Lauri Heikkinen

TIETOTURVAN TOTEUTUS SD-WAN - OPERAATTORIPALVELUISSA

JYVÄSKYLÄN YLIOPISTO
TIETOJENKÄSITTELYTIETEIDEN LAITOS
2017

TIIVISTELMÄ

Heikkinen, Lauri
Tietoturvan toteutus SD-WAN -operaattoripalveluissa
Jyväskylä: Jyväskylän yliopisto, 2017, 95 s.
Tietojärjestelmätiede, pro gradu -tutkielma
Ohjaaja(t): Hämäläinen, Timo

Tutkielmassa selvitettiin vaihtoehtoja, joiden avulla palomuuraukseen liittyvä tekninen tietoturva voidaan toteuttaa uusissa SD-WAN -pohjaisissa verkottamisratkaisuissa ja niihin liittyvissä hybridiverkoissa operaattorin näkökulmasta. Palomuuraukseen liittyvää ratkaisua tulisi voida hyödyntää operaattorin asiakasverkoissa ja sen tulee skaalautua asiakkaiden käyttämien palveluiden mukaisesti. Tutkielmassa analysoitiin, kuinka SDN- ja NFV-teknologioita voidaan hyödyntää palomuurauksen toteutuksessa ja kannattaako palomuuraus toteuttaa SDN-pohjaisessa verkossa keskitetysti vai hajautetusti. Lisäksi pohdittiin, kuinka verkon hallinta voidaan toteuttaa keskitetysti, jotta jokaista verkon laitetta ei tarvitse konfiguroida erikseen.

Osana tutkielmaa tehtiin kokeellinen tutkimus, jossa luotiin SDN-arkkitehtuurin SouthBound-rajapinnalla toimivan OpenFlow-protokollan vuosääntöihin perustuva palomuurisäännöstö. Tämän lisäksi tehtiin viisi SD-WAN -palveluihin liittyvää toteutusta, joiden yhteensopivuutta luotuun säännöstöön analysoitiin. Toteutukset antoivat kattavan kuvan SDN-teknologian ja siihen liittyvien komponenttien tarjoamista ominaisuuksista palomuuraukseen liittyen. Säännöstön avulla voidaan toteuttaa tehokkaasti asiakasverkon palomuuraus OSI-mallin 2-4 kerroksilla. Kun vuosäännöt asetetaan SDN-verkon kytkimille proaktiivisen mallin mukaisesti, saadaan samalla optimoitua kaistankäyttöä ja parannettua vikasietoisuutta. Lisäksi SDN-kontrollerien suorituskyky paranee, kun verkon kytkimet kykenevät välittämään verkkoliikennettä itsenäisesti lähettämättä paketteja kontrollerille. Tutkimuksessa todettiin, että kontrollerit ja niihin liittyvät ulkoiset sovellukset tukevat palomuurausta vielä melko heikosti. Kontrollerien graafisista käyttöliittymistä tehtävä vuosääntöjen määrittely tapahtuu yksitellen, joka ei ole kestävä ratkaisu operaattorien käyttämissä ympäristöissä.

Jotta kokonainen palomuurisäännöstö voidaan tuoda SDN-pohjaiseen asiakasverkkoon, siihen tulee olla erillinen sovellus tai toiminnallisuus tulisi sisällyttää kontrollerien toimintalogiikkaan. Tällöin vuosäännöt kyettäisiin asettamaan proaktiivisesti kaikille verkon SDN-kytkimille operaattorien asiakasverkoissa ja vuosäännöt voitaisiin tarvittaessa räätälöidä asiakkaiden käyttämien palveluiden mukaisesti. Tutkimuksessa luotuja palomuurisääntöjä voidaan käyttää vaatimusmäärittelyissä muun muassa SDN-arkkitehtuurin palomuuraukseen liittyvien sovellusten tai kontrollerien toimintalogiikan kehittämisessä.

Asiasanat: SD-WAN, SDN, NFV, OpenFlow, Software Defined Security

ABSTRACT

Heikkinen, Lauri
Implementing Security in SD-WAN operator services
Jyväskylä: University of Jyväskylä, 2017, 95 p.
Information Systems, Master's Thesis
Supervisor(s): Hämäläinen, Timo

This study examines options how firewall based security could be implemented in SD-WAN based networking services from operator's point of view. Firewall solution shall be appropriate for operator's customer network environments and it shall be scalable based on services used by the customers. The study analyses how SDN and NFV based technologies could be utilized in firewall implementations and should the firewall solution be centralized or distributed. Also, implementing centralized network management for the firewall solution was investigated to avoid configuring each device on the network individually.

An experimental research was conducted as a part of the study. In the research part, set of firewall policies were made based on OpenFlow protocol that is the most common protocol at SDN architecture's southbound interface. Five practical SD-WAN services related implementations were made and compatibilities with created set of firewall policies were analyzed. The implementations gave a broad view on SDN technology and features provided by related components for firewall functionality. Created set of firewall policies can be used for implementing firewall functionality at OSI model 2-4 layers. Flow rules can be set to SDN switches in a network proactively which also optimizes a bandwidth usage and improves a fault tolerance of the network. Performance of SDN controller is improved since the switches on the network are capable to forward network traffic without sending packets to the controller. It was found within research that SDN controllers and related external applications aren't supporting firewall functionalities well. Flow rules may be created from graphic user interface only individually which is not appropriate solution for operator environments.

To be able to import a whole set of firewall policies to SD-WAN customer environment, an external SDN application is needed. Another option is to develop SDN controller's operation logic to support the firewall functionality. In that case flows could be set and distributed to all the switches on the SDN network proactively and flow rules could be modified based on services customer is using on the network. Set of firewall policies created within the study may be utilized as a base on requirement definitions for developing SDN architecture based applications and SDN controller's operation logic.

Keywords: SD-WAN, SDN, NFV, OpenFlow, Software Defined Security

KUVIOT

TAULUKOT

SISÄLLYS

1 JOHDANTO

Digitalisaation myötä yritysten ja organisaatioiden tietovarannot sekä sovellukset ovat siirtymässä tavallisista paikallisista palveluista yhä enemmän pilvipohjaisiksi palveluiksi. Palveluita ei enää ole kannattavaa ja kustannustehokasta tuottaa itse, vaan ne ostetaan yhä useammin ulkoisilta palveluntuottajilta valmiina. Vaikka liiketoiminta yrityksissä olisi paikallista, tarjottavat IT-palvelut ovat pääasiassa globaaleja ja niiden pitää olla kaikkialla saatavilla. Palveluiden siirtyessä pilveen, myös tietoverkkojen tulee jatkossa pystyä vastaamaan liiketoiminnan nopeisiin muutoksiin ja tukea tätä kehitystä. Tietoverkon pitää pystyä kattamaan yrityksen itsensä lisäksi myös kumppanit sekä keskeiset sisällöntuottajat ja palveluntarjoajat.

Nykyiset yritysverkot ovat pääasiassa toteutettu MPLS (Multiprotocol Label Switching) -teknologian avulla, joissa verkkoliikenne usein kulkee yhden tai muutaman yhdyspisteen kautta Internettiin ja sieltä edelleen palveluntarjoajien konesaleihin.

Ohjelmistopohjainen yritysverkko (SD-WAN, Software Defined Wide Area Network) on operaattoririippumaton, julkisen Internetin päälle rakennettu tietoverkko, joka on keskitetysti hallittavissa virtuaalisten verkkokomponenttien (NFV, network function virtualized) avulla. Virtuaalisilla verkkokomponenteilla tarkoitetaan esimerkiksi reititintä, jonka toiminnallisuus on virtualisoitu palvelimelle perinteisen fyysisen laitteen sijasta. Tietoverkkojen ohjelmoitavuudella mahdollistetaan automaatio. Sen myötä kustannustehokkuus ja ketteryys esimerkiksi palveluiden provisioinneissa paranevat.

Transitiota ohjelmistopohjaisiin yritysverkkoihin ei kuitenkaan voida tehdä hetkessä, vaan se tulee vaatimaan siirtymävaiheita, joissa on yhtä aikaa käytössä sekä vanhaa että uutta verkkoteknologiaa. Puhutaan niin sanotuista hybridiverkoista. Näissä esimerkiksi osa yrityksien toimipisteistä siirretään käyttämään ohjelmistopohjaisia verkottamisratkaisuja, kun taas osa toimipisteistä vielä hyödyntää edelleen aikaisempaa MPLS-pohjaista teknologiaa.

Uudistuvat WAN-palvelut asettavat uusia haasteita myös tietoturvaan digitaalisen maailman tietoturvauhkien lisääntyessä, yritysverkkojen siirtyessä yhä enemmän julkisen Internetin päälle sekä tiedonsiirtomäärien lisääntyessä.

Tietoturvan toteutus edellyttää kokonaisvaltaista palvelua, joka voidaan toteuttaa joko paikallisesti asiakaslaitteessa (CPE, Customer premises equipment) tai keskitetysti hallittuna pilvipalveluna (engl. Cloud Security). Tietoturvan toteuttamisessa tulee ottaa huomioon olemassa olevat, pääasiassa MPLS-teknologialla toteutetut yritysverkot, siirtymävaiheen hybriditoteutukset sekä uudet, puhtaasti ohjelmistopohjaisina toteutetut yritysverkot. Tietoturvan on pystyttävä mukautumaan dynaamisesti tietoverkon muutoksiin ja kehitykseen. Tässä yhteydessä puhutaan usein ohjelmistopohjaisesta tietoturvasta (Software Defined Security), jossa ohjelmistot voivat automaattisesti asettaa tietoturvakäytäntöjä tietoverkkoon esimerkiksi uuden sovelluksen käyttöönoton yhteydessä.

1.1 Keskeiset käsitteet

Yritysverkko (engl. Enterprise Wide-Area-Network) tarkoittaa loogisesti yrityksen tai organisaation käyttöön eriytettyä suojattua verkkokokonaisuutta. Yritysverkon pääasiallinen tarkoitus on muodostaa tietoturvalliset tietoliikenneyhteydet yrityksen toimipisteiden välillä. Lisäksi osana yritysverkkoa ovat usein konesalit, joista IT-palveluita tuotetaan organisaatioon. Yritysverkko muodostaa perustan organisaation verkottamiselle yhdistämällä työasemat, tulostimet sekä muut päätelaitteet toisiinsa (Ashok, R. 19).

Ohjelmistopohjainen verkottaminen (SDN, engl. Software Defined Networking) on ohjelmistoihin perustuva verkottamisarkkitehtuuri, jossa verkon hallintataso erotetaan välitystasosta. Tämä mahdollistaa verkkokomponenttien keskitetyn hallinnan ja ohjauksen. Verkon keskitettyä hallintaa suorittavia komponentteja kutsutaan kontrollereiksi (Rajat, K. 15).

Verkkokomponenttien virtualisointi (NFV) tarkoittaa verkon laitteiden, kuten kytkimien, reitittimien tai palomuurien virtualisointia palvelimille. Verkon toiminnot nostetaan aikaisemmalta fyysiseltä laitetasolta ohjelmistotasolle. Näitä ohjelmistoja voidaan ajaa palvelimilla toimittajariippumattomasti, joka parantaa verkon dynaamisuutta merkittävästi. Suurimmat hyödyt NFV-teknologiasta saadaan yhdistämällä se ohjelmistopohjaiseen verkottamiseen, jossa SDN-arkkitehtuurin tarvitsemat komponentit virtualisoidaan (Gray, K. & Nadeau, T. 2016).

OpenFlow on SDN-arkkitehtuurissa yleisesti käytetty protokolla hallinta- ja infrastruktuurikerroksien välissä. Se mahdollistaa SDN-kontrollerin ja infrastruktuurikerroksella olevien laitteiden välisen kommunikaation sekä verkkoelementtien keskitetyn hallinnan. Kun kontrolleri keskustelee standardoidun OpenFlow-protokollan välityksellä verkkoelementtien suuntaan, voidaan usean eri laitevalmistajan verkkolaitteita hallita keskitetysti ja samanaikaisesti (Rajat, K. 15).

Ohjelmistopohjainen yritysverkko (SD-WAN, Software Defined Wide-Area-Network). SD-WAN on ohjelmistopohjaiseen verkottamiseen perustuva yrityksille tarjottava verkottamisratkaisu. Se perustuu niin kutsuttuun overlay-malliin, jossa toimipisteiden välinen verkkoliikenne kapseloidaan ja voidaan välittää useampaa verkkoteknologiaa (kuten esimerkiksi MPLS, 4G, Internet) hyödyntäen yrityksen toimipisteiden välillä. Useamman verkkoteknologian sekä ohjelmistopohjaisen ohjauksen hyödyntämisen odotetaan tuovan läpinäkyvyyttä, dynaamisuutta sekä kustannustehokkuutta yritysverkottamiseen. (Ashok, R. 19).

Hybridiverkko (engl. Hybrid WAN). Hybridiverkolla tarkoitetaan yritysverkottamisessa arkkitehtuuria, joka sisältää vähintään kahta eri teknologiaa liityntäverkossa (engl. access network) asiakaslaitteen ja operaattorin reunalaitteen (PE, provider edge) välillä. Näitä teknologioita ovat esimerkiksi MPLS, laajakaista sekä 3G/4G. SD-WAN -toteutuksissa hybridiverkot abstraktoidaan overlay -ohjelmistokerroksen avulla ja alla olevia hybridiverkkoja kutsutaan tässä yhteydessä underlay-verkoiksi (IDC, 2016).

Palomuuraus tarkoittaa tietoturvaan liittyvää suojausmekanismia, jonka avulla pyritään kontrolloimaan ja monitoroimaan verkkoliikenteeseen liittyvää pääsynhallintaa (engl. access control) eri verkkosegmenttien välillä. Tyypillisesti palomuurausta tehdään julkisen Internetin ja organisaation sisäisen tietoverkon reunalla. Toiminta perustuu ennalta määritettyihin sääntöihin, joiden avulla voidaan esimerkiksi määritellä, sallitaanko vai estetäänkö tarkastuspisteen läpi suuntautuva liikennöinti Internetistä yrityksen verkkoon. Yleisimpiä palomuurausta käyttäviä toteuttavia komponentteja ovat palomuuri, pääsylistat sekä erilaiset tunkeutumisen tunnistamiseen ja estoon (IDS, intrusion detection system/IPS, intrusion prevention system) tarkoitetut järjestelmät. MPLS-pohjaisissa yritysverkoissa yleensä kaikki verkkoliikenne kierrätetään palomuurin kautta liikennöidessä organisaation sisältä sekä julkiseen verkkoon että Internetistä organisaation verkkoon (TechTarget, 2017).

Ohjelmistopohjainen tietoturva (SDS, engl. Software Defined Security) on ohjelmistojen ohjaukseen perustuva tietoturvamalli. SDS hyödyntää ohjelmistopohjaista verkottamista esimerkiksi tietoturvakäytäntöjen ja -poliitikoiden keskitettyyn hallintaan ja monitorointiin. Ohjelmistopohjaisella tietoturvalla voidaan toteuttaa ja automatisoida useita tietoturvakontrolleja, kuten tunkeutumisenestoa, verkon segmentointia tai pääsynvalvontaa (Sdxcentral, 2016).

1.2 Tutkimuksen motiivit

Yritysverkkojen siirtyminen ohjelmistopohjaisiin ratkaisuihin on erittäin ajankohtainen aihe. Yrityksen palveluiden siirtyminen nopeutuvalla tahdilla pilvipalveluiksi luo uusia vaatimuksia myös yrityksien verkottamispalveluihin. Tutkimus- ja konsultointiyritys Gartner (2016) on arvioinut, että jopa 50 prosenttia nykyisistä reititinlaitteista korvataan ohjelmistopohjaisilla ratkaisuilla vuoteen 2020 mennessä. Lisäksi SD-WAN -markkinan odotetaan kasvavan vuosittain noin 15 prosenttia, mikä luonnollisesti vähentää nykyisten verkottamisratkaisujen markkinaa huomattavasti. Tästä syystä markkinassa toimivat yritykset ovat lähteneet voimakkaasti kehittämään omia ohjelmistopohjaisia ratkaisujaan (NetworkWorld, 2016).

Markkinointimateriaalia on paljon, mutta kuinka uudet palvelut vastaavat todellisuudessa tietoturvan haasteisiin? Nopean kehityksen myötä tietoturvan toteuttaminen verkottamispalveluissa muuttuu. Uusissa ratkaisuissa ei ole enää järkevää tai tehokasta käyttää perinteisiä tietoturvaratkaisuja, kuten keskitettyä palomuuria tai yksittäisille reitittimille konfiguroitavia pääsylistoja.

Oma motivaationi tutkielman aiheeseen liittyy aiempiin teknologisesti suuntautuneisiin tietoverkko-opintoihini, joissa tein myös opinnäytetyöni tekniseen tietoturvaan liittyen. Lisäksi vastaan työtehtävissäni operaattorilla suuryritysasiakkaiden asiakasratkaisuista, joihin muun muassa yritysverkot ja niiden kehitys kuuluvat tiiviisti.

1.3 Aikaisempi tutkimus

Koska aihealue on yleistynyt vasta lähivuosina, tutkimusta aiheesta on suhteellisen vähän. Ohjelmistopohjaisesta verkottamisesta (SDN) ja virtualisoituihin verkkokomponentteihin (NFV) liittyen löytyy paljon tutkimusta, mutta SD-WAN -palveluihin ja etenkin sen tekniseen tietoturvaan liittyvää tutkimusmateriaalia on huomattavasti vähemmän. Tieteellisiä artikkeleja kuitenkin löytyy muutamia ja lähes kaikki tutkimustieto on uutta ja ajankohtaista.

Tarkoitus on pohjata tutkimus pääasiassa aiemmin julkaistuihin tieteellisiin artikkeleihin sekä Pro Gradu -tutkielmiin. Suomessa tekniset yliopistot, kuten Tampereen teknillinen yliopisto sekä Aalto yliopisto, ovat tutkineet aihealuetta aikaisemmin. Ulkomailta löytyy tutkimustietoa tieteellisten artikkeleiden muodossa muun muassa tutkimus- ja kehitysorganisaatioiden, kuten IDC:n tai IEEE:n tuottamana.

2 TUTKIMUKSEN TAVOITTEET JA MENETELMÄT

Tutkimuksen tavoitteena on tutkia, kuinka palomuuraukseen liittyvää tietoturva voidaan toteuttaa teknologisesta näkökulmasta uusissa ohjelmistopohjaisissa WAN-palveluissa, joissa nykyään hyödynnetään pääsääntöisesti MPLS-teknologiaa.

2.1 Tutkimuskysymys

Päätutkimuskysymys:

- Kuinka palomuuraukseen liittyvä tietoturva voidaan toteuttaa teknologisesta näkökulmasta ohjelmistopohjaisissa operaattorien tarjoamissa SD-WAN -palveluissa ja niihin liittyvissä hybridiverkoissa.

Mahdolliset alakysymykset:

- Mitä mahdollisuuksia SDN-teknologia ja virtualisoidut verkkokomponentit (NFV) tarjoavat tietoturvan toteutukseen (palomuuraukseen) yritysverkoissa?
- Kannattaako palomuuraus jatkossa toteuttaa keskitetysti pilvipalveluista vai paikallisesti asiakaslaitteilla (CPE)?
- Kuinka SD-WAN -palveluihin liittyvä tietoturva voidaan hallita keskitetysti?

2.2 Tutkimusmenetelmä

Tutkimus on tarkoitus perustaa konstruktiiviseen tutkimusotteeseen. Konstruktiivisen tutkimusotteen tarkoituksena on tuottaa uutta tietoa, jonka avulla voidaan ratkaista tutkimuskysymyksessä kuvattu ongelma. Kyseinen tutkimusmenetelmä on valittu, koska se sopii rakenteeltaan hyvin kuvatun ongelman ratkaisemiseen. Tutkimusongelmana tässä tutkielmassa on, ettei tarkkaan tiedetä, kuinka palomuuraukseen liittyvä tietoturva kannattaisi toteuttaa uusissa SD-WAN-palveluissa teknologisesta näkökulmasta. Kokeellisesta tutkimusprosessista saadut tulokset tullaan hyödyntämään teorian rinnalla. Tutkimusmenetelmään liittyvänä konstruktiona luodaan testiympäristöön OpenFlow-protokollaan perustuva malli palomuurisäännöstön luomiseksi sekä varsinaiset palomuurisäännöt, joiden toimivuutta arvioidaan ja validoidaan tutkimuksen edetessä eri toteutuksiin liittyen.

Konstruktiivinen tutkimusote voi sisältää sekä määrällistä että laadullista aineistoa, mutta tämä tutkimustyö on tarkoitus tehdä laadullisena eli kvalitatiivisena tutkimuksena. Laadullinen tutkimus on valittu, koska tutkimuksessa pyritään saamaan tarkempi käsitys aihealueesta sekä tuotetaan siihen liittyvää ymmärrystä lisäävää tietoa. Lisäksi kirjallisuuskatsaukseen liittyvässä osuudessa ja tiedonkeruussa tullaan hyödyntämään sisällönanalyysiä. (Saaranen, Kauppinen & Puusniekka, 2016).

Osana tutkielmaa tehdään kokeellinen tutkimus, jossa hyödynnetään useita eri ympäristöjä SDN-verkon toteuttamisessa. Kokeellisen tutkimuksen yhtenä toteutuksena on hyödynnetty Jyväskylän ammattikorkeakoulun CyberTrust-hankkeeseen liittyvää SDN-testiympäristöä (SDN testbed). Tarkoituksena on todentaa teknisen palomuuraukseen liittyvän tietoturvan toimivuutta ja sen käytössä esiintyviä haasteita (CyberTrust 2017).

Tiedonkeruu tutkimusaineistoa varten tapahtuu aihealueeseen liittyvistä aikaisemmista tutkimuksista, kirjallisuudesta sekä tieteellisistä artikkeleista. Edellä mainituista lähteistä on tarkoitus hakea teoriapohjaa käytännön tutkimukselle.

2.3 Tutkielman aiheen tarkennus ja rajaus

Tutkimuksen aihetta tarkastellaan operaattorin näkökulmasta. Tutkimuksessa otetaan huomioon olemassa olevat yritysverkot ja tietoturvan toteutukseen liittyvät mallit sekä niiden asettamat rajoitukset uusien verkottamisratkaisujen kehityksessä. Tutkimus keskittyy SD-WAN -ratkaisuiden tietoturvan toteutukseen, pääasiassa palomuuraukseen liittyvään teknologiaan, eikä se käsittele aihealueen kaupallista näkökulmaa tai markkinoilla olevia tuotteita. Myös hallinnolliseen tietoturvaan liittyvät seikat jätetään tarkastelun ulkopuolelle. Kokeelliseen tutkimukseen pyritään ottamaan kattava otanta erilaisia SDN-verkkoon

soveltuvia komponentteja, joihin tutkimus rajataan. Kattavalla otannalla varmistetaan, että tutkimuksesta saadaan mahdollisimman luotettava.

2.4 Odotetut tulokset

Tutkimuksen tuloksena odotetaan löytyvän ratkaisuehdotus, jonka avulla tekninen tietoturva voidaan toteuttaa SD-WAN palveluissa palomuurauksen näkökulmasta. Tuloksista odotetaan myös selviävän mitä seikkoja tässä toteutuksessa tulisi ottaa huomioon. Käytännön tuotos voisi olla esiteltyihin teknologioihin pohjautuvat malli, jonka avulla palomuurisäännöstö voidaan toteuttaa. Tarkastelussa on myös havaitut haasteet ja kuinka niitä voidaan ratkaista. On mielenkiintoista nähdä, kuinka asiat, joista on aikaisemmin saanut lukea vain ylätasolla markkinassa toimivien palveluntuottajien palvelukuvauksista, voidaan toteuttaa käytännössä.

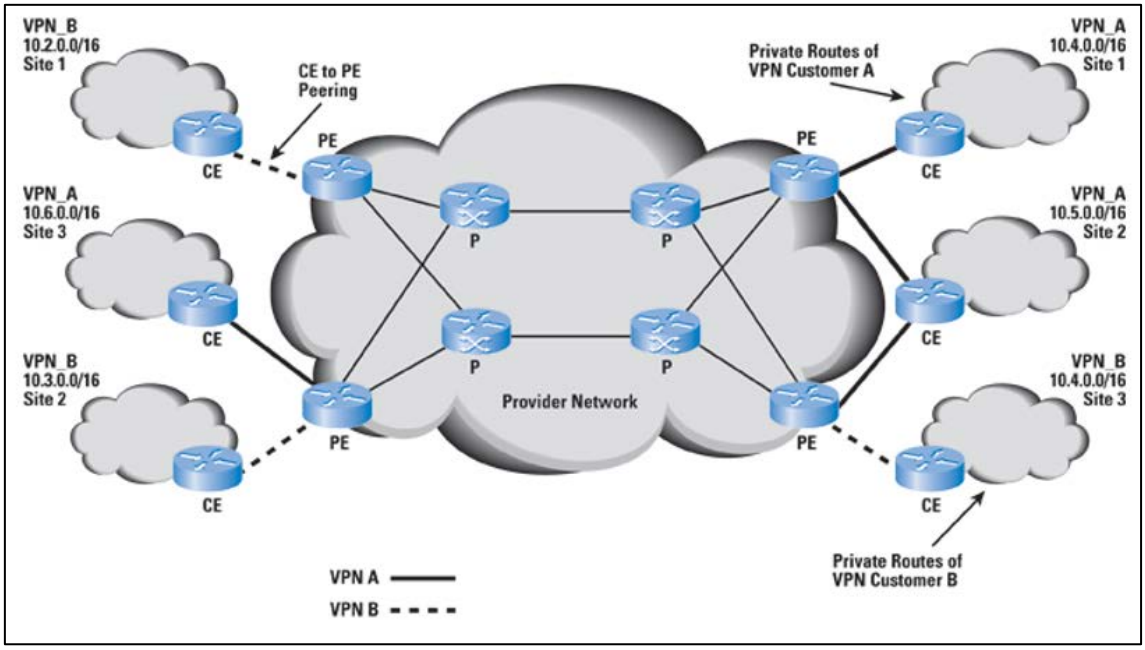
Erittäin todennäköisesti ohjelmisto-ohjattujen palveluiden merkitys myös tietoturvassa tulee kasvamaan ja tietoturvan toteuttaminen tulee automatisoitumaan entistä pidemmälle. Tutkimustuloksia voidaan hyödyntää myös muissa ohjelmistopohjaisissa palveluissa kuin ainoastaan ohjelmistopohjaisissa yritysverkoissa operaattorin näkökulmasta.

3 OPERAATTOREIDEN YRITYSVERKOTTAMIS-PALVELUT

3.1 Nykyisten verkottamisratkaisujen haasteet

Tämän päiväiset yritysverkottamisen ratkaisut perustuvat pääosin Frame Relay ja ATM -teknologioiden seuraajana tulleeseen MPLS-teknologiaan. MPLS on operaattorien runkoverkoissa käytettävä IP-pohjainen yhteydetön protokolla, jossa hyödynnetään leimakytkentää verkkoliikenteen välityksessä. Kun tavallisessa IP-reititykseen perustuvassa tietoverkossa jokainen reititinlaite analysoi IP-paketin otsikon ja tekee reitityspäätöksen sen perusteella, MPLS-verkossa paketti kytketetään eteenpäin IP-paketteihin lisättävien leimojen avulla. Leimaperusteinen kytkentä reitityspäätösten sijaan vähentää tarvetta tehdä reitityspäätöksiä runkoverkossa. Tämän avulla saavutetaan huomattavasti tehokkaampi verkkoliikenteen välitys ja parempi verkon suorituskyky. Lisäksi MPLS-teknologia tuo liikenneluokittelun ja protokollan yhdeydettömän toiminnan myötä paremman tuen reaaliaikasovelluksille, kuten VoIP:lle ja videokuvan välitykselle tietoverkoissa (NetworkWorld 2012).

Suojattujen yhteyksien muodostamiseen yritysten toimipisteiden välillä Internetin yli käytetään pääasiassa IPSec (Internet Protocol Security) -tunnelointia. Tällä saadaan toteutettua verkkoliikenteen salaus, osapuolten todennus sekä tiedon eheyden varmistaminen päästä päähän. Operaattorien asiakkaiden salatut yhteydet eristetään runkoverkossa loogisesti toisistaan VPN (engl. virtual private network) -teknologian avulla. Jokaiselle asiakkaalle on määritetty operaattorien MPLS-runkoverkossa oma VPN-leimansa, joilla he liikennöivät eristettynä toisistaan. VPN-leimattua MPLS-verkkoa kutsutaan tässä yhteydessä MPLS VPN -teknologiaksi. Kuviossa 1 on havainnollistettu MPLS-VPN -teknologian arkkitehtuuri. Asiakas A on kuvattu lihavoidulla kiinteällä viivalla ja asiakas B katkoviivalla. Asiakasliikenne kulkee runkoverkossa (P ja PE -laitteet) samojen fyysisten laitteiden kautta, mutta on loogisesti eristettynä toisistaan. (Rajendran, A. 2016).



KUVIO 1 MPLS-VPN -arkkitehtuuri (Cisco Systems 2015.)

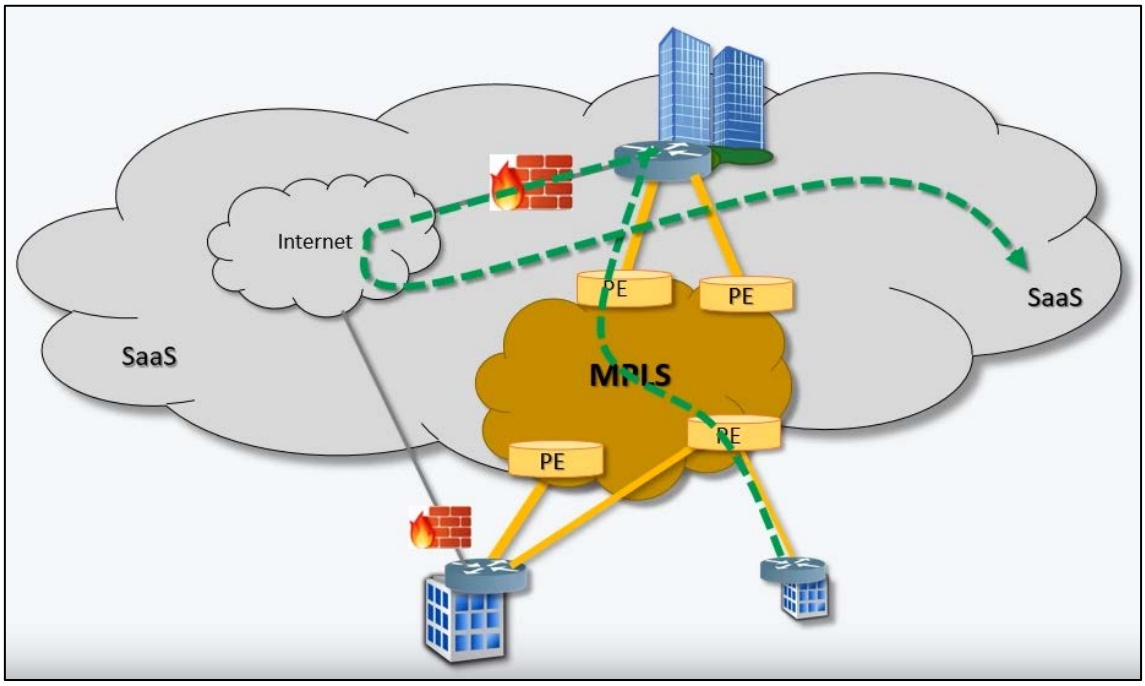
Missä MPLS-pohjaisten yritysverkkojen vahvuutena on ollut luotettavuus ja mahdollisuus viivekriittisiin sovelluksiin, haasteena ovat olleet muun muassa kustannustehokkuus, suhteellisen alhaiset siirtonopeudet, lisääntyvä kompleksisuus sekä verkkoliikenteen optimointi. Samalla yritysten liiketoiminnan vaatimukset verkon toiminnalle ovat kasvaneet merkittävästi. Muutokset nykyisiin yritysverkkoihin ovat pääsääntöisesti riippuvaisia operaattoreiden suorittamista toimenpiteistä. Nykyinen teknologia ei tue automaattisesti esimerkiksi yrityksen liiketoiminnan muutostarpeisiin liittyviä muutoksia verkossa ja näin ollen kaikki verkkoon liittyvät muutokset tehdään pääsääntöisesti manuaalisesti tilauksesta.

Tämän seurauksena yritysverkottamisen markkinaan on tullut laajakaista- sekä 3G/4G -teknologioihin perustuvia ratkaisuja. Näissä ratkaisuissa provisioinnit saadaan tehtyä aikaisempaa nopeammin, mikä vastaa yritysten liiketoiminnan kasvavaan muutosnopeuteen. Yritykset käyttävät näitä etenkin pienillä toimipisteillä ja MPLS-pohjaisten liittymien varayhteyksinä. MPLS-pohjaiset liittymät ovat pääasiassa kuitenkin säilyneet yritysten toimipisteiden ensisijaisina yhteyksinä niiden luotettavuuden asioista. Koska Laajakaista- sekä 3G/4G -pohjaisissa ratkaisuissa verkkoliikenne toimipisteiden välillä kulkee osittain salattuna julkisen Internetin yli, liittymien saatavuus ja siihen liittyvät palvelutasot (SLA) eivät ole vielä MPLS-liittymien kanssa samalla tasolla (NetworkWorld, 2012).

Digitalisaation myötä yhä useammat organisaatiot ovat siirtyneet käyttämään IT-palvelutuotannossaan kaupallisia pilvipalveluita (SaaS, engl. Software as a Service) kuten SalesForce, Microsoft Office 365 tai Amazon AWS. Tutkimusyhtiö IDC:n tekemän kyselyn (IDC, 2016) mukaan, jopa 80% yrityksistä on siirtymässä käyttämään julkisia pilvipalveluita seuraavan vuoden sisällä. Jatkuvasti vähenevät sisäiset IT-resurssit ja heikko kustannustehokkuus ovat myös

osaltaan vauhdittaneet tätä kehitystä. Kehityksen myötä palvelut on oltava saatavilla kaikkialla työntekijän sijainnista tai päälaitteesta riippumatta. Lisäksi tiedonsiirtomäärät julkisesta tietoverkosta tarjottaviin pilvipalveluihin kasvavat räjähdysmäisesti. Myös yritysverkkojen on luonnollisesti pystyttävä vastaamaan digitalisaation tuomaan kehitykseen.

Nykyiset MPLS-pohjaiset yritysverkot, joissa verkkoliikenne julkiseen verkkoon reitietään pääsääntöisesti keskitetyn yhden tai muutaman Internet liittymän sekä palomuurin kautta, eivät kuitenkaan ole enää optimaalisin vaihtoehto. Liikenteen välityksen kannalta tästä aiheutuu ylimääräisiä viiveitä sekä kuormaa yritysverkkoon. Kuviossa 2 on havainnollistettu vihreällä katkoviivalla verkkoliikenteen reititys yrityksen sivutoimipisteestä julkisessa verkossa sijaitsevaan SaaS-palveluun. Verkkoliikenne kiertää MPLS-verkon läpi päätoimipisteelle, josta se välitetään palomuurin kautta Internettiin.



KUVIO 2 MPLS -teknologiaan perustuva verkkoarkkitehtuuri (Solutions Reservoir, 2016.)

Toinen keskeinen haaste nykyisessä teknologiassa on verkkokapasiteetin heikko käyttöaste. Nykyisissä ratkaisuissa hyödynnetään pääsääntöisesti ensisijaiseen ja varayhteyteen perustuvaa mallia, jossa ainoastaan toinen yhteyksistä on aktiivisesti käytössä. Varayhteyden kapasiteettia käytetään vain tapauksissa, joissa ensisijainen yhteys vikaantuu ja ei ole saatavilla. Tämä asetelma monimutkaistaa myös huomattavasti verkon hallintaa, kun liittymien konfiguraatiossa joudutaan ottamaan huomioon myös uudelleenreititykset ja niihin liittyvät prioriteetit mahdollisissa häiriötilanteissa. Muita verkon hallintaa monimutkaistavia tekijöitä ovat muun muassa multicast, liikenteen luokittelu (QoS, engl. Quality of Service) sekä poikkeavien tietoturvakäytäntöjen toteuttaminen yrityksen toimipisteille (ONUG SD-WAN Working Group, 2014).

3.2 Tietoturvan merkitys yritysverkoissa

Internettiin kytkettävien laitteiden ja tiedonsiirron määrän kasvu ovat asettaneet uusia haasteita tietoturvalle. Tietoverkkoon kytketään yhä enemmän esimerkiksi esineiden Internetiin (IOT, Internet of Things) liittyviä laitteita, joissa tietoturva on erittäin heikolla tasolla ja näin ollen hankalasti hallittavissa. Jotta organisaation tietoturva olisi hallittavissa, täytyy tietoturvan tilannekuvan olla kattava ja ajantasainen. Tämä edellyttää, että tietoturvapalvelut ovat keskitetysti toteutettu. Lisäksi julkisen Internetin yli siirretään organisaatioiden liiketoiminnalle yhä kriittisempää informaatiota, kun yritykset siirtyvät lisääntyvässä määrin käyttämään julkisia pilvipalveluita. Tästä johtuen operaattoreiden on jatkossa pystyttävä tarjoamaan verkottamispalveluissaan tietoturvaratkaisuja, jotka skaalautuvat suuriin laite- ja datansiirtomääriin sekä tukevat Internetin hyödyntämistä siirtotienä tehokkaasti.

Digitalisaatio on tuonut mukanaan uusien palveluiden myötä myös paljon uusia haavoittuvuuksia ja tietoturvauhkia organisaatioille. ENISA:n vuosittaisen tietoturvaraportin (2017) mukaan ulkoverkosta kohdistettavat hyökkäykset, kuten malwaret, web-pohjaiset- sekä palvelunestohyökkäykset ovat jälleen kasvaneet edelliseen vuoteen verrattuna. Trendinä hyökkäyksissä on ollut, että hyökkääjät pyrkivät entistä enemmän hyötymään niistä taloudellisesti (ENISA, 2017).

Malwareihin liittyvät hyökkäykset ovat olleet pääasiassa ransomwareja, joissa kohteen kovalevyillä olevat tiedostot salataan ja niiden palauttamisesta takaisin vaaditaan käyttäjiltä yleensä kryptovaluutoilla, kuten bitcoinilla maksettavia lunnaita. Lisäksi malwaret ovat olleet informaation varastamiseen liittyviä, joissa käyttäjiltä varastetut tiedot pyritään myymään edelleen verkossa (ENISA, 2017).

Web-pohjaisissa hyökkäyksissä uhkat liittyvät käyttäjien web-selaimeen liitännäisineen sekä web-palvelimeen, johon käyttäjä ottaa yhteyden. Palvelunestohyökkäyksen malwarejen tapaan on pyritty kaupallistamaan. Niitä myydään palveluna tai niiden aiheuttamasta tietomurrosta pyritään hyötymään rahallisesti toisella tapaa. Näissä käytetään hyväksi saastuneita verkon laitteita, joiden avulla saadaan generoitua suuria, jopa kapasiteetiltaan terabittien suuruisia hyökkäyksiä. Kuviossa 3 esitetyn raportin mukaan nämä kaikki yleisimmät hyökkäykset (TOP5) ovat sellaisia, joita pyritään estämään ensisijaisesti palomuuraukseen liittyvän tietoturvan avulla (ENISA, 2017).

Top Threats 2015	Assessed Trends 2015	Top Threats 2016	Assessed Trends 2016	Change in ranking
1. Malware	↑	1. Malware	↑	→
2. Web based attacks	↑	2. Web based attacks	↑	→
3. Web application attacks	↑	3. Web application attacks	↑	→
4. Botnets	↓	4. Denial of service	↑	↑
5. Denial of service	↑	5. Botnets	↑	↓
6. Physical damage/theft/loss	⊃	6. Phishing	⊃	↑
7. Insider threat (malicious, accidental)	↑	7. Spam	↓	↑
8. Phishing	⊃	8. Ransomware	⊃	↑
9. Spam	↓	9. Insider threat (malicious, accidental)	⊃	↓
10. Exploit kits	↑	10. Physical manipulation/damage/ theft/loss	↑	↓
11. Data breaches	⊃	11. Exploit kits	↑	↓
12. Identity theft	⊃	12. Data breaches	↑	↓
13. Information leakage	↑	13. Identity theft	↓	↓
14. Ransomware	↑	14. Information leakage	↑	↓
15. Cyber espionage	↑	15. Cyber espionage	↓	→

Legend: Trends: ↓ Declining, ⊃ Stable, ↑ Increasing
Ranking: ↑ Going up, → Same, ↓ Going down

KUVIO 3 Yhteenveto ja vertailu tietoturvauhkista (Enisa, 2017)

3.3 Palomuuraus nykyisissä yritysverkoissa

ONUG SD-WAN työryhmän (2014) mukaan varsinkin useita toimipisteitä käsittävissä yritysverkoissa käytetään yleensä useita eri palomuuraukseen liittyviä tietoturvaratkaisuja. Muun muassa IT-palveluiden ja niitä käyttävien laitteiden voimakas lisääntyminen ovat johtaneet tilanteeseen, jossa pelkkä palomuuri ei ole enää riittävä suojautumiskeino. Käytössä on samanaikaisesti useampia palomuurityyppejä sekä muita tietoturvapalveluita kuten IDS/IPS-laitteita tai sisällönsuodattimia. Tämän avulla pyritään varmistamaan optimaalinen pilvipalveluiden ja verkkoyhteyksien siirtokapasiteetin käyttö pienemmillä toimipisteillä. Nopeasti kasvava verkko- ja tietoturvapalveluiden tarve on johtanut siihen, että organisaatiot ovat hankkineet usean eri toimittajan tuotteita riittävän tietoturvan toteuttamiseksi. Manuaalisen laitekohtaisen konfiguroinnin lisäksi kaikki nämä laitteet tarvitsevat jatkuvaa ylläpitoa sekä elinkaaren hallintaa. Tämä tekee niiden hallinnan monimutkaiseksi ja lisää tuotantokustannuk-

sia tietoturvaan liittyvien sisäisten henkilöstöresurssien supistuessa organisaatioissa samalla jatkuvasti (ONUG 2014).

3.3.1 Pääsynhallinta

Tietoverkkoturvallisuuden kontekstissa pääsynhallinnalla tarkoitetaan pääsyn kontrollointia verkkoliikenteen päätepisteiden, kuten esimerkiksi loppukäyttäjän työaseman sekä web-palvelimen välillä. Pääsynhallin perustuu ennalta määritettyihin tietoturvapolitiikkoihin. Organisaation tietoturvapolitiikat perustuvat tunnettuihin uhkiin ja haavoittuvuuksiin. Ne määrittävät ylätasolla kenellä on pääsy mihinkin palveluun ja millä keinoilla. Pääsynhallinnassa voidaan hyödyntää roolipohjaista pääsynhallintaa, jossa tietylle käyttäjäryhmälle määritetään samanlaiset oikeudet. Tietoverkkoturvallisuuden kontekstissa pääsynhallinta nostetaan usein ylemmälle tasolle, jossa voidaan säädellä ja rajoittaa pääsy tiettyyn verkon resurssiin tai palveluun koko organisaation tasolla. Pääsynhallinnassa hyödynnetään vähimmän oikeuksien (engl. least privilege) -periaatetta. Tämä perustuu ajatukseen, että käyttäjille annetaan mahdollisimman suppeat oikeudet, jotka ovat välttämättömät annetun tehtävän suorittamisessa (Pfleeger, P., Pfleeger, S. & Margulies, J, 2015, 2.0).

3.3.2 Palomuuri

Keskeisin palomuuraukseen liittyvä komponentti teknisen tietoturvan toteuttamiseen yritysverkoissa on 90-luvun alussa keksitty palomuuri. Sen tarkoitus on suodattaa verkkoliikennettä eri verkkosegmenttien välillä. Yleisin tapaus tästä on suodatus luotettujen ja ei-luotettujen verkkosegmenttien välillä organisaation sisäverkon ja Internetin rajalla. Tässä tapauksessa kaikki verkkoliikenne näiden verkkojen välillä reititetään palomuurin läpi ja näin ollen kaikkea organisaation verkkoliikennettä voidaan seurata ja hallita. Palomuurilla voidaan myös segmentoida ja suodattaa luotettujen sisäisten verkkojen välistä liikennettä. Verkkoliikenteen suodatus tapahtuu tietoturvapolitiikoihin perustuvien, ennalta määritettyjen palomuurisääntöjen avulla.

Palomuurit ovat perinteisesti toimineet OSI-mallin kerroksilla 2-4, mutta nykyään ne käsittelevät usein verkkoliikennettä sovelluskerrokselle 7 asti. Kerroksilla 2-3 toimivat palomuurit tarkastelevat läpi kulkevien Ethernet-kehysten tai IP-pakettien osoitetietoja, kun taas kerroksella 4 toimivat palomuurit voivat tarkastella näiden lisäksi myös TCP (Transmission Control Protocol) ja UDP (User Datagram Protocol) -portteja sekä seurata liikenteen yhteydellisyyttä. Palomuuri vertaa sen läpi meneviä paketteja asetettuihin sääntöihin ja tekee sen perusteella paketin käsittelyyn liittyvät päätökset. Näitä ovat esimerkiksi paketin välittämisen salliminen, estäminen, salaaminen, priorisointi, seuranta ja jonotus.

Palomuuri on tavanomaisesti toteutettu fyysisellä, tähän tarkoitukseen tarkoitetulla laitteella, koska se joutuu käsittelemään paljon tietoliikennettä, joka asettaa korkeat vaatimukset laitteen suorituskyvylle. Teknologian kehitty-

essä palomuureja virtualisoidaan konesalipalvelimille yhä useammin muun muassa skaalautuvuuden parantamiseksi. Palomuuri voidaan lisäksi toteuttaa pelkkänä sovelluksena. Tätä mallia käytetään esimerkiksi työasemien palomuuriratkaisuissa (Pfleeger, P., Pfleeger, S. & Margulies, J, 2015).

Toiminnallisuudet vaihtelevat laajasti palomuureissa, mutta ylätasolla ne voidaan jakaa kolmeen kategoriaan: Tilaton, tilallinen ja sovelluspalomuuri. Mikään näistä ei ole yleisesti paras, vaan jokaiselle on oma käyttötarkoituksensa ja niihin liittyvät vahvuutensa.

Tilaton palomuuri on yksinkertaisin, mutta joissain tapauksissa myös tehokkain palomuurin tyyppi. Tällaisia ovat esimerkiksi seuraavassa kappaleessa esiteltävät pääsylistat. Verkkoliikenteen suodatus perustuu staattisiin sääntöihin, joissa määritetään datan (esim. IP-paketti, Ethernet-kehys) lähde- ja kohdeosoitteet ja/tai käytettävä protokolla. Tilaton palomuuri tarkastelee ainoastaan paketin otsikkoa, ei mitä paketti sisältää. Tämän ansiosta se pystyy tehokkaaseen paketinvälitykseen. Jokainen paketti tarkistetaan säännöstöä vasten. Konfiguraatio on staattinen ja vaatii manuaalista konfiguraatiota muutosten yhteydessä. Haasteena tilattomassa palomuurissa on, että lista säännöistä kasvaa usein pitkäksi ajan myötä ja sen hallittavuus on vaikeaa ja työlästä (Pfleeger, P., Pfleeger, S. & Margulies, J, 2015).

Tilallinen palomuuri kykenee seuraamaan verkko-osoitteiden ja protokollien lisäksi yhteyksien tilaa sekä pakettien järjestystä. Tämän avulla voidaan esimerkiksi dynaamisesti sallia vastaus Internetissä olevalta web-palvelimelta sisäverkon yhteyspyyntöön. Tilallinen suodatus toimii kahden pääperiaatteen mukaan: ensiksi verkkoliikenne luokitellaan tarkastelemalla ja tämän jälkeen palomuuri pitää kirjaa yhteyden tilasta tarkastelemalla jokaista tapahtumaa siihen asti, kuin yhteys suljetaan. Näiden ominaisuuksien avulla on mahdollista laajentaa palomuurin toiminnallisuutta. Kun palomuuri saa haltuunsa IP-paketin, se käyttää tilataulua (engl. state table) tarkistaakseen, onko yhteys muodostettu (engl. established) -tilassa tai onko pyynnön saapuvalle IP-paketille tehnyt lähiverkossa sijaitseva päätelaite. Jos kumpikaan edellä mainitut kriteerit eivät täyty, IP-paketti käsitellään palomuuriin määriteltyjen sääntöjen perusteella. Tilallisen palomuurin yhteyden tilaan perustuva suodatus on skaalautuva ja läpinäkymätön käyttäjille (Pfleeger, P., Pfleeger, S. & Margulies, J, 2015).

Sovelluspalomuuri on OSI-mallin sovelluskerroksella (7) toimiva palomuurityyppi. Se pystyy suodattamaan verkkoliikennettä otsikkotietojen lisäksi myös paketin sisällön perusteella. Verkkoliikenne voidaan tunnistaa ja suodattaa varsinaisen sovelluksen tai esimerkiksi verkkosivuston perusteella. Sovelluspalomuuri pystyy tutkimaan paketin hyötykuormaa ja näin ollen suodatusta voidaan tehdä myös paketin sisältöön liittyen. Sen avulla voidaan tehdä myös verkkoliikenteen monitorointiin, muokkaamiseen tai uudelleenohjaukseen liittyviä sääntöjä. Sovelluspalomuurin avulla voidaan myös esimerkiksi seurata ja estää organisaation sisältä mahdollisesti haitallisille sivustoille tehtäviä yhteydenottoja. Lisäksi sovelluspalomuurit tarjoavat huomattavasti paremmat mahdollisuudet lokitietojen keräämiseen ja tarkasteluun jälkikäteen. Sovelluspalo-

muurien heikkoutena ovat yleisesti läpimenevään datamäärään liittyvät rajoitukset, hitaus verkkoliikenteen käsittelyssä muihin palomuurimalleihin verrattuna sekä tähän liittyvät suorituskykyvaatimukset. (Pfleeger, P., Pfleeger, S. & Margulies, J, 2015).

3.3.3 Pääsylistat

Palomuurien lisäksi palomuurausta toteutetaan pääsylistojen (engl. ACL, access control list) avulla. Suodatukseen liittyvä toimintaperiaate on sama kuin palomuurissa, mutta keskeinen ero on siinä, että pääsylistat ovat laitekohtaisia, eikä niitä yleensä ole keskitetty ainoastaan yhdelle verkkolaitteelle palomuurin tapaan. Pääsylista on käytännössä sarja verkkolaitteen, kuten reitittimen tai kytkimen, konfiguraatioon lisättyjä määrityksiä, joiden avulla voidaan kontrolloida laitteesta sisään- ja ulosmenevää verkkoliikennettä tiettyyn resurssiin tai palveluun.

Pääsylistat voidaan määrittää verkkolaitteen sekä sisään tulevaan että ulos lähtevään rajapintaan. Pääsylista sisältää palomuurin tapaan sääntöjä, joiden avulla verkkoliikennettä voidaan suodattaa, mutta pääsylistoissa suodatus tapahtuu OSI-mallin kerroksilla 3-4. Näin ollen suodatus perustuu kohde- ja lähde IP-osoitteisiin sekä TCP/UDP -portteihin, eikä esimerkiksi varsinaista käytettävää sovellusta voida palomuurin tapaan tunnistaa. Pääsylistojen avulla ei kuitenkaan voidaan tehdä esimerkiksi liikenteen priorisointia, vaan IP-paketeille tehtävät toimenpiteet rajoittuvat pääasiassa sallimiseen tai estämiseen. Lisäksi pääsylistoilla voidaan tehdä verkkoliikenteen merkkausta ja seurantaa esimerkiksi laitteiden sisäänkirjautumiseen liittyen. Pääsylistojen haasteena on, että ne ovat staattisia ja vaativat verkkomuutoksiin liittyen manuaalista, jokaiselle yksittäiselle laitteelle erikseen tehtävää konfiguraatiota. Ne eivät skaalaudu uusien SD-WAN kaltaisten uusien verkottamispalveluiden malliin, joissa hallinta on keskitetty (Orbit Computer, 2015).

3.3.4 Tunkeutumisenesto järjestelmät (IDS/IPS)

Palomuuri suojaa organisaatiota pääasiassa ulkoverkosta kohdistuvia hyökkäyksiä vastaan. Koska tietoturvauhkia muodostuu usein myös organisaation sisältä, tarvitaan muita tietoturvakontrolleja palomuurin toteuttamien tietoturvapolitiikoiden lisäksi. Organisaation sisältä tulevat tietoturvauhat voivat olla tahallisesti aiheutettuja tai tahattomia, esimerkiksi käyttäjien virheistä johtuvia. Tunkeutumisenesto järjestelmillä pyritään tunnistamaan ja estämään nämä sisäiset tieturvauhat sekä ulkoiset uhat, joita palomuuri ei kykene tunnistamaan ja estämään. Niiden vahvuutena palomuuriin verrattuna on verkkoliikenteen sisällöstä tehtävät havainnot mahdollisiin tietoturvauhkiin liittyen. Järjestelmien haasteena on täsmällinen konfigurointi niin, etteivät ne joko estä sallittua verkkoliikennettä tai generoi hallitsematonta määrää aiheettomia hälytyksiä. IDS/IPS-järjestelmän vaativat organisaation tietoturvahenkilöstöltä jatkuvaa

lokitietojen seurantaa ja hälytyksiin reagointia (Pfleeger, P., Pfleeger, S. & Margulies, J, 2015).

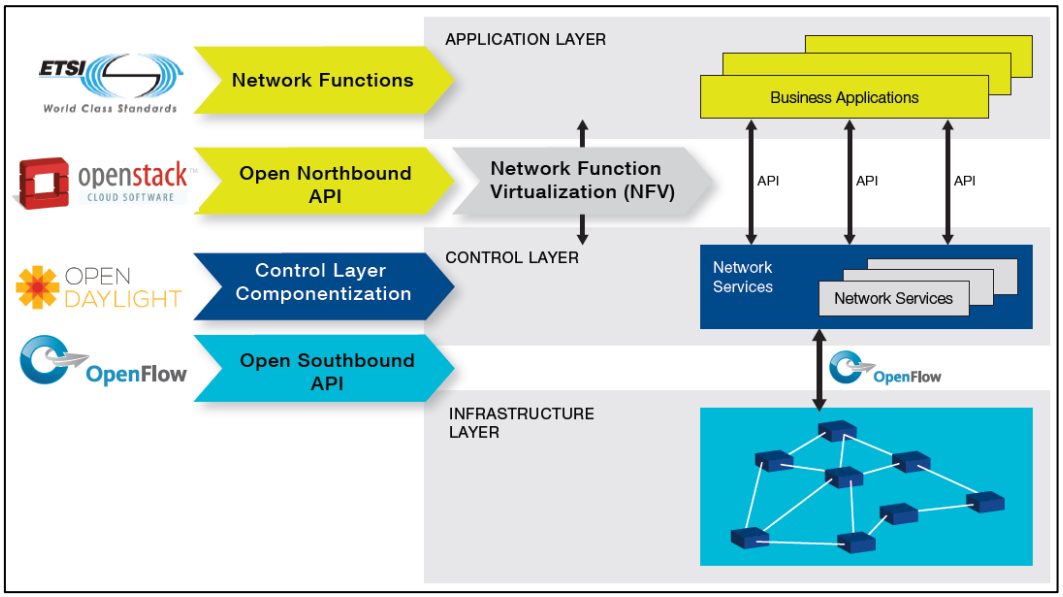
Tunkeutumisenesto järjestelmät käsittävät kaksi osa-kokonaisuutta: Tunkeutumisen havaitsemiseen tarkoitetut IDS-järjestelmät sekä tunkeutumisen varsinaiseen estämiseen tarkoitetut IPS-järjestelmät. Molemmat voivat sijaita joko omalla fyysisellä laitteellaan tai ne voivat olla toteutettuna ohjelmallisesti esimerkiksi palomuurisovelluksen yhteyteen.

IDS-järjestelmän tarkoitus on tuottaa mahdollisiin hyökkäyksien havaitsemiseen liittyvää automaatiota. Se monitoroi tietoverkon liikennettä ja pyrkii havaitsemaan ja raportoimaan poikkeamia ennalta määritettyjen sääntöjen perusteella. Sillä saadaan kerättyä tarkkaa lokitietoa verkkoliikenteestä analysointia varten. Havainnointi perustuu yleensä poikkeamiin verkon normaalitilasta (engl. baseline), jossa on määritetty tiedot verkon, työasemien, ohjelmien ja laitteiden lähtötilasta. Havainnot voivat liittyä joko aiempiin allekirjoituksiin, joissa tietty kaava verkkoliikenteessä toistuu, tai heuristiseen malliin, jossa järjestelmä oppii tunnistamaan poikkeamat normaalitilasta ajan myötä. IDS-järjestelmä ei havainnoinnin ja raportoinnin lisäksi suorita varsinaisia suojautumistoimenpiteitä, vaan ne täytyy suorittaa joko organisaation tietoturvahenkilöstön tai IPS-järjestelmän avulla (Pfleeger, P., Pfleeger, S. & Margulies, J, 2015).

IPS-järjestelmä on laajennus IDS-järjestelmästä, jonka tarkoitus on estää luvaton verkon käyttö, datan lukeminen, tai järjestelmien käyttö sekä raportoida tehdyistä toimenpiteistä. Estämisen lisäksi järjestelmä voi esimerkiksi kerätä poikkeamaan liittyviä lokitietoja normaalia tarkemmalla tasolla, kutsua muita suojautumisjärjestelmiä tai hälyttää tietoturvahenkilöstön tekemään tarvittavia manuaalisia suojaustoimenpiteitä. IPS on verkon turvallisuuden kannalta tärkeää, koska sen avulla mahdollisiin hyökkäyksiin voidaan reagoida automaattisesti ilman henkilöstön manuaalisia toimenpiteitä, joka parantaa reagointinopeutta huomattavasti. IPS kykenee suojaamaan verkkoympäristöä hyökkäyksiltä omalla säännöstöllään, vaikka palomuuri olisikin murrettu (Pfleeger, P., Pfleeger, S. & Margulies, J, 2015).

4 OHJELMISTOPOHJAISET YRITYSVERKOT

Kappaleessa esitellään keskeiset SD-WAN toteutuksiin liittyvät teknologiat sekä niiden väliset riippuvuudet. Teknologioista esitellään yleisimmin käytetyt. Näitä ovat Open Network Foundatation (ONF) yhdistyksen kehittämät, avoimeen lähdekoodiin perustuvat teknologiat; ohjelmistopohjainen verkottaminen (SDN), European Telecommunications Standard Institute:n - ETSI:n kehittämä verkkokomponenttien virtualisointi (NFV) sekä OpenFlow. ONF:n mukaan OpenFlow-pohjaisella SDN-verkolla pyritään infrastruktuurissa olevien verkkojen optimointiin, kun taas NFV-teknologia mahdollistaa kustannussäästöjä esimerkiksi laitekustannuksissa, verkkojen skaalautuvuuden sekä huomattavasti aikaisempaa nopeamman verkkopalveluiden käyttöönoton. Lisäksi teknologiat ovat optimoitu toimimaan laajasti skaalautuvissa, pilvipalveluina tuotettavissa ympäristöissä, joita operaattorit hyödyntävät. Kuviossa 4 on esitetty esimerkki arkkitehtuurista, joissa näitä teknologioita on yhdistetty (Open Network Foundation, 2014).



KUVIO 4 Esimerkki SDN- ja NFV -arkkitehtuurista (ONF, 2014).

4.1 Ohjelmistopohjainen verkottaminen (SDN)

Aikaisemmin verkkolaitteissa liikenteen välitykseen liittyvä logiikka, kuten esimerkiksi reitityspäätökset, käyttöjärjestelmä sekä liikenteen välitys on toteutettu yksittäisellä laitteella. Tästä johtuen jokainen laite on jouduttu hallitsemaan omana kokonaisuutenaan, kun liikenteen välityksen logiikka on hajautunut laitteiden välille. Lisäksi konfigurointi tehdään tyypillisesti laitekohtaisesti komentoriviltä, jolloin se on hidasta ja virheiden mahdollisuus kasvaa. Kokonaiskuvan hahmottaminen verkon toiminnasta on haastavaa (Taha, 2014).

SDN-teknologia mahdollistaa verkkoliikenteen välitys- ja hallintatason eriyttämisen toisistaan. Verkon infrastruktuuriin kuuluvat verkkolaitteet saadaan hallittua keskitetysti, jolloin laitekohtaista konfiguraatiota ei enää tarvita. Verkkolaitteiden keskitetty hallinta mahdollistaa lisäksi niiden toimittajariippumattomuuden. Hallintatason eriyttäminen ja arkkitehtuurin kerroksien väliin muodostuvat rajapinnat (API) mahdollistavat verkon ohjelmoitavuuden. ONF (2014) kuvaa SDN arkkitehtuurin kolmella eri kerroksella:

Sovelluskerros käsittää verkkoa hyödyntävät esimerkiksi verkonhallintaan liittyvät ohjelmoitavat sovellukset, joilla voidaan ohjata hallintakerroksella olevan kontrollerin toimintaa ja sitä kautta vaikuttaa esimerkiksi verkkoliikenteen ohjaukseen ja monitorointiin. Täten sovelluksien täytyy olla yhteensopivia ainoastaan verkon hallintakerroksella toimivan SDN-kontrollerin kanssa. Sovellukset ovat abstraktoitu infrastruktuurikerroksella olevilta verkkoteknologioilta. Sovelluskerrokset ja hallintakerroksen välistä rajapintaa kutsutaan Northbound-rajapinnaksi (API). Northbound-rajapinnassa käytettäviä yleisimpiä rajapintakieliä ovat REST (Representational State Transfer), C++, Java ja Python (Taha, 2014).

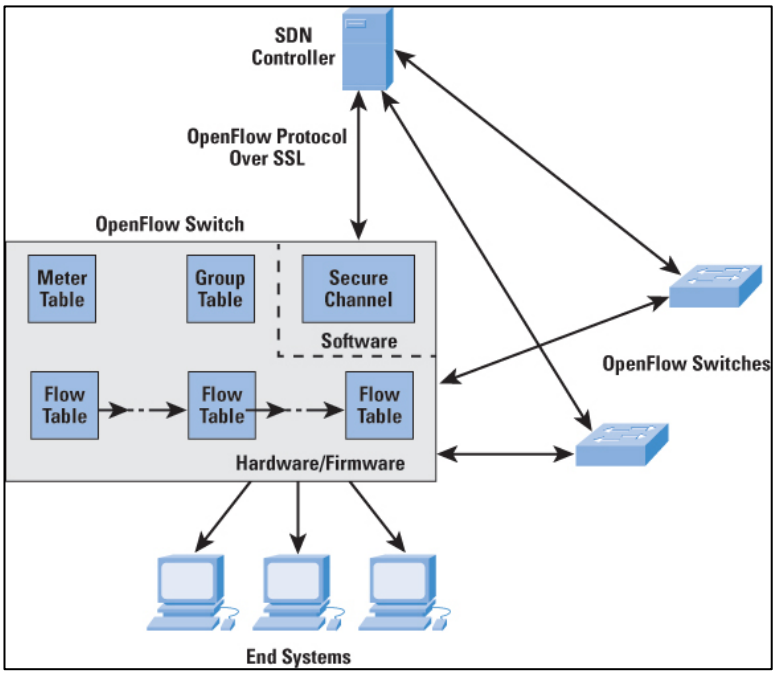
Hallintakerros käsittää keskitetyn verkon hallinnan SDN-kontrollerin avulla. Kontrolleri sisältää logiikan, joka tekee verkkoliikenteen välitykseen liittyvät päätökset. Avoimeen lähdekoodiin perustuvia kontrollereita on useita, joista tunnetuimmat ovat ONOS, OpenDayLight, Floodlight ja RYU. Kontrolleri keskustelee verkon laitteiden kanssa ja pitää yllä kokonaiskuvaa verkon topologiasta ja tilasta. Topologian muodostamiseen käytettäviä protokollia ovat muun muassa LLDP (Link Layer Discovery Protocol) sekä SNMP (Simple Network Management Protocol). Hallintakerros muuttaa sovelluskerrokselta saamansa ohjeet infrastruktuurikerrokselle välitettäviksi ohjaussäännöiksi. Hallinta- ja infrastruktuurikerroksen välistä rajapintaa kutsutaan Southbound-rajapinnaksi (API). Sen kautta kontrolleri hallitsee verkon välitykseen liittyvää käyttäytymistä. Tässä eniten käytettävät rajapintakielet ovat OpenFlow sekä OVSDB (Antikainen, 2016).

Infrastruktuurikerros käsittää kaikki liikenteen välitykseen osallistuvat verkkolaitteet, jotka osallistuvat SDN-verkon toimintaan. Näitä voi olla esimerkiksi

reitittimet, kytkimet tai palvelimet, joihin toiminnallisuus on virtualisoitu. Infrastruktuurikerroksella olevia laitteita kutsutaan SDN-teknologiassa verkkoelementeiksi (engl. NE, network element). Verkkoelementit hoitavat verkkoliikenteen ohjauksen lisäksi esimerkiksi pakettien suodatusta, laatuluokittelua sekä jonottamista (Taha, 2014).

4.1.1 OpenFlow

Eniten käytetty protokolla SDN-arkkitehtuurin southbound-rajapinnalla on ONF:n kehittämä avoin standardi OpenFlow. Sillä hallitaan kommunikaatio hallintakerroksella toimivan kontrollerin ja infrastruktuurikerroksella sijaitsevien verkkoelementtien välissä. OpenFlow kehitettiin alun perin Ethernet-kytkinten hallintaan. Sen aikaisempia toteutuksia olivat SANE sekä Ethane, jotka olivat ensimmäisiä protokollatoteutuksia, joissa hallinta ja infrastruktuurikerros saatiin erotettua toisistaan. OpenFlow toimii keskeisenä teknologiana SDN-arkkitehtuurissa. Kun SDN-kontrolleri keskustelee OpenFlow-protokollan välityksellä infrastruktuurikerroksella olevien laitteiden, kuten OVS (OpenFlow Switch) -kytkimien suuntaan standardin API-rajapinnan avulla, saadaan verkon laitteisto abstraktoitua kontrollerilta. Näin ollen verkkoinfrastruktuurissa voidaan hyödyntää useamman eri valmistajan laitteistoja yhtä aikaa. OVS-kytkimenä voivat toimia fyysiset Ethernet-kytkimet, jotka tukevat OpenFlow-protokollaa tai esimerkiksi konesalipalvelimelle tai pilvipalveluun virtualisoidut toteutukset. Kuviossa 5 esitetty arkkitehtuuri koostuu yhteensopivasta OpenFlow-yhteensopivasta kytkimestä (engl. OpenFlow Switch), kontrollerista sekä salatusta yhteyskanavasta näiden välillä (Bidaj, A., 2016).



KUVIO 5 OpenFlow-kytkimen arkkitehtuuri (Cisco Systems 2016.)

Protokolla tukee pakettien leimakytkentää ja OpenFlow:ta hyödyntävät kytkimet pystyvät tekemään MPLS-kytkentään liittyviä push-, pop ja swap -toimintoja. Näin ollen OpenFlow sopii hyvin protokollaksi hybridiverkkoihin, joissa hyödynnetään MPLS-pohjaisia yhteyksiä. Kandoin (2015) mukaan jos kaikki verkon kytkimet tukevat OpenFlow-protokollaa, kytkimet voivat toimia joko reaktiivisessa tai proaktiivisessa roolissa yhdistäessään MPLS-leimoja paketteihin. Reaktiivisessa mallissa kytkin lähettää aina ensimmäisen paketin kontrollerille. Kontrolleri lähettää tiedon leimasta ja ulosmenevästä portista kytkimelle tiettyyn vuohon liittyen. Proaktiivisessa mallissa kytkimen pyytäessä leimatietoa kontrollerilta, se laskee leimakytketyn polun kaikille verkon kytkimille samalla. Tämä säästää kytkimien ja kontrollerin välillä lähetettäviä hallintaviestejä huomattavasti. (Kandoi, R. 2015).

4.1.2 OpenFlow-viestit

SDN-verkon hallintaan käytetään kytkimen ja kontrollerien välisiä viestejä. Viestejä lähettääkseen kytkimellä täytyy olla TLS- tai TCP-yhteys kytkimelle määriteltyyn porttiin. Oletuksena käytetään TCP-porttia 6633. Hallintaliikennettä ei siis välitetä samassa OpenFlow-kanavassa kuin mitä verkon päälaitteet käyttävät liikennöintiin. Kytkimet käsittelevät kontrollerilta tulevat viestit kokonaisuudessaan ja kontrollerin näin vaatiessa vastaavat viesteihin. Mikäli viestin käsittely epäonnistuu, kytkin lähettää virheilmoituksen takaisin kontrollerille. Kontrollerit voivat yleisesti olla vastaamatta kytkimiltä saapuviin viesteihin, mutta niiden täytyy vastata ainakin Echo-viesteihin, jotta kytkimet eivät katkaise yhteyttää kontrolleriin.

Viestit voidaan luokitella symmetrisiin, asymmetrisiin sekä kontrollerilta kytkimelle välitettäviin viesteihin. Symmetrisen viestin lähettäminen voi tapahtua sekä kytkimen että kontrollerin toimesta, eikä se vaadi aiempaa pyyntö toiselta osapuolelta. Asynkronisien viestien välityksellä kytkimet ilmoittavat kontrollerille muutoksesta SDN-verkossa tai kytkimen tilassa. Kontrollerilta kytkimelle lähetettävien viestien avulla kontrolleri antaa käskyjä kytkimelle tai tiedustelee tämän tilaa verkon toimintaan liittyen. OpenFlow-version 1.3.0 viestit kuvauksineen on esitetty taulukossa 1. (Open Networking Foundation 2012).

TAULUKKO 1 OpenFlow 1.3.0 viestit

Viestityyppi	Viesti	Kuvaus
Kontrollerilta kytkimelle	Features	Kontrolleri pyytää tietoja kytkimen ominaisuuksista ja kyvykkyyksistä. Käytetään esimerkiksi OpenFlow-kanavan muodostukseen.
	Configuration	Kontrolleri muuttaa tai kysyy konfiguraatiota kytkimeltä. Kytkin vastaa ainoastaan kyselyihin.
	Modify-State	Kytkimen tilan hallinta. Käytetään vuosääntöjen muutoksiin (uusi, muutos & poisto)
	Read-State	Statistiikan tai konfiguraation kysely
	Packet Outs	Pakettien välittäminen ja niihin liittyvät toimenpiteet. Sisältää koko paketin tai puskurin tunnisteen (ID), jolla paketti kytkimellä tunnistetaan.
	Barrier	Kontrolleri varmistaa, että viestien riippuvuudet ovat oikein ja saa tarvittavat kuittaukset kytkimen suorittamista toimenpiteistä.
	Role-Request	OpenFlow-kanavan roolin asettaminen tai kysely
	Asynchronous-Configuration	Suodattimien lisäys asynkronisiin viesteihin OpenFlow-kanavassa
Asynkroninen	Packet-in	Kytkin lähettää saapuneen paketin kontrollerille kytkentäpäätöksen tekoa varten. Tämä voidaan tehdä joko lähettämällä koko paketit tai puskuroimalla osan paketista ja lähettämällä puskurintunnisteen.
	Flow-Removed	Kytkin ilmoittaa vuomerkinnän poistamisesta kontrollerille. OFPFF_SEND_FLOW_REM -lippu täytyy olla asetettuna.
	Port Status	Kytkin ilmoittaa muutoksesta portin tilassa.
	Error	Kytkin ilmoittaa virheestä
Synkroninen	Hello	Lähetetään hallintayhteyden muodostamiseksi ja ylläpitämiseksi kontrollerin ja kytkimen välillä.
	Echo	Lähetetään hallintayhteyden ylläpitämiseksi kontrollerin ja kytkimen välillä. Molemmat myös vastaavat pyyntöön *echo*-viestillä. Voidaan käyttää myös viiveen ja siirtokapasiteetin analysoinnissa.
	Experimenter	Mahdollisia tulevia lisäominaisuuksia varten. Ei aktiivisessa käytössä.

4.1.3 OpenFlow-protokollan toiminta

Verkkoliikenteen välitys OpenFlow-pohjaisessa verkkoinfrastruktuurissa perustuu vuotauluihin ja niihin määritettäviin vuosääntöihin. Näiden avulla verkkoelementeille saapuvia liikennevoita on mahdollista manipuloida. Jokai-

nen OpenFlow-kytkimen sisääntuloportti ja ulostuloportti sisältävät vähintään yhden vuotaulun (Kandoi, R. 2015).

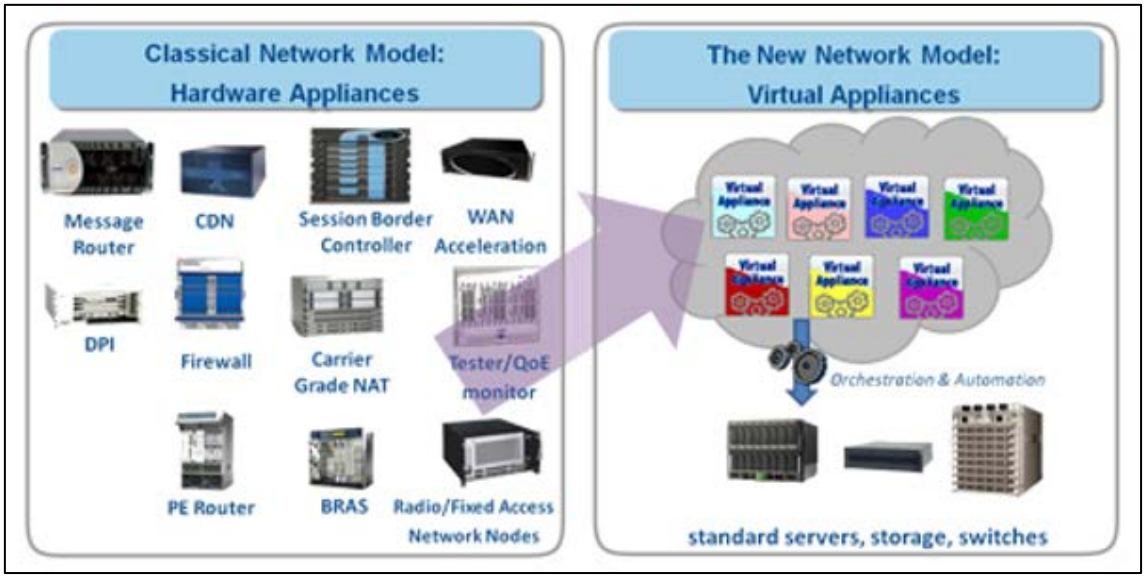
Vuosääntö koostuu seitsemästä kentästä: Osumakentät (engl. Match Fields), prioriteetti, laskurit, ohjeet, aikakatkaisu, evästeet sekä liput. Kytkin tunnistaa vuosäännöt osumakentän sekä prioriteetin avulla. Se käy vuosäännöt läpi prioriteettikentän mukaisessa järjestyksessä ja vuon sisältämiä otsikkotietoja verrataan osumakenttään sekä sisään tulevaan porttiin. Vuon ohjaus voidaan myös määrittää tehtävän toisessa vuotaulussa, jolloin aikaisempaan tauluun liittyvä metadata näytetään osumakentässä. Ohjekentällä voidaan määrittää paketille tehtäviä toimenpiteitä, kuten edelleen välittäminen tietystä kytkimen portista. Aikakatkaisun avulla voidaan määrittää vuosäännölle voimassaoloaika. Evästeitä voidaan käyttää suodattamaan vuosääntöjä, joihin on tehty ohjekentän mukaisia toimenpiteitä. Lippukentällä voidaan muuttaa vuosääntöihin liittyvää hallintatapaa ja hallinnan yhteydessä näytettäviä viestejä (Open Network Foundation 2014).

SDN-kontrolleri vastaanottaa ja lähettää paketteja verkon kytkimiltä salatun kanavan välityksellä ja manipuloi verkkoliikennettä kytkimelle lähetettävien vuosääntöjen avulla. Kytkin tekee käsketyt toimenpiteet jokaiseen yksittäiseen vuohon liittyen. Toimenpiteitä voivat olla esimerkiksi paketin välittäminen määritetystä kytkimen portista ulos, paketin lähettäminen kontrollerille tai tietoturvaan liittyvä paketin pudottaminen. Jos kytkimellä ei ole sääntöä tiedossa ennalta, se kysyy tyypillisesti tehtäviä toimenpiteitä kontrollerilta tai pudottaa paketin. Kontrolleri voi asettaa verkon kytkimille vuosääntöjä joko kytkimen sitä pyytäessä, tai proaktiivisesti etukäteen, ennen varsinaisen välityspäätöksen kohteena olevan paketin saapumista kytkimelle. Fernandezin (2013) mukaan proaktiivinen malli parantaa useissa tapauksissa kontrollerin suorituskykyä, mutta vaatii enemmän konfigurointia, ennen kuin paketteja voidaan välittää SDN-verkossa. Reaktiivisessa mallissa ei välttämättä tarvitse konfiguroida, koska IP-osoitteet ja pakettien välitys toimivat kontrollerin luodessa tarvittavat vuot automaattisesti (Bidaj, A., 2016; Fernandez, M. P. 2013).

4.2 Virtualisoidut verkkokomponentit (NFV)

Tietoverkkojen jatkuva laajentuminen sekä laitemäärän kasvu on johtanut verkojen entistä haastavampaan hallintaan. Verkkotoimintoihin dedikoidut verkkolaitteet eivät skaalaudu kehityksen asettamiin vaatimuksiin, jonka vuoksi laitteiden elinkaari on huomattavasti virtualisoituja laitteita lyhyempi.

Verkkokomponenttien virtualisoinnilla tarkoitetaan verkon resurssien ja toiminnallisuuksien, kuten reitittimien, kytkimien tai tietoturvaan liittyvien komponenttien virtualisointia. Kuviossa 6 on esitetty erot dedikoituihin laitteisiin perustuvan mallin sekä virtualisoinnin välillä. Oikealla esitetyssä mallissa verkon toiminnot ovat virtualisoitu keskitetyille konesalipalvelimille.



KUVIO 6 NFV-arkkitehtuuri verrattuna laitekohtaiseen malliin (Romero, L. 2014.)

Operaattorien näkökulmasta virtualisoinnin avulla voidaan siirtää verkkotoiminnallisuudet dedikoiduilta laitteistoilta yleisille palvelimille, joka mahdollistaa palveluiden tehokkaan skaalaamisen muuttuvien asiakastarpeiden mukaisiksi ilman ylimääräisiä laiteinvestointeja. Verkon toiminnallisuuksien toteuttaminen virtualisoituna ohjelmallisesti mahdollistaa myös sijoittelun ja siirtämisen helposti paikasta toiseen, ilman fyysisten laitteiden siirtämistä. Tästä esimerkkinä ovat toisen palvelutoimittajan konesalit tai sisällöntuottajat, joihin asiakkaiden verkot laajenevat (Ilyadis, N. 2014).

Virtualisointi mahdollistaa samoilla fyysisillä laitteistoilla ajettavien asiakkaiden verkkokokonaisuuksien eristämisen loogisesti toisistaan. Operaattoriverkoissa yleiset esimerkit ovat tietoverkkojen erottaminen VRF- (engl. Virtual routing and forwarding) sekä VLAN (engl. Virtual Local Area Network). VRF toimii OSI-mallin verkkokerroksella ja sen avulla jokaiselle asiakkaalle saadaan luotua omat reititystaulunsa ja asiakkaiden reititystoiminnot saadaan näin ollen eristettyä toisistaan. VLAN taas mahdollistaa Ethernet-verkkojen loogisen erottamisen toisistaan OSI-mallin toisella kerroksella.

SD-WAN toteutuksien kannalta virtualisoinnin avulla voidaan toteuttaa esimerkiksi verkon hallinnasta vastaava SDN-kontrolleri sekä tarvittavat tietoturvallisuuteen liittyvät komponentit, kuten palomuuri tai tunkeutumisenestojärjestelmät. Asiakkaille toimitettavan CPE-laitteen ei tarvitse enää jatkossa olla määritetyn laitevalmistajan tai mallin mukainen, vaan arkkitehtuuriin liittyvät verkkoelementit voidaan toteuttaa toimittajariippumattomasti esimerkiksi OpenFlow-rajapintaa tukevien laitteiden avulla.

Palomuurauksen näkökulmasta NFV parantaa merkittävästi kustannustehokkuutta sekä joustavuutta, kun palomuurisovellukset voidaan myös toteuttaa virtuaalisina sovelluksina fyysisien laitteiden sijaan. Uuden sukupolven palomuuriratkaisut vaativat merkittävästi laskentatehoa muun muassa OSI L7 -tason liikenteen, kuten liikenteen sisällön tunnistamiseen ja suodattamiseen.

Palvelimilla toteutettavien tietoturvapalveluiden resursseja saadaan huomattavasti helpommin skaalattua tarpeiden mukaisiksi verrattuna fyysisiin palomuurilaitteisiin (Lorenz, C., Hock, D., Scherer, J., Durner, R., Kellerer, W., Gebert, S. & Tran-Gia, P. 2017).

4.3 SD-WAN (Software-defined Wide Area Networking)

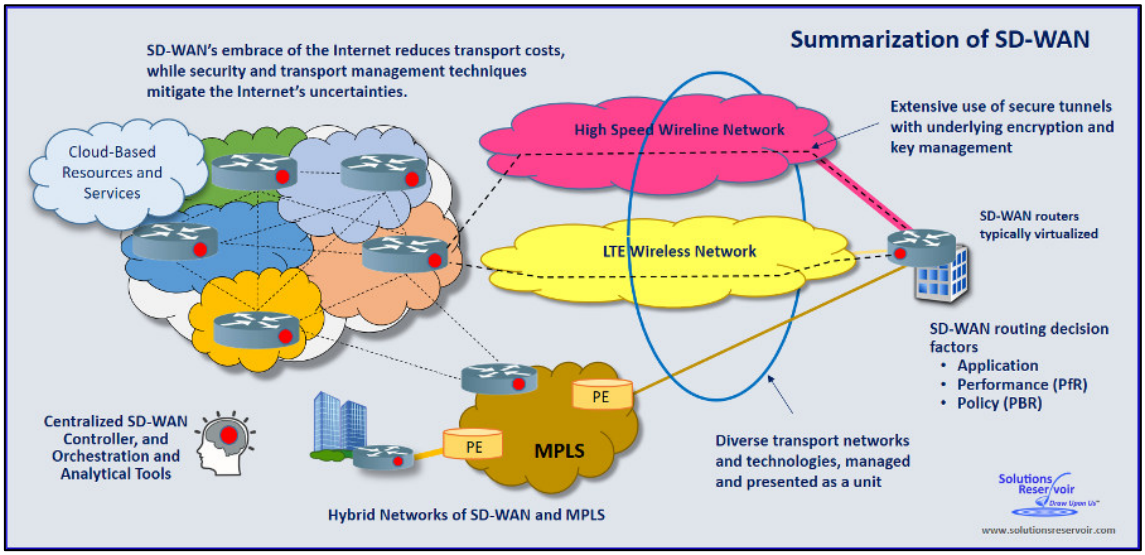
SD-WAN on laitteistosta, sovelluksista ja palveluista koostuva ekosysteemi, jolla voidaan tuottaa yritysverkottamiseen tarvittavat kyvykkyydet. Tarvittavia kyvykkyyksiä ovat muun muassa verkon suorituskyky, luotettavuus ja tietoturvallisuus. SD-WAN perustuu pääasiassa taustalla oleviin SDN- ja NFV-teknologioihin, jotka mahdollistavat laitekohtaisen hallintan ja välityskerroksen erottamisen toisistaan. Ohjelmisto-ohjauksen avulla yritysverkkojen automaatiotasoa saadaan nostettua sekä verkkoliikenteen ohjauksen että hallinnan osalta. Lisäksi hyödynnetään aikaisemmissakin verkottamisratkaisuissa käytettäviä teknologioita, kuten IPSec-tunnelointia sekä overlay-arkkitehtuuria, jonka avulla underlay-kerroksella olevat verkkoteknologiat saadaan yhdistettyä toisiinsa (Rajendran, A. 2016).

Verkon hallinta saadaan keskitettyä, joka mahdollistaa paremman verkon tilannekuvan lisäksi muun muassa dynaamisen politiikoihin perustuvan reitityksen. Jokaista päätelaitetta ei tarvitse enää konfiguroida erikseen, vaan hallinnointi suoritetaan ohjelmisto-ohjatusti keskitetyn, useimmiten pilvipalveluna toteutetun SDN-kontrollerin avulla. Yritysten toimipisteille toimitettavia laitteita kutsutaan SD-WAN termistössä reunareitittimiksi (engl. Edge Router). Ne hakevat asetuksensa automaattisesti kontrollerilta Internet-yhteyden kytkemisen yhteydessä, jonka vuoksi käyttöönotto ei välttämättä vaadi yrityksiltä välttämättä aiheeseen liittyvää osaamista tai operaattorilta asentajan lähettämistä yrityksen toimipisteelle. Aikaisemmin MPLS-VPN -pohjaisissa toteutuksissa vaadittiin usein operaattorien manuaalisia konfiguraatiotoimenpiteitä sekä yrityksien toimipisteillä paikan päällä että hallintaverkon yli tehtävinä etätoimenpiteinä (Rajendran, A. 2016).

Kontrollerin hallintaliikenne on eriytettynä verkossa reititettävästä asiakasliikenteestä. Ohjelmisto-ohjatun mallin mahdollistaa overlay-arkkitehtuuri, jossa olemassa olevien verkkoteknologioiden päälle rakennetaan looginen verkkokerros, joka on salattu IPSec-tunneleiden avulla. SD-WAN verkon reunalaitteet kapseloivat ja salaavat verkkoliikenteen, joka välitetään siirtotien, kuten esimerkiksi Internetin yli. Yleisimpiä kapsulointiin käytettäviä protokollia ovat VXLAN (Virtual Extensible Local Area Network) sekä GRE (Generic Routing Encapsulation).

Arkkitehtuurissa voidaan hyödyntää verkkoliikenteen kuorman jakamista useammalle WAN-linkille samanaikaisesti, kun aikaisempi MPLS VPN -teknologia tuki ainoastaan yhden linkin käyttämistä kerrallaan. Yritysten käyttämien sovellusten verkkoliikennettä voidaan priorisoida ja ohjata eri siirtoteille sovelluksen liiketoiminta- tai viivekriittisyyden perusteella. Esimerkiksi video-

kuvaa käyttävät sovellukset käyttävät MPLS-yhteyttä, jossa laatuluokittelu voidaan taata, kun taas web-selaus käyttää taustalla olevaa, toisen linkin 4G-tekniikkaa siirtotienä. Kuviossa 7 esitetty SD-WAN -arkkitehtuuri mahdollistaa usean eri verkkoteknologin (esim. MPLS, LTE/4G, Laajakaista) eli hybridiverkkojen sekä Internetin hyödyntämisen siirtotienä, josta seuraa tehokkuuden ja kapasiteetin käyttöasteen parantuminen (Solutions Reservoir, 2016).



KUVIO 7 SD-WAN verkkoarkkitehtuuri (Solutions Reservoir, 2016.)

Kun Internetiä voidaan hyödyntää siirtotienä, WAN-yhteyksien provisiointi on aikaisempaa nopeampaa esimerkiksi uusien yhteyksien käyttöönottoihin tai olemassa olevien yhteyksien muutoksiin liittyen. Tämän avulla operaattorit pystyvät vastaamaan huomattavasti paremmin asiakkaidensa kiihtyvään liiketoiminnan muutosnopeuteen.

Hybridiverkkojen sekä overlay-teknologian hyödyntäminen tuo verkkolaitteisiin liittyvän toimittajariippumattomuuden, kun verkon hallinta nostetaan yksittäisiltä laitteilta ohjelmistotasolle. Päätelaitteet eivät välttämättä tarvitse olla dedikoituja reititinlaitteita, vaan ne voidaan virtualisoida palvelimille, jolloin esimerkiksi julkisia pilvipalveluita voidaan tuoda osaksi yritysverkkoa (Rajendran, A. 2016).

4.3.1 VXLAN

VXLAN on aikaisemmin konesaliympäristöissä käytetty protokolla. Se on laajennus aikaisemmasta VLAN-teknologiasta, jonka porttimäärään liittyvä osoiteavaruus rajoittui 4096 osoitteeseen, joka oli riittämätön sovellusten ja asiakasverkkojen erottamiseen toisistaan. VXLAN kasvattaa osoiteavaruutta 24-bittisen VXLAN-kentän avulla 16 miljoonaan osoitteeseen ja jokainen looginen verkko tunnistetaan ja erotetaan toisistaan VXLAN Segment ID:n avulla. Näin ollen operaattorit voivat hyödyntää VXLAN-pohjaisia toteutuksia WAN-verkoissa ja

saavat erotettua asiakkaansa verkkokokonaisuudet sekä niihin liittyvät sovellukset loogisesti toisistaan (Vajaranta, M., Kannisto, J. & Harju, J. 2016).

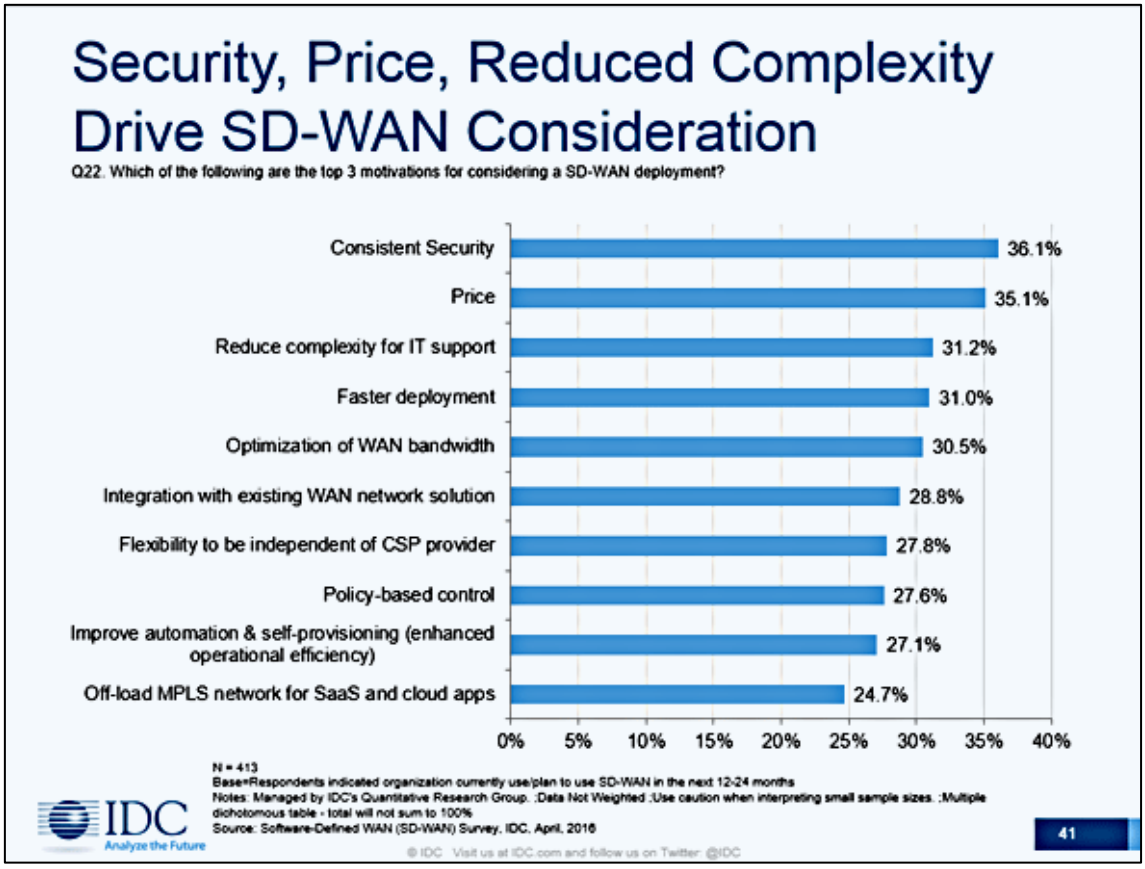
Protokollan avulla OSI-mallin 2-3 kerroksella välitettävä verkkoliikenne voidaan kapseloida L4 UDP-kehyksiksi ja näin ollen muodostaa SD-WAN teknologiassa käytetty overlay-verkkoarkkitehtuuri. Protokolla sisältää multicast-hallintakerroksen, jonka avulla verkossa esiintyvät tuntemattomat unicast paketit saadaan välitettyä kaikille verkon kytkimille. Kytkimet käyttävät omaa kytkentätauluaan yhdistääkseen unicast-liikenteen IP- ja MAC (Media Access Control) -osoitteet toisiinsa, jota voidaan hyödyntää unicast-tyyppisen liikenteen välityksessä. VXLAN muodostaa tunneleita, jotka terminoidaan IP-osoitteisiin, joita kutsutaan VTEP:ksi (Virtual Tunnel End-Point). IP-osoitteena voi olla minkä tahansa rajapinnan osoite laitteella, kunhan se on toisen VTEP-päätepisteen saavutettavissa IP-verkon yli. (Vajaranta, M., Kannisto, J. & Harju, J. 2016).

VXLAN itsessään ei sisällä mitään tietoturvaan tai salaukseen liittyviä ominaisuuksia, joten sitä ei sellaisenaan voida hyödyntää WAN-verkon muodostamiseen julkisen Internetin yli. Salaukseen käytettäviä protokollia VXLAN:n yhteydessä ovat muun muassa IPSec sekä DTLS (Datagram Transport Layer Security). Tällöin toteutus kuitenkin nojaa vahvasti salaustunneleihin ja tunnelin vikaantuessa verkkoliikenne välitetään salaamattomana (Vajaranta, M., Kannisto, J. & Harju, J. 2016).

4.3.2 Siirtyminen ohjelmistopohjaisiin yritysverkkoihin

Tutkimusyhtiö Gartnerin mukaan (2015) SD-WAN -arkkitehtuurin avulla pyritään alentamaan verkottamiseen liittyviä investointikustannuksia (Capex) sekä yksinkertaistamaan palveluiden tuottamismallia hyödyntämällä Internetiä siirtotienä aikaisempien MPLS-VPN -yhteyksien sijaan. Olemassa olevia MPLS-VPN -yhteyksiä voidaan edelleen hyödyntää SD-WAN -teknologian rinnalla, joten käyttöönotto on mahdollisesta tehdä useammassa vaiheessa. Käyttöönotto tulee kuitenkin toteuttaa niin, että uudet verkottamispalvelut tulevat osaksi organisaation olemassa olevaa WAN-kokonaisarkkitehtuuria. Tämän tulisi kattaa sekä pilvipalvelut että yrityksen konesalista ”on-site” -mallina toteutettavat IT-palvelut. Tutkimusyhtiö arvioi, että 30% yrityksien sivutoimipisteiden WAN-yhteyksistä tullaan toteuttamaan SD-WAN:n avulla vuoden 2019 loppuun mennessä. Tästä kehityksestä johtuen voidaan olettaa, että myös SD-WAN -palveluiden kehitys tulee lähivuosina olemaan nopeaa (Gartner, 2016).

Kuviossa 8 on esitetty keskeisimmän SD-WAN -palveluihin siirtymiseen ajavat seikat tutkimusyhtiö IDC:n mukaan (2016). Merkittävimpinä näistä ovat WAN-verkon yhtenäinen tietoturva, kustannustehokkuus sekä IT-palveluiden tuottamisen ja tuen kompleksisuuden vähentäminen.



KUVIO 8 Keskeisimmät ajurit SD-WAN -palveluiden käyttöönottoon (IDC 2016).

4.3.3 SD-WAN -palveluiden tietoturva

Yritysverkkojen ohjelmoitavuus tuo uusia kyvykkyyksiä verkon tietoturvan hallintaan, mutta samaan aikaan se tuo myös uusia haavoittuvuuksia, joihin operaattoreiden ja yritysten täytyy pystyä reagoimaan. MPLS-pohjaisia WAN-yhteyksiä on pidetty luotettavina ja niihin on Rajendranin (2016) mukaan kohdistunut vähän hyökkäyksiä. Koska SD-WAN -palvelut käyttävät julkista Internetiä siirtotienä ja keskitettyä SDN-kontrolleria koko verkon hallintaan, tietoturvan merkitys korostuu.

Taustalla oleva SDN-teknologia sisältää useita tunnettuja hyökkäysvektoreita, jotka jakautuvat jokaiselle arkkitehtuurin kerrokselle. Zhyuan, H., Xueqiang, Y., Emile, Y. & Zhigang, L. (2015) mukaan osa haavoittuvuuksista ovat samoja, kuin aikaisemmissa verkkoteknologioissa, mutta niiden vaikuttavuuteen ja esiintyvyyteen liittyvä riskitaso on muuttunut. Kun verkon hallinta on keskitetty kontrollerille, mahdollinen kontrolleriin kohdistuva hyökkäys on vaikutukseltaan huomattavasti yksittäistä reititintä suurempi. Kandoin (2015) mukaan kontrollerin on syytä olla sekä fyysisesti että loogisesti kahdennettu, jotta Single point of Failure -pisteet vältettäisiin. Koska kontrolleri on arkkitehtuurin haavoittuvin komponentti, suurin osa SDN-teknologiaan liittyvistä suojaustoimenpiteistä kohdistuu siihen. Tunnistettuja kontrollereihin kohdistuvia hyökkäyksiä ovat esimerkiksi palvelunestohyökkäykset, joilla pyritään lamaut-

tamaan verkon toiminta sekä kontrollerin väärentäminen, jonka avulla hyökkääjä pyrkii saamaan verkon hallintaansa. Rengarajun ja Lungin tekemässä tutkimuksessa (2017) on esitetty, että hajautetulla palomuuriratkaisulla SDN-verkossa saadaan toteutettua huomattavasti keskitettyä ratkaisua tehokkaampi palomuuraus. (Zhyuan, H., Xueqiang, Y., Emile, Y. & Zhigang, L. 2015; Rengaraju, P., V, R. R., & Lung, C. 2017.).

Arkkitehtuurin sovelluskerrokselle kohdistuvat hyökkäyksillä pyritään vaikuttamaan verkon suoraan verkon toimintaan ja siihen liittyviin poliitikoihin. Kontrolleri muuntaa sovelluksilta saamaansa politiikat vuomerkinnöiksi. Sovellukset kommunikoivat suoraan kontrollerin kanssa, ja Southbound-rajapinnassa toimiva OpenFlow-protokolla ei pysty tunnistamaan, onko vuomerkintä tullut sovellukselta vai aikaisemmasta vuomerkinnästä. Näin ollen on mahdollista, että hyökkääjän käyttämä sovellus ylikirjoittaa esimerkiksi tietoturvaan liittyviä poliitikoita (Zhyuan, H., Xueqiang, Y., Emile, Y. & Zhigang, L.,2015).

Infrastruktuurikerroksella sijaitsevaan, yksittäiseen reitittimeen kohdistuvalla hyökkäyksellä, voidaan vaikuttaa ainoastaan verkkoliikenteeseen, joka välitetään tämän yksittäisen laitteen kautta. Operaattoreiden asiakkaiden toimipisteisiin asennettavat reititinlaitteet hakevat asetuksensa verkosta automaattisen bootstrap-prosessin avulla. Tämän ansiosta laitteiden kytkemiseen verkkoon ei tarvitse välttämättä IT-alan asiantuntemusta. Toisaalta tämä asettaa asiakaslaitteet aikaisempaa alttiimmaksi esimerkiksi fyysisille hyökkäyksille, kun laitteen asentaja ei välttämättä osaa ottaa huomioon laitteen vaatimaa tietoturvaa (Rajendran, A. 2016).

Koska overlay-verkko muodostetaan Internetin yli, perinteiset, fyysiset palomuuriratkaisut eivät enää ole liikenteen välityksen kannalta optimaalisin toteutustapa. Verkkoliikenne pitää pystyä välittämään suoraan yrityksen toimipisteiltä Internetiin, reitittämättä sitä keskitetyn palomuurin kautta muun muassa viiveiden, viiveen vaihtelun (engl. Jitter) ja pakettihävikin välttämiseksi. Mainitut ongelmat johtuvat usein pitkistä maantieteellisistä etäisyyksistä liikenteen lähtöpisteestä, kuten yrityksen toimipisteestä keskitettyyn palomuurausta tekevään laitteeseen (Rajendran, A. 2016).

5 KOKEELLINEN TUTKIMUS

Tässä kappaleessa kuvataan kokeellisessa tutkimuksessa käytetty ympäristö sekä siihen liittyvät toteutukset. Toteutuksia tehtiin yhteensä viisi, jotta saatiin tarpeeksi kattava kuva eri komponenttien toimivuudesta ja yhteensopivuudesta SDN-verkon palomuurauksessa. Tutkielman yhteydessä mallinnettu verkkotopologia on esitetty liitteessä 2.

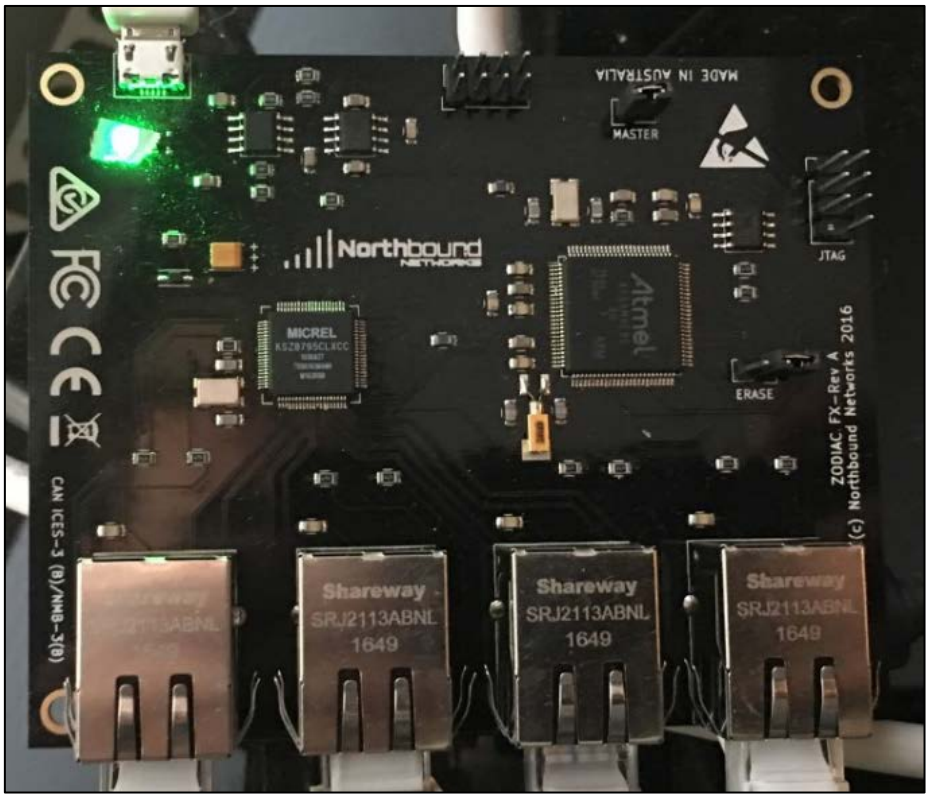
5.1 Tutkimusympäristö

Kokeellista tutkimusta tehtiin useissa eri ympäristöissä. Tarkoitus oli todentaa, että kuinka erilaiset SDN-kontrollerit, OpenFlow-kytkimet sekä sovelluskerroksella toimivat applikaatiot pystyvät toteuttamaan palomuuraukseen liittyvät tekniset tietoturvakontrollit. Näillä eri komponenteilla tehdyistä testauksista saatuja tutkimustuloksia oli tarkoitus verrata puhtaasti OpenFlow:lla ja sen suorilla komennoilla luotaviin palomuurisääntöihin. Palomuurisäännöt mallinnettiin kuvaamaan yrityksien toimialueella (engl. domain) yleisesti käytettyjä palomuuraukseen liittyviä tietoturvakontrolleja. Tutkimuksessa käytettiin oletusta, että toimialueella on käytössä Windows-työasemaympäristö sekä siihen liittyvät ylläpidossa käytettävät palvelimet. Toteutuksissa käytetyillä komponenteilla voidaan emuloida tuotantoympäristöjä hyvin pitkälle, mutta huomioitavaa on, ettei testiympäristössä tehty toteutus vastaa täysin reaalimaailman ympäristöjä, joissa operaattorit toimivat. Esimerkiksi maantieteellisistä etäisyyksistä johtuvia viiveitä tai jitteriä sekä mahdollisia odottamattomia virhetilanteita on testiolosuhteissa haastava mallintaa. Alcornin ja Meltonin (2017) mukaan myös mahdolliset virtualisoinnista johtuvat ongelmat voivat muuttaa SDN-verkon käyttäytymistä oikeaan tuotantoympäristöön verrattuna (Alcorn, J., & Melton, S. 2017).

5.2 Käytetyt laitteistot ja ohjelmistot

VirtualBox on Oraclen kehittämä virtualisointialusta. Tässä tutkimuksessa sen avulla virtualisoitiin toteutuksissa käytetyt SDN-kontrollerit sekä Mininet-verkkotopologia. VirtualBoxin avulla isäntäkoneen verkkorajapinta voidaan sillata virtuaalikoneille, jolloin ne pääsevät samaan tapaan liikennöimään Internettiin. Virtuaaliympäristölle voidaan luoda myös oma virtuaalinen verkkonsa Internal Network sekä NAT (Network Address Translation) -toiminnallisuuksien avulla.

Zodiac FX on Northbound Networksin kehittämä minimalistinen OpenFlow-kytkin, jonka avulla SDN-verkon toimivuutta voidaan testata helposti oikean laitteiston avulla. Kytkin on kuitenkin tarkoitettu nimenomaan SDN-palveluiden testikäyttöön, eikä sitä voida hyödyntää tuotantoympäristöissä. Käytännössä kytkin on rakennettu yhdelle piirilevylle ja se sisältää neljä fyysistä verkkorajapintaa: Kolme OpenFlow-käyttöön sekä yksi tavallista ethernet-liikennettä varten. Kuviossa 9 on esitetty käynnissä oleva kytkin. Vasemmassa yläkulmassa näkyy hallintaan ja virransyöttöön käytettävä Micro-USB -liitin ja alareunassa neljä RJ-45 -liitäntää OpenFlow- ja Ethernet-liikennettä varten. Kytkin tukee OpenFlow:n versioita 1.0 sekä 1.3. Zodiac FX sisältää sekä komentoriviltä operoitavan käyttöjärjestelmän että web-käyttöliittymän laitteen hallintaa varten (Northbound Networks 2017).



KUVIO 9 Zodiac FX OpenFlow -kytkin

Puikkari on Jyväskylän ammattikorkeakoulun CyberTrust-hankkeessa kehitetty SDN-alusta, joka pohjautuu avoimen lähdekoodin RYU-kontrolleriin. Puikkari sisältää varsinaisen SDN-kontrollerin lisäksi graafisen web-käyttöliittymän (GUI, engl. Graphical User Interface), jossa SDN-verkkoon liittyvää topologiaa sekä siihen kuuluvia komponentteja voidaan monitoroida ja hallita operaattorin näkökulmasta. Lisäksi pääkäyttäjät voivat luoda palveluun toimialueita, joilla voidaan mallintaa operaattorin asiakasverkkoja. Käyttöliittymä mahdollistaa verkon ylläpitäjille uusien palveluiden, kuten esimerkiksi virtuaalisen palomuurin implementoimisen luoduille asiakastoimialueille. Puikkari tukee OpenFlow:n versiota 1.3 (CyberTrust 2017).

ONOS (Open Network Operating System) on avoimeen lähdekoodiin perustuva SDN-kontrolleri, joka tarjoaa kattavat ominaisuudet SDN-verkon kontrollikerroksen hallintaan. Kontrollerin toimintaa voidaan hallita sekä oman komentorivin että graafisen käyttöliittymän kautta. ONOS-järjestelmä sisältää useita sovelluskerroksella toimivia applikaatioita SDN-verkon hallintaa varten valmiiksi asennettuna. Operaattoreiden näkökulmasta ONOS tukee usean asiakasverkon hallintaa (engl. multi-tenant) ja sen avulla asiakkaiden ympäristöt sekä niihin liittyvät asiakaskohtaiset politiikat ja konfiguraatiot voidaan eristää toisistaan. ONOS valittiin toteutukseen, koska se tukee korkean käytettävyyden implementointeja hyvän skaalautuvuuden sekä kontrollerin hajauttamismahdollisuuden ansiosta (ONOS Project 2017; Han, Y., Li, J., Hoang, D., Yoo, J. H., & Hong, J. W. K. 2017).

Aruba VAN SDN Controller on Hewlett Packardin Aruba-tuotelinjassa kehittämä kaupallinen SDN-kontrolleri. Se sisältää kattavan graafisen käyttöliittymän SDN-verkon ylläpitoon. Kontrolleri tukee OpenFlow:n versioita 1.0 ja 1.3 sekä SNMP- ja NetConf -protokollia, joita voidaan hyödyntää esimerkiksi verkon monitoroinnissa. Graafisen käyttöliittymän kautta saadaan asennettua ja käytettyä myös SDN-applikaatioita. Kontrolleri tukee sekä Hewlett Packard Enterprisen että kolmansien osapuolien ja kumppaneiden kehittämiä SDN-applikaatioita. Applikaatiot ovat ladattavissa HP:n tarjoaman sovelluskaupan kautta (Hewlett Packard 2017).

Mininet on Python-ohjelmointikieleen perustuva SDN-verkon emulointityökalu. Sen avulla voidaan helposti toteuttaa virtualisoitu SDN-verkkoympäristö. Mininet tukee oletuksena useita erilaisia verkkotopologioita, kuten lineaarinen- tai puutopologia. Lisäksi python-skriptien avulla voidaan toteuttaa kustomoituja topologioita käyttötarkoitukseen paremmin sopivaksi. Mininet luo automaattisesti topologioissa käytetyt OVS-kytkimet, päätelaitteet sekä linkit näiden välillä. Paketin välitys virtualisoidussa verkossa tapahtuu samalla semantiikalla, kuin käytetään fyysisissä OpenFlow:ta tukevissa kytkimissä. Verkon kontrolleri luodaan oletuksena Mininetin paikalliseen simulointiympäristöön, mutta myös ulkoisen kontrollerin käyttö on mahdollista. Verkon kytkinten ja päätelaitteiden

MAC- ja IP (Internet Protocol) -osoitteet voidaan asettaa haluttuun käyttötarkoitukseen sopivaksi (De Oliveira, R. L. S., Schweitzer, C. M., Shinoda, A. A., & Prete, L. R. 2014).

Floodlight on avoimeen lähdekoodiin perustuva SDN-kontrolleri, joka pohjautuu JAVA-ohjelmointikieleen. Sen arkkitehtuuri koostuu kolmesta eri komponentista: Itse kontrollerista, JAVA-moduuleihin perustuvista sovelluskerroksen applikaatioista sekä REST API -rajapinnan yli rakennetuista applikaatioista. Käytetty versio 1.2 sisältää vakio palomuuritoiminnallisuuden, jota toteutuksessa on tarkoitus hyödyntää (Zope, N., Pawar, S., & Saquib, Z. 2016).

Zodiac FX -kytkintä ja päätelaitteita lukuun ottamatta kaikki verkon komponentit toteutettiin virtualisoituna VirtualBoxin avulla. Toteutuksen verkkoympäristönä toimi NAT:n takana oleva 192.168.1.0/24 IP-verkko. Toteutus olisi voitu määrittää VirtualBox:sta myös oman NAT:n taakse, mutta silloin kaikki hallintayhteydet olisi jouduttu luomaan porttiohjauksen välityksellä.

Toteutusympäristössä käytetyt sovellus-, ohjelmisto- ja järjestelmäversiot on kuvattu taulukossa 2.

TAULUKKO 2 Toteutuksissa käytetyt komponentit ja versiot

Komponentin nimi	**Versio**
Isäntäkone	Windows 10
Linux -virtuaalikoneet	Ubuntu 14.04 LTS
OpenFlow	1.3.0 (0x04)
Zodiac FX	v0.82
Mininet	2.2.1
Puikkari	VM 0.17
Aruba VAN SDN Controller	2.8.7.0336 Trial
Flow Maker	1.1.1
ONOS	1.11.2001
Floodlight	v1.2

5.3 OpenFlow-protokollan vuosäännöt

OpenFlow-säännöillä ohjataan OVS-kytkinten toimintaa. Niillä voidaan suodattaa tehokkaasti verkkoliikennettä ja tehdä palomuurausta mahdollisimman lähellä asiakasympäristön päätelaitteita. Kun palomuuraus tehdään jo päätelaitetta lähimmillä kytkimillä, ylimääräisen liikenteen välitys yrityksen sisäverkossa saadaan minimoitua ja mahdollinen haittaliikenne saadaan torjuttua ennen sen päätymistä muihin verkon laitteisiin.

Kytkimet, jotka tukevat ainoastaan OpenFlow-protokollaa liikenteen välitykseen, eivät pysty tekemään paikallisesti kytkentään liittyviä päätöksiä, vaan ne täytyy tulla SDN-kontrollerilta. OpenFlow-hybrid -kytkimet sen sijaan sisältää älyä ja kykenee esimerkiksi tekemään kytkentäpäätöksiä, OSI L3 -reititystä sekä palomuuraukseen liittyviä toimenpiteitä myös paikallisesti ilman kontrolleria. Lisäksi niiden avulla voidaan toteuttaa operaattoriverkoissa paljon käytettyjä toimintoja, kuten VLAN-määrittelyjä tai palvelunlaatuun (QoS) liittyviä määrityksiä (Feng, T., & Bi, J. 2015).

OpenFlow-protokolla tukee sekä reaktiivista että proaktiivista tapaa palomuurauksen toteuttamiseen. Reaktiivisessa mallissa ensimmäinen kytkimen vastaanottama paketti välitetään kontrollerille *Packet In* -viestillä. Tämän jälkeen kontrolleri lisää yhden tai useamman kytkimen vuotauluun tähän liittyvän vuosäännön *Flow-mod* -viestin avulla ja tämän jälkeen kytkin käsittelee paketin luodun säännön mukaisesti. Reaktiivinen malli säästää SDN-verkon kytkinten vuotauluihin liittyvää muistia, koska reaktiivisesti luodut säännöt säilytetään kytkimellä ainoastaan rajatun ajan. Reaktiivisen mallin heikkoutena on SDN-verkon riippuvuus kontrollerista. Kontrollerin vikaantuessa kytkimet eivät pysty itsenäisesti välittämään liikennettä ja SDN-verkon toiminta lamaantuu. Näin ollen kontrollerin tulisi aina olla hajautettu useammalle palvelimelle reaktiivisessa mallissa, jolloin vältetään Single point of failure -pisteet. Lisäksi pakettien välittäminen kontrollerille luo viiveitä tiedonsiirtoon.

Proaktiivisessa mallissa SDN-kontrolleri asettaa säännöt proaktiivisesti verkon kytkimille, ennen kuin paketit saapuvat niille käsiteltäviksi. Mallin etuna on nopeampi tiedonsiirto, kun kytkin kykenee välittämään paketit itsenäisesti ilman kontrollerin ohjeita. Pakettien välitys onnistuu kytkimiltä, vaikka yhteys kontrolleriin häviäisi.

Malleja voidaan tarvittaessa yhdistää. Kaur ja Kumar yhdistivät tutkimuksessaan (2015) toiminnallisuudet niin, että palomuuraukseen liittyvät vuosäännöt luotiin proaktiivisesti kontrollerilla ja varsinaiseen paketin välitykseen liittyvät säännöt luotiin reaktiivisesti kontrollerin toimesta. Näin ollen vuotaulujen muistia saatiin säästettyä ja palomuurauksella liittyvillä säännöillä saatiin rajoitettua ylimääräistä liikennettä verkkoon jo päätelaitteita lähimpänä olevilta kytkimiltä saakka. (Kaur, K., Singh, J., Kumar, K., & Ghumman, N. S. 2015).

Vuosäännöt perustuivat OpenFlow-standardin mukaiseen osumakenttien rakenteeseen ja siihen liittyviin argumentteihin, jotka OVS-kytkin tarkistaa paketin saapuessa kytkimelle. Tarkistaminen aloitetaan vuotaulusta 0. Osumakenttien tarkistamisen jälkeen vuosäännön prioriteetin ja tekee sen jälkeen paketille vuosäännön määrittämät toimenpiteet (engl. actions).

Säännöt saadaan luotua kytkimen komentoriviltä *ovs-ofctl add-flow* -komennon ja sille annettavien parametrien avulla. Alla olevassa esimerkissä:

ovs-ofctl add-flow s3 "table=2, priority=1, in_port=3,dl_type=0x0800,nw_proto=6,nw_proto=6,tcp_dst=80,actions=resubmit(,3)

- *s3* tarkoittaa, että vuo lisätään kytkimelle s3.

- *table=2* määrittää, että vuo lisätään vuotauluun numero 2.
- *Priority* määrittää vuon prioriteetin.
- *in_port* määrittää kytkimen portin, josta paketti tulee sisään (ingress)
- *dl_type* määrittää, että kyseessä on IPv4 -liikenne
- *nw_proto=6* määrittää, että kyseessä on TCP-protokolla
- *tcp_dsp=80* määrittää, että kohdeportti on 80
- *actions=resubmit(,3)* määrittää, että paketti siirretään vuotauluun 3 käsiteltäväksi

OVS-kytkimellä olevat aktiiviset säännöt saadaan listattua *ovs-ofctl dump-flows* -komennon ja siihen määritettävien parametrien avulla. Alla olevassa esimerkissä on esitetty komennon tuloste ja yksi siihen liittyvä kytkimen s3 vuo, johon liittyvät parametrit on eritelty tulosteen alla olevaan listaan. OpenFlow version 1.3 tukemat, tutkimuksen kannalta oleelliset osumakentät sekä niiden riippuvuudet on esitelty tarkemmin taulukossa 3.

sudo ovs-ofctl dump-flows s3

NXST_FLOW reply (xid=0x4):

cookie=0x89ad361b4dabf193, duration=118.506s, table=0, n_packets=58, n_bytes=4582, idle_age=1, priority=101,in_port=1,dl_type=0x88cc actions=CONTROLLER:65509

- *cookie=0x89ad361b4dabf193* esittää vuon evästeen. Tässä tapauksessa kontrolleri on luonut *0x89ad361b4dabf193* luvun automaattisesti. Kun vuo luodaan manuaalisesti *add-flow* -komennon avulla, evästeeksi tulee 0x0.
- *duration=118.506s* esittää keston sekunteina, kuinka kauan sääntö on ollut voimassa.
- *table=0* esittää vuon koskevan vuotaulua 1.
- *n_packets=58* kuvaa pakettien määrää, jotka ovat osuneet kyseiseen vuohon.
- *n_bytes=4582* esittää liikenteen määrän kilotavuina, joka on osunut kyseiseen vuohon.
- *idle_age=1* esittää sekunteina ajan, kuinka kauan vuo on ollut käyttämättömänä
- *priority=101* esittää vuon prioriteetin (101).
- *in_port=2,* esittää kytkimen sisääntulo-portin, jota vuo koskee
- *dl_type=0x88cc* esittää, että kyseessä on LLDP-liikenne
- *actions=CONTROLLER:65509* esittää toimenpiteen, jossa liikenne ohjataan kontrollerille Packet In -viestillä.

*(*Pfarr, B., Pettit, J. & Tourrilhes, J., 1990*)*

TAULUKKO 3 Openflow 1.3.0 osumakentät ja riippuvuudet

Argumentti	Riippuvuus	Arvo	Kuvaus
in_port	-	Integer 32bit	Switch input port
in_phy_port	-	Integer 32bit	Switch physical input port
metadata	-	Integer 64bit	Metadata passed between tables
eth_dst	-	MAC address	Ethernet destination address
eth_src	-	MAC address	Ethernet source address
eth_type	-	Integer 16bit	Ethernet frame type
vlan_vid	-	Integer 16bit	VLAN id
vlan_pcp	-	Integer 8bit	VLAN priority
ip_dscp	nw_type = 0x0800	Integer 8bit	IP DSCP (6 bits in ToS field)
ip_ecn	nw_type = 0x0800	Integer 8bit	IP ECN (2 bits in ToS field)
ip_proto	nw_type = 0x0800	Integer 8bit	IP protocol
ipv4_src	nw_type = 0x0800	IPv4 address	IPv4 source address
ipv4_dst	nw_type = 0x0800	IPv4 address	IPv4 destination address
tcp_src	nw_proto = 6	Integer 16bit	TCP source port
tcp_dst	nw_proto = 6	Integer 16bit	TCP destination port
udp_src	nw_proto = 17	Integer 16bit	UDP source port
udp_dst	nw_proto = 17	Integer 16bit	UDP destination port
sctp_src	nw_proto = 132	Integer 16bit	SCTP source port
sctp_dst	nw_proto = 132	Integer 16bit	SCTP destination port
icmpv4_type	nw_proto = 1	Integer 8bit	ICMP type
icmpv4_code	nw_proto = 1	Integer 8bit	ICMP code
arp_op	dl_type = 0x0806	Integer 16bit	ARP opcode
arp_spa	dl_type = 0x0806	IPv4 address	ARP source IPv4 address
arp_tpa	dl_type = 0x0806	IPv4 address	ARP target IPv4 address
arp_sha	dl_type = 0x0806	MAC address	ARP source hardware address
arp_tha	dl_type = 0x0806	MAC address	ARP target hardware address
mpls_label	0x8847	Integer 32bit	MPLS label
mpls_tc	0x8847	Integer 8bit	MPLS TC
mpls_bos	0x8848	Integer 8bit	MPLS BoS bit
tcp_flags	nw_proto = 6	Integer 16bit	TCP flags (EXT-109 ONF Extension)

5.4 Palomuurisäännöstön luonti

Vuosäännöstöön perustuvan palomuurisäännöstön luominen aloitettiin kartoittamalla päätelaitteiden tarvitsemat palvelut Windows-ympäristössä. Säännöt tehtiin olettamasta, että verkon ulkolaidalla on erillinen palomuuri, joka suodattaa verkkoliikenteen Internetistä yrityksen sisäverkkoon. Vuosääntöjen politiikkana oli, että ainoastaan erikseen sallitut palvelut ja yhteydet sallitaan. Vuosääntöihin osumattomat yhteydet hylätään ympäristössä. Säännöt jaettiin useampaan eri vuotauluun niiden hallittavuuden parantamiseksi. Vuotaulujen logiikka rakennettiin siten, että osumakenttien tarkastaminen tapahtuu vuotauluissa 0-2 ja vuotaulu 3 sisältää vasta varsinaisen paketin kytkemiseen liittyvän toimenpiteen.

Ensimmäiseen vuotauluun (0) luotiin ainoastaan yksi vuo, josta kaikki paketit välitettiin käsiteltäväksi vuotauluun 1. Tämän avulla saatiin seurattua kokonaisuudessaan kytkimen läpimenevää liikennemäärää.

sudo ovs-ofctl add-flow s3 "table=0, priority=10, actions=resubmit(,1)"

Vuotaulu 1 määritettiin rajaamaan, mistä kytkimen porteista liikenne sallitaan sisään sekä sallimaan ARP- ja LLDP -yhteydet verkkolaitteiden välisien tarvittavien yhteyksien muodostamiseksi. Ensimmäiset kaksi vuota koskevat ulkoverkon suuntaan olevia portteja. Koska pääasiallinen tarkoitus ei ollut palomuurata OpenFlow:n vuosääntöjen avulla ulkoverkosta tulevaa liikennettä, ei tässä tapauksessa erikseen rajoitettu lähdeosoitetta. Kaksi alimmaista sääntöä koskevat portteja 3 ja 4, joissa on kytkettynä päätelaitteet. Näissä vuosäännöissä on rajattu sallituiksi MAC-osoitteeksi ainoastaan h1 ja h2 -päätelaitteiden osoitteet *dl_src* -parametrilla. Ethernet-tai muulle OSI-mallin L2 -liikenteelle ei tarvitse erikseen määrittää *dl_type* -parametria.

 # Kytkimet - Sallitaan liikenne sisään
sudo ovs-ofctl add-flow s3 "table=1, priority=1, in_port=1 ,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=1, priority=1, in_port=2 ,actions=resubmit(,3)"

 # Päätelaitteet - Sallitaan liikenne sisään ainoastaan h1 ja h2 MAC-osoitteista
sudo ovs-ofctl add-flow s3 "table=1, priority=1, in_port=3,dl_src=00:00:00:00:00:11,actions=resubmit(,2)"
sudo ovs-ofctl add-flow s3 "table=1, priority=1, in_port=4,dl_src=00:00:00:00:00:22,actions=resubmit(,2)"

Porttisuodatuksen lisäksi vuotaulu 1 sisälsi ARP- ja LLDP -protokolliin liittyvät säännöt. ARP-protokollaa käytetään IP- ja MAC-osoitteiden välisessä käännöksessä, joka on välttämätön lähiverkon toiminnan kannalta. LLDP:n avulla kontrolleri muodostaa oman topologianäkymänsä, joten sekin haluttiin sallia.

 # Sallitaan ARP
sudo ovs-ofctl add-flow s3 "table=1, priority=1,dl_type=0x0806,actions=resubmit(,3)"

 # Sallitaan LLDP kytkimien porteista
sudo ovs-ofctl add-flow s3 "table=1, priority=5, in_port=1,dl_type=0x88cc,action=CONTROLLER:65509"
sudo ovs-ofctl add-flow s3 "table=1, priority=5, in_port=2,dl_type=0x88cc,action=CONTROLLER:65509"

Viimeisenä sääntönä vuotaulussa 1 oli *drop*-sääntö, jonka avulla hylättiin kaikki liikenne, joka ei osu vuotaulun aikaisempiin sääntöihin. Säännössä käytettiin prioriteettina arvoa 0, jolloin se luettiin vasta muiden sääntöjen jälkeen.

 # Estetään kaikki vuotaulu 1:n sääntöihin osumaton liikenne
sudo ovs-ofctl add-flow s3 "table=1, priority=0, actions=drop"

Vuotaulussa 2 oli määritelty kaikki protokollat ja palvelut, jotka sallitaan päätelaitteilta Internetin suuntaan kytkimien porteista 3 ja 4. Koska kyseessä oli OSI L3 ja L3 liikennettä, vuosäännöissä piti määrittää *dl_type, nw_proto* sekä *tcp-/upd_dst* -parametrit. Yleisimmin käytetyt parametrit on kuvattu alla olevassa esimerkissä. Tarkemmat vuosäännöt on kuvattu liitteissä 3 ja 4. Mikäli paketti osui johonkin asetetuista vuosäännöistä, se välitettiin vuotauluun 3 käsiteltäväksi. Myös vuotaulussa 2 on viimeisenä pakettien hylkäämiseen liittyvä sääntö. Esimerkki mallintaa s3-kytkimelle asetettavia sääntöjä, jossa oletetaan kytkimen LAN (Local Area Network) -portteihin olevan kytkettynä päätelaitteet, kuten esimerkiksi Windows-työasemat. Liitteen 4 esimerkki koskee kytkintä s4 ja mallintaa tilannetta, jossa kytkimen porttiin 3 on kytketty web-palvelin ja porttiin 4 nimipalvelin. Koska esimerkeissä mallinnetaan palvelimilta verkkoon suuntautuvaa liikennettä, palomuuraus tehdään pääsääntöisesti lähdeporttien perusteella (Open Networking Foundation. 2014. 44; Cantrell, C., Henmi, A., Lucas, M. & Singh, A. 2006; Shinder, T. & Behrens, T. 2011; IANA. 2017).

- *dl_type= 0x0800* = IP-liikenne
- *nw_proto=1* = ICMP-liikenne (ping)
- *nw_proto=6* = TCP-liikenne
- *nw_proto=17* = UDP-liikenne
- *tcp_dst (tai tp_dst)* = TCP-kohdeportti
- *udp_dst (tai tp_dst)* = UDP-kohdeportti

Viimeisessä vuotaulussa 3 määritettiin varsinaiset kytkentätoimenpiteet paketeille. Ensimmäisessä osiossa määritettiin päätelaitteilta Internetin suuntaan kulkeva liikenne. Prioriteettien avulla liikenne voitiin ohjata porttien 1 ja 2 kautta. Ensisijaisesti liikenne ohjattiin ulos prioriteetilla 10 portin 1 kautta *actions=output:1* -toimenpiteellä alla olevan esimerkin mukaisesti.

```
## Vuotaulu 3
# Sallitaan liikenne ulos porteista 1 ja 2. Portti 1 toimii ensisijaisena.
sudo ovs-ofctl add-flow s3 "table=3, priority=10, in_port=3,actions=output:1"
sudo ovs-ofctl add-flow s3 "table=3, priority=5, in_port=3,actions=output:2"
sudo ovs-ofctl add-flow s3 "table=3, priority=10, in_port=4,actions=output:1"
sudo ovs-ofctl add-flow s3 "table=3, priority=5, in_port=4,actions=output:2"
```

Tämän jälkeen määritettiin kohde MAC-osoitteiden avulla, että kummasta portista kytketään kohti päätelaitetta suuntautuva liikenne. Lisäksi määritettiin sääntö samalla kytkimellä olevien päätelaitteiden välille. Prioriteetti tulee tässä olla korkeampi kuin edellisissä määrityksissä, jotta samaan kytkimeen liitettyjen päälaitteiden välinen liikenne ohjaudu ulos porteista 1 tai 2 Internetin suuntaan.

```
# Sallitaan liikenne sisään kytkimien porteista
sudo ovs-ofctl add-flow s3 "table=3, priority=20,
in_port=1,dl_dst=00:00:00:00:00:11,actions=output:3"
sudo ovs-ofctl add-flow s3 "table=3, priority=20,
in_port=2,dl_dst=00:00:00:00:00:22,actions=output:4"
sudo ovs-ofctl add-flow s3 "table=3, priority=20,
in_port=3,dl_dst=00:00:00:00:00:22,actions=output:4"
sudo ovs-ofctl add-flow s3 "table=3, priority=20,
in_port=4,dl_dst=00:00:00:00:00:11,actions=output:3"
```

Viimeisenä vuosääntönä oli ARP-liikenteen salliva sääntö. Toimenpiteenä käytetään *actions=output:all* -määritystä, jolloin kytkin välittää liikenteen ulos kaikista muista, paitsi portista, josta kyseinen paketti on tullut sisään. Tämä on välttämätöntä ARP-protokollan toiminnan kannata.

```
# Sallitaan ARP-liikenne
sudo ovs-ofctl add-flow s3 "table=3, priority=10,dl_type=0x0806,actions=output:all"
```

Sääntöjen toimivuutta voitiin testata OVS-kytkinsovelluksen mukana tulevalla *Appctl ofproto/trace* -työkalulla. Sen avulla voidaan helposti selvittää, kuinka kytkin käsittelee kyseistä pakettia vuosääntöjen perusteella, kun paketti tulee sisään kytkimelle. Kuvion 10 esimerkissä testataan S4-kytkimen toimenpiteitä portista 3 sisään tulevalle paketille, jonka kohde IP-osoitteena on 10.0.0.99 ja kohdeporttina UDP 53, joka on yleisesti DNS-protokollan (engl. Domain Name System) käyttämä. *OpenFlow actions* -kohdista voidaan seurata, että ensin paketti siirretään vuotauluun 1 käsiteltäväksi ja vuotaulussa 1 se ohjataan kontrollerille Packet In -viestin avulla. *Datapath actions* parametri kertoo, kuinka kytkin tällä hetkellä käsittelisi paketin. Koska kontrolleri ei ole luonut vielä tarvittavaa sääntöä paketin välittämiseksi, parametri näyttää *drop* -argumenttia.

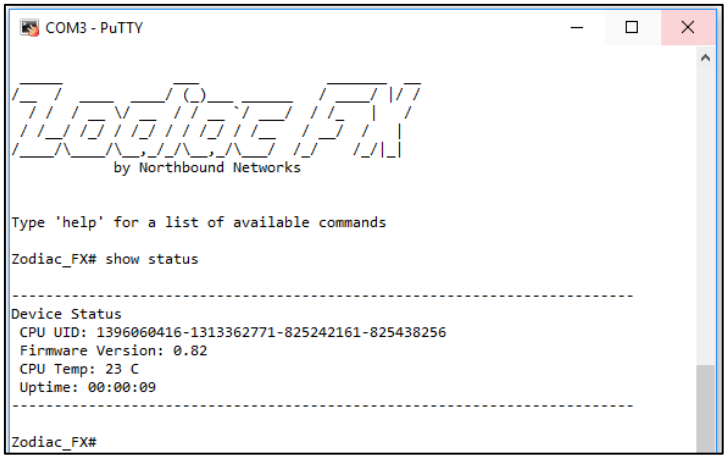
```
mininet@mininet-vm:~$ sudo ovs-appctl ofproto/trace s4 in_port=3,udp,nw_dst=10.0.0.99,udp_dst=53
Bridge: s4
Flow: udp,in_port=3,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,nw_src=0.0.0.0,nw_dst=10.0.0.99,nw_tos=0,nw_ecn=
0,nw_ttl=0,tp_src=0,tp_dst=53

Rule: table=0 cookie=0x8f54c59c43e8c0f7 priority=1,in_port=3
OpenFlow actions=goto_table:1

        Resubmitted flow: udp,in_port=3,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,nw_src=0.0.0.0,nw_dst=10.0.0
.99,nw_tos=0,nw_ecn=0,nw_ttl=0,tp_src=0,tp_dst=53
        Resubmitted regs: reg0=0x0 reg1=0x0 reg2=0x0 reg3=0x0 reg4=0x0 reg5=0x0 reg6=0x0 reg7=0x0
        Resubmitted  odp: drop
        Resubmitted megaflow: ip,in_port=3,nw_frag=no
        Rule: table=1 cookie=0x8f54c59c43e8c0f7 priority=1,in_port=3
        OpenFlow actions=CONTROLLER:65509

Final flow: udp,in_port=3,vlan_tci=0x0000,dl_src=00:00:00:00:00:00,dl_dst=00:00:00:00:00:00,nw_src=0.0.0.0,nw_dst=10.0.0.99,nw_tos=0,n
w_ecn=0,nw_ttl=0,tp_src=0,tp_dst=53
Megaflow: ip,in_port=3,nw_frag=no
Datapath actions: drop
This flow is handled by the userspace slow path because it:
        - Sends "packet-in" messages to the OpenFlow controller.
mininet@mininet-vm:~$
```

KUVIO 10 ovs-appctl ofproto/trace esimerkki

5.5 Puikkari & Zodiac FX

Ensimmäinen toteutus tehtiin fyysisen Zodiac FX -kytkimen ja Linux-pohjaisten työasemien avulla. SDN-kontrollerina käytettiin Puikkaria. Päätelaitteina käytettiin MacBook Pro ja RaspBerry Pi -päätelaitteita. Toteutus aloitettiin Zodiac FX -kytkimen konfiguroinnilla. Yhteys laitteeseen muodostettiin USB-liitännän ja XMODEM-tiedonsiirtoa tukevan terminaaliohjelman (ExtraPutty) avulla. Laitteen komentorivin käyttöjärjestelmä on hyvin yksinkertainen ja selkeä, joten tässä toteutuksessa käytettiin pääasiassa ainoastaan sitä. Käyttöjärjestelmä on jaettu kolmeen osioon: Base, Config ja OpenFlow, jossa jokaisessa on omat komentonsa. Aluksi Zodiac FX -kytkimeen asennettiin viimeisin firmware-päivitys (v0.82), jotta se tukisi paremmin OpenFlow 1.3.0 -versiota ja muita ympäristöön liittyviä toiminnallisuuksia. Aikaisempi firmware-versio oli kuitenkin sen verran uusi, että päivitys voitiin toteuttaa suoraan komentoriviltä tai web-käyttöliittymän kautta.

Päivitys tehtiin tässä tapauksessa komentorivin kautta. Uuden firmwaren .bin -tiedosto ladattiin laitteen tukisivustolta ja kytkimen käyttöjärjestelmän Base-tilassa annettiin komento *update*, jonka jälkeen .bin -tiedosto asennettiin XMODEM -tiedonsiirron avulla, terminaaliohjelmaa käyttäen. Päivityksen jälkeen laite käynnistyi uudelleen automaattisesti. Kuviossa 11 on havainnollistettu laitteen komentoriviltä toimiva käyttöjärjestelmä ja päivitetty firmware-versio *show status* -komennon avulla (Northbound Networks 2017).



KUVIO 11 Zodiac FX:n komentorivi ja päivitetty firmware

Seuraavaksi konfiguroitiin laitteen verkkorajapinnat sekä tarvittavat osoite- ja porttitiedot virtualisoidussa linux-koneessa pyörivälle SDN-kontrollerille, Puikkarille. Lopuksi konfiguraatio tallennettiin, jotta muutokset pysyvät voimassa myös laitteen uudelleenkäynnistymisen jälkeen. Määritykset tehtiin *config*-tilasta seuraavasti:

Zodiac_FX# config
Zodiac_FX(config)# set ip-address 192.168.1.14
Zodiac_FX(config)# set netmask 255.255.255.0
Zodiac_FX(config)# set gateway 192.168.1.1
Zodiac_FX(config)# set of-controller 192.168.1.133
Zodiac_FX(config)# set of-port 6633
Zodiac_FX(config)# save

Uusi asetettu konfiguraatio saatiin näkyviin *config*-tilassa *show config* -komennon avulla. Kytkimen uudet asetukset on esitetty kuviossa 12.

```
Zodiac_FX# config
Zodiac_FX(config)# show config

------------------------------------------------------------------------
Configuration
 Name: Zodiac_FX
 MAC Address: 70:B3:D5:6C:D6:99
 IP Address: 192.168.1.14
 Netmask: 255.255.255.0
 Gateway: 192.168.1.1
 OpenFlow Controller: 192.168.1.133
 OpenFlow Port: 6633
 Openflow Status: Enabled
 Failstate: Secure
 Force OpenFlow version: Disabled
 EtherType Filtering: Disabled

------------------------------------------------------------------------
```

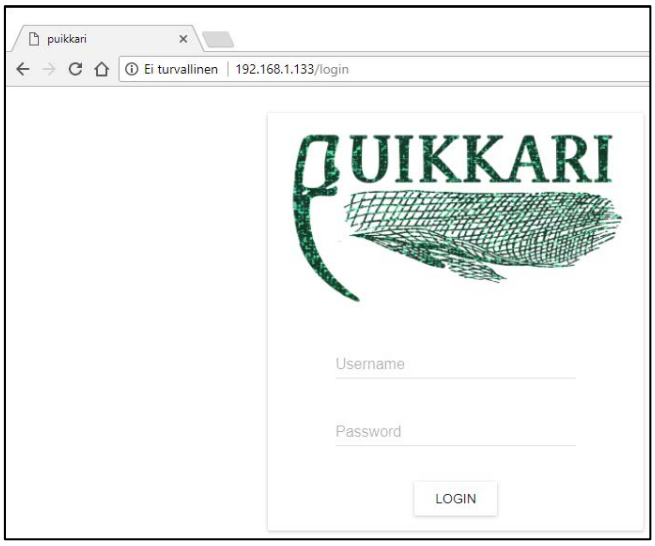
KUVIO 12 Zodiac FX -kytkimen verkkokonfiguraatio

Seuraavaksi tehtiin Puikkariin liittyvä konfiguraatio. Valmistelut aloitettiin lataamalla Puikkarin virtuaalikone VirtualBox-sovellukseen import-toiminnon avulla sekä sillattiin koneen verkkorajapinta isäntäkoneen fyysiseen rajapintaan.

Tämän jälkeen aloitettiin itse Puikkari-virtuaalikoneen konfigurointi. Aluksi koneen IP-osoite piti määrittää Puikkarin konfiguraatioon */opt/puikkari-vm/update_myip.sh* -komennolla. Seuraavaksi käynnistettiin tarvittavat palvelut verkkoliitäntöihin, topologiapalveluun, redis-tietokantaan sekä itse kontrolleriin liittyen:

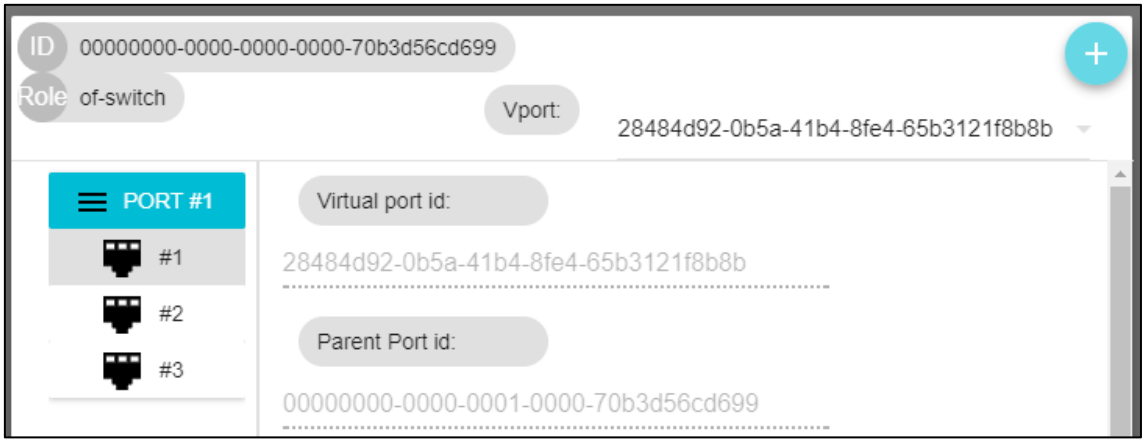
sudo /opt/puikkari-vm/start_puikkari.sh
/opt/puikkari-vm/start_redis.sh
/opt/puikkari-vm/start_socket.sh
/opt/puikkari-vm/start_topo.sh
/opt/puikkari-vm/start_ryu.sh

Tämän jälkeen Puikkaria ja siihen liitettyä SDN-verkkoa voitiin hallita web-selaimella. Kirjautuminen web-käyttöliittymään tapahtui virtuaalikoneen IP-osoitteen *192.168.1.133* kautta. Kuviossa 13 on esitetty Puikkarin kirjautumisikkuna web-selaimessa:



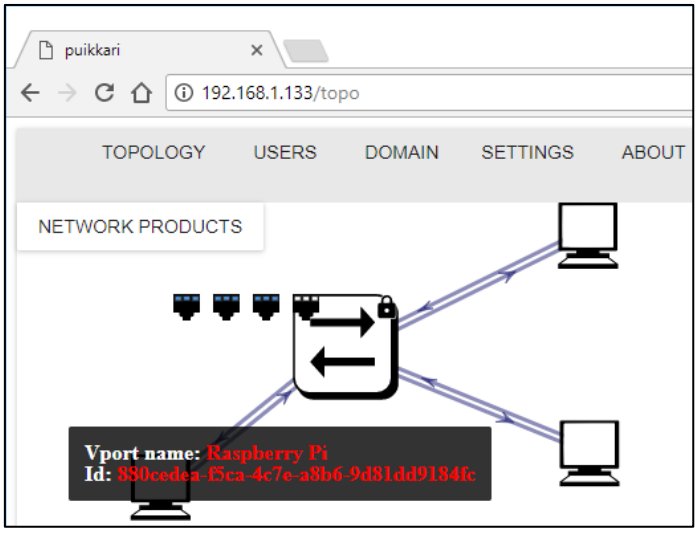
KUVIO 13 Puikkarin kirjautumisikkuna

Koska Zodiac FX -kytkimelle oli aikaisemmin asetettu kontrolleriksi Puikkari-koneen IP-osoite, kytkin ilmestyi automaattisesti Puikkarin topologianäkymään. Kytkimen fyysisiin rajapintoihin luotiin ja nimettiin virtuaaliset portit sekä tehtiin niihin tarvittavat porttimääritykset, jotta OpenFlow-verkkoliikenne kulkisi päätelaitteiden välillä. Kuviossa 14 on esitetty kytkimen fyysiseen porttiin #1 luotu virtuaalinen portti, jonka tunnisteen (*Virtual port id:28484d94-0b4a..*) Puikkari on muodostanut automaattisesti.



KUVIO 14 Virtuaaliportti Puikkarissa

Kuviossa 15 on esitetty Puikkarin topologianäkymä, jossa näkyy sekä Zodiac FX -kytkin että siihen liitetyt päätelaitteet. Esimerkissä on havainnollistettu päätelaitteena käytetyn RaspBerry Pi:n porttitunniste (id) sekä virtuaalisen portin nimi.



KUVIO 15 Puikkarin topologianäkymä

Pelkällä Zodiac FX -kytkimellä ei voida luoda vuosääntöjä, vaan ne täytyy tehdä ulkopuolisen kontrollerin ja/tai siihen liittyvän sovelluksen avulla. Puikkari generoi kytkimelle automaattiset vuosäännöt reaktiivisesti, mutta se ei kuitenkaan tukenut manuaalista vuosääntöjen luontia. Tästä johtuen tätä toteutusta päätettiin olla jatkamatta pidemmälle. Sääntöjä olisi voitu luoda manuaalisesti esimerkiksi erillisen python-skriptin avulla. Automaattisesti luodut vuosäännöt olivat nähtävissä kytkimellä *show flows* -komennolla. *Show status* -komennolla voitiin nähdä kytkimen ja Puikkarin välisen yhteyden tila, neuvoteltu OpenFlow-versio 1.3 sekä luotujen vuosääntöjen ja vuotaulujen määrät. Alla olevassa esimerkissä (Kuvio 16) on esitetty vuotaulussa 1 oleva vuo numero 9,

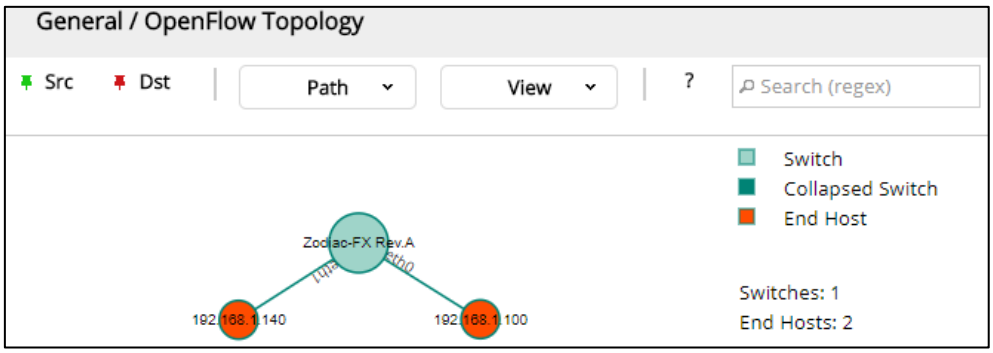
jossa paketti tulee sisään portista 1. Vuosäännössä näkyy myös sekä lähde- että kohde MAC-osoitteet.

```
Flow 9
 Match:
  In Port: 1
  Destination MAC: 01:00:5E:7F:FF:FA
  Source MAC: 98:DE:D0:1A:27:2B
 Attributes:
  Table ID: 0                          Cookie:0xae398cc16d0ea13d
  Priority: 10                         Duration: 4 secs
  Hard Timeout: 20 secs                Idle Timeout: 5 secs
  Byte Count: 651                      Packet Count: 3
  Last Match: 00:00:01
 Instructions:
  Apply Actions:

Zodiac_FX(openflow)# show status

------------------------------------------------------------------------------
 Status: Connected
 Version: 1.3 (0x04)
 No tables: 2
 No flows: 7
 Total Lookups: 13621
 Total Matches: 13621
```

KUVIO 16 Esimerkki vuosäännöistä sekä OpenFlow-yhteydestä

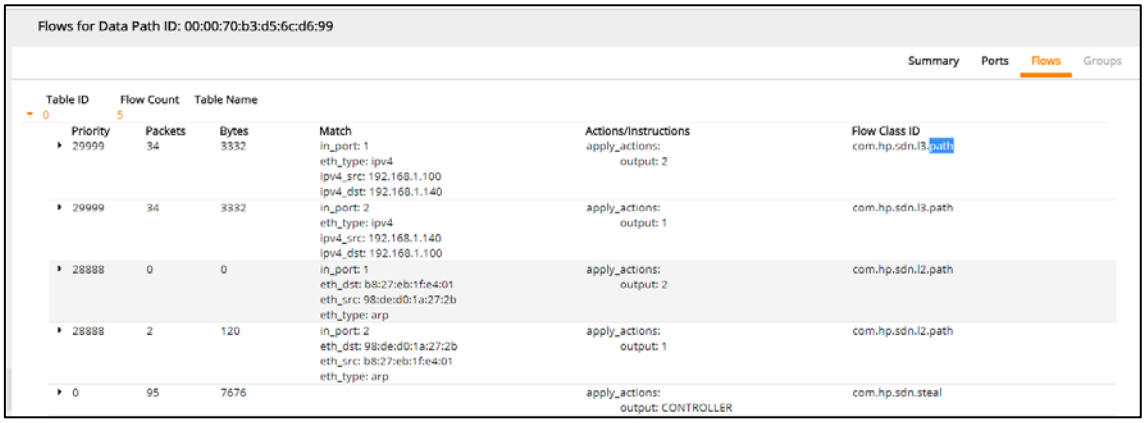
5.6 Aruba VAN SDN Controller & Zodiac FX

Toisena toteutuksena Zodiac FX -kytkintä lähdettiin kokeilemaan uuden kontrollerin, Aruban VAN SDN Controller -ohjelmiston kanssa. Verkkotopologian uudeksi kontrolleriksi määritettiin uuden kontrollerin virtuaalikoneen IP-osoite *192.168.1.129*. Kontrollerista oli saatavilla ilmainen trial-versio, joka oli ladattavissa HP:n sivuilta virtuaalikoneen muodossa. Virtuaalikoneeseen tehtiin VirtualBox:ssa vastaavat määritykset kuin aikaisemmassa toteutuksessa Puikkarilla. Kirjautuminen JAVA-pohjaiseen web-portaaliin, josta kontrolleria hallittiin, tapahtui selaimella *https://192.168.1.129:8443/sdn/ui/* -osoitteesta. Aruban kontrollerin topologianäkymä on kattava ja se näyttää havainnollisesti esimerkiksi laitteiden IP-osoitteet, portit sekä käytettävissä olevat datapolut. Kontrollerin topologianäkymä on esitetty kuviossa 17.



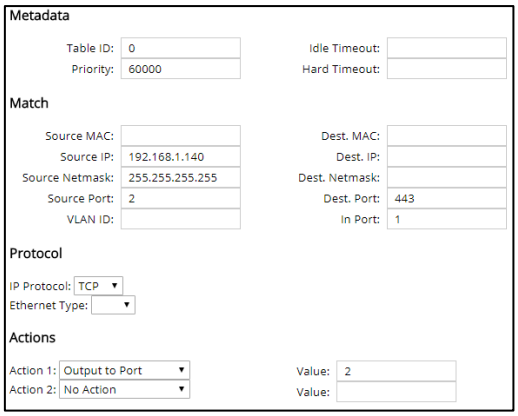
KUVIO 17 Aruba VAN SDN Controller -topologianäkymä

Koska Zodiac FX oli puhtaasti OpenFlow-kytkin, eikä näin ollen tukenut tavallista ethernet-kytkentää, kontrollerin hybriditila piti aluksi kytkeä pois. Kontrolleri loi automaattisesti tarvittavat vuosäännöt reaktiivisesti päätelaitteiden välille. Kuviossa 18 on esitetty päätelaitteiden välille reaktiivisesti l3.path ja l3.path -sovelluksilla luodut vuosäännöt sekä IPv4-liikenteeseen että ARP-protokollaan (Address Resolution Protocol) liittyen.



KUVIO 18 Reaktiivisesti generoidut vuosäännöt

Oletusasetuksilla kontrollerin graafisesta käyttöliittymästä ei voida luoda sääntöjä manuaalisesti. Sääntöjen luontia varten kontrollerille asennettiin Flow Maker -sovellus HP:n sovelluskaupasta. Kuviossa 19 on esitetty vuosääntöjen luontiin liittyvät käytettävissä olevat parametrit. Kontrollerin Trial-versio ei kuitenkaan sallinut vuosääntöjen tallentamista, joten vuot hävisivät aina kontrollerin uudelleenkäynnistyksen yhteydessä. Kontrollerin käyttöliittymässä esiintyi useita ohjelmointivirheitä (engl. Bug) muun muassa Flow Maker -sovelluksen toimivuudessa, jotka haittasivat toteutuksen tekemistä. Lisäksi jokainen sääntö piti luoda yksitellen ilman kopiointimahdollisuutta, joka teki sääntöjen luonnista erittäin työlästä.



KUVIO 19 Vuosääntöjen luontiin liittyvät parametrit

5.7 Puikkari & Mininet

Kolmannessa toteutuksessa Zodiac FX sekä fyysiset päätelaitteet jätettiin kokonaan pois ja ne korvattiin mininetin virtuaalisilla päätelaitteilla ja OVS-kytkimillä. Tämä mahdollisti huomattavasti laajemman topologian sekä kaikkien laitteiden hallinnan yhdeltä työasemalta. Asennus aloitettiin lataamalla mininetin virtuaalikone ja tuomalla se VirtualBoxiin. Koska mininetin verkkotopologian haluttiin mallintavan operaattorin verkkoa, mininetin topologia tehtiin kustomoituna Python-skriptin (LIITE 1) avulla. Topologiassa on mallinnettu operaattorin runko-, keräily-, liityntäverkot, päätelaitteet (h1-h6), kytkimet (s1-s6) sekä yhdyskäytävä (engl. Gateway) Internetin suuntaan. Tarkoituksena oli, että topologia mallintaa operaattorin verkkoympäristöä, mutta on samalla manuaalisilla palomuuraukseen liittyvillä vuosäännöillä hallittavissa. Samalla määritettiin sekä päälaitteiden että kytkinten MAC- ja IP-osoitteet, jotta ne voitiin tunnistaa myöhemmin testauksen yhteydessä. Lisäksi samalla skriptillä saatiin helposti määriteltyä verkkotopologialle ulkoinen SDN-kontrolleri. Python-skriptin nimeksi annettiin *sdn_network.py* ja siihen liittyvä Mininet-topologia käynnistettiin komennolla *sudo ./sdn_network.*

Mininet-ympäristö sisältää useita työkaluja kytkinten ja päätelaitteiden, sekä niiden välisten linkkien tarkastelemiseen. Näitä ovat esimerkiksi *net*, *nodes* ja *links.* Mininetin konsoli tukee XTERM-toiminnallisuutta, jonka avulla voidaan avata omia komentoriviyhteyksiä (engl. shell) topologiassa oleville laitteille. Tämä ominaisuus on erittäin hyödyllinen esimerkiksi komponenttien välisten yhteyksien testauksissa päästä päähän. Lisäksi komentoriveiltä voidaan käyttää muita hyödyllisiä työkaluja verkkoliikenteen seurannassa ja testauksessa, kuten esimerkiksi Wireshark, Telnet tai NetCat.

Verkkotopologiassa olevien komponenttien välisiä yhteyksiä voidaan testata esimerkiksi *ping* ja *pingall* -komentojen avulla. Pingall testaa ICMP-echolla kaikkien komponenttien välisten yhteyksien toimivuuden ja samalla kontrolleri havaitsee SDN-verkossa olevat työasemat. Ennen virtuaaliporttien luontia kytkimillä oli ainoastaan vuosääntö pakettinen pudottamiselle (drop), joten yhteydet komponenttien välillä eivät toimineet.

Kun Mininet-topologia on luotu ja kontrolleri on havainnut siinä käytettävät komponentit, ne tulevat näkyviin Puikkarin topologianäkymään. Virtuaaliporttien luonnin yhteydessä Puikkari luo kytkimelle porttikohtaiset vuosäännöt. Kuviossa 20 on kytkimelle *s5* luotu virtuaaliportti, mutta koska portti on vielä Puikkarissa *disabled*-tilassa, säännön toimenpiteenä on paketin pudottaminen.

```
mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s5
NXST_FLOW reply (xid=0x4):
 cookie=0xb08456e1c66033f7, duration=57.081s, table=0, n_packets=0, n_bytes=0, idle_age=57, priority=1,in_port=3 actions=drop
 cookie=0x0, duration=1169.154s, table=0, n_packets=48, n_bytes=2016, idle_age=453, priority=0 actions=drop
mininet@mininet-vm:~$
```

KUVIO 20 Drop-sääntö Puikkarissa

Kun virtuaalinen portti laitetaan aktiiviseen tilaan, Puikkari luo automaattisesti porttiin liittyvät vuosäännöt. Uudet vuosäännöt luodaan korkeammalla prioriteetilla 1, jotta ne huomioidaan ennen prioriteetilla 0 olevaa drop-sääntöä. Kytkimen portit, joihin on kytkettynä päätelaite, määritetään *Edge Port* -tilaan. Kuviossa 21 on esitetty Puikkarin *Edge Port* -tilaiseen porttiin automaattisesti tekemät vuosäännöt sekä lisätyt päätelaitteet ja niihin liittyvät linkit topologianäkymään. Ensimmäinen sääntö ohjaa siirtymään vuotauluun numero 1 ja alin vuotaulussa 1 oleva sääntö ohjaa verkkoliikenteen kontrollerille.

```
mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s3
NXST_FLOW reply (xid=0x4):
 cookie=0x8195951bebab8c38, duration=5.679s, table=0, n_packets=0, n_bytes=0, idle_age=5, priority=1,in_port=3 actions=resubmit(,1)
 cookie=0x0, duration=1489.243s, table=0, n_packets=72, n_bytes=3024, idle_age=845, priority=0 actions=drop
 cookie=0x8195951bebab8c38, duration=5.679s, table=1, n_packets=0, n_bytes=0, idle_age=5, priority=1,in_port=3 actions=CONTROLLER:6550
9
```

KUVIO 21 Puikkarin reaktiivisesti luomat vuosäännöt

Kytkinten portit, joihin on liitetty toinen kytkin, määritetään *Internal* -tilaan. Näihin portteihin täytyi konfiguroida LLDP-määritykset, jotta portit osaavat kuunnella ja lähettää OpenFlow-protokollan käyttämiä LLDP-sanomia, jonka avulla SDN-kontrolleri pitää yllä verkkotopologiaa. Konfiguraatio tehtiin seuraavien parametrien avulla:

{"lldp_listen":true,"lldp_send":true,"lldp_send_interval":2}

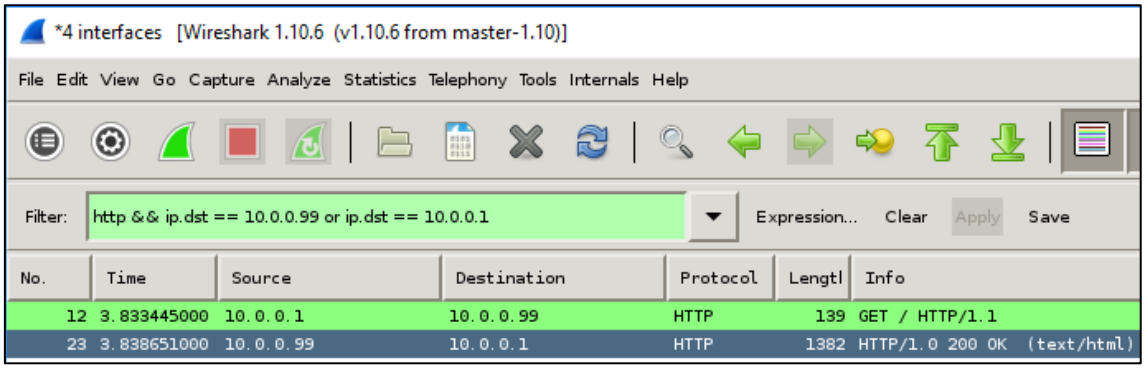
Samalla Puikkari loi tarvittavan vuosäännön, jotta kytkin hyväksyy kontrollerilta tulevan ja sille välitettävän LLDP-liikenteen. Luodun LLDP-vuosäännön tunnistaa Ethernet *0x88cc* -tyypistä.

cookie=0x89ad361b4dabf193, duration=118.506s, table=0, n_packets=58, n_bytes=4582, idle_age=1, priority=101,in_port=2,dl_type=0x88cc actions=CONTROLLER:65509

Tämän jälkeen yhteyksien toimivuutta voitiin testata. Puikkari loi reaktiivisesti tarvittavat säännöt virtuaalilaitteiden välille. Alla on kuvattu automaattisesti luodut vuosäännöt työaseman h1 ja gw:n välillä. Ensimmäinen sääntö koskee paketin tuloa päätelaitteelta kytkimelle. Kytkin lisää pakettiin *vlan 4097* -tagin ja kytkee sen ulos portista 2. Alimmainen vuosääntö koskee paketin vastaanottoa gw:n suunnasta portista 2. Tässä tapauksessa kytkin poistaa vlan-tagin ja kytkee paketin ulos portista 3 kohti päätelaitetta h1.

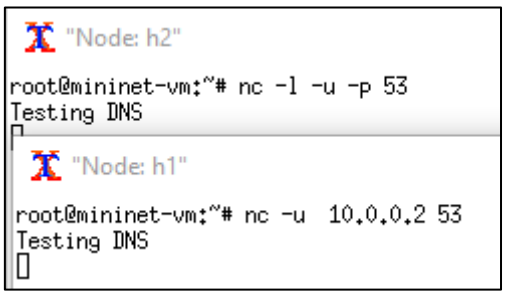
cookie=0x0, duration=1.046s, table=0, n_packets=1, n_bytes=74, idle_timeout=5, hard_timeout=20, priority=10,in_port=3,dl_src=00:00:00 actions=push_vlan:0x8100,set_field:4097->vlan_vid,output:2
cookie=0x0, duration=1.034s, table=0, n_packets=1, n_bytes=58, idle_timeout=5, hard_timeout=20, priority=10,in_port=2,dl_vlan=1,dl_sr0:00:00:11 actions=pop_vlan,output:3

Pingin lisäksi yhteyksien testaamisessa käytettiin muun muassa Curl-työkalua, jonka avulla saadaan generoitua HTTP (Hypertext Transfer Protocol) -pyyntöjä komentoriviltä. Yhteyttä HTTP-protokollan yleisesti käyttämään TCP 80 -porttiin voitiin testata *curl 10.0.0.99 80* -komennon avulla. Alla esitetystä Wireshark-seurannasta (Kuvio 22) voidaan nähdä onnistunut HTTP GET -pyyntö ja siihen liittyvä palvelimen 200 OK -vastaus.



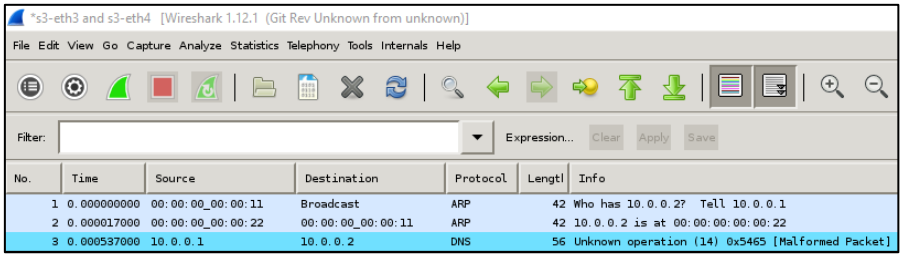
KUVIO 22 Onnistunut HTTP-pyyntö ja -vastaus

UDP-protokollan toimivuutta testattiin NetCat-työkalulla. Sen avulla voitiin luoda kuuntelija kohdepalvelimelle ja ottaa siihen päätelaitteelta UDP-yhteys määritettyyn kohdeporttiin 53. Alla olevassa kuviossa 23 on esitetty kaksi terminaalinäkymää, jossa ylimmäisessä määritetään Netcat-kuuntelija UDP-porttiin 53 *nc -l -u -p 53* -komennon avulla. Alemmassa terminaalissa otetaan päätelaitteelta yhteys kuuntelijaan *nc -u 10.0.0.2* -komennolla. Tämän jälkeen yhteys muodostui ja toiminta testattiin kirjoittamalla päätelaitteelle ”Testing DNS”, joka tuli näkyviin palvelimen kuuntelijaan.



KUVIO 23 Yhteyden testaaminen NetCat-työkalulla

Yhteys voitiin todentaa myös Wiresharkin avulla. Alla olevassa kuviossa 24 on esitetty *10.0.0.2* ja *10.0.0.2* välinen ARP-kysely (kaksi ensimmäistä riviä) sekä H1-päätelaitteen lähettämä UDP 53 -porttiin lähettämä paketti, jonka Wireshark tulkitsee DNS-liikenteeksi (alin rivi).



KUVIO 24 UDP 53 -paketti Wiresharkissa

5.8 ONOS & Mininet

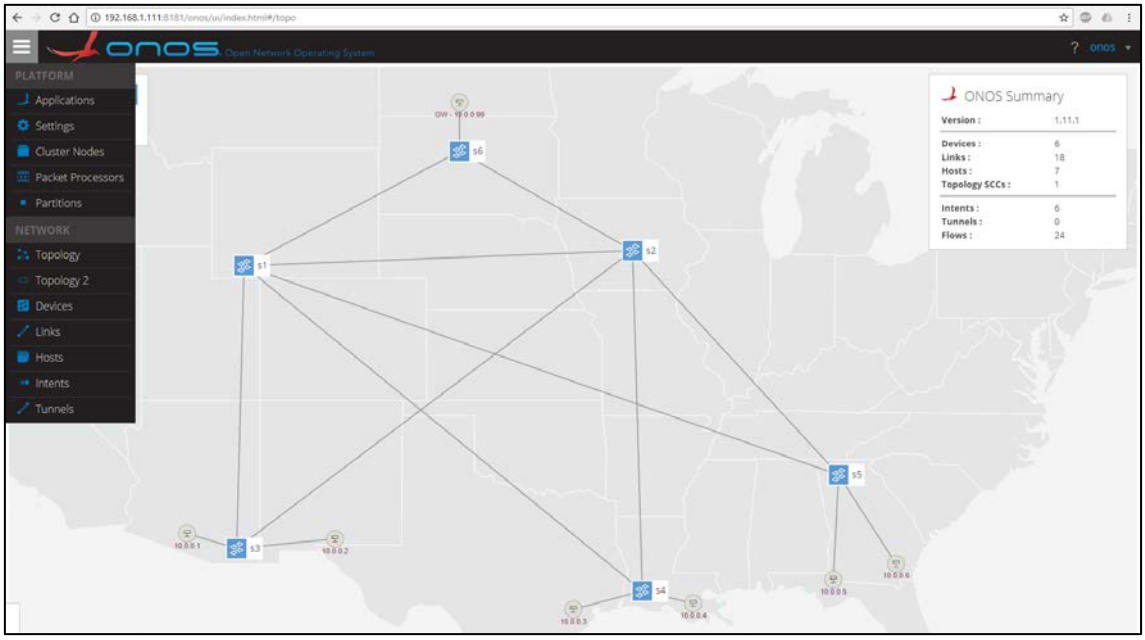
Tässä toteutuksessa käytettiin edellisen tapaan Mininettiä ja siihen edellisessä toteutuksessa luotua verkkotopologiaa. Tällä kertaa SDN-kontrollerina toimi JAVA-pohjainen ONOS. Ennen asennusta piti varmistaa, että Javan versio 8 oli asennettu oletukseksi. Asennus aloitettiin hakemalla ONOS-projektin verkkosivustolta valmis virtuaalikone versio 1.5.1 VirtualBoxiin ja tämän virtuaalikone päivitettiin uusimaan ONOS-versioon 1.11.1 manuaalisesti.

wget https://downloads.onosproject.org/release/onos-1.11.1.tar.gz
tar xvkf onos-1.11.1.tar.gz
mv onos-1.11.1 onos

ONOS-palvelimen konfiguraatioon asetettiin seuraavat ympäristömuuttujat, jotta palvelu saatiin toimimaan halutusti IP-osoitteessa *192.168.1.111*.

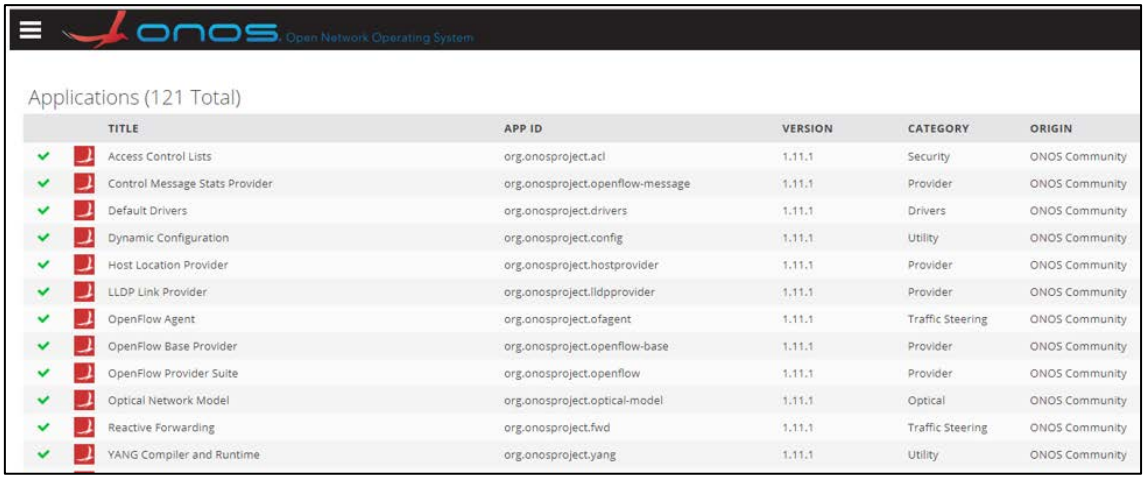
export HOME=/root
export JAVA_HOME=/usr/lib/jvm/java-8-oracle
export ONOS_ROOT=/root/onos
export KARAF_VERSION=3.0.5
export KARAF_ROOT=/root/onos/apache-karaf-3.0.5
export KARAF_LOG=/root/onos/apache-karaf-3.0.5/data/log/karaf.log
export PATH=$PATH:$KARAF_ROOT/bin
export ONOS_IP=192.168.1.111

Tämän jälkeen palvelu voitiin käynnistää */root/onos/bin/onos-service* -komennon avulla. ONOS-kontrolleri sisältää palvelun oman käyttöjärjestelmän ja siihen liittyvän komentorivin. Kirjautuminen graafiseen käyttöliittymään tapahtui http://192.168.1.111:8181/onos/ui/ osoitteen kautta. Graafiseen käyttöliittymän topologianäkymään saatiin tuotua taustalle karttapohjia, jolloin SDN-verkon maantieteellinen esittäminen sekä visualisointi helpottui. Kuviossa 25 on esitetty toteutettu SDN-verkko USA:n karttapohjalla.



KUVIO 25 ONOS-kontrollerin topologianäkymä

Sovelluksien asennus onnistuu joko graafisen käyttöliittymän tai ONOS-komentorivin avulla. Oletuksena reaktiivinen pakettienvälitys ja LLDP-viestien välittäminen eivät ole käytössä, joten kytkimien väliset linkit tai päätelaitteet eivät näy topologiassa automaattisesti. Tärkeimmät sovellukset, kuten *Reactive forwarding* sekä *LLDP Link Provider* asennettiin graafisen käyttöliittymän Applications -osiosta. Kuviossa 26 on esitetty asennettuja ja käytössä olevia sovelluksia. Sovellukset saadaan käyttöön helposti osoittamalla listasta haluttu sovellus ja valitsemalla ”*Activate selected Application*”.



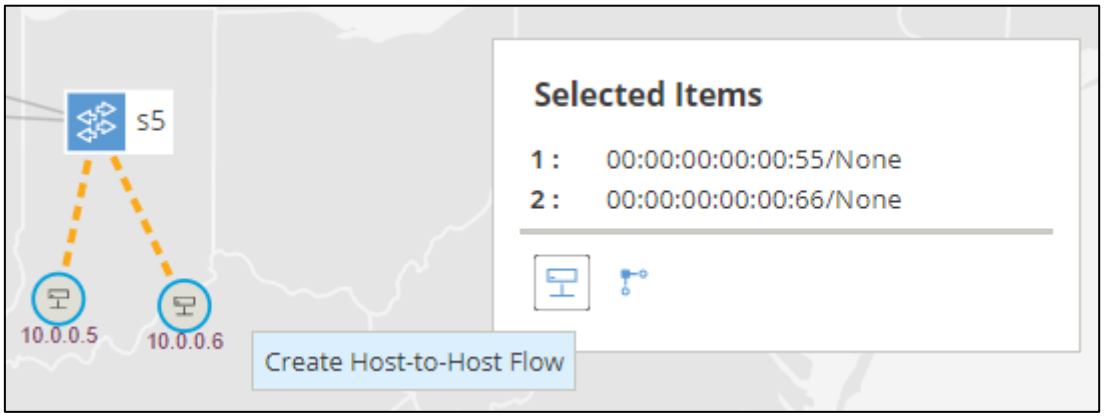
KUVIO 26 ONOS-kontrollerin sovellusnäkymä

Vuosääntöjen luonti ONOS-kontrollerissa tapahtuu intenttien (engl. intent) avulla. Intentit ovat liiketoiminta- tai järjestelmälähtöisiä, korkean tason poliitikoita ja vaatimusmäärityksiä SDN-verkon toimintaan. Intentit mahdollistavat

SDN-verkon teknisten yksityiskohtien abstraktoinnin niin, että verkon ylläpitäjien tarvitsee ainoastaan määrittää verkotukseen liittyvä tarve. Ylläpitäjien ei tarvitse määritellä esimerkiksi tarkkojen parametrien avulla järjestelmälle, että miten yhteydet toteutetaan teknisesti. Perusajatuksena on, että SDN-verkon ylläpitäjien, kuten operaattorin, tarvitsee ainoastaan tietää haluttu verkkoliikenteen palvelunlaatu, käyttäytyminen sekä päätepisteet asiakasverkoissa. Lisäksi intenttien avulla saadaan automatisoitua operaattorien verkonhallintaan liittyviä konfiguraatiotoimenpiteitä. Intentit eivät ole valmistaja- tai teknologiariippuvaisia, joten niitä voidaan käyttää universaalisti esimerkiksi useiden laitevalmistajien laitteilla toteutetuissa SDN-verkoissa (Han, Y., Li, J., Hoang, D., Yoo, J. H., & Hong, J. W. K. 2017).

Toteutuksessa testattiin kahta erilaista Intent-tyyppiä: *point-intent* ja *host-intent*. *Point-intent* luo yhteyden kahden topologiassa olevan kytkimen portin välille. Sillä saadaan määritettyä haluttu polku SDN-verkon liikenteelle huomattavasti tarkemmin, mutta se vaatii verkon toimintaan huomattavasti enemmän vuosääntöjä, kun jokaisen kytkimen välille täytyy määrittää *point-intentit* molempiin suuntiin. *Host-Intent* taas määritetään kahden päätepisteen välille ja ONOS-kontrolleri selvittää automaattisesti päätepisteiden välisen polun sekä sen varrella olevat kytkimet. Tämän jälkeen kontrolleri luo automaattisesti tähän liittyvät vuot SDN-verkon kytkimille. Tämä helpottaa merkittävästi vuosääntöjen hallintaa, koska verkon ylläpitäjän tarvitsee tietää ainoastaan yhteyden päätepisteet sekä yhteydessä käytettävät parametrit.

ONOS-kontrollerin graafisesta käyttöliittymästä pystyi luomaan *host-intenttejä*. Kuvion 27 esimerkissä on esitetty h5 ja h6 -päätelaitteiden välille luotu *host-intent*. Graafinen käyttöliittymä ei kuitenkaan tukenut palomuuraukseen liittyvien vuosääntöjen luontia, joten sitä jatkettiin manuaalisesti ONOS-komentoriviltä.

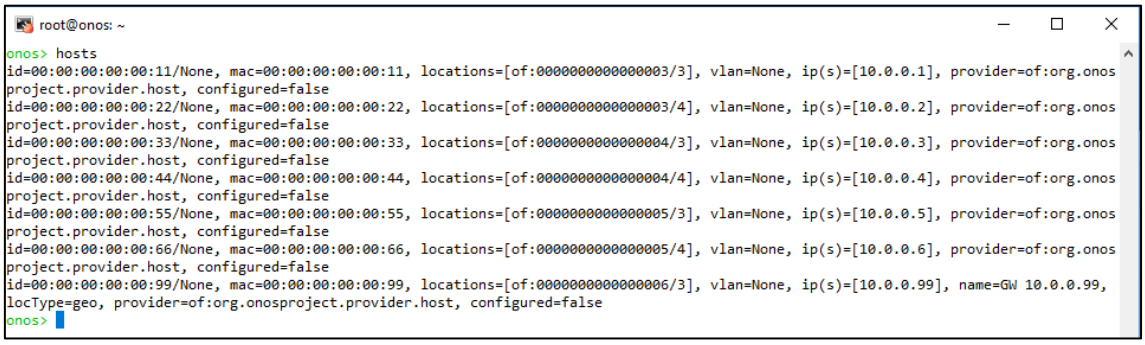


KUVIO 27 Host-intent lisäys ONOS-kontrollerin topologianäkymästä

Intenttien luominen manuaalisesti aloitettiin poistamalla reaktiivinen pakettien välitykseen liittyvä *org.onosproject.fwd* -sovellus käytöstä, jotta palomuuraukseen liittyvien intenttien toimivuus voitiin testata.

app deactivate org.onosproject.fwd

Seuraavaksi selvitettiin päätepisteiden tunnukset (id), joita käytetään intenttien luonnissa. *Hosts* -komento listaa kaikki kontrollerin tunnistamat päätelaitteet. Kuviossa 28 on esitetty käytetyn topologian päätelaitteet. Tunniste on muotoa *MAC-osoite/Vlan*. Esimerkiksi h1-päätelaitteen tapauksessa tunniste on *00:00:00:00:00:11/None*. None tarkoittaa, ettei kyseistä intenttiä ole määritetty mihinkään VLAN-verkkosegmenttiin.

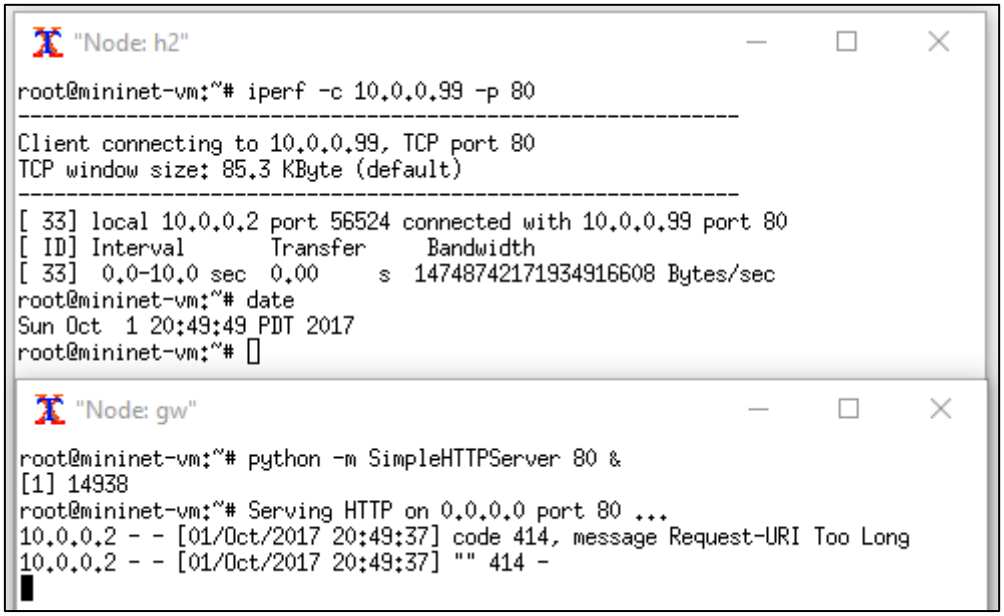


KUVIO 28 ONOS-kontrollerin tunnistamat päätelaitteet

Sääntöjen luonti tehtiin *add-host-intent* -komennolla ja siihen asetetuilla parametrivalinnoilla. Alla olevassa esimerkissä *host-intent* määritetään h1 ja gw -päätelaitteiden välille TCP 80 (HTTP) -porttiin. Ensimmäinen intentti on varsinaista yhteydenottoa varten ja toinen siihen liittyviä paluupaketteja, jolloin TCP 80 määritetään lähdeportiksi.

add-host-intent --tcpDst 80 00:00:00:00:00:22/None 00:00:00:00:00:99/None
add-host-intent --tcpSrc 80 00:00:00:00:00:99/None 00:00:00:00:00:22/None

Yhteyttä testattiin tällä kertaa *iperf*-sovelluksen avulla, joka mittaa yhteysnopeuden sekä yhteyden laadun kahden kohteen välillä. Kuviossa 29 on esitetty alemmassa terminaalissa GW:llä toimiva HTTP-palvelin, johon h2 ottaa yhteyden. Ylemmässä terminaalissa nähdään h2-päätelaitteelta tehty yhteydenotto TCP 80 -porttiin *iperf -c 10.0.0.99 -p 80* -komennolla.



KUVIO 29 Yhteyden testaus Iperf-työkalulla

5.9 Floodlight & Mininet

Viimeinen toteutus tehtiin Mininetin sekä Floodlight-kontrollerin avulla. Asennus aloitettiin hakemalla Floodlightin virtuaalikone osoitteesta http://opennetlinux.org/binaries/floodlight-vm.zip ja tuomalla se VirtualBoxiin. Mininet-topologia sammutettiin ja siihen liittyvään Python-skriptiin päivitettiin Floodlight-kontrollerin IP-osoite *192.168.1.134*. Lisäksi Floodlight käytti oletuksena eri TCP-porttia (6653), joka piti päivittää python-skriptiin. Tämän jälkeen kontrolleri päivitettiin viimeisimpään versioon git-versionhallinnan avulla.

cd ~/floodlight
git pull origin master
git submodule init
git submodule update

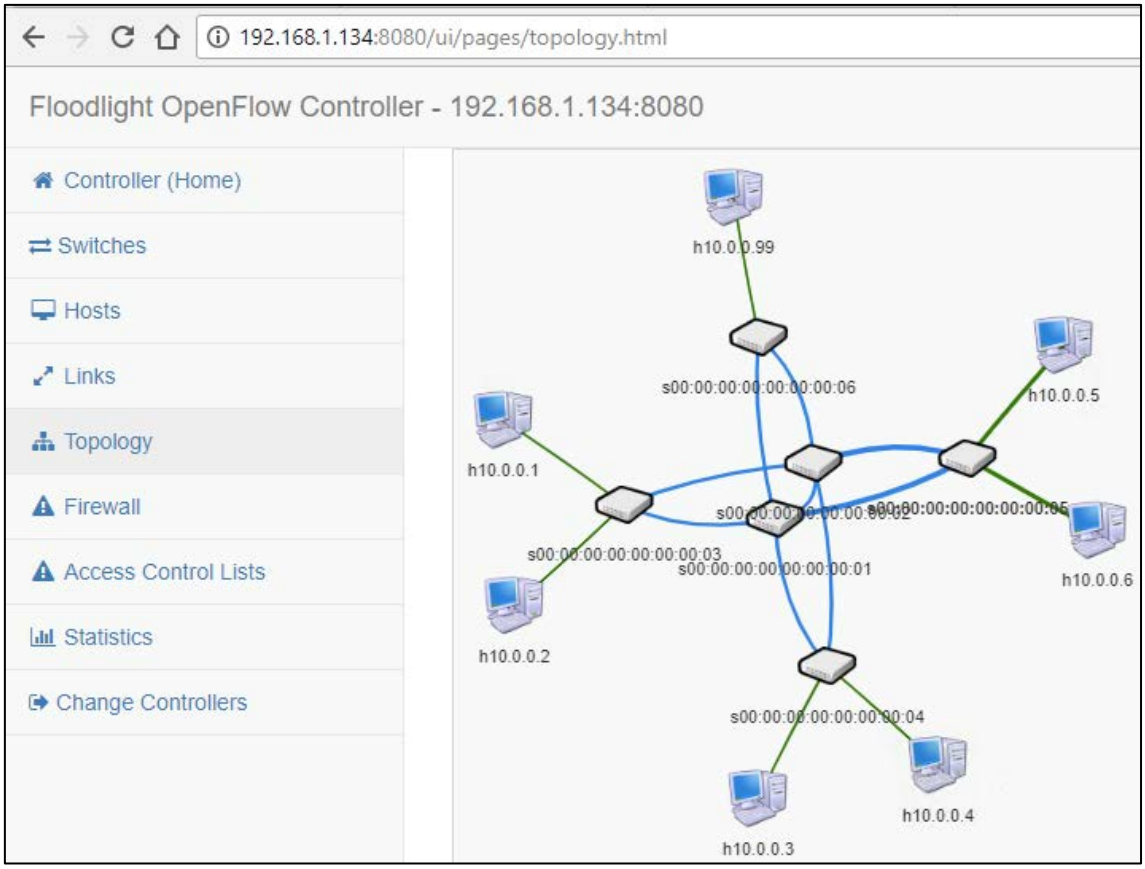
Ennen kontrollerin käynnistämistä, piti päivittää Floodlightin virtuaalikoneelle viimeisimmät pakettilistat sekä asettaa Javan oletusversioksi 8.

sudo apt-get update
sudo apt-get install oracle-java8-installer
sudo apt install oracle-java8-set-default

Kontrolleri käynnistettiin ja samalla siihen liittyvät lokitiedot ohjattiin tiedostoon log.txt. Lokitietoja tarkasteltiin *sudo watch tail log.txt* -komennolla toisesta terminaali-ikkunasta.

ant
java -jar target/floodlight.jar > log.txt
sudo watch tail log.txt (toinen terminaali-ikkuna)

Floodlight sisältää pelkistetyn graafisen käyttöliittymän, jonka avulla kontrolleria voidaan hallita. Kuviossa 30 on esitetty Floodlight-kontrollerin yksinkertainen topologianäkymä, joka ei sisällä verkkokomponenttien siirtämisen lisäksi muita toiminnallisuuksia. Reaktiivinen pakettien välitys on oletuksena päällä.



KUVIO 30 Floodlight-kontrollerin topologianäkymä

Kontrolleri sisältää vakiona REST-api -pohjaisen palomuurisovelluksen, jota hallitaan joko graafisesta käyttöliittymästä tai komentoriviltä Curl-työkalun avulla. Palomuuri-toiminnallisuus on oletuksena pois käytöstä ja näin ollen kaikki liikenne sallitaan verkkolaitteiden välillä. Kun palomuuri kytketään aktiiviseksi, reaktiivinen pakettien välitys kytkeytyy pois päältä ja sallivat palomuurisäännöt luodaan manuaalisesti, jotta pakettien välittäminen onnistuu (Zope, N., Pawar, S., & Saquib, Z. 2016).

Palomuurisääntöjen luontia testattiin ensin graafisesta käyttöliittymästä. Sääntöjen luonti tapahtui *Add New Rule* -painikkeella. Aluksi lisättiin ARP-protokollaan liittyvät sääntö kaikille kytkimille, jotta ne kykenevät liikennöimään OSI L3 -liikennettä. Testausta varten kytkimelle s3 tehtiin säännöt TCP

443 -porttia varten. Säännöt piti tehdä molempiin suuntiin, jotta myös paluuliikenne sallitaan. Kuviossa 31 on esitetty graafisen käyttöliittymän palomuurinäkymä sekä siihen lisätyt vuosäännöt. Säännöissä olevat nollat tarkoittavat, että parametrille sallitaan kaikki arvot. Esimerkiksi ensimmäinen sääntö on tehty kytkimelle *00:00:00:00:00:00:00:00*, joka tarkoittaa, että sääntö lisätään kaikille topologian kytkimille. Alimmat kaksi sääntöä koskevat TCP 433 -porttiin kohdistuvaa liikennettä ja siihen liittyviä paluupaketteja.

KUVIO 31 Palomuurisäännöt Floodlight-kontrollerilla

Seuraavaksi palomuurisääntöjen luontia testattiin komentorivin ja Curlin avulla. Aikaisemmin tehdyt säännöt poistettiin ja ensimmäisenä tarkistettiin palomuurin tila. Esimerkissä näkyy, että palomuuri on käytössä (*firewall enabled*).

curl http://192.168.1.134:8080/wm/firewall/module/status/json
{"result" : "firewall enabled"}

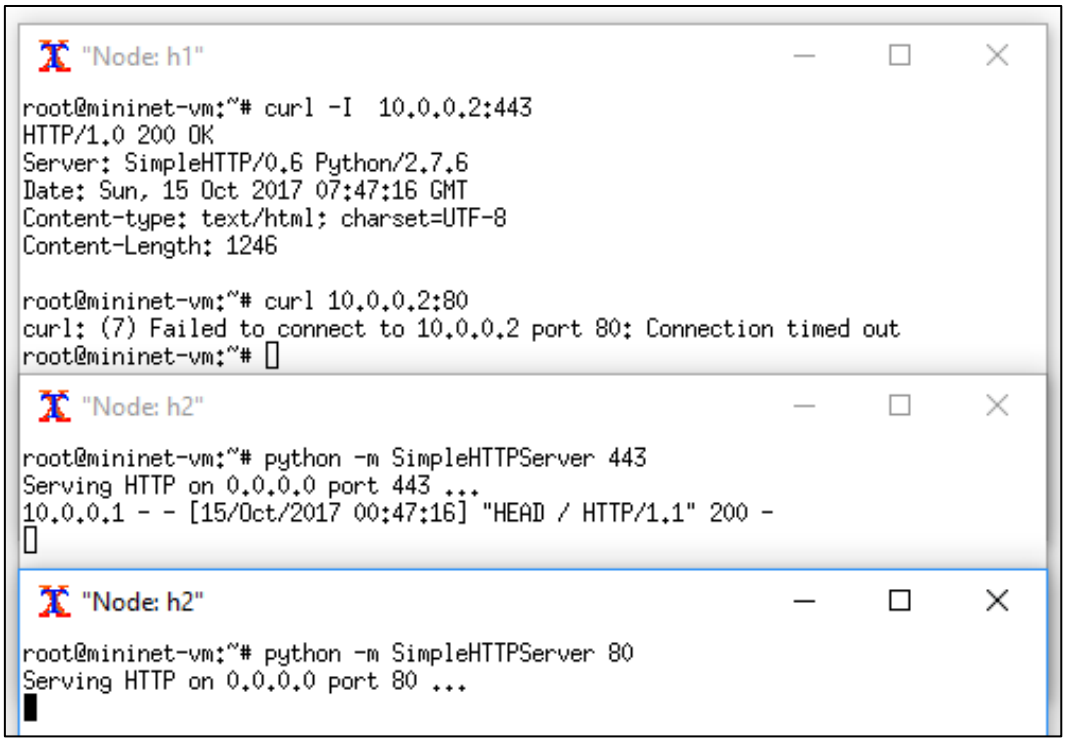
Sääntöjen luonti tapahtuu Curlilla tehtävän HTTP-post -metodin avulla. Esimerkissä luodaan ensin yleinen ARP-sääntö ja alimmaisilla riveillä TCP 443 -sääntö kytkimelle s3. Kontrolleri ilmoittaa Rule Added -ilmoituksella, jos säännön luonti onnistuu ja ilmoittaa samalla säännölle tunnisteen (id). Esimerkiksi viimeisessä säännössä palautettiin *{"status" : "Rule added", "rule-id" : "-1727927717"}.*

curl -X POST -d '{"dl-type":"ARP", "action":"ALLOW"}' http://192.168.1.134:8080/wm/firewall/rules /json

curl -X POST -d '{"dpid":"00:00:00:00:00:00:00:03", "src-ip": "10.0.0.1/32", "dst-ip": "10.0.0.2/32", "nw-proto":"TCP" "tp-dst":"443", "action":"ALLOW"}' http://192.168.1.134:8080/wm/firewall/rules/json
curl -X POST -d '{"dpid":"00:00:00:00:00:00:00:03", "src-ip": "10.0.0.2/32", "dst-ip": "10.0.0.1/32", "nw-proto":"TCP" "tp-src":"443", "action":"ALLOW"}' http://192.168.1.134:8080/wm/firewall/rules/json

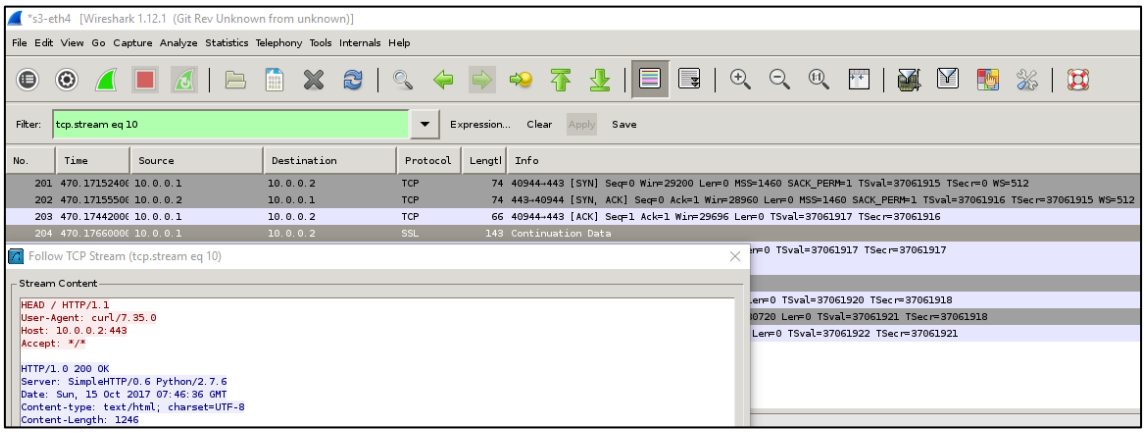
Tehtyjä palomuurisääntöjä pääsi tarkastelemaan *curl http://192.168.1.134:8080/wm/firewall/rules/json* -komennolla komentoriviltä, mutta selkeämmin ne sai näkyville web-selaimella samasta osoitteesta.

Sääntöjen toimivuus voitiin testata myös Curlin avulla (Kuvio 32). Esimerkissä palvelimena toimivalle h2:lle käynnistettiin HTTP-palvelimet portteihin TCP 80 ja TCP 443. Alla olevassa kuviossa on ylimmässä terminaalissa lähetetty HTTP-head -pyynnöt h2-palvelimelle portteihin 443 ja 80. 443-porttiin tehty pyyntö menee perille ja keskimmäisestä terminaali-ikkunasta voidaan havaita onnistunut http-head-vastaus (200). Sen sijaan yhteydenotto porttiin 80 päätyy virheeseen (*Connection timed out*) ja alimmassa ikkunassa olevalle palvelimelle ei pyyntö tule perille johtuen siitä, että sallivaa palomuurisääntöä porttiin 80 ei ole tehty.



KUVIO 32 Floodlight-palomuurin todennus

TCP 443 -yhteyden muodostuminen todennettiin lisäksi Wiresharkin avulla. Alla olevassa kuviossa 33 on kolmella ensimmäisellä rivillä esitetty TCP-kättely (SYN, SYN-ACK, ACK) päätelaitteen ja palvelimen välillä. Neljännellä rivillä Wireshark on tulkinnut automaattisesti yhteyden SSL (Secure Sockets Layer) -verkkoliikenteeksi. Koska yhteys ei todellisuudessa kuitenkaan ollut salattu, voitiin lähetettyä HTTP-header -pyyntöä tarkastella Wiresharkin *Follow TCP stream* -toiminnallisuuden avulla selkokielisenä.



KUVIO 33 TCP 433 -yhteyden todennus

Edellä esitettyjen toteutuksien lisäksi ympäristön toimivuutta testattiin samalla Aruba VAN SDN -kontrollerilla ja Zodiac FX:llä sekä python-ohjelmointikielellä kirjoitetulla POX-kontrollerilla. Aruban ja mininetin tapauksessa samoista Flow Maker -sovelluksen rajoittuneisuudesta johtuen tämä toteutus ei olisi tuonut tutkimukseen lisäarvoa. POX-kontrollerin ja Mininetin tapauksessa testauksia ei päätetty jatkaa pidemmälle, koska kyseinen kontrolleri tuki ainoastaan OpenFlow 1.0 versiota, jonka osumakenttärakenne on huomattavasti suppeampi muissa toteutuksissa käytettyyn 1.3.0 versioon verrattuna. Lisäksi kontrolleriin liittyvä dokumentaatio oli muita suppeampi. Toteutuksissa alkoi toistua hyvin pitkälle samat rajoitukset, joten sen perusteella voisi todeta, että tutkimuksessa käytetty otanta erilaisten toteutuksien muodostamasta populaatiosta oli tarpeeksi kattava (Shieha, A. 2014).

5.10 Palomuurisäännöstön jatkokehitys

Tutkielman yhteydessä luotua palomuurisäännöstöä voidaan hyödyntää useaan eri jatkokehitystapaukseen. Säännöstöjen perusteella voidaan esimerkiksi kerätä lokitietoja ja asettaa ne generoimaan hälytyksiä, kun verkkoympäristössä havaitaan poikkeamia. Poikkeamiin liittyvät suojautumistoimenpiteet on myös mahdollista automatisoida, jolloin SDN-kontrolleri kykenee asettamaan tarvittavat palomuurisääntöjen avulla tehtävät tietoturvakontrollit automaattisesti ilman ylläpitäjän manuaalisia toimenpiteitä. Toumi, K., Idrees, M. S., Charmet, F., Yaich, R., ja Blanc, G (2017) hyödyntivät OpenFlown tietoturvan kehitykseen tarkoitetun FRESCO-viitekehykseen perustuvia tietoturvamoduuleita, joiden avulla hälytyksien generointi sekä mahdolliset suojautumistoimenpiteet saatiin automatisoitua (Shin, S., Porras, P., Yegneswaran, V., Fong, M., & Gu, G. 2013).

Säännöstöä voitaisiin hyödyntää NFV-pohjaisten palomuuripalveluiden käyttöönotossa. Hu, Z., ja Yin, Y. ovat tutkineet (2017) tällaisten tietoturvapalveluiden käyttöönottoa tietoturvaketjujen (engl. Security Chain) avulla. Heidän

tutkimuksessaan palomuuraukseen liittyviä tietoturvapalveluita voidaan ottaa käyttöön tarpeen tullen dynaamisesti. Tämän avulla operaattorit voisivat jatkossa tarjota tietoturvapalveluita, jotka provisioituisivat dynaamisesti asiakasverkkoon, kun siinä havaitaan jokin uusi riskitekijä tai kun ympäristöön otetaan uusia palveluita käyttöön. Palomuurisäännöstön vuotauluun 1 liittyviä porttimäärityksiä voidaan skaalata erilaisiin ympäristöihin sopiviksi. Sen avulla voidaan estää muun muassa luvattomasti asiakasverkkoon kytketyistä laitteista kohdistuvat hyökkäykset.

Al-Zewairi, M., Suleiman, D., ja Almajali, S. (2017) esittivät tutkimuksessaan SDN-kontrollerin laajentamista SDsec-toiminnallisuudella. Sen avulla voitiin torjua MAC-osoitteiden väärennyksiä (engl. Spoofing) sekä SDN-verkkoon kohdistuvia palvelunestohyökkäyksiä. Chi, Y. on tutkinut (2017) myös IDS/IPS -toiminnallisuuksien sulauttamista SDN-kontrollerille erillisen moduulin avulla. IDS-moduuli pystyi välittämään kontrollerille tiedon havaitusta poikkeamasta, jonka perusteella kontrolleri pystyi asettamaan tarvittavan tietoturvapolitiikan verkon kytkimille dynaamisesti. Tätä toiminnallisuutta hyödyntäen luotuja palomuurisääntöjä voidaan automaattisesti kytkeä verkkoon ja näin ollen estää haitallista verkkoliikennettä.

Yakasai ja Guy (2017) ovat tutkineet päätelaitteiden vahvaa tunnistautumista SDN-verkossa. Tunnistautuminen tehtiin 802.1X-protokollan eli porttikohtaisen tunnistaumisen avulla, jonka yhteydessä palomuurisäännöstöt voitiin pakottaa päätelaitteille. Jatkokehityksenä tämän tutkielman puitteissa tehtyä palomuurisäännöstöä voidaan hyödyntää myös porttikohtaisessa tunnistautumisessa ja näin ollen palomuurisäännöt voidaan räätälöidä päätelaite- tai profiilikohtaisesti Windows Active Directory -ryhmiin perustuen (Yakasai, S. T., & Guy, C. G. 2017).

6 TUTKIMUSTULOKSET

Yleisenä huomiona kokeellisesta tutkimuksesta oli, että kaikki kontrollerit tukivat graafista, web-selaimella käytettävää käyttöliittymää (GUI). Myös Zodiac FX-kytkintä voidaan uusimmissa firmware-versioissa hallita graafisen käyttöliittymän avulla. Tällä hetkellä saatavilla olevat kontrollerit ja niiden topologianäkyvät eivät yleisesti tue palomuuraukseen liittyvää toiminnallisuutta muutamaa poikkeusta lukuun ottamatta. Kuitenkin sovelluksia näiden toiminnallisuuksien toteuttamiseen on jossain määrin saatavilla. Sovellukset ovat kuitenkin huonosti skaalautuvia isompiin ympäristöihin. Palomuuraukseen liittyvien vuosääntöjen luonti kontrollerien graafisista käyttöliittymistä tapahtui tässä tutkimuksessa käytetyissä kontrollereissa ainoastaan yksittäisinä. Näin ollen huomattavasti tehokkaampi tapa olisi tuoda säännöt kontrollerille massana ulkoisen sovelluksen tai pienemmissä toteutuksissa komentoriviltä ajettavan skriptin avulla.

Puikkari ja Zodiac FX eivät olleet palomuurauksen kannalta sopiva yhdistelmä. Kytkin tukee ainoastaan puhdasta OpenFlow-kytkentää, joten se tarvitsee ulkoisen kontrollerin, joka kykenee luomaan tarvittavat vuosäännöt palomuurausta varten. Puikkarin avulla saatiin reaktiivisesti luotua vuosäännöt SDN-verkkoon, mutta palomuuraukseen liittyvien vuosääntöjen luominen sen sijaan ei onnistunut. Tätä varten jatkossa tulisi kehittää Puikkarin graafisen käyttöliittymän toiminnallisuutta. Puikkarille tulisi lisätä mahdollisuus luoda vuosääntöjä suoraan topologianäkymästä eri parametrien avulla. Lisäksi valmiin vuosäännöstön tuominen ulkopuolisesta tiedostosta olisi hyvä ratkaisu. Pelkkä ulkoinen python-skripti ei ole hallittavuuden kannalta toimiva ratkaisu, etenkään operaattoriympäristöihin. Tätä toteutusta ei juurikaan voi verrata pelkillä OpenFlow-vuosäännöillä tehtävään palomuurisäännöstöön, koska sääntöjen luonti ei ilman erillistä skriptiä tai sovellusta ollut mahdollista.

Aruba VAN SDN Controller ja Zodiac FX -toteutuksen tulokset olivat edellistä parempia. Kontrolleri tuki sääntöjen lisäämistä erillisen Flow Maker -sovelluksen avulla graafisesta käyttöliittymästä. Vuosääntöjen lisääminen tapahtui kuitenkin yksittäin, joka on erittäin työlästä kontrollerin vaatiessa useita eri parametreja jokaiselle vuosäännölle. Tutkimuksessa käytetyssä Trial -

versiossa sääntöjä ei pystynyt tallentamaan. Kontrollerille ei ole tällä hetkellä saatavilla sovellusta, jonka avulla palomuuritoiminnallisuudet voitaisiin toteuttaa vuosääntöjen avulla.

Puikkariin ja Mininet-verkkoemulaattoriin liittyvässä toteutuksessa oli luonnollisesti sama haaste kuin ensimmäisessä toteutuksessa. Sääntöjä ei pystynyt luomaan graafisesta käyttöliittymässä. Se vaatisi käytännössä joko kontrollerin ja topologianäkymän toiminnallisuuksien kehitystä tai ulkoisen sovelluksen. Tässä tapauksessa kuitenkin vuosääntöjä voitiin luoda manuaalisesti Mininet-palvelimen komentoriviltä. Komentoriviltä saadaan lisättyjä sääntöjä massana, joka tukee laajempiin ympäristöihin liittyviä toteutuksia erittäin hyvin. Kontrolleri ei kuitenkaan tue tällä hetkellä palomuuraukseen liittyvää toiminnallisuutta tai sen esittämistä graafisessa käyttöliittymässä. Tässä toteutuksessa päästiin hyvin lähelle puhtaasti OpenFlow:n vuosäännöillä tehtävää toteutusta.

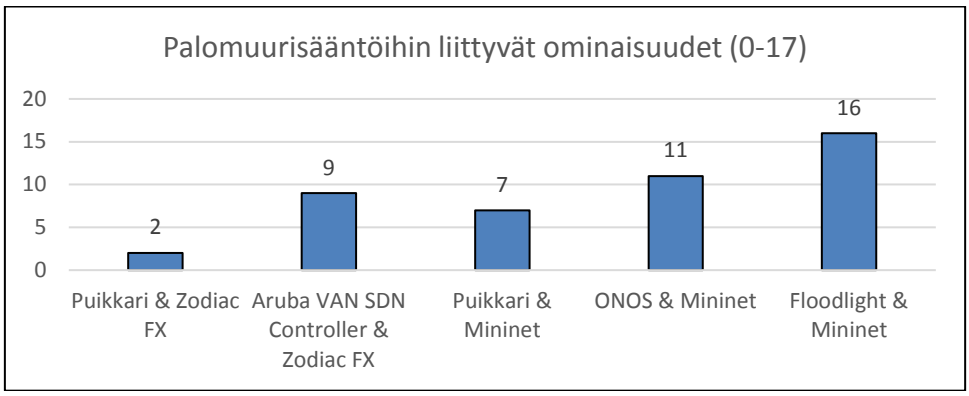
ONOS-toteutuksessa kävi ilmi, että kontrolleri olisi skaalautuvuutensa ja korkean käytettävyyden toteutuksien myötä toimiva vaihtoehto myös operaattoriympäristöihin. Vuosääntöihin perustuvaan palomuuraukseen liittyen se kuitenkin sisälsi useita rajoitteita OpenFlow:n osumakentissä, jotka hankaloittaisivat sen hyödyntämistä. Intentit eivät muun muassa tue natiivisti UDP-protokollaa, joten vuosäännöstö voitiin tehdä vain TCP-protokollaa käyttäville palveluille. Näin ollen toteutus tuki vain osittain vuosäännöillä tehdyn palomuurisäännöstön käyttöönottoa. Graafisesta käyttöliittymästä luotaville säännöille ei voinut määrittää tarkempia parametreja, vaan kontrolleri määritti ne automaattisesti sallien samalla liikenteen kaikkiin portteihin.

Floodlight-kontrolleri ei tue sääntöjen muokkaamista, vaan muokkaaminen vaatii olemassa olevan vuosäännön poistamisen ja uuden säännön luonnin. Vuosääntöjen luonti graafisesta käyttöliittymästä yksitellen on erittäin työlästä ja sitä on vaikea saada automatisoitua. Tämän takia sääntöjen hallinta kannattaa toteuttaa komentoriviltä, jolloin Curl-työkalulla lisättävät vuosäännöt voidaan tarvittaessa automatisoida esimerkiksi Python-skriptien avulla. Sääntöjen monitorointi ja tarkistustoimenpiteet sen sijaan voidaan tehdä graafisesta käyttöliittymästä tai selaimella palomuurin REST-api -osoitteista. Muuttamalla vertailukohteena olevan palomuurisäännöstön Curlille sopivaksi, se saadaan Floodlight-kontrollerilla toteutettua lähes kokonaisuudessaan. Ainoastaan kontrollerin logiikkaan sisältyvä olemassa olevien sääntöjen tarkistus saattaa vaatia sääntöjen parametrien muokkaamista. Floodlightin palomuuri ei esimerkiksi sallinut päätelaitteiden välisiä sääntöjä molempiin suuntiin kohde- ja lähde MAC-osoitteita hyödyntäen. MAC-osoitteita voitiin kuitenkin hyödyntää muun muassa sisäverkossa olevan päätelaitteen ja palvelimen välisessä palomuurisäännöstössä.

Toteutuksiin liittyvät tarkemmat ominaisuudet sekä kyvykkyydet on esitetty taulukossa 4. Siihen on kerätty palomuurauksen toteuttamisen kannalta keskeiset ominaisuudet. Ominaisuudet ovat pisteytetty asteikolle 0-17 ominaisuuksiensa perusteella ja pisteytyksestä on tehty yhteenveto kuvioon 34. Ominaisuuksiin liittyvät pisteet ovat merkittyinä sulkeisiin.

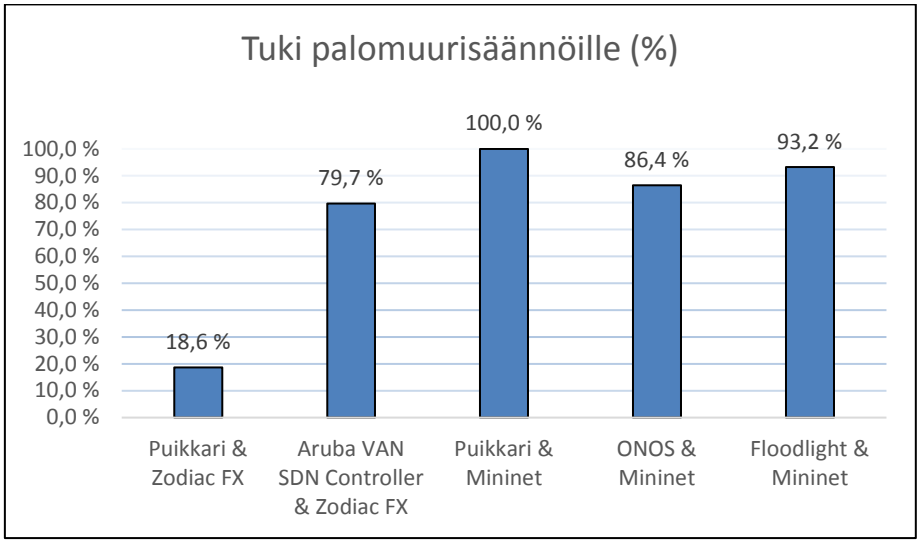
TAULUKKO 4 Yhteenveto toteutuksien ominaisuuksista

	Puikkari & Zodiac FX	Aruba VAN SDN Controller & Zodiac FX	Puikkari & Mininet	ONOS & Mininet	Floodlight & Mininet
Avoin lähdekoodi (0-1)	Kyllä (1)	Ei	Kyllä (1)	Kyllä (1)	Kyllä (1)
Dokumentaatio (0-2)	Heikko (0)	Kohtalainen (1)	Kohtalainen (1)	Kattava (2)	Kattava (2)
Käyttöönoton helppous (0-2)	Monimutkainen (0)	Helppo (2)	Monimutkainen (0)	Keskitasoinen (1)	Helppo (2)
Vuosääntöjen luonti graafisesta käyttöliittymästä (0-2)	Ei mahdollinen (0)	Flow Maker -sovelluksen avulla (1)	Ei mahdollinen (0)	Ainoastaan kaiken sallivat päästä-päähän vuot (1)	Onnistuu yksittäin (1)
Vuosääntöjen luonti komentoriviltä (0-2)	Ei mahdollinen, vaatii ulkoisen sovelluksen (0)	Haastava (1)	Onnistuu helposti OVS-komentojen avulla (2)	Intenttien avulla, ei kata kaikkia protokollia (1)	REST-apin kautta (2)
Vuosäännöstön ylläpidettävyys (0-2)	Ei mahdollinen (0)	Flow Maker -sovelluksen avulla (1)	Manuaalisesti OVS-komentojen avulla (1)	Intenttien avulla, ei kata kaikkia protokollia (1)	Sekä GUI:n että komentoriviltä REST-apin avulla (2)
Palomuuri toiminnallisuus (0-2)	Ei mahdollinen (0)	Ei mahdollinen (0)	Ei mahdollinen (0)	Ei mahdollinen (0)	Palomuuri- sekä pääsylista (ACL) -toiminnallisuudet (2)
Tuki reaktiiviselle ja proaktiiviselle mallille (0-2)	Ei (0)	Molemmat (2)	Vain reaktiivinen (1)	Molemmat. Reaktiivinen fwd-sovelluksen avulla (2)	Molemmat. Proaktiivinen palomuuri-toiminnallisuuden avulla (2)
OpenFlow-tuki (0-2)	Vain 1.3 (1)	1.0 & 1.3 (1)	Vain 1.3 (1)	1.3 versioon asti (2)	1.4 versioon asti (2)

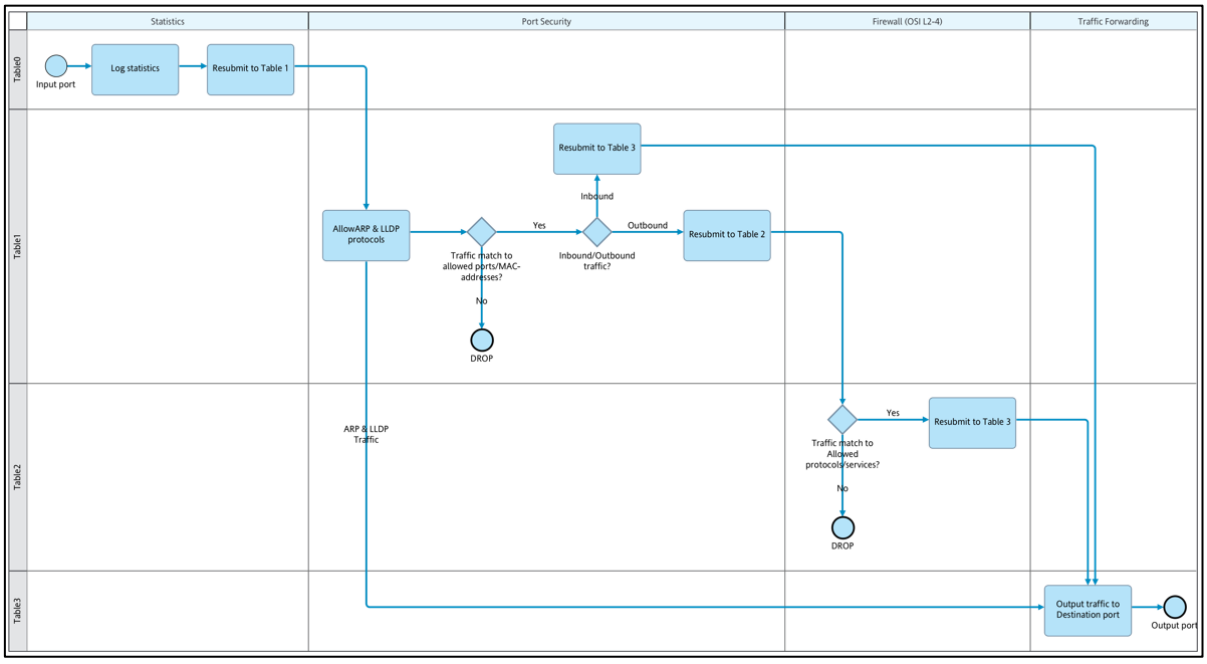


KUVIO 34 Tuki palomuuraukseen liittyville ominaisuuksille

Lopuksi jokaisesta toteutuksesta tulostettiin *ovs-ofctl dump-flows* -komennon avulla s3-kytkimen vuotaulut ja verrattiin niitä tutkielman yhteydessä tehtyyn vuosäännöstöön. Säännöstössä oli yhteensä 59 vuosääntöä. Koska kahdessa ensimmäisessä toteutuksessa ei käytetty Mininet-verkkoemulaattoria, näiden tulokset konvertoitiin bash-skriptien avulla Zodiac FX -kytkimiltä vastaavaan muotoon, jolloin niitä voitiin ajaa Mininet-ympäristössä ja näin ollen tuloksista saatiin vertailukelpoisia. Kuviossa 35 on esitetty prosentuaalinen osuus vuosäännöistä, jotka eri toteutuksissa saatiin luotua. Tarkemmat vuotulosteet toteutuksittain on esitetty liitteessä 5. Tutkielman yhteydessä toteutetun palomuurisäännöstöön liittyvä toimintalogiikka on esitetty kuviossa 36.



KUVIO 35 Prosentuaalinen tuki palomuurisäännöstölle



KUVIO 36 Palomuurisäännöstön toimintalogiikka

7 YHTEENVETO JA POHDINTA

Tässä tutkimuksessa tarkasteltiin SD-WAN -operaattoripalveluiden tietoturvaa ja sen toteuttamista käytännössä. Tutkimus keskittyi vahvasti SDN-arkkitehtuurin välityskerroksella toteutettavaan palomuuraukseen pääasiassa OpenFlow-protokollaa hyödyntäen. Teknisesti palomuuraus toteutettiin OpenFlown vuosääntöjen avulla. Toteutusympäristöön rakennettiin viisi erilaista toteutusta sekä niihin liittyvät operaattorin verkkoa mallintavat topologiat. Tutkimuksessa luotiin kattava OpenFlow-pohjainen palomuurisäännöstö, johon toteutuksien yhteensopivuuksia verrattiin.

OpenFlow-pohjaisella palomuurisäännöstöllä saatiin luotua kattavat tietoturvakontrollit verkkoon. Operaattorin tapauksessa säännöt pitää räätälöidä asiakaskohtaisesti käytettyihin sovelluksiin ja verkon käyttötarkoitukseen perustuen.

Vuosääntöjen rajoituksena on, ettei palomuurausta saa niillä tehokkaasti ulotettua OSI-mallin kerroksia 2-4 ylemmäksi, joten esimerkiksi pakettien sisällön analysoiminen (engl. deep packet inspection) tai vastaavat sovellustason palomuuritoiminnallisuudet eivät vuosääntöjen avulla onnistu. Näissä tapauksissa verkkoliikenne voidaan kuitenkin välittää erilliselle, esimerkiksi ulkoverkon reunalla toimivalle palomuurilaitteelle analysoitavaksi.

Vuosääntöihin perustuva palomuuraus on tehokas keino erityisesti yritysten sisäverkossa verkkosegmenttien ja päätelaitteiden välisessä palomuurauksessa. Sen avulla voidaan estää mahdolliset sisäverkosta kohdistuvat verkkohyökkäykset sekä mahdollinen hyökkääjän pivotointi, eli eteneminen saastuneen päätelaitteen kautta sisäverkossa saastuneiden päätelaitteiden välillä.

Vuosääntöihin perustuvan palomuuritoiminnallisuuden keskittäminen ei ole hyvä ratkaisu, koska silloin kaikki verkkoliikenne täytyy reitittää sen kautta ja samalla siitä muodostuu Single point of Failure -piste verkkoon. Keskitetyssä mallissa myös sääntöjen määrä yhdellä verkkokomponentilla kasvaa isoksi ja vaikeammin hallittavaksi. Lisäksi keskitetty malli aiheuttaa turhaa verkkoliikennettä, kun kaikki liikenne välitetään SDN-verkon läpi kyseiselle palomuurille sen sijaan, että se esimerkiksi estettäisiin lähimmällä kytkimellä. Näin ollen keskitetty malli ei ole optimaalinen myöskään siirtokapasiteetin tehokkaan hyödyntämisen näkökulmasta.

Tämän perustella voidaan todeta, että palomuuraukseen liittyvä toiminnallisuus kannattaa tässä tapauksessa hajauttaa jokaiselle SDN-verkon kytkimelle. Tällöin jokainen kytkin verkossa toimii itsenäisenä palomuurina ja suodattaa verkkoliikenteen mahdollisimman läheltä sen lähettäjää, kuten päätelaitetta tai palvelinta. Hajautetussa mallissa vuosäännöt saadaan määriteltyä helposti tarkalle tasolle vastaamaan verkon tarpeita. SDN-verkon palomuuritoiminnot tulisi erillisen ulkoverkon reunalla toimivien palomuurien lisäksi olla toteutettu hajautetusti sisäverkon kytkimille. Operaattorin tapauksessa SD-WAN -verkon hallinta voidaan keskittää kahdennetuille kontrollereille, jotta jokaista verkkolaitetta ei tarvitsisi konfiguroida erikseen. Palomuuritoiminnot voidaan hajauttaa ulkoisen palomuurin lisäksi asiakaslaitteille. Samalla verkon kapasiteettia saadaan optimoitua. Tämä edellyttää, että vuosäännöt luodaan SDN-verkon kytkimille proaktiivista mallia hyödyntäen, jolloin samalla saadaan parannettua kontrollerin suorituskykyä.

Operaattorin näkökulmasta siirtymävaiheessa MPLS VPN -pohjaisista verkoista on käytettävä ehdottomasti sekä OpenFlow että Ethernet -yhteensopivia hybrid-kytkimiä, jotta kytkimet kykenevät välittämään verkkoliikennettä uuden ja vanhan teknologian välillä. Lisäksi hybridikytkimet kykenevät vuosääntöjen avulla toimimaan itsenäisesti ja välittämään liikennettä esimerkiksi tilanteessa, jossa SDN-kontrolleri ei ole käytettävissä. Lisäksi tällä estetään, ettei kontrolleri muodosta verkkoon pullonkaulaa tilanteissa, joissa verkko on poikkeuksellisen kuormitettu tai kontrolleri olisi mahdollisen hyökkäyksen kohteena.

Jatkokehityksenä tässä tutkimuksessa luotuja palomuurisääntöjä voitaisiin lähteä rakentamaan kontrollereiden logiikkaan tai ulkoiseen sovellukseen. Näin ollen kontrollereista voitaisiin määrittää automaattisesti yleiset palomuurisäännöt SDN-verkon kytkimille proaktiivisen mallin mukaisesti. Kontrollereilla pitäisi olla mahdollisuus olemassa olevien säännöstöjen muokkaamisen, jotta palomuurisäännöt voidaan toteuttaa aina jokaiseen verkkokokonaisuuteen sopivaksi. Tämän avulla operaattori voisi määrittää palomuurisäännöstön asiakaskohtaisesti heidän käyttämiinsä palveluihin perustuen. SD-WAN-toteutuksiin liittyvissä hybridiverkoissa operaattorit voisivat hyödyntää samoja tietoturvapolitiikoita sekä olemassa olevissa MPLS-VPN- pohjaisissa että uusissa SDN- ja NFV -pohjaisissa verkon osa-alueissa.

Tämän tutkimuksen yhteydessä luotua vuosääntöihin pohjautuvaa palomuurisäännöstöä voidaan käyttää pohjana vaatimusmäärittelyille kontrollerin logiikkaa tai ulkoista SDN-verkon palomuuraukseen liittyvää sovellusta kehittäessä. Säännöstö voitaisiin abstraktoida niin, että ylläpitäjän tarvitsee tietää ainoastaan verkkoympäristössään sallitut palvelut ja teknisten yksityiskohtien määrittäminen tapahtuu taustalla. Säännöistä voitaisiin kerätä lokitietoja ja lähettää niitä esimerkiksi SIEM-järjestelmän analysoitavaksi. Tämän avulla voitaisiin parantaa verkon tilannekuvaa ja havaita ympäristössä esiintyviä poikkeamia.

LÄHTEET

Alcorn, J., & Melton, S. (2017). SDN On-The-Go (OTG) Physical Testbed, 202-208.

Al-Zewairi, M., Suleiman, D., & Almajali, S. (2017). An experimental Software Defined Security controller for Software Defined Network. 2017 4th International Conference on Software Defined Systems, SDS 2017, 32–36. https://doi.org/10.1109/SDS.2017.7939137

Antikainen, M. (2016). Security in Emerging Networking Technologies: Views on Internet of Things and Software-Defined Networking.

Bidaj, A. (2016). Security Testing SDN Controllers.

Cantrell, C., Henmi, A., Lucas, M. & Singh, A. (2006). Firewall policies and VPN configurations, s.92-93.

Chapman, C. (2016). Network Performance and Security: Testing and Analyzing Using Open Source and Low-Cost Tools.

Chi, Y. (2017). Design and Implementation of Cloud Platform Intrusion Prevention System based on SDN, 847–852.

Chou, L. Der, Tseng, C. W., Huang, Y. K., Chen, K. C., Ou, T. F., & Yen, C. K. (2016). A Security Service on-demand Architecture in SDN. 2016 International Conference on Information and Communication Technology Convergence, ICTC 2016, 287-291. https://doi.org/10.1109/ICTC.2016.7763487

Cisco Systems. (2015). Haettu 01.03.2017 osoitteesta http://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-50/134-bgp.html

CyberTrust. (2017). Securing Platforms and Networks. Haettu 12.10.2017 osoitteesta http://cybertrust.fi/research-themes/securing-platforms-and-networks/

De Oliveira, R. L. S., Schweitzer, C. M., Shinoda, A. A., & Prete, L. R. (2014). Using Mininet for emulation and prototyping Software-Defined Networks. 2014 IEEE Colombian Conference on Communications and Computing, COLCOM 2014 - Conference Proceedings. https://doi.org/10.1109/ColComCon.2014.6860404

ENISA. (2016). Threat Landscape Report 2016.

Feghali, A., Kilany, R., & Chamoun, M. (2015). SDN security problems and solutions analysis. International Conference on Protocol Engineering, ICPE 2015 and International Conference on New Technologies of Distributed Systems, NTDS 2015 - Proceedings. https://doi.org/10.1109/NOTERE.2015.7293514

Feng, T., & Bi, J. (2015). OpenRouteFlow: Enable legacy router as a software-defined routing service for hybrid SDN. Proceedings - International Conference on Computer Communications and Networks, ICCCN, 2015-Octob. https://doi.org/10.1109/ICCCN.2015.7288441

Fernandez, M. P. (2013). Comparing OpenFlow controller paradigms scalability: Reactive and proactive. Proceedings - International Conference on Advanced Information Networking and Applications, AINA, 1009-1016. https://doi.org/10.1109/AINA.2013.113

Gartner. Market Guide for SD-WAN. (2016).

Gartner. Predicting SD-WAN Adoption. (2015). Haettu 06.02.2017 osoitteesta http://blogs.gartner.com/andrew-lerner/2015/12/15/predicting-sd-wan-adoption/.

Gray, K. & Nadeau, T. 2016. Network Function Virtualization.

Göransson, P., Black, C. & Culver, T. (2016). Software Defined Networks: A Comprehensive Approach, Second Edition.

Han, Y., Li, J., Hoang, D., Yoo, J. H., & Hong, J. W. K. (2017). An intent-based network virtualization platform for SDN. 2016 12th International Conference on Network and Service Management, CNSM 2016 and Workshops, 3rd International Workshop on Management of SDN and NFV, ManSDN/NFV 2016, and International Workshop on Green ICT and Smart Networking, GISN 2016, 353-358. https://doi.org/10.1109/CNSM.2016.7818446

Hewlett Parkard. (2017). Aruba VAN SDN Controller Software. Haettu 12.10.2017 osoitteesta https://www.hpe.com/us/en/product-catalog/networking/networking-software/pip.hpe-van-sdn-controller-software.5443866.html

Hu, Z., Wang, M., Yan, X., Yin, Y., & Luo, Z. (2015). A comprehensive security architecture for SDN. 2015 18th International Conference on Intelligence in Next Generation Networks, ICIN 2015, 30-37. https://doi.org/10.1109/ICIN.2015.7073803.

Hu, Z., & Yin, Y. (2017). A framework for security on demand. 2017 13th International Wireless Communications and Mobile Computing Conference, IWCMC 2017, 378–383. https://doi.org/10.1109/IWCMC.2017.7986316

Hussein, A., Elhajj, I. H., Chehab, A., & Kayssi, A. (2016). SDN security plane: An architecture for resilient security services. Proceedings - 2016 IEEE International Conference on Cloud Engineering Workshops, IC2EW 2016, 54-59. https://doi.org/10.1109/IC2EW.2016.15

IANA. (2017). Service Name and Transport Protocol Port Number Registry. Haettu 05.10.2017 osoitteesta: https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml

Ilyadis, N. (2014). 5 NFV Benefits & The Trends Driving Them. Haettu 24.03.2017 osoitteesta http://www.networkcomputing.com/networking/5-nfv-benefits-trends-driving-them/1187036662

International Data Corporation (IDC). (2016). SD-WAN: Guidance on WAN Transformation.

Kandoi, R. (2015). Deploying Software-Defined Networks: A Telco Perspective. Master"s Thesis (Master"s Degree Programme in Security and Mobile Computing - School of Science), Aalto University, 77.

Kaur, K., Singh, J., Kumar, K., & Ghumman, N. S. (2015). Programmable Firewall Using Software Defined Networking. IEEE INDIACom, 5-9.

Kulmala, M. (2016). Improving network security with software-defined networking.

Lorenz, C., Hock, D., Scherer, J., Durner, R., Kellerer, W., Gebert, S., … Tran-Gia, P. (2017). An SDN/NFV-Enabled Enterprise Network Architecture Offering Fine-Grained Security Policy Enforcement. IEEE Communications Magazine, 55(3), 217–223. https://doi.org/10.1109/MCOM.2017.1600414CM

NetworkWorld. (2012). Haettu 01.03.2017 osoitteesta http://www.networkworld.com/article/2222089/cisco-subnet/a-brief-history-of-the-enterprise-wan.html

NetworkWorld. Gartner predicts: SD-WANs to replace routers, but which SD-WAN is the question (2016). Haettu 06.02.2017 osoitteesta http://www.networkworld.com/article/3142053/lan-wan/gartner-predicts-sd-wans-to-replace-routers-but-which-sd-wan-is-the-question.html.

Northbound Networks. (2017). Zodiac Fx. Haettu 12.10.2017 osoitteesta https://northboundnetworks.com/collections/zodiac-fx/products/zodiac-fx

Northbound Networks. (2017). Zodiac FX User Guide. Haettu 12.10.2017 osoitteesta http://forums.northboundnetworks.com/downloads/zodiac_fx/guides/ZodiacFX_UserGuide_0517.pdf, 5-8.

ONOS Project. (2017). Open Network Operating System (ONOS) Wiki Haettu 12.10.2017 osoitteesta https://wiki.onosproject.org/

Open Networking Foundation. (2014). OpenFlow-enabled SDN and Network Functions Virtualization, s. 44.

Open Networking Foundation. (2012). OpenFlow Switch Specification. Version 1.3.0 (Wire Protocol 0x04). Haettu 08.10.2017 osoitteesta https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf. s. 22-24.

Open Networking Foundation. (2014). OpenFlow Switch Specification. Version 1.5.0 (Protocol version 0x06). Haettu 03.04.2017 osoitteesta https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf.

Pfarr, B., Pettit, J. & Tourrilhes, J. (1990). Open vSwitch Manual. ovs−fields(7). Haettu 08.10.2017 osoitteesta http://openvswitch.org/support/dist-docs/ovs-fields.7.pdf

Pfarr, B., Pettit, J. & Tourrilhes, J. (1990). Open vSwitch Manual. ovs-ofctl(8). Haettu 08.10.2017 osoitteesta http://openvswitch.org/support/dist-docs/ovs-ofctl.8.pdf

Orbit Computer. (2015). Haettu 13.03.2017 osoitteesta http://www.orbit-computer-solutions.com/access-control-lists-acls-explained/

Park, H. W., Hwang, I. S., & Lee, J. R. (2016). Study on the sustainable migration to software defined network for nation-wide R&E network. Proceedings - 2016 10th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, IMIS 2016, 392-396. https://doi.org/10.1109/IMIS.2016.117

Rengaraju, P., V, R. R., & Lung, C. (2017.). Detection and Prevention of DoS attacks in Software-Defined Cloud Networks, 217-223.

Pfleeger, P., Pfleeger, S. & Margulies, J. (2015). Security in Computing - 5th edition.

PKC. (2016). SD-WAN for service providers: Threat or opportunity?

Ranjbar, A. (2015). Domain Isolation in a Multi-Tenant Software-Defined Network.

Rajendran, A. (2016). Security Analysis of a Software Defined Wide Area Network Solution.

Reyes, G. P., Dammers, M., & Kastanja, M. (2014). Security assessment on a VXLAN-based network. Haettu 10.03.2017 osoitteesta http://www.delaat.net/rp/2013-2014/p57/report.pdf

Saaranen-Kauppinen, A. & Puusniekka, A. (2016). Haettu 06.02.2017 osoitteesta: http://www.fsd.uta.fi/menetelmaopetus/kvali/L1.html.

Salisbury, B. (2012). Configuring VXLAN and GRE Tunnels on OpenvSwitch. Haettu 06.02.2017 osoitteesta http://networkstatic.net/configuring-vxlan-and-gre-tunnels-on-openvswitch/.

Saxena, M., & Kumar, R. (2016). A recent trends in software defined networking (SDN) security. 2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom), 851-855.

Sdxcentral. (2016). Haettu 27.03.2017 osoitteesta https://www.sdxcentral.com/security/definitions/what-is-software-defined-security/

Shin, S., Porras, P., Yegneswaran, V., Fong, M., & Gu, G. (2013). FRESCO: Modular Composable Security Services for Software-Defined Networks. Network and Distributed System Security Symposium (NDSS 2013), 2(February), 1–16. https://doi.org/10.1.1.297.7129

Shin, S., Xu, L., Hong, S., & Gu, G. (2016). Enhancing Network Security through Software Defined Networking (SDN). 2016 25th International Conference on Computer Communication and Networks (ICCCN), 1-9. https://doi.org/10.1109/ICCCN.2016.7568520

Shieha, A. (2014). Application Layer Firewall Using OpenFlow. Interdisciplinary Telecommunications Graduate Theses & Dissertations. Haettu 12.10.2017 osoitteesta http://scholar.colorado.edu/tlen_gradetds/1, s.23.

Shinder, T. & Behrens, T. (2011). The Best Damn Firewall Book Period, s. 390-391, 776-777.

Solution Reservoir. (2016). Haettu 01.03.2017 osoitteesta http://solutionsreservoir.com/resources/sdn-sd-wan/tutorial-introduction-sd-wan/

Taha, A. (2014). Software-Defined Networking and its Security.

TechTarget (2017). Haettu 01.03.2017 osoitteesta http://searchsecurity.techtarget.com/definition/firewall

Toumi, K., Idrees, M. S., Charmet, F., Yaich, R., & Blanc, G. (2017). Usage Control Policy Enforcement in SDN-Based Clouds: A Dynamic Availability Service Use Case. Proceedings - 18th IEEE International Conference on High Performance Computing and Communications, 14th IEEE International Conference on Smart City and 2nd IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2016, 578–585. https://doi.org/10.1109/HPCC-SmartCity-DSS.2016.0087

Vajaranta, M., Kannisto, J., & Harju, J. (2016). Implementation Experiences and Design Challenges for Resilient SDN Based Secure WAN Overlays. Proceedings - 11th Asia Joint Conference on Information Security, AsiaJCIS 2016, 17-23. https://doi.org/10.1109/AsiaJCIS.2016.25

Yakasai, S. T., & Guy, C. G. (2017). Towards Policy Unification for Enterprise Network Security.

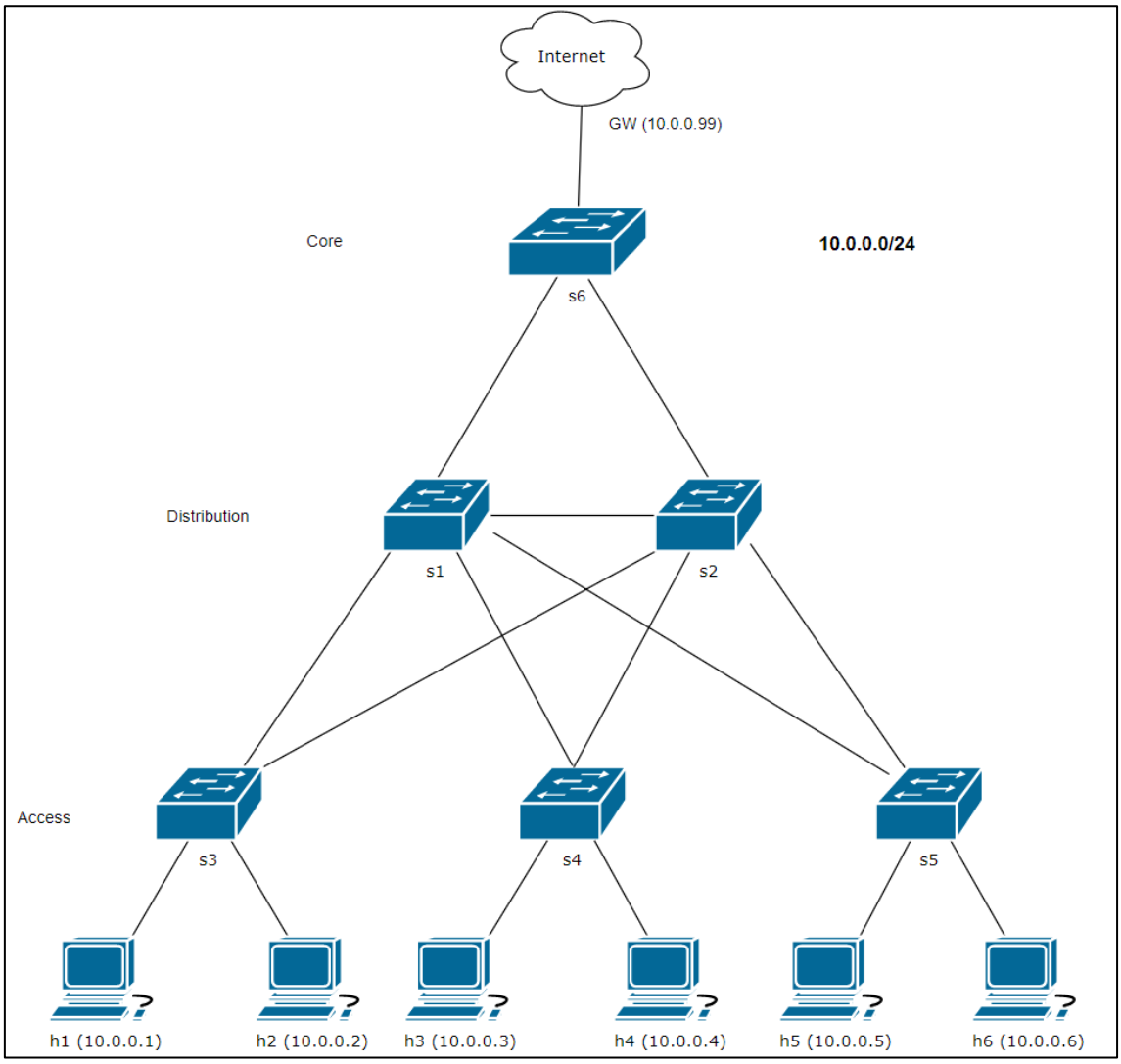
Zope, N., Pawar, S., & Saquib, Z. (2016). Firewall and load balancing as an application of SDN. Conference on Advances in Signal Processing, CASP 2016, 354-359. https://doi.org/10.1109/CASP.2016.7746195

LIITE 1 PYTHON-SKRIPTI MININET-TOPOLOGIAA VARTEN

```
#!/usr/bin/python

from mininet.net import Mininet
from mininet.node import Controller, RemoteController
from mininet.cli import CLI
from mininet.log import setLogLevel, info

def emptyNet():

   "Network topology with Access, Distribution and access layers"

   net = Mininet( controller=None )

#    info( '*** Configuring controller\n' )
   Controller_IP='192.168.1.133'
   Ctrl = net.addController( 'c0', controller=RemoteController, ip=Controller_IP,
port=6633)

#    net.addController( 'c0' )

   info( '*** Creating hosts\n' )
   h1 = net.addHost( 'h1', ip='10.0.0.1', mac='00:00:00:00:00:11' )
   h2 = net.addHost( 'h2', ip='10.0.0.2', mac='00:00:00:00:00:22' )
   h3 = net.addHost( 'h3', ip='10.0.0.3', mac='00:00:00:00:00:33' )
   h4 = net.addHost( 'h4', ip='10.0.0.4', mac='00:00:00:00:00:44' )
   h5 = net.addHost( 'h5', ip='10.0.0.5', mac='00:00:00:00:00:55' )
   h6 = net.addHost( 'h6', ip='10.0.0.6', mac='00:00:00:00:00:66' )
   gw = net.addHost( 'gw', ip='10.0.0.99', mac='00:00:00:00:00:99' )

   info( '*** Creating switches\n' )
   s1 = net.addSwitch( 's1' )
   s2 = net.addSwitch( 's2' )
   s3 = net.addSwitch( 's3' )
   s4 = net.addSwitch( 's4' )
   s5 = net.addSwitch( 's5' )
   s6 = net.addSwitch( 's6' )


   info( '*** Creating links\n' )

#Core Network
```

```
    net.addLink ( s1, s2 )

#Distribution Network
    net.addLink ( s1, s3 )
    net.addLink ( s1, s4 )
    net.addLink ( s1, s5 )
    net.addLink ( s1, s6 )

    net.addLink ( s2, s3 )
    net.addLink ( s2, s4 )
    net.addLink ( s2, s5 )
    net.addLink ( s2, s6 )

#Access Network
    net.addLink( s3, h1 )
    net.addLink( s3, h2 )
    net.addLink( s4, h3 )
    net.addLink( s4, h4 )
    net.addLink( s5, h5 )
    net.addLink( s5, h6 )
    net.addLink( s6, gw)

    info( '*** Starting network\n')
    net.start()

    info( '*** Running CLI\n' )
    CLI( net )

    info( '*** Stopping network' )
    net.stop()

if __name__ == '__main__':
    setLogLevel( 'info' )
    emptyNet()
```

LIITE 2 TOTEUTUSYMPÄRISTÖN VERKKOTOPOLOGIA



LIITE 3 VUOSÄÄNNÖSTÖ PÄÄTELAITEKYTKIMELLE (S3)

```
# Poistetaan aikaisemmat vuosäännöt
sudo ovs-ofctl del-flows s3

## Vuotaulu 0
# Käytetään kytkimen statistiikkaa varten, ohjataan kaikki vuotauluun 1
sudo ovs-ofctl add-flow s3 "table=0, priority=10, actions=resubmit(,1)"

## Vuotaulu 1 - Kytkimelle ulkoverkosta tuleva liikenne (Inbound)
# Kytkimet - Sallitaan liikenne sisään ainoastaan kytkimistä s1 ja s2
sudo ovs-ofctl add-flow s3 "table=1, priority=1,
in_port=1,dl_src=62:94:05:ff:26:d4,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=1, priority=1,
in_port=2,dl_src=3e:60:5f:91:de:10,actions=resubmit(,3)"

# Päätelaitteet - Sallitaan liikenne sisään ainoastaan h1 ja h2 MAC-osoitteista
sudo ovs-ofctl add-flow s3 "table=1, priority=1,
in_port=3,dl_src=00:00:00:00:00:11,actions=resubmit(,2)"
sudo ovs-ofctl add-flow s3 "table=1, priority=1,
in_port=4,dl_src=00:00:00:00:00:22,actions=resubmit(,2)"

# Sallitaan ARP
sudo ovs-ofctl add-flow s3 "table=1, priority=1,dl_type=0x0806,actions=resubmit(,3)"

# Sallitaan LLDP kytkimien porteista
sudo ovs-ofctl add-flow s3 "table=1, priority=5,
in_port=1,dl_type=0x88cc,action=CONTROLLER:65509"
sudo ovs-ofctl add-flow s3 "table=1, priority=5,
in_port=2,dl_type=0x88cc,action=CONTROLLER:65509"

# Estetään kaikki vuotaulu 1:n sääntöihin osumaton liikenne
sudo ovs-ofctl add-flow s3 "table=1, priority=0, actions=drop"

## Vuotaulu 2 - Päätelaitteilta verkkoon suuntautuva liikenne (Outbound)
# Sallitaan PING (ICMP echo)
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=1,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=1,actions=resubmit(,3)"

# Sallitaan FTP (TCP 21)
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=6,tcp_dst=21,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=6,tcp_dst=21,actions=resubmit(,3)"
```

```
# Sallitaan SSH (TCP 22)
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=6,tcp_dst=22,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=6,tcp_dst=22,actions=resubmit(,3)"

# Sallitaan SMTP (TCP 25)
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=6,tcp_dst=25,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=6,tcp_dst=25,actions=resubmit(,3)"

# Sallitaan DNS (UDP 53)
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=17,udp_dst=53,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=17,udp_dst=53,actions=resubmit(,3)"

# Sallitaan DHCP (UDP 67)
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=17,udp_dst=67,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=17,udp_dst=67,actions=resubmit(,3)"

# Sallitaan HTTP (TCP 80) ja HTTPS (TCP 443)
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=6,nw_proto=6,tcp_dst=80,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=6,nw_proto=6,tcp_dst=443,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=6,nw_proto=6,tcp_dst=80,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=6,nw_proto=6,tcp_dst=443,actions=resubmit(,3)"

# Sallitaan Kerberos (TCP 88)
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=6,tcp_dst=88,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=6,tcp_dst=88,actions=resubmit(,3)"

# Sallitaan POP3 (TCP 110)
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=6,tcp_dst=110,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=6,tcp_dst=110,actions=resubmit(,3)"

# Sallitaan NTP (UDP 123)
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=17,udp_dst=123,actions=resubmit(,3)"
```

```
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=17,udp_dst=123,actions=resubmit(,3)"

# Sallitaan MSrpc (TCP 135)
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=6,tcp_dst=135,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=6,tcp_dst=135,actions=resubmit(,3)"

# Sallitaan NetBIOS (TCP 139)
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=6,tcp_dst=139,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=6,tcp_dst=139,actions=resubmit(,3)"

# Sallitaan IMAP (UDP 143) and IMAPS (TCP 993)
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=6,tcp_dst=143,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=6,tcp_dst=143,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=6,tcp_dst=993,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=6,tcp_dst=993,actions=resubmit(,3)"

# Sallitaan SMB (TCP 445)
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=6,tcp_dst=445,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=6,tcp_dst=445,actions=resubmit(,3)"

# Sallitaan LPD (TCP 515)
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=6,tcp_dst=515,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=6,tcp_dst=515,actions=resubmit(,3)"

# Sallitaan MySQL (TCP 3306)
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=6,tcp_dst=3306,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=6,tcp_dst=3306,actions=resubmit(,3)"

# Sallitaan RDP (TCP 3389)
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=3,dl_type=0x0800,nw_proto=6,tcp_dst=3389,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1,
in_port=4,dl_type=0x0800,nw_proto=6,tcp_dst=3389,actions=resubmit(,3)"

# Muihin sääntöihin osumaton liikenne ohjataan kontrollerille
```

```
sudo ovs-ofctl add-flow s3 "table=2, priority=1, in_port=4,dl_type=0x0800,nw_proto=6,tcp_dst=3389,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s3 "table=2, priority=1, in_port=3,dl_type=0x0800,nw_proto=6,tcp_dst=3389,actions=resubmit(,3)"

# Estetään kaikki vuotaulu 2:n muihin sääntöihin osumaton liikenne
sudo ovs-ofctl add-flow s3 "table=2, priority=0, actions=drop"

## Vuotaulu 3
# Sallitaan liikenne ulos porteista 1 ja 2. Portti 1 toimii ensisijaisena.
sudo ovs-ofctl add-flow s3 "table=3, priority=10, in_port=3,actions=output:1"
sudo ovs-ofctl add-flow s3 "table=3, priority=5, in_port=3,actions=output:2"
sudo ovs-ofctl add-flow s3 "table=3, priority=10, in_port=4,actions=output:1"
sudo ovs-ofctl add-flow s3 "table=3, priority=5, in_port=4,actions=output:2"

# Sallitaan liikenne sisään kytkimien porteista
sudo ovs-ofctl add-flow s3 "table=3, priority=20, in_port=1,dl_dst=00:00:00:00:00:11,actions=output:3"
sudo ovs-ofctl add-flow s3 "table=3, priority=20, in_port=2,dl_dst=00:00:00:00:00:22,actions=output:4"
sudo ovs-ofctl add-flow s3 "table=3, priority=20, in_port=3,dl_dst=00:00:00:00:00:22,actions=output:4"
sudo ovs-ofctl add-flow s3 "table=3, priority=205, in_port=4,dl_dst=00:00:00:00:00:11,actions=output:3"

# Sallitaan ARP-liikenne
sudo ovs-ofctl add-flow s3 "table=3, priority=10,dl_type=0x0806,actions=output:all"
```

LIITE 4 VUOSÄÄNNÖSTÖ PALVELINKYTKIMELLE (S4)

```
# Poistetaan aikaisemmat vuosäännöt
sudo ovs-ofctl del-flows s4

## Vuotaulu 0
# Käytetään kytkimen statistiikkaa varten, ohjataan kaikki vuotauluun 1
sudo ovs-ofctl add-flow s4 "table=0, priority=10, actions=resubmit(,1)"

## Vuotaulu 1 - Kytkimelle ulkoverkosta tuleva liikenne (Inbound)
# Kytkimet - Sallitaan liikenne sisään ainoastaan kytkimistä s1 ja s2
sudo ovs-ofctl add-flow s4 "table=1, priority=1,
in_port=1,dl_src=62:94:05:ff:26:d4,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s4 "table=1, priority=1,
in_port=2,dl_src=3e:60:5f:91:de:10,actions=resubmit(,3)"

# Päätelaitteet - Sallitaan liikenne sisään ainoastaan h1 ja h2 MAC-osoitteista
sudo ovs-ofctl add-flow s4 "table=1, priority=1,
in_port=3,dl_src=00:00:00:00:00:33,actions=resubmit(,2)"
sudo ovs-ofctl add-flow s4 "table=1, priority=1,
in_port=4,dl_src=00:00:00:00:00:44,actions=resubmit(,2)"

# Sallitaan ARP
sudo ovs-ofctl add-flow s4 "table=1, priority=1,dl_type=0x0806,actions=resubmit(,3)"

# Estetään kaikki vuotaulu 1:n sääntöihin osumaton liikenne
sudo ovs-ofctl add-flow s4 "table=1, priority=0, actions=drop"

## Vuotaulu 2 - Palvelimilta verkkoon suuntautuva liikenne (Outbound)

# Sallitaan DNS (UDP & TCP 53) & Bind (TCP 953) nimipalvelimelle
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=4,dl_type=0x0800,udp_src=53,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=4,dl_type=0x0800,tcp_src=53,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=4,dl_type=0x0800,tcp_src=953,actions=resubmit(,3)"

# Sallitaan HTTP (TCP 80) ja HTTPS (TCP 443) web-palvelimelle
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=3,dl_type=0x0800,tcp_src=80,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=3,dl_type=0x0800,tcp_src=443,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=3,dl_type=0x0800,tcp_src=80,actions=resubmit(,3)"
```

```
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=3,dl_type=0x0800,tcp_src=443,actions=resubmit(,3)"

# Sallitaan SSH (TCP 22)
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=3,dl_type=0x0800,tcp_src=22,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=4,dl_type=0x0800,tcp_src=22,actions=resubmit(,3)"

# Sallitaan sunrpc UDP & TCP 111)
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=3,dl_type=0x0800,tcp_src=111,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=3,dl_type=0x0800,udp_src=111,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=4,dl_type=0x0800,tcp_src=111,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=4,dl_type=0x0800,udp_src=111,actions=resubmit(,3)"

# Sallitaan Rpcbind (UDP 688) nimipalvelimelle
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=4,dl_type=0x0800,udp_src=688,actions=resubmit(,3)"

# Sallitaan NTP (UDP 123)
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=3,dl_type=0x0800,udp_dst=123,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=4,dl_type=0x0800,udp_dst=123,actions=resubmit(,3)"

# Sallitaan MySQL (TCP 3306) web-palvelimelle
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=3,dl_type=0x0800,tcp_dst=3306,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=4,dl_type=0x0800,tcp_dst=3306,actions=resubmit(,3)"

# Sallitaan MSrpc (TCP 135)
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=3,dl_type=0x0800,tcp_src=135,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=4,dl_type=0x0800,tcp_src=135,actions=resubmit(,3)"

# Sallitaan avahi-daemon (UDP 5353 & 56426 & 60684)
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=3,dl_type=0x0800,udp_src=5353,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=3,dl_type=0x0800,udp_src=56426,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=4,dl_type=0x0800,udp_src=5353,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s4 "table=2, priority=1,
in_port=4,dl_type=0x0800,udp_src=60684,actions=resubmit(,3)"
```

```
# Muihin sääntöihin osumaton liikenne ohjataan kontrollerille
sudo ovs-ofctl add-flow s4 "table=2, priority=1, in_port=4,dl_type=0x0800,tcp_dst=3389,actions=resubmit(,3)"
sudo ovs-ofctl add-flow s4 "table=2, priority=1, in_port=3,dl_type=0x0800,tcp_dst=3389,actions=resubmit(,3)"

# Estetään kaikki vuotaulu 2:n muihin sääntöihin osumaton liikenne
sudo ovs-ofctl add-flow s4 "table=2, priority=0, actions=drop"

## Vuotaulu 3
# Sallitaan liikenne ulos porteista 1 ja 2. Portti 1 toimii ensisijaisena.
sudo ovs-ofctl add-flow s4 "table=3, priority=10, in_port=3,actions=output:1"
sudo ovs-ofctl add-flow s4 "table=3, priority=5, in_port=3,actions=output:2"
sudo ovs-ofctl add-flow s4 "table=3, priority=10, in_port=4,actions=output:1"
sudo ovs-ofctl add-flow s4 "table=3, priority=5, in_port=4,actions=output:2"

# Sallitaan ARP-liikenne
sudo ovs-ofctl add-flow s4 "table=3, priority=0, in_port=3,dl_type=0x0806,actions=output:4"
sudo ovs-ofctl add-flow s4 "table=3, priority=0, in_port=4,dl_type=0x0806,actions=output:3"
```

LIITE 5 VUOSÄÄNNÖT TOTEUTUKSITTAIN

Puikkari & Zodiac FX

mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s3
NXST_FLOW reply (xid=0x4):
cookie=0x99861e4bfc6a5fed, duration=719.231s, table=0, n_packets=283, n_bytes=22361, idle_age=1, priority=101,in_port=2,dl_type=0x88cc actions=CONTROLLER:65509
cookie=0xb683e302e954c827, duration=709.709s, table=0, n_packets=286, n_bytes=22594, idle_age=1, priority=101,in_port=1,dl_type=0x88cc actions=CONTROLLER:65509
cookie=0x0, duration=2.836s, table=0, n_packets=2, n_bytes=196, idle_timeout=5, hard_timeout=20, idle_age=2, priority=10,in_port=3,dl_src=00:00:00:00:00:11,dl_dst=00:00:00:00:00:22 actions=output:4
cookie=0x0, duration=2.831s, table=0, n_packets=2, n_bytes=196, idle_timeout=5, hard_timeout=20, idle_age=2, priority=10,in_port=4,dl_src=00:00:00:00:00:22,dl_dst=00:00:00:00:00:11 actions=output:3
cookie=0x0, duration=2.795s, table=0, n_packets=2, n_bytes=196, idle_timeout=5, hard_timeout=20, idle_age=2, priority=10,in_port=3,dl_src=00:00:00:00:00:11,dl_dst=00:00:00:00:00:33 actions=mod_vlan_vid:1,output:1
cookie=0x0, duration=2.743s, table=0, n_packets=2, n_bytes=196, idle_timeout=5, hard_timeout=20, idle_age=2, priority=10,in_port=3,dl_src=00:00:00:00:00:11,dl_dst=00:00:00:00:00:44 actions=mod_vlan_vid:1,output:1
cookie=0x0, duration=2.692s, table=0, n_packets=2, n_bytes=196, idle_timeout=5, hard_timeout=20, idle_age=1, priority=10,in_port=3,dl_src=00:00:00:00:00:11,dl_dst=00:00:00:00:00:55 actions=mod_vlan_vid:1,output:1
cookie=0x0, duration=2.641s, table=0, n_packets=2, n_bytes=196, idle_timeout=5, hard_timeout=20, idle_age=1, priority=10,in_port=3,dl_src=00:00:00:00:00:11,dl_dst=00:00:00:00:00:66 actions=mod_vlan_vid:1,output:1
cookie=0x0, duration=2.593s, table=0, n_packets=2, n_bytes=196, idle_timeout=5, hard_timeout=20, idle_age=1, priority=10,in_port=3,dl_src=00:00:00:00:00:11,dl_dst=00:00:00:00:00:99 actions=mod_vlan_vid:1,output:1
cookie=0x0, duration=2.527s, table=0, n_packets=2, n_bytes=196, idle_timeout=5, hard_timeout=20, idle_age=2, priority=10,in_port=4,dl_src=00:00:00:00:00:22,dl_dst=00:00:00:00:00:33 actions=mod_vlan_vid:1,output:1
cookie=0x0, duration=2.477s, table=0, n_packets=2, n_bytes=196, idle_timeout=5, hard_timeout=20, idle_age=2, priority=10,in_port=4,dl_src=00:00:00:00:00:22,dl_dst=00:00:00:00:00:44 actions=mod_vlan_vid:1,output:1
cookie=0x0, duration=2.427s, table=0, n_packets=2, n_bytes=196, idle_timeout=5, hard_timeout=20, idle_age=1, priority=10,in_port=4,dl_src=00:00:00:00:00:22,dl_dst=00:00:00:00:00:55 actions=mod_vlan_vid:1,output:1
cookie=0x0, duration=2.376s, table=0, n_packets=2, n_bytes=196, idle_timeout=5, hard_timeout=20, idle_age=1, priority=10,in_port=4,dl_src=00:00:00:00:00:22,dl_dst=00:00:00:00:00:66 actions=mod_vlan_vid:1,output:1

cookie=0x0, duration=2.327s, table=0, n_packets=2, n_bytes=196, idle_timeout=5, hard_timeout=20, idle_age=1, priority=10,in_port=4,dl_src=00:00:00:00:00:22,dl_dst=00:00:00:00:00:99 actions=mod_vlan_vid:1,output:1
cookie=0x0, duration=2.785s, table=0, n_packets=2, n_bytes=200, idle_timeout=5, hard_timeout=20, idle_age=2, priority=10,in_port=2,dl_vlan=1,dl_src=00:00:00:00:00:33,dl_dst=00:00:00:00:00:11 actions=strip_vlan,output:3
cookie=0x0, duration=2.733s, table=0, n_packets=2, n_bytes=200, idle_timeout=5, hard_timeout=20, idle_age=2, priority=10,in_port=2,dl_vlan=1,dl_src=00:00:00:00:00:44,dl_dst=00:00:00:00:00:11 actions=strip_vlan,output:3
cookie=0x0, duration=2.683s, table=0, n_packets=2, n_bytes=200, idle_timeout=5, hard_timeout=20, idle_age=1, priority=10,in_port=2,dl_vlan=1,dl_src=00:00:00:00:00:55,dl_dst=00:00:00:00:00:11 actions=strip_vlan,output:3
cookie=0x0, duration=2.633s, table=0, n_packets=2, n_bytes=200, idle_timeout=5, hard_timeout=20, idle_age=1, priority=10,in_port=2,dl_vlan=1,dl_src=00:00:00:00:00:66,dl_dst=00:00:00:00:00:11 actions=strip_vlan,output:3
cookie=0x0, duration=2.583s, table=0, n_packets=2, n_bytes=200, idle_timeout=5, hard_timeout=20, idle_age=1, priority=10,in_port=2,dl_vlan=1,dl_src=00:00:00:00:00:99,dl_dst=00:00:00:00:00:11 actions=strip_vlan,output:3
cookie=0x0, duration=2.517s, table=0, n_packets=2, n_bytes=200, idle_timeout=5, hard_timeout=20, idle_age=2, priority=10,in_port=2,dl_vlan=1,dl_src=00:00:00:00:00:33,dl_dst=00:00:00:00:00:22 actions=strip_vlan,output:4
cookie=0x0, duration=2.467s, table=0, n_packets=2, n_bytes=200, idle_timeout=5, hard_timeout=20, idle_age=2, priority=10,in_port=2,dl_vlan=1,dl_src=00:00:00:00:00:44,dl_dst=00:00:00:00:00:22 actions=strip_vlan,output:4
cookie=0x0, duration=2.417s, table=0, n_packets=2, n_bytes=200, idle_timeout=5, hard_timeout=20, idle_age=1, priority=10,in_port=2,dl_vlan=1,dl_src=00:00:00:00:00:55,dl_dst=00:00:00:00:00:22 actions=strip_vlan,output:4
cookie=0x0, duration=2.367s, table=0, n_packets=2, n_bytes=200, idle_timeout=5, hard_timeout=20, idle_age=1, priority=10,in_port=2,dl_vlan=1,dl_src=00:00:00:00:00:66,dl_dst=00:00:00:00:00:22 actions=strip_vlan,output:4
cookie=0x0, duration=2.318s, table=0, n_packets=2, n_bytes=200, idle_timeout=5, hard_timeout=20, idle_age=1, priority=10,in_port=2,dl_vlan=1,dl_src=00:00:00:00:00:99,dl_dst=00:00:00:00:00:22 actions=strip_vlan,output:4
cookie=0xaa37394cc05725bc, duration=759.697s, table=0, n_packets=25, n_bytes=2394, idle_age=2, priority=1,in_port=3 actions=resubmit(,1)
cookie=0x9facb62fb04972f3, duration=752.796s, table=0, n_packets=25, n_bytes=2338, idle_age=2, priority=1,in_port=4 actions=resubmit(,1)
cookie=0x99861e4bfc6a5fed, duration=719.231s, table=0, n_packets=0, n_bytes=0, idle_age=719, priority=1,in_port=2 actions=resubmit(,1)
cookie=0xb683e302e954c827, duration=709.709s, table=0, n_packets=0, n_bytes=0, idle_age=709, priority=1,in_port=1 actions=resubmit(,1)
cookie=0x0, duration=920.837s, table=0, n_packets=0, n_bytes=0, idle_age=920, priority=0 actions=drop

cookie=0xaa37394cc05725bc, duration=759.697s, table=1, n_packets=25, n_bytes=2394, idle_age=2, priority=1,in_port=3 actions=CONTROLLER:65509
cookie=0x9facb62fb04972f3, duration=752.796s, table=1, n_packets=25, n_bytes=2338, idle_age=2, priority=1,in_port=4 actions=CONTROLLER:65509
cookie=0x99861e4bfc6a5fed, duration=719.231s, table=1, n_packets=0, n_bytes=0, idle_age=719, priority=1,in_port=2 actions=CONTROLLER:65509
cookie=0xb683e302e954c827, duration=709.709s, table=1, n_packets=0, n_bytes=0, idle_age=709, priority=1,in_port=1 actions=CONTROLLER:65509

Aruba VAN SDN Controller & Zodiac FX

mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s3
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=7.097s, table=0, n_packets=6, n_bytes=474, idle_age=1, priority=10 actions=resubmit(,1)
cookie=0x0, duration=7.044s, table=1, n_packets=0, n_bytes=0, idle_age=7, priority=1,arp actions=resubmit(,3)
cookie=0x0, duration=6.991s, table=1, n_packets=6, n_bytes=474, idle_age=1, priority=0 actions=drop
cookie=0x0, duration=6.938s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,icmp,in_port=3 actions=resubmit(,3)
cookie=0x0, duration=6.885s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,icmp,in_port=4 actions=resubmit(,3)
cookie=0x0, duration=6.831s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,tcp,in_port=3,tp_dst=21 actions=resubmit(,3)
cookie=0x0, duration=6.775s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,tcp,in_port=4,tp_dst=21 actions=resubmit(,3)
cookie=0x0, duration=6.724s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,tcp,in_port=3,tp_dst=22 actions=resubmit(,3)
cookie=0x0, duration=6.672s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,tcp,in_port=4,tp_dst=22 actions=resubmit(,3)
cookie=0x0, duration=6.619s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,tcp,in_port=3,tp_dst=25 actions=resubmit(,3)
cookie=0x0, duration=6.567s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,tcp,in_port=4,tp_dst=25 actions=resubmit(,3)
cookie=0x0, duration=6.515s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,udp,in_port=3,tp_dst=53 actions=resubmit(,3)
cookie=0x0, duration=6.462s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,udp,in_port=4,tp_dst=53 actions=resubmit(,3)
cookie=0x0, duration=6.411s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,udp,in_port=3,tp_dst=67 actions=resubmit(,3)
cookie=0x0, duration=6.359s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,udp,in_port=4,tp_dst=67 actions=resubmit(,3)
cookie=0x0, duration=6.307s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,tcp,in_port=3,tp_dst=80 actions=resubmit(,3)
cookie=0x0, duration=6.254s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,tcp,in_port=3,tp_dst=443 actions=resubmit(,3)
cookie=0x0, duration=6.200s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,tcp,in_port=4,tp_dst=80 actions=resubmit(,3)
cookie=0x0, duration=6.146s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,tcp,in_port=4,tp_dst=443 actions=resubmit(,3)
cookie=0x0, duration=6.092s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,tcp,in_port=3,tp_dst=88 actions=resubmit(,3)
cookie=0x0, duration=6.038s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,tcp,in_port=4,tp_dst=88 actions=resubmit(,3)

cookie=0x0, duration=5.980s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=3,tp_dst=110 actions=resubmit(,3)
cookie=0x0, duration=5.928s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=4,tp_dst=110 actions=resubmit(,3)
cookie=0x0, duration=5.876s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,udp,in_port=3,tp_dst=123 actions=resubmit(,3)
cookie=0x0, duration=5.821s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,udp,in_port=4,tp_dst=123 actions=resubmit(,3)
cookie=0x0, duration=5.769s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=3,tp_dst=135 actions=resubmit(,3)
cookie=0x0, duration=5.714s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=4,tp_dst=135 actions=resubmit(,3)
cookie=0x0, duration=5.659s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=3,tp_dst=139 actions=resubmit(,3)
cookie=0x0, duration=5.606s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=4,tp_dst=139 actions=resubmit(,3)
cookie=0x0, duration=5.553s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=3,tp_dst=143 actions=resubmit(,3)
cookie=0x0, duration=5.499s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=4,tp_dst=143 actions=resubmit(,3)
cookie=0x0, duration=5.447s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=3,tp_dst=993 actions=resubmit(,3)
cookie=0x0, duration=5.394s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=4,tp_dst=993 actions=resubmit(,3)
cookie=0x0, duration=5.342s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=3,tp_dst=445 actions=resubmit(,3)
cookie=0x0, duration=5.289s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=4,tp_dst=445 actions=resubmit(,3)
cookie=0x0, duration=5.237s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=3,tp_dst=515 actions=resubmit(,3)
cookie=0x0, duration=5.183s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=4,tp_dst=515 actions=resubmit(,3)
cookie=0x0, duration=5.131s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=3,tp_dst=3306 actions=resubmit(,3)
cookie=0x0, duration=5.078s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=4,tp_dst=3306 actions=resubmit(,3)
cookie=0x0, duration=4.921s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,tcp,in_port=4,tp_dst=3389 actions=resubmit(,3)
cookie=0x0, duration=4.869s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,tcp,in_port=3,tp_dst=3389 actions=resubmit(,3)
cookie=0x0, duration=4.816s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=0 actions=drop
cookie=0x0, duration=4.764s, table=3, n_packets=0, n_bytes=0, idle_age=4, priority=10,in_port=3 actions=output:1
cookie=0x0, duration=4.712s, table=3, n_packets=0, n_bytes=0, idle_age=4, priority=5,in_port=3 actions=output:2
cookie=0x0, duration=4.660s, table=3, n_packets=0, n_bytes=0, idle_age=4, priority=10,in_port=4 actions=output:1
cookie=0x0, duration=4.608s, table=3, n_packets=0, n_bytes=0, idle_age=4, priority=5,in_port=4 actions=output:2
cookie=0x0, duration=4.556s, table=3, n_packets=0, n_bytes=0, idle_age=4, priority=10,arp actions=ALL

Puikkari & Mininet
mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s3

NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=11.277s, table=0, n_packets=11, n_bytes=869, idle_age=0, priority=10 actions=resubmit(,1)
cookie=0x0, duration=10.964s, table=1, n_packets=6, n_bytes=474, idle_age=0, priority=5,in_port=1,dl_type=0x88cc actions=CONTROLLER:65509
cookie=0x0, duration=10.911s, table=1, n_packets=5, n_bytes=395, idle_age=2, priority=5,in_port=2,dl_type=0x88cc actions=CONTROLLER:65509
cookie=0x0, duration=11.226s, table=1, n_packets=0, n_bytes=0, idle_age=11, priority=1,in_port=1,dl_src=62:94:05:ff:26:d4 actions=resubmit(,3)
cookie=0x0, duration=11.174s, table=1, n_packets=0, n_bytes=0, idle_age=11, priority=1,in_port=2,dl_src=3e:60:5f:91:de:10 actions=resubmit(,3)
cookie=0x0, duration=11.121s, table=1, n_packets=0, n_bytes=0, idle_age=11, priority=1,in_port=3,dl_src=00:00:00:00:00:11 actions=resubmit(,2)
cookie=0x0, duration=11.069s, table=1, n_packets=0, n_bytes=0, idle_age=11, priority=1,in_port=4,dl_src=00:00:00:00:00:22 actions=resubmit(,2)
cookie=0x0, duration=11.016s, table=1, n_packets=0, n_bytes=0, idle_age=11, priority=1,arp actions=resubmit(,3)
cookie=0x0, duration=10.860s, table=1, n_packets=0, n_bytes=0, idle_age=10, priority=0 actions=drop
cookie=0x0, duration=10.809s, table=2, n_packets=0, n_bytes=0, idle_age=10, priority=1,icmp,in_port=3 actions=resubmit(,3)
cookie=0x0, duration=10.758s, table=2, n_packets=0, n_bytes=0, idle_age=10, priority=1,icmp,in_port=4 actions=resubmit(,3)
cookie=0x0, duration=10.708s, table=2, n_packets=0, n_bytes=0, idle_age=10, priority=1,tcp,in_port=3,tp_dst=21 actions=resubmit(,3)
cookie=0x0, duration=10.657s, table=2, n_packets=0, n_bytes=0, idle_age=10, priority=1,tcp,in_port=4,tp_dst=21 actions=resubmit(,3)
cookie=0x0, duration=10.601s, table=2, n_packets=0, n_bytes=0, idle_age=10, priority=1,tcp,in_port=3,tp_dst=22 actions=resubmit(,3)
cookie=0x0, duration=10.550s, table=2, n_packets=0, n_bytes=0, idle_age=10, priority=1,tcp,in_port=4,tp_dst=22 actions=resubmit(,3)
cookie=0x0, duration=10.495s, table=2, n_packets=0, n_bytes=0, idle_age=10, priority=1,tcp,in_port=3,tp_dst=25 actions=resubmit(,3)
cookie=0x0, duration=10.442s, table=2, n_packets=0, n_bytes=0, idle_age=10, priority=1,tcp,in_port=4,tp_dst=25 actions=resubmit(,3)
cookie=0x0, duration=10.389s, table=2, n_packets=0, n_bytes=0, idle_age=10, priority=1,udp,in_port=3,tp_dst=53 actions=resubmit(,3)
cookie=0x0, duration=10.337s, table=2, n_packets=0, n_bytes=0, idle_age=10, priority=1,udp,in_port=4,tp_dst=53 actions=resubmit(,3)
cookie=0x0, duration=10.285s, table=2, n_packets=0, n_bytes=0, idle_age=10, priority=1,udp,in_port=3,tp_dst=67 actions=resubmit(,3)
cookie=0x0, duration=10.234s, table=2, n_packets=0, n_bytes=0, idle_age=10, priority=1,udp,in_port=4,tp_dst=67 actions=resubmit(,3)
cookie=0x0, duration=10.182s, table=2, n_packets=0, n_bytes=0, idle_age=10, priority=1,tcp,in_port=3,tp_dst=80 actions=resubmit(,3)
cookie=0x0, duration=10.130s, table=2, n_packets=0, n_bytes=0, idle_age=10, priority=1,tcp,in_port=3,tp_dst=443 actions=resubmit(,3)
cookie=0x0, duration=10.077s, table=2, n_packets=0, n_bytes=0, idle_age=10, priority=1,tcp,in_port=4,tp_dst=80 actions=resubmit(,3)
cookie=0x0, duration=10.024s, table=2, n_packets=0, n_bytes=0, idle_age=10, priority=1,tcp,in_port=4,tp_dst=443 actions=resubmit(,3)
cookie=0x0, duration=9.971s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,tcp,in_port=3,tp_dst=88 actions=resubmit(,3)
cookie=0x0, duration=9.919s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,tcp,in_port=4,tp_dst=88 actions=resubmit(,3)

cookie=0x0, duration=9.867s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,tcp,in_port=3,tp_dst=110 actions=resubmit(,3)
cookie=0x0, duration=9.813s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,tcp,in_port=4,tp_dst=110 actions=resubmit(,3)
cookie=0x0, duration=9.761s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,udp,in_port=3,tp_dst=123 actions=resubmit(,3)
cookie=0x0, duration=9.708s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,udp,in_port=4,tp_dst=123 actions=resubmit(,3)
cookie=0x0, duration=9.657s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,tcp,in_port=3,tp_dst=135 actions=resubmit(,3)
cookie=0x0, duration=9.603s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,tcp,in_port=4,tp_dst=135 actions=resubmit(,3)
cookie=0x0, duration=9.551s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,tcp,in_port=3,tp_dst=139 actions=resubmit(,3)
cookie=0x0, duration=9.499s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,tcp,in_port=4,tp_dst=139 actions=resubmit(,3)
cookie=0x0, duration=9.447s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,tcp,in_port=3,tp_dst=143 actions=resubmit(,3)
cookie=0x0, duration=9.393s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,tcp,in_port=4,tp_dst=143 actions=resubmit(,3)
cookie=0x0, duration=9.341s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,tcp,in_port=3,tp_dst=993 actions=resubmit(,3)
cookie=0x0, duration=9.289s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,tcp,in_port=4,tp_dst=993 actions=resubmit(,3)
cookie=0x0, duration=9.237s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,tcp,in_port=3,tp_dst=445 actions=resubmit(,3)
cookie=0x0, duration=9.184s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,tcp,in_port=4,tp_dst=445 actions=resubmit(,3)
cookie=0x0, duration=9.133s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,tcp,in_port=3,tp_dst=515 actions=resubmit(,3)
cookie=0x0, duration=9.081s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,tcp,in_port=4,tp_dst=515 actions=resubmit(,3)
cookie=0x0, duration=9.030s, table=2, n_packets=0, n_bytes=0, idle_age=9, priority=1,tcp,in_port=3,tp_dst=3306 actions=resubmit(,3)
cookie=0x0, duration=8.977s, table=2, n_packets=0, n_bytes=0, idle_age=8, priority=1,tcp,in_port=4,tp_dst=3306 actions=resubmit(,3)
cookie=0x0, duration=8.821s, table=2, n_packets=0, n_bytes=0, idle_age=8, priority=1,tcp,in_port=4,tp_dst=3389 actions=resubmit(,3)
cookie=0x0, duration=8.769s, table=2, n_packets=0, n_bytes=0, idle_age=8, priority=1,tcp,in_port=3,tp_dst=3389 actions=resubmit(,3)
cookie=0x0, duration=8.716s, table=2, n_packets=0, n_bytes=0, idle_age=8, priority=0 actions=drop
cookie=0x0, duration=8.665s, table=3, n_packets=0, n_bytes=0, idle_age=8, priority=10,in_port=3 actions=output:1
cookie=0x0, duration=8.611s, table=3, n_packets=0, n_bytes=0, idle_age=8, priority=5,in_port=3 actions=output:2
cookie=0x0, duration=8.558s, table=3, n_packets=0, n_bytes=0, idle_age=8, priority=10,in_port=4 actions=output:1
cookie=0x0, duration=8.503s, table=3, n_packets=0, n_bytes=0, idle_age=8, priority=5,in_port=4 actions=output:2
cookie=0x0, duration=8.241s, table=3, n_packets=0, n_bytes=0, idle_age=8, priority=10,arp actions=ALL
cookie=0x0, duration=8.449s, table=3, n_packets=0, n_bytes=0, idle_age=8, priority=5,in_port=1,dl_dst=00:00:00:00:00:11 actions=output:3

cookie=0x0, duration=8.396s, table=3, n_packets=0, n_bytes=0, idle_age=8, priority=5,in_port=2,dl_dst=00:00:00:00:00:22 actions=output:4
cookie=0x0, duration=8.345s, table=3, n_packets=0, n_bytes=0, idle_age=8, priority=5,in_port=3,dl_dst=00:00:00:00:00:22 actions=output:4
cookie=0x0, duration=8.293s, table=3, n_packets=0, n_bytes=0, idle_age=8, priority=5,in_port=4,dl_dst=00:00:00:00:00:11 actions=output:3

ONOS & Mininet

mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s3
NXST_FLOW reply (xid=0x4):
cookie=0x0, duration=9.008s, table=0, n_packets=9, n_bytes=711, idle_age=1, priority=10 actions=resubmit(,1)
cookie=0x0, duration=8.694s, table=1, n_packets=4, n_bytes=316, idle_age=1, priority=5,in_port=1,dl_type=0x88cc actions=CONTROLLER:65509
cookie=0x0, duration=8.642s, table=1, n_packets=4, n_bytes=316, idle_age=1, priority=5,in_port=2,dl_type=0x88cc actions=CONTROLLER:65509
cookie=0x0, duration=8.955s, table=1, n_packets=0, n_bytes=0, idle_age=8, priority=1,in_port=1,dl_src=62:94:05:ff:26:d4 actions=resubmit(,3)
cookie=0x0, duration=8.902s, table=1, n_packets=0, n_bytes=0, idle_age=8, priority=1,in_port=2,dl_src=3e:60:5f:91:de:10 actions=resubmit(,3)
cookie=0x0, duration=8.851s, table=1, n_packets=0, n_bytes=0, idle_age=8, priority=1,in_port=3,dl_src=00:00:00:00:00:11 actions=resubmit(,2)
cookie=0x0, duration=8.799s, table=1, n_packets=0, n_bytes=0, idle_age=8, priority=1,in_port=4,dl_src=00:00:00:00:00:22 actions=resubmit(,2)
cookie=0x0, duration=8.746s, table=1, n_packets=0, n_bytes=0, idle_age=8, priority=1,arp actions=resubmit(,3)
cookie=0x0, duration=8.591s, table=1, n_packets=0, n_bytes=0, idle_age=8, priority=0 actions=drop
cookie=0x0, duration=8.538s, table=2, n_packets=0, n_bytes=0, idle_age=8, priority=1,icmp,in_port=3 actions=resubmit(,3)
cookie=0x0, duration=8.484s, table=2, n_packets=0, n_bytes=0, idle_age=8, priority=1,icmp,in_port=4 actions=resubmit(,3)
cookie=0x0, duration=8.433s, table=2, n_packets=0, n_bytes=0, idle_age=8, priority=1,tcp,in_port=3,tp_dst=21 actions=resubmit(,3)
cookie=0x0, duration=8.381s, table=2, n_packets=0, n_bytes=0, idle_age=8, priority=1,tcp,in_port=4,tp_dst=21 actions=resubmit(,3)
cookie=0x0, duration=8.329s, table=2, n_packets=0, n_bytes=0, idle_age=8, priority=1,tcp,in_port=3,tp_dst=22 actions=resubmit(,3)
cookie=0x0, duration=8.277s, table=2, n_packets=0, n_bytes=0, idle_age=8, priority=1,tcp,in_port=4,tp_dst=22 actions=resubmit(,3)
cookie=0x0, duration=8.223s, table=2, n_packets=0, n_bytes=0, idle_age=8, priority=1,tcp,in_port=3,tp_dst=25 actions=resubmit(,3)
cookie=0x0, duration=8.170s, table=2, n_packets=0, n_bytes=0, idle_age=8, priority=1,tcp,in_port=4,tp_dst=25 actions=resubmit(,3)
cookie=0x0, duration=8.119s, table=2, n_packets=0, n_bytes=0, idle_age=8, priority=1,tcp,in_port=3,tp_dst=80 actions=resubmit(,3)
cookie=0x0, duration=8.066s, table=2, n_packets=0, n_bytes=0, idle_age=8, priority=1,tcp,in_port=3,tp_dst=443 actions=resubmit(,3)
cookie=0x0, duration=8.014s, table=2, n_packets=0, n_bytes=0, idle_age=8, priority=1,tcp,in_port=4,tp_dst=80 actions=resubmit(,3)
cookie=0x0, duration=7.963s, table=2, n_packets=0, n_bytes=0, idle_age=7, priority=1,tcp,in_port=4,tp_dst=443 actions=resubmit(,3)
cookie=0x0, duration=7.910s, table=2, n_packets=0, n_bytes=0, idle_age=7, priority=1,tcp,in_port=3,tp_dst=88 actions=resubmit(,3)

cookie=0x0, duration=7.859s, table=2, n_packets=0, n_bytes=0, idle_age=7, priority=1,tcp,in_port=4,tp_dst=88 actions=resubmit(,3)
cookie=0x0, duration=7.807s, table=2, n_packets=0, n_bytes=0, idle_age=7, priority=1,tcp,in_port=3,tp_dst=110 actions=resubmit(,3)
cookie=0x0, duration=7.755s, table=2, n_packets=0, n_bytes=0, idle_age=7, priority=1,tcp,in_port=4,tp_dst=110 actions=resubmit(,3)
cookie=0x0, duration=7.704s, table=2, n_packets=0, n_bytes=0, idle_age=7, priority=1,tcp,in_port=3,tp_dst=135 actions=resubmit(,3)
cookie=0x0, duration=7.653s, table=2, n_packets=0, n_bytes=0, idle_age=7, priority=1,tcp,in_port=4,tp_dst=135 actions=resubmit(,3)
cookie=0x0, duration=7.598s, table=2, n_packets=0, n_bytes=0, idle_age=7, priority=1,tcp,in_port=3,tp_dst=139 actions=resubmit(,3)
cookie=0x0, duration=7.541s, table=2, n_packets=0, n_bytes=0, idle_age=7, priority=1,tcp,in_port=4,tp_dst=139 actions=resubmit(,3)
cookie=0x0, duration=7.487s, table=2, n_packets=0, n_bytes=0, idle_age=7, priority=1,tcp,in_port=3,tp_dst=445 actions=resubmit(,3)
cookie=0x0, duration=7.435s, table=2, n_packets=0, n_bytes=0, idle_age=7, priority=1,tcp,in_port=4,tp_dst=445 actions=resubmit(,3)
cookie=0x0, duration=7.383s, table=2, n_packets=0, n_bytes=0, idle_age=7, priority=1,tcp,in_port=3,tp_dst=515 actions=resubmit(,3)
cookie=0x0, duration=7.331s, table=2, n_packets=0, n_bytes=0, idle_age=7, priority=1,tcp,in_port=4,tp_dst=515 actions=resubmit(,3)
cookie=0x0, duration=7.279s, table=2, n_packets=0, n_bytes=0, idle_age=7, priority=1,tcp,in_port=3,tp_dst=3306 actions=resubmit(,3)
cookie=0x0, duration=7.226s, table=2, n_packets=0, n_bytes=0, idle_age=7, priority=1,tcp,in_port=4,tp_dst=3306 actions=resubmit(,3)
cookie=0x0, duration=7.070s, table=2, n_packets=0, n_bytes=0, idle_age=7, priority=1,tcp,in_port=4,tp_dst=3389 actions=resubmit(,3)
cookie=0x0, duration=6.941s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=1,tcp,in_port=3,tp_dst=3389 actions=resubmit(,3)
cookie=0x0, duration=6.886s, table=2, n_packets=0, n_bytes=0, idle_age=6, priority=0 actions=drop
cookie=0x0, duration=6.835s, table=3, n_packets=0, n_bytes=0, idle_age=6, priority=10,in_port=3 actions=output:1
cookie=0x0, duration=6.782s, table=3, n_packets=0, n_bytes=0, idle_age=6, priority=5,in_port=3 actions=output:2
cookie=0x0, duration=6.731s, table=3, n_packets=0, n_bytes=0, idle_age=6, priority=10,in_port=4 actions=output:1
cookie=0x0, duration=6.679s, table=3, n_packets=0, n_bytes=0, idle_age=6, priority=5,in_port=4 actions=output:2
cookie=0x0, duration=6.416s, table=3, n_packets=0, n_bytes=0, idle_age=6, priority=10,arp actions=ALL
cookie=0x0, duration=6.626s, table=3, n_packets=0, n_bytes=0, idle_age=6, priority=5,in_port=1,dl_dst=00:00:00:00:00:11 actions=output:3
cookie=0x0, duration=6.574s, table=3, n_packets=0, n_bytes=0, idle_age=6, priority=5,in_port=2,dl_dst=00:00:00:00:00:22 actions=output:4
cookie=0x0, duration=6.521s, table=3, n_packets=0, n_bytes=0, idle_age=6, priority=5,in_port=3,dl_dst=00:00:00:00:00:22 actions=output:4
cookie=0x0, duration=6.469s, table=3, n_packets=0, n_bytes=0, idle_age=6, priority=5,in_port=4,dl_dst=00:00:00:00:00:11 actions=output:3

Floodlight & Mininet

mininet@mininet-vm:~$ sudo ovs-ofctl dump-flows s3
NXST_FLOW reply (xid=0x4):

cookie=0x0, duration=6.009s, table=0, n_packets=6, n_bytes=474, idle_age=0, priority=10 actions=resubmit(,1)
cookie=0x0, duration=5.623s, table=1, n_packets=3, n_bytes=237, idle_age=0, priority=5,in_port=1,dl_type=0x88cc actions=CONTROLLER:65509
cookie=0x0, duration=5.567s, table=1, n_packets=3, n_bytes=237, idle_age=0, priority=5,in_port=2,dl_type=0x88cc actions=CONTROLLER:65509
cookie=0x0, duration=5.891s, table=1, n_packets=0, n_bytes=0, idle_age=5, priority=1,in_port=1,dl_src=62:94:05:ff:26:d4 actions=resubmit(,3)
cookie=0x0, duration=5.833s, table=1, n_packets=0, n_bytes=0, idle_age=5, priority=1,in_port=2,dl_src=3e:60:5f:91:de:10 actions=resubmit(,3)
cookie=0x0, duration=5.780s, table=1, n_packets=0, n_bytes=0, idle_age=5, priority=1,in_port=3,dl_src=00:00:00:00:00:11 actions=resubmit(,2)
cookie=0x0, duration=5.726s, table=1, n_packets=0, n_bytes=0, idle_age=5, priority=1,in_port=4,dl_src=00:00:00:00:00:22 actions=resubmit(,2)
cookie=0x0, duration=5.674s, table=1, n_packets=0, n_bytes=0, idle_age=5, priority=1,arp actions=resubmit(,3)
cookie=0x0, duration=5.515s, table=1, n_packets=0, n_bytes=0, idle_age=5, priority=0 actions=drop
cookie=0x0, duration=5.463s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,icmp,in_port=3 actions=resubmit(,3)
cookie=0x0, duration=5.412s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,icmp,in_port=4 actions=resubmit(,3)
cookie=0x0, duration=5.360s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=3,tp_dst=21 actions=resubmit(,3)
cookie=0x0, duration=5.308s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=4,tp_dst=21 actions=resubmit(,3)
cookie=0x0, duration=5.257s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=3,tp_dst=22 actions=resubmit(,3)
cookie=0x0, duration=5.205s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=4,tp_dst=22 actions=resubmit(,3)
cookie=0x0, duration=5.153s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=3,tp_dst=25 actions=resubmit(,3)
cookie=0x0, duration=5.101s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,tcp,in_port=4,tp_dst=25 actions=resubmit(,3)
cookie=0x0, duration=5.041s, table=2, n_packets=0, n_bytes=0, idle_age=5, priority=1,udp,in_port=3,tp_dst=53 actions=resubmit(,3)
cookie=0x0, duration=4.988s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,udp,in_port=4,tp_dst=53 actions=resubmit(,3)
cookie=0x0, duration=4.933s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,udp,in_port=3,tp_dst=67 actions=resubmit(,3)
cookie=0x0, duration=4.880s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,udp,in_port=4,tp_dst=67 actions=resubmit(,3)
cookie=0x0, duration=4.825s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,tcp,in_port=3,tp_dst=80 actions=resubmit(,3)
cookie=0x0, duration=4.772s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,tcp,in_port=3,tp_dst=443 actions=resubmit(,3)
cookie=0x0, duration=4.721s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,tcp,in_port=4,tp_dst=80 actions=resubmit(,3)
cookie=0x0, duration=4.669s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,tcp,in_port=4,tp_dst=443 actions=resubmit(,3)
cookie=0x0, duration=4.617s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,tcp,in_port=3,tp_dst=88 actions=resubmit(,3)
cookie=0x0, duration=4.563s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,tcp,in_port=4,tp_dst=88 actions=resubmit(,3)

cookie=0x0, duration=4.509s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,tcp,in_port=3,tp_dst=110 actions=resubmit(,3)
cookie=0x0, duration=4.454s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,tcp,in_port=4,tp_dst=110 actions=resubmit(,3)
cookie=0x0, duration=4.402s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,udp,in_port=3,tp_dst=123 actions=resubmit(,3)
cookie=0x0, duration=4.350s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,udp,in_port=4,tp_dst=123 actions=resubmit(,3)
cookie=0x0, duration=4.298s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,tcp,in_port=3,tp_dst=135 actions=resubmit(,3)
cookie=0x0, duration=4.241s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,tcp,in_port=4,tp_dst=135 actions=resubmit(,3)
cookie=0x0, duration=4.186s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,tcp,in_port=3,tp_dst=139 actions=resubmit(,3)
cookie=0x0, duration=4.133s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,tcp,in_port=4,tp_dst=139 actions=resubmit(,3)
cookie=0x0, duration=4.080s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,tcp,in_port=3,tp_dst=143 actions=resubmit(,3)
cookie=0x0, duration=4.027s, table=2, n_packets=0, n_bytes=0, idle_age=4, priority=1,tcp,in_port=4,tp_dst=143 actions=resubmit(,3)
cookie=0x0, duration=3.974s, table=2, n_packets=0, n_bytes=0, idle_age=3, priority=1,tcp,in_port=3,tp_dst=993 actions=resubmit(,3)
cookie=0x0, duration=3.922s, table=2, n_packets=0, n_bytes=0, idle_age=3, priority=1,tcp,in_port=4,tp_dst=993 actions=resubmit(,3)
cookie=0x0, duration=3.869s, table=2, n_packets=0, n_bytes=0, idle_age=3, priority=1,tcp,in_port=3,tp_dst=445 actions=resubmit(,3)
cookie=0x0, duration=3.817s, table=2, n_packets=0, n_bytes=0, idle_age=3, priority=1,tcp,in_port=4,tp_dst=445 actions=resubmit(,3)
cookie=0x0, duration=3.764s, table=2, n_packets=0, n_bytes=0, idle_age=3, priority=1,tcp,in_port=3,tp_dst=515 actions=resubmit(,3)
cookie=0x0, duration=3.711s, table=2, n_packets=0, n_bytes=0, idle_age=3, priority=1,tcp,in_port=4,tp_dst=515 actions=resubmit(,3)
cookie=0x0, duration=3.658s, table=2, n_packets=0, n_bytes=0, idle_age=3, priority=1,tcp,in_port=3,tp_dst=3306 actions=resubmit(,3)
cookie=0x0, duration=3.602s, table=2, n_packets=0, n_bytes=0, idle_age=3, priority=1,tcp,in_port=4,tp_dst=3306 actions=resubmit(,3)
cookie=0x0, duration=3.444s, table=2, n_packets=0, n_bytes=0, idle_age=3, priority=1,tcp,in_port=4,tp_dst=3389 actions=resubmit(,3)
cookie=0x0, duration=3.391s, table=2, n_packets=0, n_bytes=0, idle_age=3, priority=1,tcp,in_port=3,tp_dst=3389 actions=resubmit(,3)
cookie=0x0, duration=3.339s, table=2, n_packets=0, n_bytes=0, idle_age=3, priority=0 actions=drop
cookie=0x0, duration=3.281s, table=3, n_packets=0, n_bytes=0, idle_age=3, priority=10,in_port=3 actions=output:1
cookie=0x0, duration=3.228s, table=3, n_packets=0, n_bytes=0, idle_age=3, priority=5,in_port=3 actions=output:2
cookie=0x0, duration=3.176s, table=3, n_packets=0, n_bytes=0, idle_age=3, priority=10,in_port=4 actions=output:1
cookie=0x0, duration=3.122s, table=3, n_packets=0, n_bytes=0, idle_age=3, priority=5,in_port=4 actions=output:2
cookie=0x0, duration=3.070s, table=3, n_packets=0, n_bytes=0, idle_age=3, priority=10,arp actions=ALL