelmät ovat hyvin toistensa kaltaisia.

Taulukko 2. *Yhteenveto virtualisointisovelluksista*

Sovellus	Kaupallinen	Virtuaalikytkin	OpenFlow-tuki	Hajautettu
ONOS	Ei	Kyllä	Kyllä	Kyllä
VMware NSX	Kyllä	Kyllä	Ei	Kyllä

ONOS on verkkokäyttöjärjestelmä (Network Operating System), joka tarjoaa jaetun ytimen, pohjoissuuntaisen ohjelmointirajapinnan, eteläsuuntaisen ohjelmointirajapinnan, sekä modulaarisen ohjelmistotuen [70]. Jaettu ydin on toteutettu Java-virtuaalikoneella, joka voidaan skaalata useammalle alustalle klusteroimalla. Pohjoissuuntainen API tarjoaa rajapinnan ylöspäin, jonka avulla voidaan esimerkiksi hallita, sekä seurata verkon resursseja. Etelänsuuntaisen API:n kautta tuettuna ovat useimmat yleiset protokollat, sekä uudet protokollat, kuten OpenFlow 1.5 ja OVSDB. Ohjelmallisia moduuleja voidaan kehittää pohjoissuuntaisen API:n kautta. Esimerkki tällaisesta moduulista on CORD-projektin vRouter [71], joka mahdollistaa fyysisen verkon kytkemisen osaksi virtuaalista verkkoa. Virtuaalikytkin vRouter on ohjelmistopohjainen toteutus, joten liikennöintinopeus virtuaalisesta verkosta fyysiseen verkkoon rajoittuu käytössä olevan alustan mukaan. Nopeampaa liikennöintiä fyysiseen verkkoon voidaan saavuttaa, jos käytössä on OF-DPA (OpenFlow Data Plane Abstraction) -yhteensopivia kytkinlaitteita [72]. OF-DPA-moduuli laajentaa Broadcom:n piirisarjaan perustuvia Linux-pohjaisia kytkinlaitteita OpenFlow 1.3.4 -tuella, jolloin niitä voidaan käyttää suoraan esimerkiksi ONOS-hallintakonsolista.

VMware NSX on osa VMwaren virtualisointituoteperhettä. Se toimii VMwaren oman ESXi hypervisorin päällä tai vaihtoehtoisesti KVM [73] -hypervisorin päällä. Järjestelmä on jaettu viiteen eri rooliin, jotka voidaan hajauttaa maksimissaan kolmeen eri kontrolleriin käyttäen Paxos-algoritmiä [47]. Paxos-algoritmin ansiosta saavutetaan korkea tavoitettavuus (High Availability), sillä jos joku kolmesta kontrollerista tai jokin siinä ajettavasta roolista lakkaa toimimasta, siirretään roolit toiselle kontrollerille. Järjestelmä käyttää verkon virtualisointiin avoimen lähdekoodin Open vSwitch -virtuaalikytkintä, jota ajetaan suoraan hypervisorin sisällä. Open vSwitch osaa hyödyntää myös tietyillä alustoilla DPDK (Data Plane Development Kit) [74] -kirjastoa, joka pienentää pakettien käsittelystä aiheutuvaa palvelimen kuormitusta. Toisin kuin ONOS, NSX ei käytä verkkojen väliseen kommunikointiin OpenFlow-protokollaa. Verkkojen välinen liikennöinti tapahtuu OVSDB-protokollan kautta. NSX-ympäristössä, virtuaaliset ja fyysiset verkot liitetään yhteen VXLAN (Virtual eXtensible Local Area Network) [75] -protokollan avulla. Open vSwitch tukee VXLAN–VLAN -yhdistämistä, joka tarkoittaa, että virtuaalisesta VXLAN-verkosta voidaan liikennöidä fyysiseen VLAN-verkkoon. Kuvassa 4 on havainnollistettu tilanne, jossa neljä virtuaalipalvelinta on yhdistetty virtuaalikytkimen kautta fyysiseen kytkimeen, jossa on kytkettynä kaksi fyysistä palvelinta. Esimerkin tapauksessa VXLAN-VLAN -liikenteen reititys tehdään ohjelmallisesti virtuaalikytkimessä. Fyysinen kytkin voi olla myös suoraan VXLAN-yhteensopiva [76], jolloin reititys ei kuormita virtuaalisointialustaa.



Kuva 4. *VXLAN-VLAN -tunneli virtuaalisesta verkosta fyysiseen verkkoon*

Ympäristöä on mahdollista laajentaa kolmannen osapuolen laajennoksilla, joiden avulla virtuaaliverkkoon voidaan tuoda esimerkiksi palomuuri- tai antivirus -palvelu [77]. VMware NSX tukee myös ECMP (Ecual Cost Multi-Path routing) -protokollaa [52], jonka avulla liikennettä voidaan reitittää vaihtoehtoisia reittejä [78]. Näin verkossa saavutetaan parempi viansietokyky, suurempi kaistanleveys ja kuormituksen tasaaminen.

Olemassa olevassa konesalissa VMware NSX:n kaltaisella järjestelmällä saavutetaan keskitetty hallinta ja verkkojen virtualisointi. VXLAN–VLAN -yhdistämisen ansiosta olemassa olevien fyysisten kytkimien jatkona voidaan käyttää virtuaalisia kytkimiä. Tämän tekniikan ansiosta NSX soveltuu myös yhdistämis-projekteihin, joissa olemassa oleva konesali siirretään osaksi uutta konesalia. Kustannussäästöjä saavutetaan käyttämällä jo olemassa olevaa laitekantaa. Lisää kustannussäästöjä muodostuu henkilötyötunneissa, sillä verkkojen hallinta on ketterämpää.

Uudessa konesalissa lähtökohdaksi voidaan ottaa haluttu SDN-sovellus, jonka mukaan valitaan käytettävät fyysiset verkkolaitteet. ONOS sopiikin paremmin juuri uuden konesalin käyttöönottoon. Kytkimiksi voidaan valita halvempia ns. tyhmiä kytkimiä, kuten esimerkiksi OF-DPA -yhteensopivat kytkimet. Kustannussäästöjä saavutetaan tällöin pienemmillä laitekustannuksilla [79]. Liikenteen vaihtoehtoiseen reitittämiseen voidaan käyttää esimerkiksi SDN-IP ONOS-moduulia [80], jolloin reititys ei perustu pelkästään ECMP-protokollaan, josta saattaa olla tietyissä tilanteissa jopa haittaa [81].

6. YHTEENVETO

Työ alkoi lyhyellä katsauksella tietoverkkojen historiaan. Historian tuntemus auttaa lukijaa paremmin ymmärtämään perinteisten tietoverkkojen haasteet, joita käsiteltiin luvussa 2. Samassa luvussa kerrattiin myös verkon peruskomponentin, kytkimen toimintaperiaate, sekä määriteltiin mitä termillä kytkin tarkoitetaan tässä työssä. Luvussa 3 käytiin läpi muutamia tapoja, miten perinteisen tietoverkon haasteita on pyritty ratkaisemaan. Tarkastelukohteiksi valikoitui tapoja, jotka ovat osaltaan vaikuttaneet merkittävästi Software-Defined Networkig -arkkitehtuurin kehitykseen. Luku 4 aloitettiin käymällä läpi, mitä SDN-arkkitehtuurilla tarkoitetaan, sekä listaamalla siitä saatavia hyötyjä. Tämän työn kannalta oli mielekästä jakaa SDN-toteutukset avoimen lähdekoodin sovelluksiin, ja kaupallisiin sovelluksiin, sillä tutkitut avoimen lähdekoodin sovellukset olivat toteutuksiltaan hyvin toistensa kaltaisia. Tutkituista avoimen lähdekoodin SDN-toteutuksista kaikki perustuivat OpenFlow-protokollaan. Luvussa 5 käsiteltiin verkkojen virtualisointia sekä virtualisoinnin liittymäkohtaa SDN-arkkitehtuuriin ja varsinkin OpenFlow-protokollaan. Toimintaperiaatteiden lisäksi tarkasteltiin haasteita, joita verkkojen virtualisointiin liittyy. Lopuksi esiteltiin kaksi SDN-sovellusta, ja miten niitä voidaan käyttää hyödyksi konesaliverkkojen virtualisoinnissa.

Tutkimuksen perusteella voitaneen sanoa, että suurin haaste verkkojen virtualisoinnissa OpenFlow-protokollan avulla lienee sitä tukevien laitteiden ja sovellusten laaja kirjo. OpenFlow-protokollan käyttöönotto jo olemassa olevassa konesalissa vaatii tarkkaa selvitystä fyysisten verkkolaitteiden kyvystä tukea OpenFlow-protokollan eri versioita. Tämän lisäksi OpenFlow-protokollan toteutuksessa on vielä valmistajakohtaisia eroavaisuuksia, joka saattaa johtaa tilanteeseen, jossa tietyn laitevalmistajan OpenFlow 1.3 -kykin ei ole yhteensopiva toisen valmistajan OpenFlow 1.3 -kytkimen kanssa. Verkkojen ollessa puhtaasti virtuaalisia, saavutetaan OpenFlow-protokollan avulla kuitenkin ketterämpi verkkojen hallinta sekä älykkäämpi liikenteen reititys. Älykkään reitityksen tarve puhtaasti virtuaalisissa verkoissa voitaneen kuitenkin kyseenalaistaa, sillä virtuaalinen kytkin ei kärsi laitevioista.

Tutkituista sovelluksista avoimen lähdekoodin ONOS toteuttaa sekä SDN-, että NFV -arkkitehtuurin periaatteita ja luo pohjan modulaariselle verkkokäyttöjärjestelmälle. Modulaarisuuden ansiosta ONOS-järjestelmään on mahdollista liittää OpenStack-liitännäisiä, jolloin on mahdollista käyttää myös Open vSwitch -virtuaalikytkintä [77]. ONOS ja OpenStack käyttävät kuitenkin erillisiä hallintakonsoleita, joten verkon OVS-kytkinten hallinta tehdään tällöin OpenStack-hallintakonsolista. Verkon kompleksisuuden mukaan voitaneen pohtia, toteutuuko tällöin enää SDN-arkkitehtuurin hengen mukainen keskitetty hallinta. ONOS soveltuukin paremmin uuteen konesaliin, joka on rakennettu valitun SDN-sovelluksen rajoitteet huomioiden.

Toisena tutkittavana sovelluksena oli kaupallinen VMware NSX. Se toteuttaa SDN-arkkitehtuurin perusperiaatteita irrottamalla verkon hallintakerroksen tiedonsiirtokerroksesta sekä NFV-arkkitehtuurin perusteita luomalla fyysisen verkon päälle virtuaalisen verkon. Tämän toteutuksen etu OpenFlow-protokollaan verrattuna on sen riippumattomuus OpenFlow-yhteensopivista laitteista. Järjestelmä voidaan ottaa käyttöön olemassa olevassa konesalissa helpommin, sillä IEEE 802.1Q [52] VLAN -standardi on vanhempi ja siksi myös yleisempi kuin OpenFlow-protokolla. VMware NSX soveltuu myös paremmin konesaleihin, joissa käyttöönoton ensisijainen tarkoitus on fyysisten palvelimien ja niihin liittyvän verkkotopologian siirtäminen virtuaalisille alustoille kokonaisuudessaan.

Tutkimuksen pohjalta voitaneen sanoa, että SDN-arkkitehtuuri yleensä ottaen soveltuu hyvin osaksi konesaliverkkojen virtualisointia. Saatava konkreettinen hyöty vaihtelee kuitenkin käyttökohteen ja valittavan toteutustavan mukaan. Olemassa olevassa konesalissa SDN-arkkitehtuurista saadaan hyötyjä esimerkiksi siirrettäessä fyysisiä kokonaisuuksia virtuaaliseen ympäristöön. Uudessa konesalissa SDN-arkkitehtuurista saadaan kuitenkin huomattavin hyöty. Uutta konesalia voidaan lähteä rakentamaan halutun SDN-ratkaisun pohjalta. Valittu SDN-ratkaisu määrittelee, minkälaisia fyysisiä laitteita toteutuksessa voidaan käyttää. Tulevaisuudessa SDN-arkkitehtuuria tullaan ottamaan entistä laajemmin käyttöön. Näin onkin tehnyt jo esimerkiksi Google [83], jonka SDN-arkkitehtuuria hyödyntävä verkko on yksi maailman suurimmista.

LÄHTEET

[1] A. Ghiasi, R. Baca, Overview of Largest Data Centers 2014. Saatavilla: http://www.ieee802.org/3/bs/public/14_05/ghiasi_3bs_01b_0514.pdf 9.10.2017

[2] I. Gahsinsky, Warehouse scale datacenters: the case for a new approach to networking 2011. Saatavilla: http://opennetsummit.org/archives/apr12/site/talks/gahinsky-tue.pdf 9.10.2017

[3] J. Kacprzyk, T. Tronco, New Network Architectures : The Path to the Future Internet, Springer, Berlin/Heidelberg, 2010.

[4] Open vSwitch VXLANs. Saatavilla: http://docs.openvswitch.org/en/latest/faq/vxlan/ 14.12.2017

[5] Open vSwitch QoS. Saatavilla: http://docs.openvswitch.org/en/latest/faq/qos/ 14.12.2017

[6] M.F. Bari, R. Boutaba, R. Esteves, L.Z. Granville, M. Podlesny, M.G. Rabbani, Q. Zhang, M.F. Zhani, Data Center Network Virtualization: A Survey, IEEE Communications Surveys & Tutorials, Vol. 15, Iss. 2, 2013, pp. 909-928.

[7] Internet Hall of Fame. Paul Baran. Saatavilla: http://internethalloffame.org/inductees/paul-baran 9.10.2017

[8] P. Baran, On Distributed Communications Networks, IEEE Transactions on Communications Systems, Vol. 12, Iss. 1, 1964, pp. 1-9.

[9] Hargrave's Communications Dictionary, Wiley, 2001.

[10] ISO, Protocol for providing the connectionless mode network services, RFC 926, http://doi.org/10.17487/RFC0926, 1984.

[11] A.V. Stokes, An annotated bibliography of communications standards, in: Anonymous (ed.), Communications Standards, Pergamon, 1986, pp. 457-478.

[12] B. Watson, Network protocol design with Machiavellian robustness. Macquarie University, 2010.

[13] Hedrick, C., Routing Information Protocol, RFC 1058, http://doi.org/10.17487/RFC1058, 1988.

[14] A.Kaushalram. Designing Fast And Programmable Routers. Saatavilla: https://cs.nyu.edu/~anirudh/anirudh_dissertation.pdf 16.12.2017

[15] IEEE Std 802.3-2015 (Revision of IEEE Std 802.3-2012): IEEE Standard for Ethernet, IEEE, 2016.

[16] K. Ramasamy, D. Medhi, The Morgan Kaufmann Series in Networking: Network Routing: Algorithms, Protocols, and Architectures (2), Elsevier Science, 2017.

[17] P. Almquist, F. Kastenholz. Towards Requirements for IP Router, RFC 1716, http://doi.org/10.17487/RFC1716, 1994.

[18] P. Goransson, C. Black, T. Culver, Software Defined Networks: A Comprehensive Approach, Second Edition, Second ed. Elsevier Science and Technology Books, Inc, Place of publication not identified, 2017.

[19] A. Campbell, I. Katzela, K. Miki, J. Vicente, Open Signaling for ATM, INTERNET and Mobile Networks (OPENSIG'98), ACM SIGOPS Operating Systems Review, Vol. 33, Iss. 2, 1999, pp. 15–28.

[20] J. Biswas, A.A. Lazar, J.-. Huard, K. Lim, S. Mahjoub, L.-. Pau, M. Suzuki, S. Torstensson, W. Wang, S. Weinstein, The IEEE P1520 standards initiative for programmable network interfaces, IEEE Communications Magazine, Vol. 36, Iss. 10, 1998, pp. 64–70.

[21] A. Doria, F. Hellstrand, K. Sundell, T. Worster. General Switch Management Protocol (GSMP) V3, RFC 3292, http://doi.org/10.17487/RFC3292, 2002.

[22] A. Doria, K. Sundell. General Switch Management Protocol (GSMP) Applicability, RFC 3294, http://doi.org/10.17487/RFC3294, 2002.

[23] R. Enns. NETCONF Configuration Protocol, RFC 4741, http://doi.org/10.17487/RFC4741, 2006.

[24] J. Case, M. Fedor, M. Schofftall, J. Davin. A Simple Network Management Protocol (SNMP) (Historic), 1990.

[25] Essential SNMP, 2d ed, in: Scitech Book News, Ringgold Inc, Portland, 2005.

[26] B.A.A. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, T. Turletti, A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks, IEEE Communications Surveys & Tutorials, Vol. 16, Iss. 3, 2014, pp. 1617–1634.

[27] OpenFlow: enabling innovation in campus networks, in: ACM SIGCOMM Computer Communication Review, ACM, NEW YORK, 2008, pp. 69–74.

[28] OpenFlow Switch Specification, Version 1.0.0 (Wire Protocol 0x01). Saatavilla: http://www.openflow.org/documents/openflowspec-v1.0.0.pdf 8.11.2017

[29] A. Lara, A. Kolasani, B. Ramamurthy, Network Innovation using OpenFlow: A Survey, IEEE Communications Surveys & Tutorials, Vol. 16, Iss. 1, 2014, pp. 493–512.

[30] Building a crash-proof internet, in: New Scientist, 2009, pp. 38-41.

[31] P. Dutta, Internet object caching, IN'98. 7th IEEE Intelligent Network Workshop Proceedings (Cat. No.98TH8364), pp. 95-118.

[32] M.H. Raza, S.C. Sivakumar, A. Nafarieh, B. Robertson, A comparison of software defined network (SDN) implementation strategies, Procedia Computer Science, pp. 1050-1055.

[33] Cisco NX-OS. Saatavilla: https://www.cisco.com/c/en/us/products/ios-nx-os-software/nx-os/index.html 15.11.2017

[34] NOX: towards an operating system for networks, in: ACM SIGCOMM Computer Communication Review, ACM, NEW YORK, 2008, pp. 105–110.

[35] The NOX Controller. Saatavilla: https://github.com/noxrepo/nox 15.11.2017

[36] Floodlight Is an Open SDN Controller. Saatavilla: http://www.projectfloodlight.org/floodlight/ 15.11.2017

[37] Floodlight Comptible Switches. Saatavilla: https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343519/Compatible+Switches 15.11.2017

[38] OpenStack. Saatavilla: https://www.openstack.org/ 15.11.2017

[39] Ryu. Saatavilla: https://osrg.github.io/ryu/ 15.11.2017

[40] Nutron Plugins and Drivers. Saatavilla: https://wiki.openstack.org/wiki/Neutron_Plugins_and_Drivers 15.11.2017

[41] OpenDaylight. Saatavilla: https://www.opendaylight.org/ 15.11.2017

[42] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, G. Parulkar, ONOS: towards an open, distributed SDN OS, Proceedings of the third workshop on hot topics in software defined networking, ACM, pp. 1–6.

[43] ONOS Wiki. Saatavilla: https://wiki.onosproject.org/display/ONOS/Wiki+Home 15.11.2017

[44] Open Networking Foundation and ON.Lab to Merge to Accelerate Adoption of SDN. Saatavilla: https://onosproject.org/2016/10/19/open-networking-foundation-and-on-lab-to-merge-to-accelerate-adoption-of-sdn/ 15.11.2017

[45] Nicira Networks: Disruptive Network Virtualization. Stanford CasePublisher 204-2012-1. 2012.

[46] The History of NSX and the Future of Network Virtualization. Saatavilla: https://www.vmware.com/radius/history-nsx-future-network-virtualization/ 15.11.2017

[47] Sreejith.C, VMware NSX Network Essentials, 1st ed. Packt Publishing, 2016.

[48] Is Cisco Application Centric Infrastructure an SDN Technology? Saatavilla: https://www.cisco.com/c/en/us/solutions/collateral/data-center-virtualization/application-centric-infrastructure/white-paper-c11-733456.html 27.11.2017

[49] Representational State Transfer (REST). Saatavilla: http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm 27.11.2017

[50] VMware and Nicira. Saatavilla: https://www.vmware.com/company/acquisitions/nicira.html 9.11.2017

[51] The Seven Properties of Network Virtualization. Saatavilla: https://ctovision.com/the-seven-properties-of-network-virtualization/ 9.11.2017

[52] IEEE Standard for Local and metropolitan area networks, IEEE Std 802.1Q-2014 (Revision of IEEE Std 802.1Q-2011), 2014, pp.1–1832.

[53] K. Iniewski, Internet Networks: Wired, Wireless, and Optical Technologies, 1st ed. CRC Press, GB, 2010

[54] B. Lantz, B. Heller, N. McKeown, A network in a laptop: rapid prototyping for software-defined networks, Proceedings of the 9th ACM SIGCOMM Workshop on hot topics in networks, ACM, pp. 1-6.

[55] Introduction to Mininet. Saatavilla: https://github.com/mininet/mininet/wiki/Introduction-to-Mininet 16.11.2017

[56] Production Quality, Multilayer Open Virtual Switch. Saatavilla: http://openvswitch.org/ 16.11.2017

[57] FlowVisor. Saatavilla: https://github.com/OPENNETWORKINGLAB/flowvisor/wiki 22.11.2017

[58] C. Chang, Y. Lin, OpenFlow Version Roadmap. Saatavilla: http://speed.cis.nctu.edu.tw/~ydlin/miscpub/indep_frank.pdf 20.11.2017

[59] OpenDaylight User Interface DLUX. http://docs.opendaylight.org/en/stable-nitrogen/getting-started-guide/common-features/dlux.html 16.12.2017

[60] The ONOS Web GUI. https://wiki.onosproject.org/display/ONOS/The+ONOS+Web+GUI 16.12.2017

[61] HyperGlance. https://www.hyperglance.com/product/ 16.12.2017

[62] J. Wickboldt, W. De Jesus, P. Isolani, C. Both, J. Rochol, L. Granville, Software-defined networking: management requirements and challenges, IEEE Communications Magazine, Vol. 53, Iss. 1, 2015, pp. 278-285.

[63] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, R. Boutaba, Network Function Virtualization: State-of-the-Art and Research Challenges, IEEE Communications Surveys & Tutorials, Vol. 18, Iss. 1, 2015, pp. 236–262.

[64] OF-CONFIG 1.2. Saatavilla: https://www.opennetworking.org/wp-content/uploads/2013/02/of-config-1.2.pdf 22.11.2017

[65] OF-CONFIG server for Open vSwitch. Saatavilla: https://github.com/openvswitch/of-config 22.11.2017

[66] B. Sonkoly, A. Gulyas, F. Nemeth, J. Czentye, K. Kurucz, B. Novak, G. Vaszkun, OpenFlow Virtualization Framework with Advanced Capabilities, 2012 European Workshop on Software Defined Networking, IEEE, pp. 18-23.

[67] Network Functions Virtualisation. Saatavilla: http://portal.etsi.org/NFV/NFV_White_Paper.pdf 23.11.2017

[68] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, A. Akella, OpenNF: enabling innovation in network function control, Proceedings of the 2014 ACM conference on sigcomm, ACM, NEW YORK, pp. 163-174.

[69] Open Source MANO. Saatavilla: https://osm.etsi.org/images/OSM-Whitepaper-TechContent-ReleaseONE-FINAL.pdf 23.11.2017

[70] Introducing ONOS - a SDN network operating system for Service Providers. Saatavilla: http://onosproject.org/wp-content/uploads/2014/11/Whitepaper-ONOS-final.pdf 23.11.2017

[71] CORD. Saatavilla: https://wiki.opencord.org/ 23.11.2017

[72] Broadcom, OF-DPA Software. Saatavilla: https://www.broadcom.com/products/ethernet-connectivity/software/of-dpa 23.11.2017

[73] Kernel Virtual Machine. Saatavilla: https://www.linux-kvm.org/page/Main_Page 24.11.2017

[74] Data Plane Development Kit. Saatavilla: https://www.intel.com/content/www/us/en/communications/data-plane-development-kit.html 24.11.2017

[75] Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks, RFC 7348, http://doi.org/10.17487/RFC7348, 2014.

[76] Hardware VXLAN Gateway. Saatavilla: https://www.vmware.com/resources/compatibility/pdf/vi_hvxg_guide.pdf 13.12.2017

[77] VMware NSX Technology Partners. Saatavilla: https://www.vmware.com/products/nsx/technology-partners.html 13.12.2017

[78] VMware® NSX for vSphere Network Virtualization Design Guide ver 3.0. Saatavilla: https://communities.vmware.com/servlet/JiveServlet/downloadBody/27683-102-10-41631/NSX Reference Design Version 3.0.pdf 13.12.2017

[79] White boxes are now ready for prime time, in: Network World (Online), Network World Inc, Southborough, 2016.

[80] SDN-IP Reactive Routing. Saatavilla: https://wiki.onosproject.org/display/ONOS/SDN-IP+Reactive+Routing 13.12.2017

[81] Multipath Issues in Unicast and Multicast Next-Hop Selection, RFC 2991, http://doi.org/10.17487/RFC2991, 2000.

[82] SONA. Saatavilla: https://wiki.onosproject.org/display/ONOS/SONA+Architecture 27.11.2017

[83] Software Defined Networking at Scale. Saatavilla: https://research.google.com/pubs/pub42948.html 21.12.2017