Sinikka Siironen

Palvelunestohyökkäyksen vaikutukset ohjelmisto-ohjatun tietoverkon ohjaimiin

Tietotekniikan pro gradu -tutkielma

4. toukokuuta 2018

Jyväskylän yliopisto

Informaatioteknologian tiedekunta

Tekijä: Sinikka Siironen

Yhteystiedot: sinikka.a.siironen@student.jyu.fi

Ohjaaja: Timo Hämäläinen

Työn nimi: Palvelunestohyökkäyksen vaikutukset ohjelmisto-ohjatun tietoverkon ohjaimiin

Title in English: Impact of Denial-of-Service Attack on Controllers in Software-Defined Networking

Työ: Pro gradu -tutkielma

Suuntautumisvaihtoehto: Ohjelmistotekniikka

Sivumäärä: 91+0

Tiivistelmä: Ohjelmisto-ohjattu tietoverkkoarkkitehtuuri on erityisesti pilvipalveluissa hyödynnettävä, perinteistä tietoverkkoa joustavampi arkkitehtuuri. Tutkielmassa esitellään kirjallisuuskartoituksessa löydettyjä ohjelmisto-ohjatun tietoverkon ohjaimen haavoittuvuuksia. Haavoittuvuudet liittyivät verkon tietojen keräämiseen, verkon topologiatietojen väärentämiseen, ohjaimen käyttämiin verkkosovelluksiin ja arkkitehtuurin rajapintoihin ohjaimen, sovellusten ja verkkolaitteiden välillä. Tutkimuksen toisena tavoitteena oli testata käytännössä, miten ohjelmisto-ohjatun tietoverkkoarkkitehtuurin ohjainohjelmistot selviytyvät palvelunestohyökkäyksestä. Tutkimus osoitti, että pahimmassa tapauksessa hyökkäys voi kaataa ohjainohjelmiston. Hyökkäyksen aikana ohjain kuluttaa enemmän suoritintehoa ja muistia, eikä se voi palvella oikeita verkon käyttäjiä kunnolla. Hyökkäysliikenne kuormittaa myös verkon käyttäjien välisiä yhteyksiä. Tutkimuksessa käytetyistä ohjaimista Floodlight suoriutui kokeista heikoiten, koska se kaatui eikä palautunut toimintakykyiseksi. OpenDaylight käytti melko paljon resursseja, mutta myös toimi tehokkaasti. Ryu käytti ohjaimista vähiten suoritintehoa. Ohjainten tuloksia vertailtaessa todettiin, että koetta suunniteltaessa valitut työkalut eivät kaikilta osin soveltuneet tehtävään. Tutkimuksen anti on vertailevan kokeen suunnittelusta tunnistetut hyvät ja huonot valinnat ja niiden hyödyntäminen jatkotutkimuksessa.

Avainsanat: haavoittuvuus, ohjain, ohjelmisto-ohjattu tietoverkko, palvelunestohyökkäys,

tietoturva

Abstract: Software-defined networking (SDN) is used especially in cloud computing, because it is more flexible than traditional networks. In this study, first, vulnerabilities found in SDN controllers are described based on literature research. The vulnerabilities found are related to network fingerprinting, forged topology information, applications used by the controller and the interfaces between controller, switches and applications. Second, a comparative experiment is conducted, where three controllers are put under Denial-of-Service attack and their performance is measured. It was found that in the worst case the attack can crash the controller. During the attack the controller consumes more CPU time and memory and can't serve legit users of the network properly. The attack traffic strains also connections between network users. Out of the controllers used in this study, Floodlight's performance was the worst because the controller crashed and couldn't recover. OpenDaylight consumed quite a lot resources but performed effectively. Ryu used the least amount of CPU time. Comparing the controllers' results it was aknowledged that not all the tools picked for this experiment were up to the task. The good and bad choices in designing the experiment can be used in further research to conduct better studies.

Keywords: controller, denial-of-service attack, security, software-defined network, vulnerability

Termiluettelo

Aktiiviset tietoverkot	Ohjelmoitavia verkkolaitteita tarjoava tietoverkkomalli, joka edelsi ohjelmisto-ohjattuja tietoverkkoja.
Ethane	Tietoverkon hallinnan keskittävä tietoverkkoarkkitehtuuri, jossa tietoturva on kiinteä osa arkkitehtuuria.
Ethernet-kytkin	Verkkolaite, joka välittää liikennettä siirtoyhteyskerroksessa.
Hajautettu palvelunestohyökkäys	Palvelunestohyökkäys, jossa hyökkäysliikennettä generoidaan monessa laitteessa. Engl. Distributed Denial-of-Service, DDoS.
Hallintakerros	Reititinarkkitehtuurin kerros. Engl. control plane.
Hallintataso	Ohjelmisto-ohjatun tietoverkkoarkkitehtuurin keskimmäinen taso. Engl. control layer.
IEEE Standards Association	IEEE:n (Institute of Electrical and Electronics Engineers) alainen järjestö, joka kehittää kansainvälisiä standardeja monille tekniikan aloille.
Internet Engineering Task Force	IETF. Internet-protokollia standardisoiva organisaatio.
Isäntäkone	Verkkoon liitetty tietokone. Engl. host.
Kansainvälinen televiestintäliitto	YK:n alainen televiestintäverkkoja ja -palveluja koordinoiva järjestö. Engl. International Telecommuncation Union, ITU.
Latenssi	Aika, joka paketilta kuluu matkaan lähettäjältä vastaanottajalle ja takaisin.
LLDP	Protokolla, jolla kytkimiä yhdistävät linkit löydetään. Engl. Link Layer Discovery Protocol.
Middlebox	Verkkoon kytketty laite, joka voi toimia esimerkiksi palomuurina tai tunkeilijan havaitsemisjärjestelmänä.
Nimipalvelujärjestelmä	Järjestelmä, jonka avulla IP-osoitteet muunnetaan verkko-osoitteiksi. Engl. Domain Name System, DNS.

Ohjain	Ohjelmisto, joka hallitsee ohjelmisto-ohjattua tietoverkkoa. Engl. controller.
Ohjelmisto-ohjattu tietoverkko	Tietoverkko, jonka verkkolaitteita hallitsee ohjainohjelmisto. Engl. Software-Defined Network, SDN.
Ohjelmisto-ohjattu tietoverkkoarkkitehtuuri	Arkkitehtuuri, jossa tietoverkon laitteita hallitaan keskitetysti ohjaimella. Engl. Software-Defined Networking, SDN.
OpenFlow	Ohjelmisto-ohjatun tietoverkon ohjaimen ja verkkolaitteen välinen viestintäprotokolla.
OSI-viitemalli	kuvaa tiedonsiirtoprotokollien yhteistoimintaa tietoliikennejärjestelmissä. Engl. Open System Interconnections model.
Palvelunestohyökkäys	Tietoturvahyökkäys, joka estää käyttäjiä käyttämästä palvelua. Engl. Denial-of-Service, DoS.
PKI	Public Key Infrastructure. TLS-salauksessa käytettävä julkisten avainten hallintajärjestelmä.
SDN	Software-Defined Networking. Ohjelmisto-ohjattu tietoverkkoarkkitehtuuri.
Siirtoyhteyskerros	OSI-viitemallin toinen kerros, joka yhdistää kaksi verkon solmua. Engl. data link layer.
TCP SYN -hyökkäys	Palvelunestohyökkäyksen muoto, jossa kohteelle lähetetään TCP SYN -viestejä, joihin vastaamiseen se joutuu kuluttamaan paljon resursseja.
Tiedonvälityskerros	Reititinarkkitehtuurin kerros. Engl. forwarding plane, data plane.
TLS	Transport Layer Security. Tietoliikenteen salausprotokolla.
Verkkoelementtitaso	Ohjelmisto-ohjatun tietoverkkoarkkitehtuurin alin taso. Engl. infrastructure layer.
Verkkokerros	OSI-viitemallin kolmas kerros, joka välittää tietoliikennepaketteja tietokoneiden välillä. Engl. network layer.
Verkkokäyttöjärjestelmä	Ohjelmisto-ohjatun tietoverkon ohjainohjelmisto. Engl. Network Operating System, NOS.

Verkkolaite	Verkkoliikennettä käsittelevä laite, esim. kytkin tai reititin.
Verkkosovellus	Ohjelmisto-ohjatun tietoverkon ohjaimessa ajettava sovellus.
Verkkosovellustaso	Ohjelmisto-ohjatun tietoverkkoarkkitehtuurin ylin taso. Engl. application layer.
Virtualisointi	Erillisten fyysisten ja ohjelmallisten resurssien ja toimintojen yhdistäminen virtuaalisiksi kokonaisuuksiksi.
Vuo	Tietoverkossa kulkeva pakettien virta, jota käsitellään yhtenä kokonaisuutena. Engl. flow.
Väliintulohyökkäys	Tietoturvahyökkäys, jossa hyökkääjä asettaa itsensä kahden koneen väliin vakoilemaan liikennettä. Engl. Man-in-the-Middle, MitM.

Kuviot

Taulukot

Sisältö

1 Johdanto

Tutkielmassa käsitellään ohjelmisto-ohjatun tietoverkkoarkkitehtuurin tietoturvaa. Tutkimuksen kohteeksi valittiin ohjaimen tietoturva, koska ohjain on keskeisin osa arkkitehtuuria ja näin ollen myös sen turvallisuus on tärkeintä.

Tietoverkot ovat luonteeltaan dynaamisia ja monimutkaisia ja siten myös vaikeasti hallinnoitavia ja muokattavia. Perinteiset tietoverkot eivät tarjoa juuri mitään mahdollisuuksia mukauttaa verkkoa ja sen käytäntöjä automaattisesti esimerkiksi verkon käyttömäärän muutoksen tai tunkeilijan havaitsemisen perusteella. Tämä johtuu siitä, että tietoverkon hallinta on hajautettu yksittäisiin verkkolaitteisiin, jotka pitää konfiguroida yksitellen. Nykyiset verkkolaitteet vaikeuttavat verkonlaajuisten käytäntöjen konfigurointia ja uusien ominaisuuksien kehittämistä. (Kim ja Feamster 2013) Näistä syistä perinteisen tietoverkon perustaminen ja ylläpitäminen on paitsi työlästä, myös kallista. (Kreutz ym. 2015)

Ohjelmisto-ohjattu tietoverkkoarkkitehtuuri (engl. Software Defined Networking, SDN) on kehitetty ratkaisuksi edellä kuvattuihin ongelmiin. Arkkitehtuurin olennainen osa on keskitetty ohjainohjelmisto, jolla voidaan hallita koko tietoverkkoa. Verkkolaitteet sisältävät vain liikenteenvälitykseen vaadittavat toiminnot. Uusien ominaisuuksien toteuttaminen on helpompaa ohjelmallisesti kuin käyttäen verkkolaitteiden tarjoamia rajallisia toimintoja. Lisäksi SDN-tietoverkkoa voidaan ohjata keskitetysti yhdestä paikasta, ohjaimesta, jolloin voidaan myös hyödyntää tietoja koko verkon tilasta. (Kim ja Feamster 2013)

Ohjelmisto-ohjattuja tietoverkkoja ovat jo hyödyntäneet muun muassa Google, Microsoft ja NTT (Nippon Telegraph and Telephone). Google on ottanut SDN-arkkitehtuurin käyttöön WAN (Wide Area Network) -verkossaan yhdistämään datakeskuksia ympäri maailman ja on saavuttanut lähes 100 prosentin käyttöasteen (Jain ym. 2013). Microsoft käyttää SDN-arkkitehtuuria Windows Azure -pilvipalvelussaan kuormantasaajana (Patel ym. 2013). NTT hyödyntää arkkitehtuuria asiakasyhteyksissä: yhteyden muodostaminen tapahtuu automaattisesti ja siihen voidaan dynaamisesti lisätä ominaisuuksia, kuten esimerkiksi tunkeilijan havaitsemisjärjestelmän (Natarajan, Ramaiah ja Mathen 2013). Merkittävä ohjelmisto-ohjatun tietoverkkoarkkitehtuurin kehittäjä ja sen käyttöönoton edistäjä on Open Networking Foun-

dation (ONF) -yhteisö, jota tukevat muun muassa Google, Cisco, Intel, Nokia ja Verizon (*Open Networking Foundation* 2017).

Ohjelmisto-ohjattu tietoverkkoarkkitehtuuri on ajankohtainen tutkimusaihe. Julkisia pilvipalveluita hyödynnetään nykyään esimerkiksi tallennustilana, data-analyysissa ja ulkopuolisten palveluiden tarjoamisessa. Tämän takia palvelinten välinen liikenne on merkittävä osa datakeskusten liikenteestä ja tulevaisuudessa se jatkaa kasvuaan. (Patel ym. 2013) Ohjelmisto-ohjattu tietoverkkoarkkitehtuuri soveltuu juuri datakeskusten pilvipalvelu- ja virtualisointitarpeisiin (Scott-Hayward, Natarajan ja Sezer 2016). Sitä on ehdotettu hyödynnettäväksi myös kotien tietoverkoissa, joihin nykyään kuuluu perinteisten tietokoneiden lisäksi esimerkiksi teollisen Internetin (engl. Internet of Things, IoT) laitteita (Taylor ym. 2017). Network Worldin kyselytutkimuksessa selvisi, että 49 prosenttia kyselyyn vastanneista harkitsevat tai pilotoivat SDN-ympäristöä ja 18 prosenttia on jo ottanut sen käyttöönsä. Vastaajat olivat kuitenkin huolissaan uuden teknologian tuomista ongelmista, erityisesti tietoturvallisuudesta. (Network World 2017)

Arkkitehtuurin turvallisuudessa onkin vielä parantamisen varaa. Tällä hetkellä turvallinen SDN-verkko tarkoittaisi pitäytymistä yhden yrityksen tuotteissa, yhteyksien rajoittamista luotettujen laitteiden välille ja tiukkoja turvallisuuskäytäntöjä. Arkkitehtuurin täydet hyödyt jäisivät silloin saavuttamatta. Dynaaminen ja avoin SDN-tietoverkko voisi kuitenkin olla perinteistä tietoverkkoa turvallisempi, jos jo tunnistetut tietoturvaongelmat pystytään ratkaisemaan. (Scott-Hayward, Natarajan ja Sezer 2016) SDN-verkon ongelmia ovat muun muassa keskitetyn hallinnan turvallisuuden takaaminen, ohjaimen ja verkkolaitteiden viestinnän turvaaminen ja verkkosovellusten vahingollisen toiminnan estäminen. Tietoturva on ratkaisevassa osassa uuden verkkoteknologian menestymisessä. (Ahmad ym. 2015)

1.1 Tutkimusongelma

Oman tutkimukseni kohteena ovat ohjelmisto-ohjatun tietoverkkoarkkitehtuurin ohjainohjelmistot. Tutkimuksen tarkoitus on arvioida, miten ohjelmistot selviytyvät palvelunestohyökkäyksestä, ja verrata niiden suoriutumista toisiinsa. Palvelunestohyökkäys (Denial-of-Service, DoS) valittiin hyökkäysmuodoksi siksi, että se on yksi suurimpia Internetin tieto-

turvaongelmia: Arbor Networksin vuoden 2017 turvallisuusraportin mukaan Internet-palveluntarjoajien suurin ja yritysten toiseksi suurin uhka oli hajautetut palvelunestohyökkäykset. Teollisen Internetin laitteiden hyödyntäminen hyökkäyksissä ja tarvittavien resurssien ja tekniikoiden helppo saatavuus antavat syytä huoleen. Palvelunestohyökkäykset aiheuttavat sekä rahallisia menetyksiä että vahinkoa yritysten ja palveluntarjoajien maineelle. (Alcoy ym. 2018) Tutkijoitten Yan ym. (2016) mukaan kaikki ohjelmisto-ohjatun tietoverkkoarkkitehtuurin tasot ja rajapinnat ovat alttiita palvelunestohyökkäyksille. Koska ohjaimen turvallisuus on erityisen kriittinen verkon toiminnan kannalta, se on houkutteleva palvelunestohyökkäyksen kohde. (Yan ym. 2016) Tässä tutkimuksessa vastataan seuraaviin kysymyksiin:

1. Millaisia tietoturvauhkia ohjelmisto-ohjatun tietoverkkoarkkitehtuurin ohjaimessa on?
2. Paljonko ohjain käyttää suoritintehoa ja muistia palvelunestohyökkäyksen aikana?
3. Miten palvelunestohyökkäys vaikuttaa verkon toimintaan?
4. Miten ohjain selviytyy hyökkäyksestä verrattuna muihin testattuihin ohjaimiin?

Tutkimus toteutetaan käytännön kokeena eli konstruktiivisena tutkimuksena. Tutkijoitten Wohlin ym. (2012) mukaan käytännön kokeita voidaan tehdä silloin, kun halutaan kontrolloida tilannetta ja suoraan ja systemaattisesti vaikuttaa tapahtumiin. Koe voidaan helposti toistaa ja siten osoittaa, että saadut tulokset pätevät laajemmastikin. (Wohlin ym. 2012) Tässä tutkimuksessa käytännön kokeen etuna on koeympäristön helppo kontrollointi, jolloin ohjelmistojen tulokset ovat keskenään vertailukelpoisia. Tarkoitus on tarkkailla ohjelmistojen resurssienkäyttöä palvelunestohyökkäyksen aikana ja sitä, haittaako palvelunestohyökkäys verkon normaalia toimintaa. Oletus on, että hyökkäys näkyy ohjelmistojen tarvitsemien resurssien määrien kasvussa ja hitaana verkkoyhteytenä. Lopuksi kunkin ohjelmiston tuloksia verrataan muihin ja pohditaan, voiko joku ohjelmiston ominaisuus selittää yhteneväisyyksiä tai eroja. Tämä tutkimus hyödyttää sekä ohjainohjelmistojen käyttäjiä että niiden kehittäjiä tietoturvan näkökulmasta. Tulokset voivat myös motivoida tulevaa tutkimusta ja antaa tietoa koejärjestelyn suunnittelusta.

Tutkielman seuraavassa luvussa kerrotaan ohjelmisto-ohjatusta tietoverkosta ja kolmannessa luvussa tietoverkkojen tietoturvasta. Ohjaimen haavoittuvuuksia kuvaillaan luvussa neljä ja palvelunestohyökkäyksiä luvussa viisi. Luku kuusi esittelee tutkimusstrategian, koejärjes-

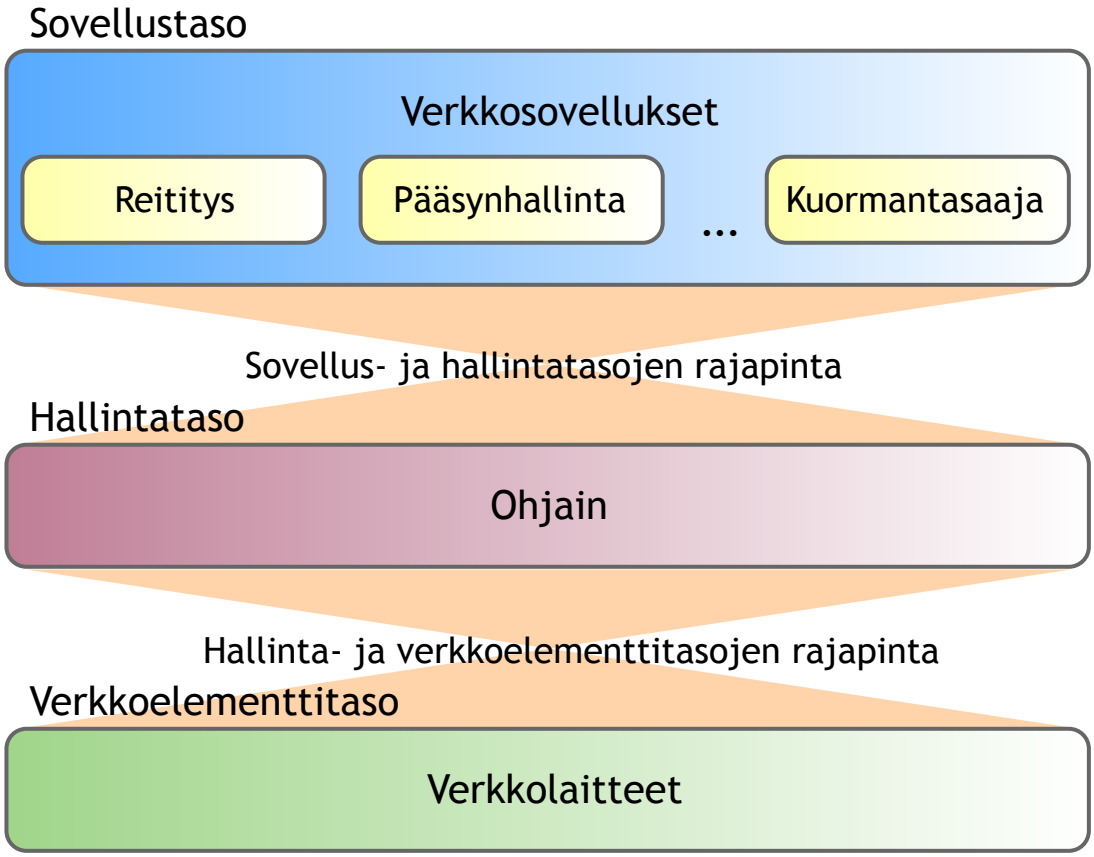
telyt ja kokeen toteutuksen. Luvussa seitsemän kerrotaan ohjainten tuloksista ja vertaillaan niitä. Luvussa kahdeksan esitetään johtopäätökset ja pohdinta.

2 Ohjelmisto-ohjattu tietoverkkoarkkitehtuuri

Perinteisessä tietoverkossa reititinarkkitehtuurin hallintakerros (engl. control plane) ja tiedonvälityskerros (engl. data plane, forwarding plane) on toteutettu samaan fyysiseen laitteeseen. Internetin alkuaikoina haluttiin varmistaa verkon vikasietoisuus, joka oli silloin tärkeä tavoite. Vaikka kyseinen arkkitehtuuri parantaakin verkkojen suorituskykyä, se on muuttumaton ja monimutkainen rakenne. Se on myös pohjimmainen syy siihen, miksi tietoverkot ovat jäykkiä ja vaikeita hallita. Verkko-operaattorit joutuvat käyttämään erilaisia middleboxeja, kuten palomuureja ja tunkeutumisen havainnointijärjestelmiä, lisätäkseen toimintoja tiedonvälitysreittien varrelle. Pääoma- ja toiminnalliset kulut ovat suuret ja haittaavat verkkojen uudistumista ja uusien toimintojen toteuttamista. (Kreutz ym. 2015)

Ohjelmisto-ohjattu tietoverkkoarkkitehtuuri (engl. Software-Defined Networking, SDN) eroaa perinteisestä tietoverkosta siten, että reititinarkkitehtuurin hallinta- ja tiedonvälityskerros on erotettu toisistaan. Arkkitehtuuri on jaettu kolmeen tasoon, joista alin on verkkoelementtitaso (engl. infrastructure layer). Kuten perinteisessä tietoverkossa, se koostuu reitittimistä, kytkimistä ja middleboxeista. Laitteista on kuitenkin poistettu kaikki logiikka, jolloin niihin jää vain välttämättömät tiedonvälityksessä tarvittavat toiminnallisuudet. Verkon älykkyys on siis keskitetty ylemmille hallinta- ja verkkosovellustasoille. (Kreutz ym. 2015) Arkkitehtuurin rakenne on kuvattu kuviossa 1.

Verkkoelementtitason yläpuolella on hallintataso. Tason tärkein osa on ohjain (engl. controller), jota voidaan kutsua myös verkkokäyttöjärjestelmäksi (engl. network operating system, NOS). Vaikka perinteisten käyttöjärjestelmien avulla on hallittu alemman tason laitteita ja resursseja jo pitkään, tietoverkoissa tällainen näkökulma on melko uusi. Ohjelmisto-ohjatun tietoverkkoarkkitehtuurin ohjain toimii ikään kuin käyttöjärjestelmä, joka piilottaa verkkolaitteiden erot ja tarjoaa ylemmälle kerrokselle laitteiden hallinnassa tarvittavat yleiset toiminnallisuudet. Ohjain kerää siis esimerkiksi tietoja verkon yleisestä tilasta, topologiasta, verkon laitteista ja konfiguraatiosta. Hallinta- ja verkkoelementtitasojen välinen ohjelmointirajapinta perustuu avoimeen standardiin, jonka selvästi suosituin toteutus on OpenFlow-protokolla (McKeown ym. 2008). Hallintatasolla voi olla rajapinta myös ohjaimen ja jonkin perinteisen verkon välillä (engl. eastbound API). (Kreutz ym. 2015)



Kuvio 1: Ohjelmisto-ohjatun tietoverkkoarkkitehtuurin rakenne.

Ohjaimet voidaan jakaa keskitettyihin ja hajautettuihin ohjaimiin. Keskitetty ohjain hallitsee kaikkia verkon laitteita yhdestä paikasta käsin, ja sen toimintavarmuus on siten kriittinen. (Kreutz ym. 2015) Keskitettyjä ohjaimia ovat esimerkiksi NOX-MT (Tootoonchian ym. 2012), Maestro (Cai, Cox ja Ng 2010), Beacon (Erickson 2013), Ryu (*Ryu* 2017), Rosemary (Shin ym. 2014) ja Floodlight (*Floodlight* 2017). Hajautetut ohjaimet voivat muodostaa keskitetyn klusterin tai fyysisesti hajautetun ryhmän. Hajautetut ohjaimet keskustelevat ohjainten välisen rajapinnan (engl. westbound API) läpi. (Kreutz ym. 2015) Esimerkiksi Onix (Koponen ym. 2010), ONOS (Berde ym. 2014) ja Fleet (Matsumoto, Hitz ja Perrig 2014) ovat hajautettuja ohjaimia.

Hallintatason yläpuolella on verkkosovellustaso (engl. application layer). Verkkosovelluksilla toteutetaan tietoverkon logiikka, esimerkiksi reititys, kuormantasaus ja tietoturvapalvelut. Sovelluksia voidaan hyödyntää myös esimerkiksi verkon virtualisoinnissa ja energian säästämisessä. (Kreutz ym. 2015)

Hallinta- ja sovellustason rajapinnalla ei vielä ole vakiintunutta toteutusta kuten OpenFlow on hallinta- ja verkkoelementtitasojen välillä. Ohjaimen toteutuksen abstrahointi kuitenkin helpottaisi sovellusten kehittämistä merkittävästi. (Kreutz ym. 2015) Nyt sovelluskehittäjät joutuvat käyttämään ohjainkohtaisia käskyjä ohjelmakoodissa, mikä heikentää koodin monikäyttöisyyttä. Rajapinnan standardointi mahdollistaisi samankaltaiset sovelluskaupat kuin esimerkiksi älypuhelimille. (Vasconcelos ym. 2017) Tällä hetkellä monet hallinta- ja sovellustason rajapinnat on toteutettu REST (REpresentational State Transfer) -arkkitehtuurimallin mukaisesti. (Li ym. 2016)

2.1 Historia

Feamster, Rexford ja Zegura (2013) ovat tutkineet ohjelmisto-ohjatun tietoverkkoarkkitehtuurin suhdetta sitä edeltäneisiin ohjelmoitaviin tietoverkkoihin. Jo varhaisissa puhelinverkoissa erotettiin hallinta- ja tiedonvälityskerros toisistaan. 1990-luvun puolivälissä kehiteltiin aktiivisia tietoverkkoja, joiden avulla voitiin ohjelmoida yksittäisten verkkolaitteiden resursseja ja näin käsitellä läpikulkevia paketteja monipuolisemmin. Aktiivisilla verkoilla haluttiin pienentää laskennan kustannuksia ja kehittää ja ottaa käyttöön uusia palveluita entistä helpommin.

2000-luvulla tietoliikenteen määrän kasvaessa tietoverkkojen haluttiin olevan entistä luotettavampia ja suorituskykyisempiä. Ongelmaa alettiin ratkaista hallinta- ja tiedonvälityskerroksien erottamisella ja hallintakerroksen ohjelmoitavuudella. Tästä syntyivät loogisesti keskitetty verkon hallinta ja tiedonvälityskerroksen avoin rajapinta, jotka ovat keskeisiä asioita nykyisessä ohjelmisto-ohjatussa tietoverkkoarkkitehtuurissa. OpenFlow (McKeown ym. 2008) sai nopeasti jalansijaa teollisuudessa, koska se oli helppo ottaa käyttöön vain verkkolaitteen ohjelmistoa päivittämällä ja se tarjosi haluttua ohjelmoitavuutta. Seuraavaksi alettiin kehittää koko verkkoa hallitsevia verkkokäyttöjärjestelmiä aikaisemman verkkolaitteen käyttöjärjestelmän sijaan. NOX (Gude ym. 2008) oli yksi ensimmäisistä verkkokäyttöjärjestelmistä eli ohjaimista. (Feamster, Rexford ja Zegura 2013)

2.2 Hyödyt

Kansainvälinen televiestintäliitto (International Telecommunication Union, ITU) on kehittänyt ohjelmisto-ohjatun tietoverkkoarkkitehtuurin viitekehyksen (*Framework of software-defined networking* 2014), joka asettaa arkkitehtuurille viisi tavoitetta:

1. Tietoverkko-operaattorit voivat vastata asiakkaiden tarpeisiin nopeammin ja saada vastinetta sijoituksille lyhyemmässä ajassa.
2. Uuden luominen ja kokeileminen on helpompaa verkon joustavuuden ansiosta.
3. Verkon asiakkaiden vaatimuksiin voidaan vastata hallitsemalla verkkolaitteita ja lisäämällä palvelun ominaisuuksia dynaamisesti.
4. Tietoverkon laitteet ovat helposti saatavilla ja niitä voidaan käyttää tehokkaasti esimerkiksi virtualisoinnin avulla.
5. Verkkolaitteita voidaan räätälöidä tiettyä palvelua varten ohjelmoimalla, jolloin esimerkiksi käytäntöjä voidaan muuttaa dynaamisesti.

Ohjelmisto-ohjattu tietoverkkoarkkitehtuuri tarjoaa siis useita hyötyjä verrattuna perinteiseen tietoverkkoon. Vaikka yritys ostaisi pilvipalveluntarjoajalta verkko- ja laskentaresursseja, se voisi silti säilyttää käyttämiensä resurssien hallinnan käyttämällä omaa ohjaintaan (Bring Your Own Controller, BYOC) (Wang ym. 2017). Hallinta- ja tiedonvälityskerroksia voidaan kehittää erillään, jolloin tuotteita voidaan tuoda markkinoille pienemmällä vaivalla. Verkkolaitetoimittajat voivat tarjota vain tiedonvälityksen toteuttavia laitteita ja asiakkaat saavat valita vapaammin eri valmistajien välillä, mikä on kasvattanut kilpailua ja luonut uusia keksintöjä. Myös avoimet rajapinnat parantavat arkkitehtuurin sovellettavuutta ja joustavuutta, kun verkkolaitteita ja hallintakerroksen toteutusta voidaan vaihtaa helposti. (Jarschel ym. 2014)

Logiikan keskittämisen ansiosta ohjain voi hyödyntää koko verkon kattavia tietojaan reitityksessä, mikä auttaa sitä sopeuttamaan verkon käytäntöjä liikenteen muutoksiin nopeammin ja paremmin kuin perinteisessä tietoverkossa on mahdollista. Ohjain voi hallita liikennettä hyvin yksityiskohtaisesti esimerkiksi yhden käyttäjän kokemuksen parantamiseksi. Ohjain voidaan myös toteuttaa hajautettuna järjestelmänä, jolloin skaalautuvuus paranee, tai se voidaan hajauttaa väliaikaisesti useammalle fyysiselle laitteelle esimerkiksi siinä tapauksessa

sa, että verkkoliikenteen ohjaus vaatii hetkellisesti tavallista enemmän resursseja. (Jarschel ym. 2014)

Ohjelmisto-ohjattu tietoverkkoarkkitehtuuri sopii myös tietoverkkojen virtualisoinnin välineeksi. Verkon ohjelmoitavuuden ansiosta koko verkkoa voidaan pitää ohjelmoitavana kokonaisuutena sen sijaan, että se koostuisi verkkolaitteista, joita pitää yksitellen konfiguroida. Tämä helpottaa esimerkiksi virtuaalisten verkkojen ja niiden liikenteen hallintaa. (Jarschel ym. 2014)

2.3 OpenFlow

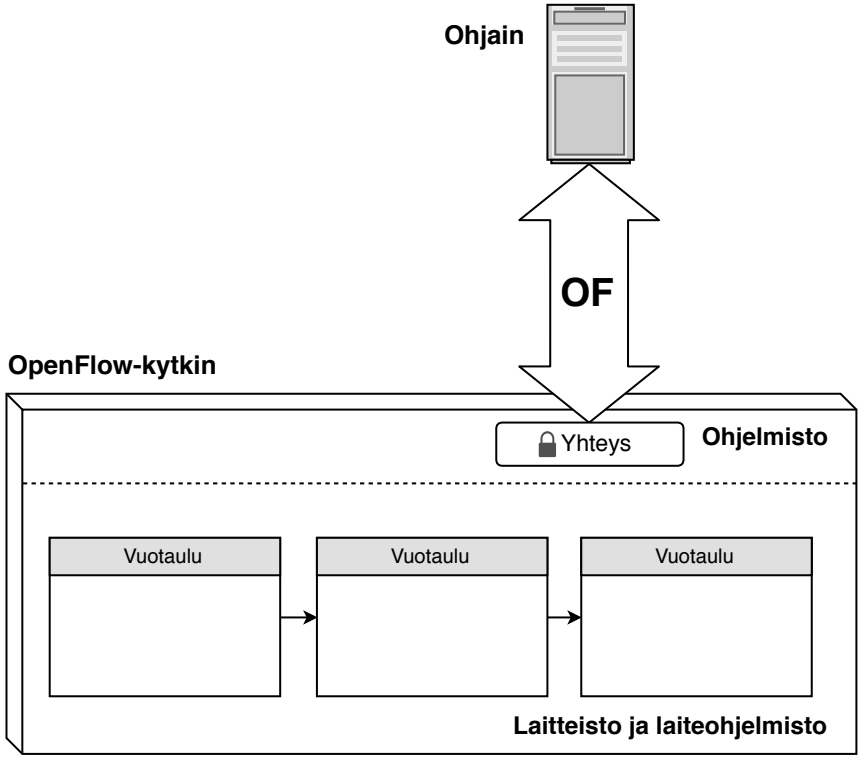
OpenFlow-kytkin kehitettiin alunperin helpottamaan uusien ideoiden kokeilua ohjelmoitavissa verkoissa. Sen periaatteita ovat hyvä suorituskyky ja matalat kustannukset, monipuolisen akateemisen tutkimuksen tukeminen ja koeliikenteen erottaminen oikeasta liikenteestä. OpenFlow hyödyntää eri valmistajien Ethernet-kytkinten yhteisiä toimintoja ja tarjoaa protokollan, jonka avulla näitä toimintoja voidaan käyttää kytkimen reititystaulun muokkaamiseen. Esimerkiksi tutkimukseen kuuluvat ja varsinaiset liikennevuot voidaan erottaa reitityssäännöillä. Uusia reititysprotokollia, tietoturvamalleja ja muita innovaatioita voidaan siis testata oikeassa tietoverkossa. (McKeown ym. 2008) Nykyään OpenFlow-standardia tukee ja kehittää Open Networking Foundation (*Open Networking Foundation* 2017).

OpenFlow-kytkimet voidaan jakaa OpenFlow'ta tukeviin yleiskäyttöisiin kytkimiin, joihin protokolla on lisätty, ja erityisiin OpenFlow-kytkimiin, jotka eivät tue tavallista siirtoyhteys- ja verkkokerrosten liikenteenvälitystä. Yleiskäyttöinen kytkin voi toimia yhtä aikaa sekä OpenFlow-kytkimenä että tavallisena siirtoyhteys- ja verkkokerrosten kytkimenä. Kytkimen rakenne on esitetty kuviossa 2. Se koostuu kolmesta osasta (McKeown ym. 2008):

- Vuotaulut, joiden mukaan kytkin käsittelee paketteja.
- Yhteys, joka yhdistää kytkimen ohjaimeen (suojausta ei vaadita).
- OpenFlow-protokolla, jonka mukaan kytkin ja ohjain kommunikoivat.

Jokaiseen liikennevuohon liitetään toiminto, joka voi olla esimerkiksi liikenteen välittäminen annettuun porttiin, liikenteen välittäminen ohjaimelle tai pakettien pudottaminen. Yleensä

vuon ensimmäinen paketti lähetetään ohjaimelle, jonka jälkeen ohjain antaa kytkimelle käsittelyohjeet vuolle. (McKeown ym. 2008) Vuosäännöt tallennetaan vuotauluihin, joita kytkimessä voi olla useita. Vuosääntöön liitetään myös osumakentät (engl. match field), joiden perusteella sääntöä käytetään pakettiin. Kun paketti saapuu kytkimelle, sen tietoja verrataan ensimmäisen vuotaulun vuosääntöihin alkaen ensimmäisestä säännöstä. Sääntöön sopiva paketti reititetään säännön ohjeiden mukaan joko toisen vuotaulun käsiteltäväksi tai ulos jostain kytkimen portista. Pakettia voidaan myös muokata säännön mukaan. Jos paketti ei sovi mihinkään vuosääntöön, se voidaan lähettää ohjaimelle reititysohjeiden saamiseksi. (*OpenFlow Switch Specification Version 1.3.0 (Wire Protocol 0x04)* 2012)



Kuvio 2: OpenFlow-kytkimen rakenne.

Tämän tutkimuksen koejärjestelyissä käytetään OpenFlow-protokollan versiota 1.3. Protokollaan kuuluu viestejä, jotka voivat liikkua ohjaimelta kytkimelle, asynkronisesti kytkimeltä ohjaimelle tai symmetrisesti. Ohjaimen kytkimelle lähettämiä viestejä käytetään kytkimen hallintaan ja tilan tarkkailuun. Ne eivät välttämättä vaadi vastausta kytkimeltä. Asynkroniset viestit lähettää aina kytkin; niissä kerrotaan ohjaimelle verkon muutoksista tai kytkimen tilasta. Symmetrisen viestin voi lähettää joko ohjain tai kytkin, ja siihen odotetaan aina vas-

tausta. (*OpenFlow Switch Specification Version 1.3.0 (Wire Protocol 0x04)* 2012) OpenFlow-protokollan version 1.3 viestit ja niiden merkitykset on koottu taulukkoon 1.

Viestit ohjaimelta kytkimelle	
Features	Ohjain kysyy kytkimeltä sen ominaisuuksia, usein yhteyttä muodostettaessa.
Configuration	Ohjain asettaa tai kysyy kytkimen asetuksia.
Modify-State	Ohjain muokkaa kytkimen vuosääntöjä tai porttien asetuksia.
Read-State	Ohjain kerää tietoja kytkimeltä.
Packet-out	Ohjain lähettää paketin kytkimelle ja antaa sille reititysohjeet.
Barrier	Ohjain varmistaa, että kytkin on käsitellyt saamansa viestit.
Role-Request	Ohjain asettaa tai kysyy sen ja kytkimen välisen yhteyden tilaa useamman ohjaimen ympäristössä.
Asynchronous-Configuration	Ohjain voi asettaa suodattimen kytkimeltä tuleville viesteille.
Asynkroniset viestit	
Packet-in	Kytkin lähettää ohjaimelle paketin tai paketin tiedot, koska se ei sovi kytkimen vuosääntöihin.
Flow-Removed	Kytkin ilmoittaa ohjaimelle, että vuo on poistettu joko ohjaimen pyynnöstä tai annetun aikarajan umpeutumisen takia.
Port-Status	Kytkin ilmoittaa ohjaimelle portin tilan tai asetusten muutoksesta.
Error	Kytkin ilmoittaa ohjaimelle ongelmasta.
Symmetriset viestit	
Hello	Viestejä käytetään ohjaimen ja kytkimen yhteyden muodostuksessa.
Echo	Viestejä käytetään yhteyden varmistamiseksi tai latenssin ja kaistanleveyden mittaamiseksi.
Experimenter	Viestiä käytetään uusien OpenFlow-ominaisuuksien testaamiseen.

Taulukko 1: OpenFlow-protokollan version 1.3 viestit ohjaimen ja kytkimen välillä (*OpenFlow Switch Specification Version 1.3.0 (Wire Protocol 0x04)* 2012).

3 Tietoverkkojen tietoturva

Internet on elintärkeä nykypäivän yrityksille, korkeakouluille ja valtionhallinnolle. Myös tavalliset ihmiset ovat riippuvaisia Internetistä, jossa he tekevät töitä, hoitavat sosiaalisia suhteita ja yksityisasioita. Rikollisuutta on myös verkossa: hyökkääjä voi loukata ihmisten yksityisyyttä tai estää jonkin palvelun käytön Internetissä. Tietoverkkoturvallisuus tarkoittaa sitä, miten tietoverkkoja vastaan voidaan hyökätä ja miten näitä hyökkäyksiä voidaan ehkäistä ja torjua. Tietoverkkoturvallisuus on keskeinen aihe tietoverkkotutkimuksessa, koska verkkohyökkäyksiä tapahtuu jatkuvasti ja uusia hyökkäyksiä kehitellään koko ajan lisää. (Kurose ja Ross 2013) Tietoturvan kehittäminen on kuitenkin hankalaa, koska verkot ovat monimutkaisia, niiden tietoturvaratkaisuja on vaikea hallinnoida ja IP-verkoissa henkilöä on hankala tunnistaa. Myös Internetin infrastruktuurissa on omat ongelmansa. (Ahmad ym. 2015)

Tietoturvallisuuden tarkoituksena on suojata resursseja, joita ovat esimerkiksi fyysiset laitteet ja infrastruktuurit, tieto ja ohjelmistot. Myös viestinnän on oltava turvallista: tieto pitää suojata myös tiedonsiirron aikana. Resursseja voidaan suojella tunnistamalla ja ehkäisemällä tietoturvauhkia ja -haavoittuvuuksia. Tietoturvauhka on jokin hyökkäys, jolla hyökkääjä pyrkii vahingoittamaan resurssia. Haavoittuvuus taas on järjestelmän ominaisuus, joka altistaa jollekin uhalle. Teknisten ratkaisujen lisäksi tietoturvaa voi parantaa hyvillä menettelytavoilla ja organisaatiotason toimenpiteillä. (Kotzanikolaou ja Douligeris 2007)

Perinteisesti tietoturvallisuus on määritelty luottamuksellisuudeksi, eheydeksi ja käytettävyydeksi. Luottamuksellisuus (engl. confidentiality) tarkoittaa, että tieto pysyy niillä luotetuilla käyttäjillä, joille se on tarkoitettukin. Eheys (engl. integrity) takaa, että tietoa ei muuteta luvattomasti. Saatavuus (engl. availability) tarkoittaa, että tietoa ei voida luvattomasti salata tai pidättäytyä antamasta. Myös todennus ja kiistämättömyys ovat tietoturvallisuuden perusasioita. Todennus (engl. authentication) voi olla esimerkiksi käyttäjän, koneen tai tietolähteen identiteetin varmistaminen. Kiistämättömyys (engl. nonrepudiation) takaa, että teko voidaan aina kiistatta liittää tekijään esimerkiksi käyttäjätunnuksen avulla. (Kotzanikolaou ja Douligeris 2007)

Kotzanikolaou ja Douligeris (2007) antavat seuraavanlaisia esimerkkejä tietoturvahyökkäyksistä:

Salakuuntelu	Hyökkääjä voi kaapata liikennettä tai salakuunnella yhteyttä.
Väärentäminen	Hyökkääjä voi väärentää IP-osoitteensa ja tekeytyä siten luotetuksi henkilöksi tai laitteeksi.
Tunkeutuminen	Hyökkääjä voi päästä sisälle järjestelmään hyödyntämällä jotakin järjestelmän haavoittuvuutta esimerkiksi verkkopalvelimessa.
Kaappaus	Hyökkääjä voi päästä sisälle järjestelmään kaappaamalla luotetun laitteen yhteyden, esimerkiksi verkkoistunnon.
Palvelunestohyökkäys	Hyökkääjä kuluttaa mahdollisimman paljon verkon tai palvelun resursseja, jolloin oikeat käyttäjät eivät pääse käyttämään palvelua.
Sovelluskerroksen hyökkäykset	Hyökkääjä hyödyntää sovelluskerroksen heikkouksia esimerkiksi verkkopalvelimessa tai käytetyssä teknologiassa. Tällaisia hyökkäyksiä ovat esimerkiksi haittaohjelmat, SQL-injektiot ja Cross-Site Scripting (XSS).

3.1 Ohjelmisto-ohjatun tietoverkon tietoturva

Ohjelmisto-ohjatun tietoverkkoarkkitehtuurin tietoturvasta on tehty viime vuosina useita kirjallisuuskatsauksia, esimerkiksi Kreutz ym. (2015), Scott-Hayward, Natarajan ja Sezer (2016), Ahmad ym. (2015) ja Alsmadi ja Xu (2015). Arkkitehtuurin OpenFlow-toteutuksen turvallisuutta ovat tutkineet Klöti, Kotronis ja Smith (2013). Arkkitehtuurin turvallisuutta on myös standardoitu: Kansainvälisen televiestintäliiton standardointiosasto (ITU-T) on laatinut omat suosituksensa arkkitehtuurin tietoturvan parantamiseksi (*Security requirements and reference architecture for software-defined networking* 2016), samoin IEEE Standards Association kehittää omaa standardiaan (*Standard for Software Defined Networking and Network Function Virtualization Security* 2017).

Tietoturvaa on pyritty parantamaan ohjelmoitavilla tietoverkoilla jo pitkään. Ethane-arkkitehtuuria (Casado ym. 2007) voidaan pitää SDN-verkon OpenFlow-toteutuksen edeltäjänä. Myös Ethanessa on loogisesti keskitetty ohjain, joka hallitsee yksinkertaisia kytkimiä. Tie-

toturvallisuus on toteutettu kiinteänä osana arkkitehtuuria esimerkiksi pääsynhallinnan muodossa. Ethane myös sitoo paketin ja sen lähettäjän tiukasti yhteen, jolloin käyttäjien seuraaminen on mahdollista, vaikka sijainnit muuttuisivatkin. Ohjelmisto-ohjatussa tietoverkossa turvallisuutta ei ole suunniteltu osaksi arkkitehtuuria, joten Ethanen tietoturvan toteutuksesta voitaisiin ottaa opiksi ohjelmisto-ohjatussa tietoverkkoarkkitehtuurissa. (Ahmad ym. 2015)

Perinteisillä tietoturvapalveluilla on hankalaa torjua nykyaikaisia tietoverkkohyökkäyksiä (Jeong ym. (2015), Morzhov, Alekseev ja Nikitinskiy (2016)), mihin voi löytyä ratkaisu ohjelmisto-ohjatusta tietoverkkoarkkitehtuurista. Arkkitehtuurin verkkosovellukset voivat hyödyntää ohjaimelta saatavia tietoja verkon tilasta ja liikenteestä. Sovellus voi esimerkiksi analysoida pakettivirtaa reaaliaikaisesti ja ohjata liikenteen uudelle reitille verkon tietoturvakäytäntöjen mukaisesti. Ne voivat hyödyntää ohjaimen tarjoamaa yhteistä abstraktiota verkon tilasta, jolloin ne voivat toimia johdonmukaisesti ja ristiriidattomasti. Ohjelmisto-ohjatun verkon tietoturva on myös perinteistä tietoverkkoa helpompi pitää ajan tasalla päivittämällä sovelluksia sen sijaan, että vaihdettaisiin fyysisiä verkkolaitteita tai päivitettäisiin niitä yksittäin. Lisäksi arkkitehtuurissa uusien ominaisuuksien toteuttaminen on nopeampaa. (Ahmad ym. 2015)

Toisaalta ohjelmisto-ohjattu tietoverkko aiheuttaa uusia tietoturvauhkia. Arkkitehtuurin tasot luovat suuremman hyökkäyspinnan kuin perinteisessä tietoverkossa (Betgé-Brezetz, Kamga ja Tazi 2015), joten tietoturvaa on tarkasteltava kaikilla tasoilla ja niiden rajapinnoilla. Tutkijoitten Scott-Hayward, Natarajan ja Sezer (2016) mukaan arkkitehtuuri on myös erityisen altis palvelunestohyökkäyksille verkon keskitetyn hallinnan, kytkinten vuotaulujen rajallisuuden, avointen rajapintojen ja hyvien käytänteiden kehittymättömyyden vuoksi.

Ohjelmisto-ohjatun tietoverkon tietoturvallisuudessa on vielä kehittämisen varaa. Scott-Hayward, Natarajan ja Sezer (2016) uskovat, että ohjelmisto-ohjattua tietoverkkoarkkitehtuuria ei voida ottaa yksittäistä datakeskusta tai yritysverkkoa laajempaan käyttöön ennen kuin tietoturvallisuuteen on kiinnitetty enemmän huomiota. Tutkijoitten Jeong ym. (2015) mukaan ohjaimen turvallisuus täytyy varmistaa ennen kuin sen päälle voidaan rakentaa muita turvallisuuspalveluita, sillä ohjaimella on arkkitehtuurissa niin keskeinen asema. Ohjaimella ja hallintatasolla on siis suuri merkitys koko arkkitehtuurin turvallisuudelle.

Ohjelmisto-ohjattu tietoverkkoarkkitehtuuri on synnyttänyt myös ohjelmisto-ohjatun tietoturvallisuuden (engl. Software-Defined Security, SDSec) käsitteen. Tällä tarkoitetaan turvallisuuteen liittyvien toimintojen loogista keskittämistä yhteen paikkaan sen sijaan, että ne sijaitsisivat erillisissä laitteissa ympäri verkkoa. Al-Zewairi, Suleiman ja Almajali (2017) ovat kehittäneet erityisen SDSec-ohjaimen, joka osaa autentikoida verkkolaitteet ennen niiden yhdistämistä verkkoon. Tutkijoitten mukaan SDSec-ohjain voitaisiin myös erottaa tavallisesta SDN-ohjaimesta, jolloin se voisi hoitaa tietoturvallisuutta itsenäisesti. Yanbing ym. (2016) taas ovat esitelleet ohjelmisto-ohjatun tietoturva-arkkitehtuurin (engl. Software-Defined Security Architecture, SDSA), joka erottaa turvallisuustoiminnot ja niiden hallinnan toisistaan. Turvallisuusominaisuuksia tarvitsevat sovellukset voivat kutsua niitä alemmalta tasolta, jolloin ominaisuuksien hyvällä toteutuksella voidaan taata sovellusten turvallisuus. Sekä SDSec-ohjain että SDSA ovat esimerkkejä siitä, miten ohjelmisto-ohjatun tietoverkon tärkeintä ominaisuutta, hallinnan ja toimintojen erottamista, voidaan hyödyntää tietoturvan parantamisessa.

Taulukossa 2 on esitelty ITU-T:n tunnistamia ohjelmisto-ohjatun tietoverkkoarkkitehtuurin tietoturvauhkia tasojen ja rajapintojen mukaan luokiteltuna. ITU-T on antanut myös suositukset uhkien ehkäisemiseksi.

Uhka	**Kuvaus**
Verkkosovellustaso	
Väärentäminen	Hyökkääjä esiintyy ohjaimena.
Kiistäminen	Käyttäjä voi toimia haitallisesti ja kieltää tekonsa.
Tiedon paljastuminen	Hyökkääjä voi käyttää oikean käyttäjän tunnuksia ja lähettää verkkoon väärennettyä liikennettä verkkosovelluksen kautta.
Verkkosovellusten haavoittuvuudet	Hyökkääjä voi päästä käsiksi sovelluksen resursseihin ja käyttää niitä muissa hyökkäyksissä.
Hallintataso	
Vuosääntöjen ristiriidat	Verkkosovellus voi korvata toisen sovelluksen antaman vuosäännön.
Haitallinen vuosääntö	Hyökkääjä voi kaapata verkkosovelluksen ja asentaa haitallisen vuosäännön salakuunnellakseen tietoja.
Väärentäminen	Hyökkääjä voi esiintyä pääkäyttäjänä tai verkkosovelluksena ja muokata tai poistaa tietoja, käsitellä topologia- ja reititystietoja tai saada koko ohjaimen haltuunsa. Väärentämällä ohjaimen osoitteen hyökkääjä voi käyttää verkossa omaa ohjaintansa. Hyökkääjä voi luoda kytkimen ja saada tietoja verkosta tarkkailemalla, miten ohjain reagoi erilaisiin paketteihin.
Palvelunestohyökkäys Viive hyökkäyksen torjunnassa	Hyökkääjä voi luoda liikennettä, joka saa kytkimen kysymään ohjaimelta vuosääntöjä ja näin kuormittaa ohjainta. Vuosääntöjä päivitetään tavallisesti tietyin väliajoin suorituskyvyn parantamiseksi, jolloin kiireelliset päivitykset hyökkäyksen pysäyttämiseksi viivästyvät.

Kiistäminen	Pääkäyttäjä tai verkkosovellus voi asentaa haitallisia vuosääntöjä ja kiistää tekonsa.
Tiedon paljastuminen	Hyökkääjä voi saada haltuunsa tietoja järjestelmästä tulevaa hyökkäystä varten.
Käyttöjärjestelmän haavoittuvuudet	Ohjainohjelmistoa ajetaan jossain käyttöjärjestelmässä, jolloin käyttöjärjestelmän haavoittuvuudet ovat myös ohjaimen haavoittuvuuksia.
Ohjelmiston haavoittuvuudet	Hyökkääjä voi hyödyntää virheitä, puutteita ja heikkouksia ohjainohjelmistossa hyökkäyksissään.
Laitteistoviat	Ohjaimen tai kytkinten laitteistoviat voivat vaarantaa turvallisuuden tai kaataa verkon.
Verkkoelementtitaso	
Väärentäminen	Hyökkääjä voi esiintyä pääkäyttäjänä tai ohjaimena päästäkseen käsiksi kytkimen tietoihin.
Salakuuntelu	Hyökkääjä voi salakuunnella kytkinten välistä liikennettä ja saada tietää, millaista tietoa verkossa liikkuu, mikä liikenne on sallittua ja millaisia voita on käytössä.
Tiedon paljastuminen	Hyökkääjä voi saada haltuunsa tietoja järjestelmästä tulevaa hyökkäystä varten.
Vuotaulun ylivuoto	Kytkimellä on tavallisesti rajallinen vuotaulu, jota hyökkääjä voi hyödyntää oikeiden vuosääntöjen ylikirjoittamisessa tai palvelunestohyökkäyksessä.
Kiistäminen	Pääkäyttäjä tai ohjain voi muuttaa laitteen asetuksia ja kiistää tekonsa.
Verkkosovellus- ja hallintatason rajapinta	
Salakuuntelu	Salakuunneltujen tietojen perusteella hyökkääjä voi päätellä verkon käytäntöjä ja hyödyntää niitä hyökkäyksissä.
Viestien muuttaminen ja sieppaaminen	Hyökkääjä voi siepata tai muuttaa ohjaimen ja sovelluksen välisiä viestejä esimerkiksi käytäntöjen muuttamiseksi.

Hallinta- ja verkkoelementtitason rajapinta	
Salakuuntelu	Hyökkääjä voi salakuunnella ohjaimen ja kytkimen välisiä viestejä ja päätellä verkon reitityskäytännöt.
Viestien muuttaminen ja sieppaaminen	Hyökkääjä voi siepata tai muuttaa ohjaimen ja kytkimen välisiä viestejä esimerkiksi lähettääkseen omia viestejä kytkimille.

Taulukko 2: Ohjelmisto-ohjatun tietoverkkoarkkitehtuurin tietoturvauhkia (*Security requirements and reference architecture for software-defined networking* 2016).

4 Ohjaimen haavoittuvuudet

Ohjelmisto-ohjatussa tietoverkossa ohjaimen tietoturvallisuus on ensiarvoisen tärkeää, kuten monet tutkijat ovat huomauttaneet (Akhunzada ym. (2015), Ahmad ym. (2015)). Tutkijoitten Ahmad ym. (2015) mukaan hallintatason turvallisuus vaikuttaa myös verkkoelementtitason turvallisuuteen, koska kytkinten ja ohjaimen välillä on niin vahva riippuvuus. Scott-Hayward (2015) huomauttaa kuitenkin, että arkkitehtuurin kehittyessä keskitetty hallinta ei enää ole suurin ongelma: turvallisuusuhkia aiheuttavat myös kolmannen osapuolen verkkosovellukset, tietoturvakäytäntöjen ristiriidat ja monimutkaiset usean ohjaimen ympäristöt. Edellä mainittuja ongelmia voidaan kuitenkin ratkaista kehittämällä ohjaimen toimintaa.

Seuraavaksi esitellään ohjaimen haavoittuvuuksia. Tunnistetut haavoittuvuudet on jaettu neljään eri kategoriaan: verkon tietojen kerääminen, topologiatietojen väärentäminen, haitalliset verkkosovellukset ja rajapintojen turvattomuus.

4.1 Verkon tietojen kerääminen

Ohjelmisto-ohjattujen verkkojen ja niiden ominaisuuksien tunnistamiseksi on kehitetty erilaisia tekniikoita. Hyökkääjää kiinnostavat erityisesti käytetty ohjainohjelmisto ja sen versio, tietoturvakäytännöt ja ohjainta kuormittavat liikennevuot. Ne kaikki voidaan tunnistaa tutkimalla kytkinten vuosääntöjen osumakenttiä (engl. match field), joiden selvittämiseen M. Zhang ym. (2017) ovat kehittäneet menetelmän.

Shin ja Gu (2013) ovat kehittäneet SDN Scanner -työkalun, jolla voidaan tunnistaa, käyttääkö verkko ohjelmisto-ohjattua tietoverkkoarkkitehtuuria. Tunnistaminen perustuu verkkoon lähetettyjen pakettien vasteaikaan: jos kytkin joutuu lähettämään paketin ohjaimelle reititystä varten, vasteaika pitenee. Tutkijat Azzouni ym. (2016) kuitenkin huomauttavat, että vasteaikojen vertailu oikeassa WAN (Wide Area Network) -verkossa on todella vaikeaa, koska vasteaikoihin vaikuttaa moni muukin asia. Myös Bifulco ym. (2015) tunnistavat ohjelmisto-ohjatun verkon paketin kiertoviivettä (round-trip time, RTT) mittaamalla, mutta huomauttavat, että tunnistuksen varmuuteen vaikuttaa erityisesti verkon koko. Samankaltaista tekniikkaa uusien vuosääntöjen asentamisen tunnistamiseksi ovat tutkineet Sonchack, Aviv ja

Keller (2016) sekä Klöti, Kotronis ja Smith (2013).

Azzouni ym. (2016) ovat kehittäneet useampia tekniikoita, joita yhdistelemällä voidaan tunnistaa, mitä ohjainohjelmistoa verkko käyttää. Hyökkääjä voi kerätä tietoja verkosta edellä esitetyillä tekniikoilla ja käyttää niitä erilaisten hyökkäysten suunnittelussa ja toteutuksessa (Hoque ym. 2014). Esimerkiksi tutkijoitten Sonchack, Aviv ja Keller (2016) mukaan kerättyjä tietoja voidaan käyttää palvelunestohyökkäyksen suunnittelussa: hyökkääjä voi lähettää suuria määriä paketteja, joiden käsittelyyn ohjain joutuu käyttämään kaikki resurssinsa.

4.2 Topologiatietojen väärentäminen

Ohjain kerää tietoja verkon topologiasta suoraan verkkolaitteilta. OpenFlow-protokollaa tukevat ohjaimet selvittävät kytkinten välisiä linkkejä seuraavasti: Ohjain lähettää kaikkien kytkinten kaikkiin portteihin LLDP (Link Layer Discovery Protocol) -paketin, joka sisältää vastaanottavan kytkimen ja portin tunnisteet. Vastaanottaessaan LLDP-paketin toiselta kytkimeltä kytkin lähettää sen tietojen kera ohjaimelle. Tästä ohjain voi päätellä linkin yhdistävän kyseisiä kytkimiä. (Alharbi, Portmann ja Pakzad 2015)

Tutkijat Alharbi, Portmann ja Pakzad (2015) esittävät yksinkertaisen linkinväärennöshyökkäyksen. Hyökkääjä voi helposti antaa ohjaimelle tiedon kahden kytkimen välisestä linkistä, jota ei tosiasiassa ole olemassa. Hyökkääjän tarvitsee vain lähettää kytkimelle LLDP-paketti ja merkitä sen lähettäjäksi haluamansa kytkin ja sen portti. Väärennetyn paketin vastaanottanut kytkin lähettää sen edelleen ohjaimelle, joka päättelee paketin tietojen perusteella, että näiden kahden kytkimen välillä on linkki. Tutkijat Khan ym. (2017) esittävät samankaltaisen hyökkäyksen, jossa kytkin lähettää saamansa LLDP-paketin ohjaimen sijaan eteenpäin toiselle kytkimelle, mikä myös vääristää ohjaimen topologiatietoja.

Tutkijoitten Khan ym. (2017) mukaan hyökkääjän on myös mahdollista kaapata verkossa sijaitsevan isäntäkoneen liikenne. Ohjain päivittää isäntäkoneiden sijaintitietoja niiltä saamiensa pakettien perusteella. Jos hyökkääjä voi lähettää ohjaimelle paketin, jonka lähettäjän IP-osoite on kohdekoneen osoite, ohjain tunnistaa kohdekoneen muiden tunnisteiden perusteella ja päättelee sen vaihtaneen osoitetta. Tällöin kohdekoneen liikenne ohjautuu suoraan hyökkääjälle.

Jotta verkkoa voidaan ylläpitää ja hallita, topologiatietojen pitää olla oikeellisia ja täydellisiä. Monet ohjaimen perustoiminnot ja verkkosovellukset tarvitsevat topologiatietoja toimiakseen. (Khan ym. 2017) Linkkejä väärentääkseen tai isäntäkoneen liikenteen kaapatakseen hyökkääjän tarvitsee hallita vain yhtä verkkoon kuuluvaa, fyysistä tai virtuaalista konetta (Alharbi, Portmann ja Pakzad 2015), joten hyökkäys on helposti toteutettavissa.

4.3 Haitalliset verkkosovellukset

Lee, Yoon ja Shin (2016) ovat tunnistaneet verkkosovellukset yhdeksi arkkitehtuurin hyökkäysvektoreista. Ohjaimen ja verkkosovellustason välinen avoin rajapinta tarkoittaa, että kuka tahansa voi kehittää sovelluksia ja levittää niitä sovelluskaupassa. Suurin osa ohjaimista ei kuitenkaan autentikoi sovelluksia ennen niiden suorittamista, jolloin haitallinen ohjelma saatetaan ottaa käyttöön. Haittaohjelma voi olla esimerkiksi piilohallintaohjelmisto (engl. rootkit), joka piiloutuu ohjainohjelmistoon ja voi ohjelmoida verkkoa haluamallaan tavalla salassa (Röpke ja Holz 2015).

Tutkijoitten Ahmad ym. (2015) mukaan verkkosovellusten autentikoinnin lisäksi ohjaimen pitäisi rajoittaa niiden oikeuksia. Sovellusten pitäisi päästä käsiksi vain niihin ohjaimen tarjoamiin resursseihin, joita ne oikeasti tarvitsevat. Lisäksi kolmansien osapuolten tuottamien sovellusten oikeuksia pitäisi rajoittaa niiden epäluotettavuuden vuoksi. Wen ym. (2013) esittävät, että rajoittamattomat oikeudet voivat olla vaarallisia, jos sovellus on haitallinen tai siinä on ohjelmointivirheitä.

Lee, Yoon ja Shin (2016) demonstroivat kolme hyökkäystä, jotka perustuvat sovellusten rajoittamattomiin oikeuksiin. Floodlight-ohjaimessa (*Floodlight* 2017) haitallinen sovellus voi väärentää saapuvan paketin viestin ennen kuin se päätyy muiden sovellusten käyttöön, sillä Floodlight ei takaa viestien eheyttä eikä viestien saapumisjärjestystä sovelluksille. ONOS-ohjain (Berde ym. 2014) sallii tiettyjen ohjaimen komponenttien parametrien muokkaamisen rajoituksetta, jolloin haitallinen sovellus voi esimerkiksi manipuloida pakettien välitystä. OpenDaylight-ohjaimen (Medved ym. 2014) tapauksessa Lee, Yoon ja Shin (2016) havaitsivat, että haitallinen sovellus voi estää toista sovellusta käyttämästä tiettyä ohjaimen palvelua.

Sovellusten rajoittamattomat oikeudet ja hallitsematon resurssien käyttö voivat pahimmillaan

johtaa ohjaimen toimimattomuuteen. Shin ym. (2014) onnistuivat kaatamaan Floodlight-ohjaimen kahdella virheellisellä sovelluksella, joista toinen kutsuu järjestelmän exit-funktiota ja toinen varaa rajattomasti muistia. Kolmas sovellus pääsi käsiksi Floodlightin sisäiseen tietorakenteeseen ja pystyi muuttamaan verkon linkkitietoja. Shin ym. (2014) löysivät samankaltaisia haavoittuvuuksia myös OpenDaylight-, POX- (*POX* 2017) ja Beacon (Erickson 2013) -ohjaimista.

Scott-Hayward (2015) uskoo, että sovellusten autentikointi yhdistettynä niiden suorituksen eristämiseen riittäisi suojaamaan ohjainta haitallisilta tai virheellisiltä sovelluksilta.

4.4 Rajapintojen turvattomuus

Ohjelmisto-ohjatun tietoverkkoarkkitehtuurin tasot viestivät toistensa kanssa rajapintojen kautta. OpenFlow on yleinen toteutus hallinta- ja verkkoelementtitasojen rajapintana, mutta hallinta- ja verkkosovellustasojen rajapinnalla ei ole yleistä standardia. Useamman ohjaimen ympäristössä voidaan tunnistaa myös rajapinta kahden ohjaimen välillä.

Jotkut ohjaimet tarjoavat web-käyttöliittymän, jonka kautta verkkoa voidaan hallita. Esimerkiksi OpenDaylight-ohjaimeen voidaan asentaa DLUX-niminen web-käyttöliittymä. Brooks ja Yang (2015) toteuttivat sen avulla väliintulohyökkäyksen (engl. Man-in-the-Middle, MitM): He käyttivät *ettercap*-työkalua kaapatakseen ohjaimen liikenteen ja saivat selville web-käyttöliittymän kirjautumistiedot. Tutkijoita ihmetyttää, että kirjautumistiedot lähetetään salaamattomalla yhteydellä. Käyttöliittymästä voidaan hallita koko verkkoa, joten haavoittuvuus on vakava.

Tutkijoitten Padekar ym. (2016) mielestä ohjaimen rajapinnat pitäisi suojata sovellusvirheitä ja verkkohyökkäyksiä vastaan. He ehdottavat, että jokainen sovelluksen tekemä rajapintakutsu tarkistettaisiin tarkastelemalla kutsujan ja kutsuttavan suhdetta sekä niiden syötettä ja tulostetta ennen kuin kutsu sallitaan. Padekar ym. (2016) pitävät rajapinnan käytön valvontaa tärkeänä, koska sillä voidaan ehkäistä erilaisia uhkia, kuten muistin väärinkäyttöä, palvelunestohyökkäyksiä ja koneen sijainnin kaappausta.

Kerroksellisessa arkkitehtuurissa täytyy siis kiinnittää huomiota kerrosten tietoturvan lisäk-

si myös niiden välisten rajapintojen tietoturvaan. Scott-Hayward (2015) toteaa, että uudemmat ohjaimet tukevatkin SSL/TLS-salausta hallinta- ja verkkoelementtikerroksien viestinnän salaamiseksi, mutta lisäksi tarvitaan suojaa IP-kerroksen hyökkäyksiltä, kuten IP-osoitteen väärentämiseltä, TCP SYN -hyökkäykseltä ja palvelunestohyökkäyksiltä.

5 Palvelunestohyökkäykset

Palvelunestohyökkäykset (engl. Denial-of-Service, DoS) ovat yksi suurimpia tietoturvaongelmia Internetissä. Palvelunestohyökkäyksen perusperiaatteena on, että hyökkääjä lähettää palvelimelle niin paljon dataa, että oikeat käyttäjät eivät voi hyödyntää palvelua. Hajautetut palvelunestohyökkäykset (engl. Distributed Denial-of-Service, DDoS) ovat vieläkin vaikeampia ehkäistä ja torjua. Saatavilla on ilmaisia ja käyttäjäystävällisiä työkaluja, joiden avulla palvelunestohyökkäyksen toteuttaminen on helppoa kenelle tahansa. (Mitrokotsa ja Douligeris 2007) Suurin palvelunestohyökkäyksen motivaattori on verkkopelaaminen, toisena rikollisten kykyjen esittely ja kolmantena kiristys. (Alcoy ym. 2018)

Palvelunestohyökkäykset ja niiltä suojautuminen on ajankohtainen tutkimusaihe. Arbor Networksin vuoden 2017 turvallisuusraportin mukaan Internet-palveluntarjoajien suurin ja yritysten toiseksi suurin uhka oli hajautetut palvelunestohyökkäykset. Teollisen Internetin laitteiden hyödyntäminen hyökkäyksissä ja tarvittavien resurssien ja tekniikoiden helppo saatavuus antavat syytä huoleen. Palvelunestohyökkäykset aiheuttavat sekä rahallisia menetyksiä että vahinkoa yritysten ja palveluntarjoajien maineelle. (Alcoy ym. 2018)

Yritykset halusivat hyödyntää turvallisuuden parantamisessa myös ohjelmisto-ohjattua tietoverkkoarkkitehtuuria. Arkkitehtuurin käyttöönottoa jarrutti merkittävästi turvallisuuteen liittyvät huolet, joten sen tietoturvaan on aihetta panostaa. (Alcoy ym. 2018)

5.1 Tavat

Palvelunestohyökkäyksen tavoitteena on haitata jonkin tietoverkon resurssin käyttöä, esimerkiksi kuluttaa tietoverkon kaistanleveyttä tai estää palvelimeen yhdistäminen. Hyökkääjä voi lähettää verkkoon suuren määrän liikennettä, jolloin kaistanleveyttä jää vähemmän oikeiden käyttäjien käyttöön. Hyökkääjä voi myös tehdä monia palvelupyyntöjä samalle palvelimelle, jolloin palvelin ei ehdi vastaamaan oikeiden käyttäjien pyyntöihin. Jos palvelunestohyökkäyksessä käytetään hyväksi useita Internetiin kytkeytyneitä koneita hyökkäämään yhtä kohdetta vastaan, puhutaan hajautetusta palvelunestohyökkäyksestä. Tällöin hyökkäysliikenteen määrää voidaan kasvattaa, jolloin kohteen on entistä hankalampi käsitellä sitä. (Mitrokotsa

ja Douligeris 2007)

Palvelunestohyökkäyksessä voidaan hyödyntää esimerkiksi verkkolaitteiden ohjelmistojen haavoittuvuuksia tai fyysisten resurssien rajallisuutta, käyttöjärjestelmien protokollatoteutuksia ja palvelimen sovellusten haavoittuvuuksia. Myös IP-osoitteen väärentämistä ja nimipalvelujärjestelmän (engl. Domain Name System, DNS) palvelinten saastuttamista käytetään hyväksi hyökkäyksissä. (Mitrokotsa ja Douligeris 2007)

Teollisen Internetin laitteiden määrä kasvaa, ja samalla kasvaa laitteiden haavoittuvuuksien määrä. Internetiin kytketyt laitteet ovat usein heikommin suojattuja kuin työpöytätietokoneet. IoT-laitteista voidaan koota bottiverkko (engl. botnet), jolla voidaan generoida suuria määriä liikennettä kohdetta vastaan. Laitteiden hyödyntäminen palvelunestohyökkäyksissä vaatii osaamista, jota asiantuntijat myyvät hyökkääjille. Uusimmilla tekniikoilla hyökkääjä voi valjastaa käyttöönsä myös laitteita, jotka ovat suojassa organisaation tietoverkon sisäpuolella. (Alcoy ym. 2018)

5.2 Tunnistaminen ja torjunta

Internet itsessään ei osaa puolustautua palvelunestohyökkäyksiä vastaan, joten yhden Internetiin liittyneen koneen suojaaminen ei vielä riitä. Sekin on hankalaa, koska jokaisella koneella on rajalliset resurssit, jotka hyökkääjä pystyy halutessaan kuluttamaan loppuun. Tehokkaan suojausjärjestelmän kehittäminen on vaikeaa, koska sen pitäisi olla itsessään suojattu, tunnistaa hyökkäykset tarkasti ja soveltua nykyisessä Internet-verkossa käytettäväksi. (Mitrokotsa ja Douligeris 2007)

Palvelunestohyökkäyksen liikenteen tunnistaminen on hankalaa, koska hyökkääjä yleensä vaihtelee pakettien otsaketietoja. Siksi hyökkäykseen kuuluvia paketteja ei voida erotella minkään tietyn piirteen perusteella. IP-osoitteita väärentämällä hyökkääjä voi peittää jälkensä, eikä hyökkäystä pystytä yhdistämään tiettyyn henkilöön. (Mitrokotsa ja Douligeris 2007)

5.3 Palvelunestohyökkäykset ohjelmisto-ohjatussa tietoverkossa

Tutkijoitten Yan ym. (2016) mukaan ohjelmisto-ohjattu tietoverkko on palvelunestohyökkäyksien kannalta kahtalainen: toisaalta se on itse altis hyökkäyksille, toisaalta sitä voidaan käyttää hyväksi hyökkäyksiltä suojautumisessa. He ovat koonneet arkkitehtuurille ominaisia palvelunestohyökkäyksen muotoja ja huomanneet, että kaikki arkkitehtuurin tasot ja rajapinnat ovat niille alttiita. Koska hallintataso on erityisen kriittinen verkon toiminnan kannalta, se on houkutteleva palvelunestohyökkäyksen kohde. (Yan ym. 2016)

Ohjelmisto-ohjatussa tietoverkossa reitityssääntöjä voidaan asentaa kytkimiin etukäteen (proaktiivisesti) tai vain tarvittaessa (reaktiivisesti). Reaktiivista vuosääntöjen luomista voidaan hyödyntää palvelunestohyökkäyksissä. Reaktiivisessa tavassa kytkimen vastaanottaessa paketin, jota se ei osaa reitittää, se tallentaa paketin puskuriinsa ja lähettää paketin tiedot ohjaimelle. Ohjain lähettää kytkimelle uuden vuosäännön, jonka mukaan kytkin voi reitittää kyseisen vuon paketit jatkossa. (P. Zhang ym. 2016)

Suuri määrä liikennettä ohjelmisto-ohjatussa verkossa voi saada kytkimet lähettämään monia paketteja ohjaimelle reitityspäätöstä varten, jos niiden reitityssääntöjä ei ole etukäteen lisätty kytkimeen. Tällöin ohjaimen prosessointiteho ei välttämättä riitä, ja liikenteen kulku hidastuu kytkinten odottaessa reititysohjeita. Tilannetta voi auttaa ohjaimen hajauttaminen, jolloin kytkimet voidaan jakaa useamman ohjaimen vastuulle. (Ahmad ym. 2015)

Tri ja Kim (2015) sekä Klöti, Kotronis ja Smith (2013) kuvaavat palvelunestohyökkäystä, joka täyttää kytkimen vuotaulut generoimalla tekaistuja paketteja. Ohjain asentaa kytkimeen uuden vuomerkinnän jokaiselle paketille, jonka otsakkeet eroavat edellisistä, mikä lopulta kuluttaa kytkimen muistin loppuun, eikä uusille vuosäännöille ole enää tilaa. Tri ja Kim (2015) ehdottavat kahta ratkaisua: joko ohjaimen pitäisi pystyä pitämään verkko toiminnassa kytkimen muistin loppumisesta huolimatta tai ohjain voisi väliaikaisesti tallentaa vuomerkintöjä itse ja vaihtaa niitä kytkimeen tarpeen mukaan. P. Zhang ym. (2016) kuitenkin huomauttavat, että OpenFlow-protokollan määritelmä sallii kytkinten poistaa vuosääntöjä itsenäisesti versiosta 1.4 eteenpäin.

Smyth ym. (2016) ovat kehittäneet tavan toteuttaa hajautettu palvelunestohyökkäys käyttämällä hyväksi ohjaimen toimintaa, kun sen pitäisi välittää paketti, jota se ei osaa reitittää.

Tällöin ohjain antaa jokaiselle kytkimelle ohjeen lähettää kyseinen paketti kaikille siihen kytkeytyneille laitteille, jolloin se saattaisi päätyä oikealle vastaanottajalle. Ohjaimen vastaanottaessa suuren määrän tällaisia paketteja verkko kuormittuu laitteiden lähetellessä niitä eteenpäin. Hyökkäys hyödyntää siis verkkoon jo kuuluvia kytkimiä liikenteen lisäämisessä. Se saattaa täyttää kytkinten pakettipuskurit, ylikuormittaa ohjainta ja viedä verkon kaistanleveyttä. Tutkijoitten Smyth ym. (2016) mukaan hyökkäystä voidaan pitää todellisena uhkana ja suurissa verkoissa sillä on luultavasti vakavat vaikutukset.

Kuten Khan ym. (2017) ovat esittäneet, hyökkääjän on mahdollista esiintyä jonain verkon isäntäkoneena huijaamalla ohjainta luulemaan, että kone on vaihtanut paikkaa verkossa. Nguyen ja Yoo (2016) ovat käyttäneet isäntäkoneen sijainnin kaappausta palvelunestohyökkäyksen toteutuksessa: jos hyökkääjä voi ohjata jonkun palvelimen liikenteen itselleen, palvelin ei voi palvella sen asiakkaita.

Tässä tutkielmassa tunnistetut ohjelmisto-ohjatun tietoverkon palvelunestohyökkäykset on koottu taulukkoon 3. Palvelunestohyökkäykset haittaavat ohjaimen tai kytkimen toimintaa kuluttamalla niiden resursseja. Palvelimen sijainnin kaappaus ei varsinaisesti kuluta mitään resurssia, mutta estää silti palvelun saamisen. Hyökkäykset perustuvat yleensä tietynlaisten pakettien lähettämiseen verkkoon suurella nopeudella.

Hyökkäys	**Kohderesurssi**	**Vaikutukset**
Suuri määrä uusia voita	Ohjaimen ja kytkimen välinen kaistanleveys	Vaikeuttaa ohjaimen ja kytkimen viestintää.
	Ohjaimen prosessointiteho	Hidastaa ohjaimen toimintaa.
	Kytkimen muisti	Täyttää kytkimen vuosääntötaulua.
Suuri määrä paketteja tuntemattomille vastaanottajille	Ohjaimen ja kytkimen välinen kaistanleveys	Vaikeuttaa ohjaimen ja kytkimen viestintää.
	Ohjaimen prosessointiteho	Hidastaa ohjaimen toimintaa.
	Kytkimen muisti	Täyttää kytkimen pakettipuskuria.
Palvelimen sijainnin kaappaus		Oikea palvelin ei saa sille tarkoitettua liikennettä.

Taulukko 3: Ohjelmisto-ohjatun tietoverkon palvelunestohyökkäyksiä

6 Tutkimuksen esittely

Luvussa kuvataan tutkimusstrategiaa, koejärjestelyä ja kokeen toteutusta.

6.1 Tutkimusstrategia

Tutkimus toteutettiin käytännön kokeena eli konstruktiivisena tutkimuksena. Konstruktiivinen tutkimus pyrkii löytämään vastauksen reaalimaailman ongelmiin, minkä vuoksi sitä käytetään kasvavassa määrin tekniikan ja tietojärjestelmätieteen aloilla. Tutkimuksen lopputuloksena on konstruktio, joka voi olla mikä tahansa ihmisen keksimä ja kehittämä artefakti. Konstruktio on siis abstrakti käsite, joka voi toteutua lukemattomilla tavoilla. Konstruktiivinen tutkimus keskittyy todellisen maailman ongelmiin, joiden ratkaisemiseksi on käytännön tarve ja tuottaa uuden konstruktion eli ratkaisun ongelmaan. Parhaassa tapauksessa konstruktiivisesta tutkimuksesta on hyötyä sekä teorian että käytännön näkökulmasta. (Lukka 2001)

Tutkijoitten Wohlin ym. (2012) mukaan käytännön kokeita voidaan tehdä silloin, kun halutaan kontrolloida tilannetta ja suoraan ja systemaattisesti vaikuttaa tapahtumiin. Kokeita voidaan tehdä todellisessa maailmassa tai laboratoriossa, joista jälkimmäisessä kokeen tekijällä on yleensä suurempi kontrolli. Koe voi olla ihmissuuntautunut, jolloin ihminen tekee jotain kohteelle; tällöin ihmisen ennustamaton toiminta vähentää kokeen tekijän kontrollia. Teknologiasuuntautuneessa kokeessa kohdetta käsittelee yleensä jokin työkalu, jonka toiminta on determinististä ja siis ennustettavissa. Kokeilla voidaan testata esimerkiksi teorioita tai yleisiä käsityksiä, suhteita asioiden välillä tai mallin tai mittaustavan tarkkuutta. Niillä voidaan myös osoittaa, missä tilanteissa tietyt standardit, toimintatavat tai työkalut ovat käyttökelpoisia. Verrattuna kysely- tai tapaustutkimuksiin käytännön kokeen etuna on mahdollisuus kontrolloida toteutusta ja mittausta. Lisäksi koe voidaan helposti toistaa ja siten osoittaa, että saadut tulokset pätevät laajemmastikin. (Wohlin ym. 2012)

Ohjelmisto-ohjatun tietoverkon ohjaimilla on tehty jonkin verran käytännön kokeita (ks. esim. (Nguyen ja Yoo 2016) ja (Polat ja Polat 2017)). Tootoonchian ym. (2012) ovat kehittäneet erityisen *cbench*-työkalun, jonka avulla voi arvioida vuon alkuasennuksen suoritustehoa ja latenssia. *cbench* emuloi haluttua määrää OpenFlow-kytkimiä, jotka on yhdistetty yhteen

ohjaimeen. Se mittaa vuon alkuasennukseen kuluvaa aikaa, joka todennäköisimmin aiheuttaa ohjaimen suorituskykyyn pullonkaulan. Ohjainten vertailemiseksi on myös kehitetty erityinen menetelmä, jolla voidaan arvioida suorituskyvyn lisäksi myös skaalautuvuutta, turvallisuutta ja käyttövarmuutta (Vengainathan ym. 2018). Esimerkiksi ohjaimen palvelunhyökkäyksestä selviytymistä voidaan testata neljällä mittarilla: 1. reitin alkuasennukseen kuluva aika, 2. suurin määrä reittejä, jotka ohjain pystyy samanaikaisesti asentamaan, 3. ohjaimelta topologian muutoksen havaitsemiseen kuluva aika ja 4. ohjaimen hallitseman verkon suurin koko. Generoimalla TCP SYN -viestejä hallinta- ja verkkoelementtitasojen rajapinnassa voidaan testata ohjaimen kykyä käsitellä suurta määrää kytkinten lähettämiä PacketIn-viestejä.

Oman tutkimukseni kohteena ovat arkkitehtuurin ohjainohjelmistot. Tutkimuksen tarkoitus on arvioida, miten ohjelmistot selviytyvät palvelunestohyökkäyksestä ja verrata niiden suoriutumista toisiinsa. Käytännön kokeen etuna on koeympäristön helppo kontrollointi, jolloin ohjelmistojen tulokset ovat keskenään vertailukelpoisia. Tutkimus hyödyttää sekä ohjainohjelmistojen käyttäjiä että niiden kehittäjiä tietoturvan näkökulmasta. Kokeen tulokset voivat myös motivoida tulevaa tutkimusta ja antaa tietoa koejärjestelyn suunnittelusta. Tarkoitus on tarkkailla ohjelmistojen resurssienkäyttöä palvelunestohyökkäyksen aikana ja sitä, haittaako palvelunestohyökkäys ohjelmistojen normaalia toimintaa. Oletus on, että hyökkäys näkyy ohjelmistojen tarvitsemien resurssien määrien kasvussa ja hitaana toimintana. Lopuksi kunkin ohjelmiston tuloksia verrataan muihin ja katsotaan, voiko joku ohjelmiston ominaisuus selittää yhteneväisyyksiä tai eroja. Tutkimuskysymykset ovat seuraavat:

1. Millaisia tietoturvauhkia ohjelmisto-ohjatun tietoverkkoarkkitehtuurin ohjaimessa on?
2. Paljonko ohjain käyttää suoritintehoa ja muistia palvelunestohyökkäyksen aikana?
3. Miten palvelunestohyökkäys vaikuttaa verkon toimintaan?
4. Miten ohjain selviytyy hyökkäyksestä verrattuna muihin testattuihin ohjaimiin?

6.2 Koejärjestelyt

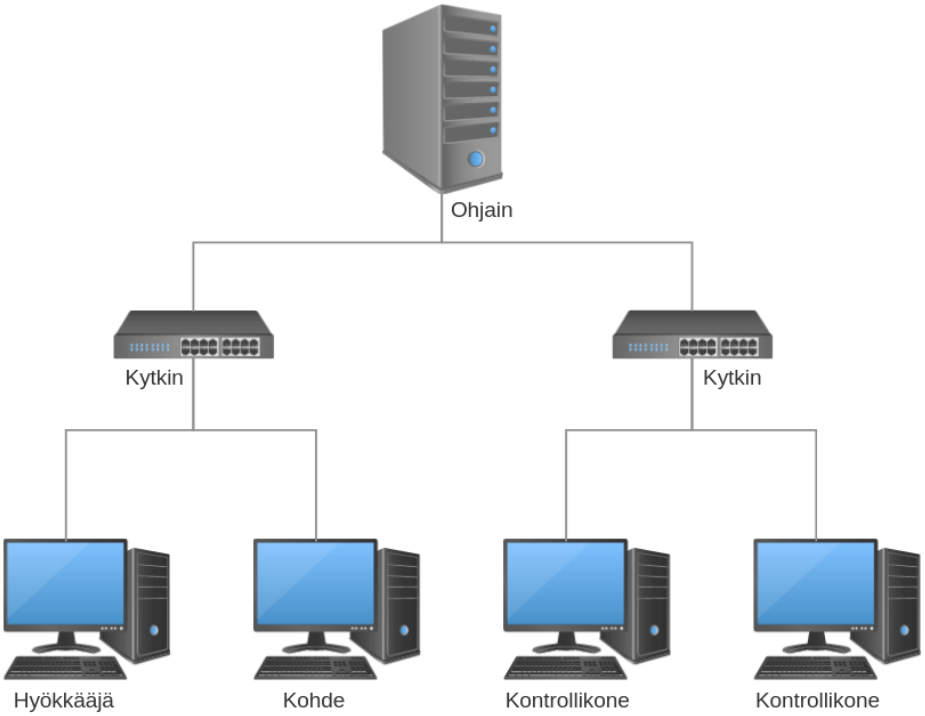
Tutkimukseen valittiin ohjainohjelmistoja, jotka ovat vapaasti saatavissa ja helposti otettavissa käyttöön. Valitut ohjaimet ovat Floodlight, OpenDaylight ja Ryu, joiden tiedot on esitetty taulukossa 4.

Ohjain	Versio	Ohjelmointikieli	Säikeisyys	OF-versiot
Floodlight	1.2	Java	Monisäikeinen	1.0, 1.3
OpenDaylight	Nitrogen SR1	Java	Monisäikeinen	1.0, 1.3
Ryu	4.22	Python	Yksisäikeinen	1.0–1.5

Taulukko 4: Kokeeseen valitut ohjaimet

Koeympäristöön kuuluu kaksi fyysistä tietokonetta. Toinen kone on ohjainkone, jossa ohjainohjelmistoa ajetaan. Siinä on suorittimena 2x Intel Celeron N2840 2.16 GHz, keskusmuistia 8 gigatavua ja käyttöjärjestelmänä Ubuntu Server 16.04.3. Toisessa koneessa on suorittimena 4x Intel Core i3-2367M 1.40 GHz, keskusmuistia 8 gigatavua ja käyttöjärjestelmänä Ubuntu 16.10. Tähän koneeseen asennettiin Mininet-verkkoemulaattori. Mininet (Lantz, Heller ja McKeown 2010) on kehitetty helpottamaan uusien prototyyppien testaamista virtuaaliverkossa simulaattoreiden ja virtuaalikoneiden sijaan. Mininetin etuna on, että sillä voidaan luoda satojen ja tuhansien laitteiden verkkoja yhdellä tietokoneella ja testata prototyyppejä realistisessa ympäristössä. Mininet tukee erityisesti ohjelmisto-ohjatun tietoverkon kehittämistä: verkkoon voidaan lisätä OpenFlow'ta tukevia kytkimiä ja emuloitu ohjain. Ohjain voi olla myös oikea ohjelmisto, jota ajetaan esimerkiksi toisessa koneessa tai pilvessä. Ohjaimelle kytkimet näyttäytyvät fyysisten laitteiden verkkona. (Lantz, Heller ja McKeown 2010)

Kuviossa 3 on esitetty kokeessa käytetty verkko, joka luodaan Mininetin versiolla 2.3.0. Mininetin oletuskytkin on Open vSwitch (*Open vSwitch* 2016) (versio 2.6.1), joka tukee OpenFlow'n versioita 1.0–1.3. Ohjain toimii siis omassa fyysisessä koneessaan, jotta sillä on kaikki koneen resurssit käytettävissään ja muut kokeen osat eivät häiritse mittausta. Kytkimet ja neljä isäntäkonetta luodaan virtuaalisesti toisella fyysisellä tietokoneella. Yksi isäntäkone toimii hyökkääjän koneena ja toinen on hyökkäysliikenteen kohde. Kaksi ylimääräistä konetta toimivat kontrollikoneina, joilla voidaan testata verkon toimivuutta hyökkäyksen aikana.



Kuvio 3: Kokeessa käytetyn verkon topologia.

6.3 Ohjainten asennus

Floodlight-ohjaimen asennuksessa käytettiin ohjelmistosivuston omia ohjeita (*Floodlight Installation Guide* 2016). Ensin asennettiin Java Development Kitin versio 8 ja Ant-työkalu ohjelmiston koostamista (engl. build) varten. Ohjaimen uusin versio ladattiin ja koostettiin suoraan Git-tietovarastosta seuraavilla komennoilla:

```
git clone git://github.com/floodlight/floodlight.git
cd floodlight
git submodule init
git submodule update
ant
```

Kokeissa ohjaimen konsoli tulostettiin tiedostoon. Ohjain käynnistettiin siis seuraavasti:

```
java -jar target/floodlight.jar > <lokitiedosto>
```

Myös OpenDaylight tarvitsee Java Development Kitin version 8 toimiakseen. Ohjain asen-

nettiin OpenDaylight-sivuston ohjeilla (*Installing OpenDaylight* 2018). Ohjelmisto ladattiin sivustolta, jonka jälkeen se käynnistetään Apache Karaf -kontissa. Karaf-komentorivillä asennettin L2 Switch -ominaisuus, joka mahdollistaa Ethernet-kerroksen liikenteenvälityksen.

```
./bin/karaf
feature:install odl-l2switch-switch
```

Ohjaimen loki luetaan suoraan Karafin lokitiedostosta */karaf/data/log/karaf.log*.

Ryu asennettiin suoraan Git-tietovarastosta:

```
git clone git://github.com/osrg/ryu.git
cd ryu; python ./setup.py install
```

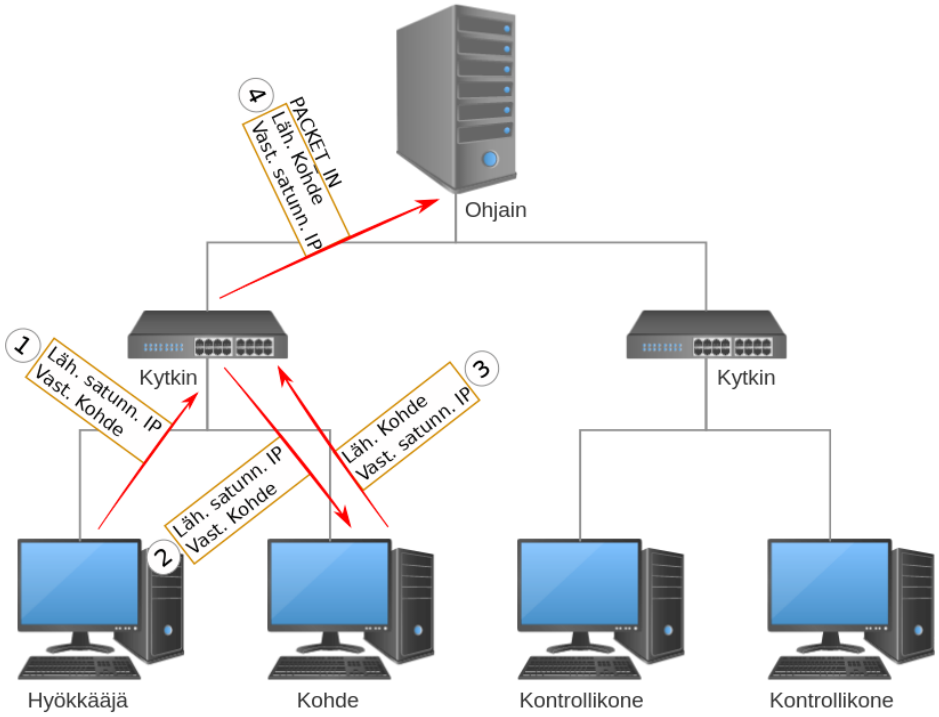
Ryu käynnistetään ohjaimen mukana tulevan L2-tason kytkimen kanssa, ja loki tallennettiin tiedostoon antamalla käynnistyskomennossa valitsin:

```
ryu-manager --verbose ryu/app/simple_switch_13.py
--log-file <lokitiedosto>
```

6.4 Toteutus

Palvelunestohyökkäys voidaan toteuttaa monella tavalla. Tähän kokeeseen on valittu TCP SYN -tulvahyökkäys, jota voidaan soveltaa ohjelmisto-ohjatun tietoverkon ohjaimen kuormittamiseen. Hyökkääjä lähettää TCP-protokollan yhteydenmuodostuspaketteja (SYN) kohdekoneelle porttiin 80. Jokaiseen pakettiin generoidaan satunnaiset lähettäjän MAC- ja IP-osoitteet ja portti. Kun hyökkääjäkoneen liikennettä välittävä kytkin saa tällaisen paketin, se välittää sen kohdekoneelle. Kohdekone lähettää vastauspaketin satunnaisesti generoituun osoitteeseen. Kytkin, joka hoitaa koneen liikennettä, ei tiedä, miten kohdekoneen lähettämä vastauspaketti pitäisi reitittää, koska vastaanottajan osoite on tekaistu. Niinpä se kysyy neuvoa ohjaimelta, jolle lähetetään OpenFlow-protokollan mukainen PacketIn-viesti, joka sisältää reititettävän paketin tiedot. Ohjain vastaa kytkimelle PacketOut-viestillä. Kun hyökkääjä lähettää paketteja tekaistulla lähettäjän tiedoilla tarpeeksi nopeasti, ohjain saa paljon

reititysohjepyyntöjä kytkimeltä ja kuormittuu. Pakettien kulkua on kuvattu kuviossa 4.



Kuvio 4: Pakettien kulku hyökkäyksessä.

Koe alkaa ohjaimen käynnistyksellä. Ohjaimen toiminnan tallentamiseen käytetään ohjaimen tarjoamia lokitiedostoja tai konsolin tulostamista tiedostoon. Samassa koneessa ajetaan ohjaimen resurssien käyttöä tallentava skripti, joka käyttää Linuxin *top*-komentorivityökalua ja */sys*-tiedostojärjestelmää. *top* (*procps* 2018) näyttää reaaliaikaisia tietoja tietokoneessa käynnissä olevista prosesseista. Työkalun avulla mitattiin ohjaimen suoritintehon ja RAM-muistin käyttöä sekä säikeitten määrää. */sys*-tiedostojärjestelmän (Mochel ja Murphy 2011) avulla saadaan tietoja muun muassa tietokoneen laitteista, tässä tapauksessa verkkoliitynnän lähettämistä ja vastaanottamista paketti- ja tavumääristä. Käytetyt polut ovat */sys/class/net/<liityntä>/statistics/rx_bytes*, */sys/class/net/<liityntä>/statistics/rx_packets*, */sys/class/net/<liityntä>/statistics/tx_bytes* ja */sys/class/net/<liityntä>/statistics/tx_packets*. Sekä resurssien käyttöä että liikennemääriä mitataan sekunnin välein.

Toisessa fyysisessä koneessa käynnistetään Mininet-virtuaaliverkko, joka yhdistetään ohjaimeen. Verkon topologiaa varten tehtiin yksinkertainen Python-skripti, jonka avulla Mininet osaa luoda halutunlaisen verkon:

```
from mininet.topo import Topo

class MyTopo( Topo ):
    "Two switches plus two hosts for each switch."

    def __init__( self ):
        "Create custom topo."

        # Initialize topology
        Topo.__init__( self )

        # Add hosts and switches
        h1 = self.addHost( 'h1' )
        h2  = self.addHost( 'h2' )
        h3  = self.addHost( 'h3' )
        h4  = self.addHost( 'h4' )
        leftSwitch = self.addSwitch( 's1' )
        rightSwitch = self.addSwitch( 's2' )

        # Add links
        self.addLink( h1, leftSwitch )
        self.addLink( h2, leftSwitch )
        self.addLink( rightSwitch, h3)
        self.addLink( rightSwitch, h4)

topos = { 'mytopo': ( lambda: MyTopo() ) }
```

Lisäksi Mininetille annetaan komento pitää MAC-osoitteet staattisina hyökkäystyökalun konfiguroinnin helpottamiseksi. Mininet käynnistetään seuraavalla komennolla:

```
sudo mn --custom topo-2sw-4host.py --topo mytopo
--controller=remote,ip=<ohjaimen IP-osoite>,port=<ohjaimen portti> --mac
```

Mininetin käynnistyttyä sen toimintaa voidaan ohjata Mininetin komentorivillä. Komentorivin kautta kahdessa kontrollerikoneessa käynnistetään *iPerf*-työkalu (*iPerf - The ultimate speed test tool for TCP, UDP and SCTP* 2018), jolla voidaan mitata kaistanleveyttä, latenssin vaihtelua (engl. jitter) ja pakettihäviötä kahden koneen välillä. Toinen kontrollikone toimii

palvelimena ja toinen asiakkaana, joiden välistä yhteyttä testataan. *iPerf*-työkalua käytetään sekä TCP- että UDP -asetuksilla, jolloin TCP-yhteydestä voidaan mitata kaistanleveyttä ja UDP-yhteydestä latenssin vaihtelua ja pakettihäviötä. Mittaukset tehdään 30 sekunnin välein.

Lisäksi käytetään Wireshark-pakettianalysaattorin *TShark*-komentorivityökalua (*tshark – The Wireshark Network Analyzer 2.4.6* 2018), jolla kaapataan OpenFlow-protokollaa käyttävät lähetetyt ja vastaanotetut paketit. Kytkimen ja ohjaimen väliseen viestintään kuuluvat Echo-viestit, joiden avulla ne voivat testata yhteyttä. Viestejä tutkimalla voidaan päätellä, milloin ohjain ei enää kuormitukseltaan pysty vastaamaan kytkimen Echo-pyyntöihin.

Ohjaimen annetaan toimia kokeen alussa noin 10 minuuttia tyhjäkäynnillä, siis luomatta mitään liikennettä verkkoon. Tänä aikana ohjain luo yhteyksiä verkon kytkimiin ja selvittää verkon topologiaa, esimerkiksi muiden ohjainten olemassaoloa. Kymmenen minuutin jälkeen hyökkäys käynnistetään. Hyökkäys toteutetaan *Hyenae*-työkalulla (*Hyenae* 2016), joka on verkkohyökkäyksiin erikoistunut pakettigeneraattori. Sen avulla voi helposti toteuttaa erityyppisiä hyökkäyksiä vain muutamaa asetusta vaihtamalla. Esimerkiksi tässä kokeessa käytetty TCP SYN -tulvahyökkäys on sisällytetty työkaluun jo valmiiksi. *Hyenae* käynnistetään komennolla, jonka valitsimet on kuvattu taulukossa 5. Oletuksena työkalu generoi liikennettä niin nopeasti kuin mahdollista.

`hyenae`	**Selitys**
`-i h1-eth0`	Käytettävä verkkoliityntä hyökkäyskoneessa.
`-a tcp`	Hyökkäystyyppi TCP-tulva.
`-f s`	TCP-paketin lippu SYN.
`-s %-%@%%%%`	Satunnainset lähettäjän MAC- ja IP-osoitteet ja portti.
`-d 00:00:00:00:00:02-10.0.0.0.2@80`	Vastaanottajan MAC- ja IP-osoite ja portti.

Taulukko 5: Hyenae-työkalun hyökkäyksessä käytetyt valitsimet ja niiden selitykset.

Hyökkäys kestää noin kahden tunnin ajan, jonka jälkeen *Hyenae*-työkalu pysäytetään. Mit-

tausta jatketaan vielä noin kymmenen minuuttia, jonka jälkeen ohjainohjelmisto ja mittaustyökalut pysäytetään. Kunkin ohjaimen kohdalla koe toistetaan samanlaisena kolme kertaa, ja kokeiden tuloksista lasketaan keskiarvo.

7 Tulokset

Seuraavassa esitetään kunkin ohjainohjelmiston koetulokset omassa aliluvussaan. Kokeen alusta puhuttaessa tarkoitetaan ensimmäistä kymmentä minuuttia, jonka aikana ei generoida mitään liikennettä. Hyökkäykseksi katsotaan kokeen aikaväli kymmenestä minuutista 130 minuuttiin. Kokeen loppu tarkoittaa aikaväliä 130–140 minuuttia. Kokeista puhuttaessa tarkoitetaan kolmen kokeen keskiarvoisia tuloksia. Kuvaajissa näkyy kolmesti toistetun kokeen keskiarvoinen tulos. Suoritintehon käytön kuvaajissa on huomattava, että Floodlight ja OpenDaylight ovat monisäikeisiä ja käyttävät kahta ydintä (suoritintehon käytön maksimi 200 %), Ryu yksisäikeisenä vain yhtä (100 %).

7.1 Floodlight

Kokeissa *iPerf*-työkalun kaistanleveysmittaus päättyi noin 23,7 minuutin kohdalla ja pakettihäviö- ja latenssinvaihtelumittaukset noin 28,3 minuutin kohdalla. Toisessa kokeessa UDP-yhteys säilyi 35 minuuttiin asti, kun muissa kokeissa se päättyi 22 ja 28 minuutin kohdalla. TCP-yhteyden katkeamisessa erot olivat pienemmät, 25, 23 ja 23 minuuttia. Kontrollikoneiden välinen yhteys voidaan siis katsoa katkenneeksi viimeistään 35 minuutin kohdalla. Ohjain lähetti viimeisen PacketOut-viestin noin 37, 56 ja 27 minuutin kohdalla, joten se oli toimintakykyinen vielä kontrollikoneiden välisen yhteyden katkeamisen jälkeen. Kokeen aikana vastaanotetut ja lähetetyt pakettimäärät on koottu taulukkoon 6.

Myös Alharbi, Layeghy ja Portmann (2017) mittasivat Floodlightin pakettihäviötä ja suoritintehon käyttöä samankaltaisessa palvelunestohyökkäyksessä. He mittasivat Floodlightin suoritintehon käytöksi noin 40–45 prosenttia, kun hyökkäyksen pakettien määrä oli 1000–2000 pakettia sekunnissa. Tässä tutkimuksessa Floodlight vastaanotti enimmillään 1500–2000 pakettia sekunnissa, jona aikana suoritintehon käyttö nousi maksimiinsa. Tutkijoitten Alharbi, Layeghy ja Portmann (2017) kokeessa pakettihäviö alkoi nousta vasta hyökkäysnopeuden ylittäessä 2000 pakettia sekunnissa; tässä tutkimuksessa mitattiin suurempia pakettihäviöitä hyökkäysnopeuden ollessa 500–1500 pakettia sekunnissa. Koe suoritettiin Mininetverkossa ja kaikki isäntäkoneet olivat yhden kytkimen takana. Erot tämän tutkimuksen ko-

keen ja tutkijoitten Alharbi, Layeghy ja Portmann (2017) kokeen välillä saattaa johtua käytettyjen fyysisten tietokoneiden tai verkon topologian eroista.

Kaikkien kokeiden loppupuolella Floodlight-ohjaimen konsoliin alkoi tulostua Javan virheilmoitus muistin loppumisesta: "GC overhead limit exceeded". Tämä tarkoittaa, että ohjelma käyttää 98 prosenttia ajastaan roskienkeruuseen eikä onnistu siinä. Ohjelman käyttämään Java-kekoon ei siis saada tehtyä tilaa uusille allokoinneille. (*Java Platform, Standard Edition Troubleshooting Guide: Understand the OutOfMemoryError Exception* 2018)

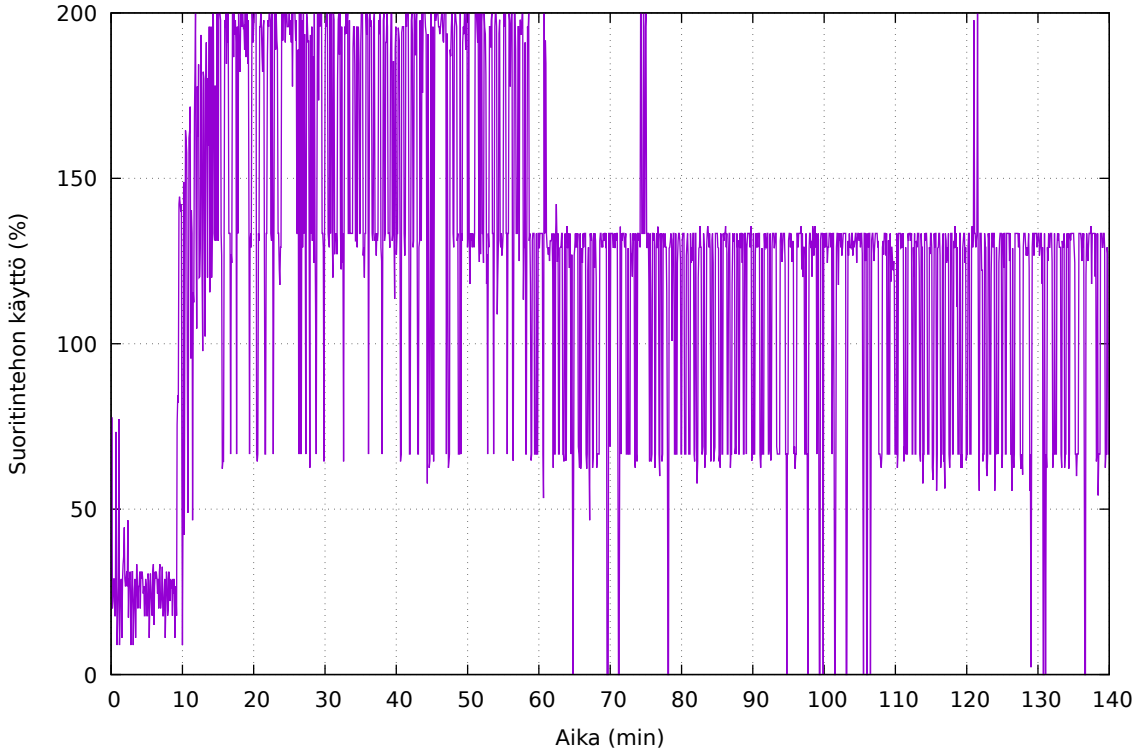
Hyökkäyksen kestettyä 120 minuuttia se pysäytettiin. Floodlight-ohjain ei palannut toimintakykyiseksi kokeen viimeisten 10 minuutin aikana. Ohjaimen tulokset on koottu taulukkoon 7 ja kuvioihin 5–14.

	Koe 1	**Koe 2**	**Koe 3**	**Keskiarvo**
Vastaanotetut paketit	95 928	92 759	91 739	93 475
Lähetetyt paketit	4085	4409	3070	3835
PacketIn-viestit	86 926	78 231	78 941	80 916
PacketOut-viestit	2777	1917	1316	1833
PacketIn-viestit hyökkäyksessä	86 516	77 292	78 565	80 791
PacketOut-viestit hyökkäyksessä	2370	1407	907	1561
Viimeinen PacketIn	36,68 min	79,91 min	31,85 min	49,50 min
Viimeinen PacketOut	36,53 min	56,39 min	26,32 min	39,75 min

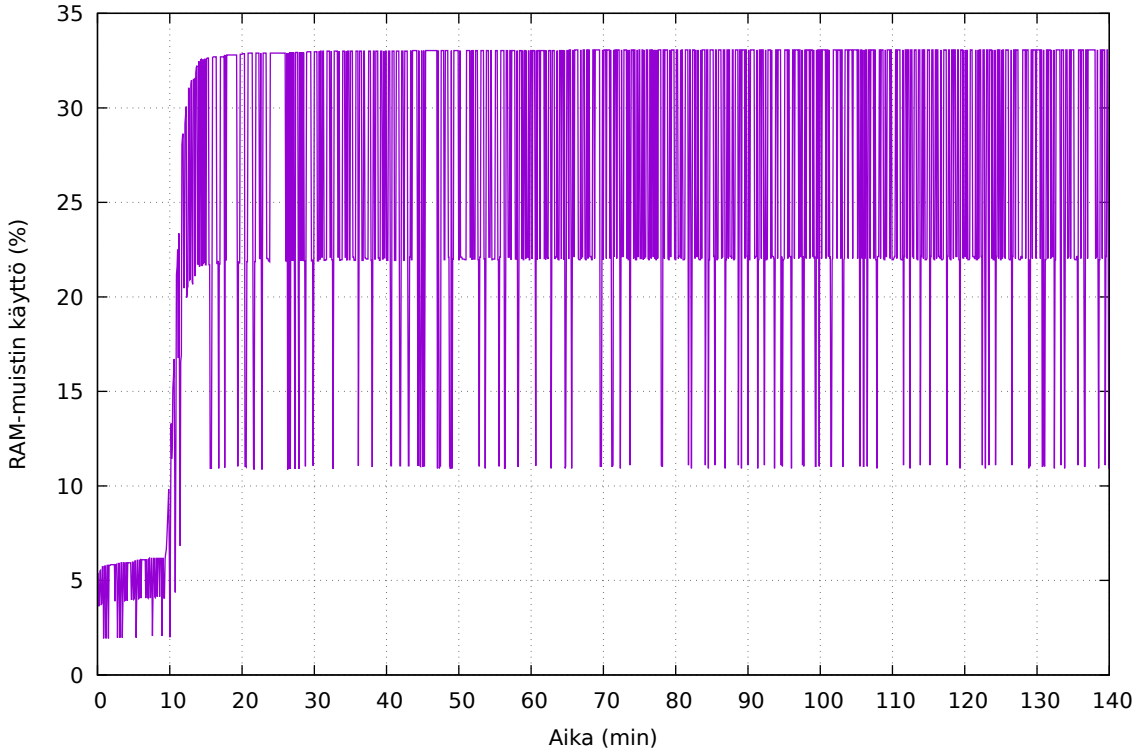
Taulukko 6: Floodlight-ohjaimen vastaanottamat ja lähettämät paketit kokeittain.

	Kokeen alku	**Hyökkäys**	**Kokeen loppu**
Suoritintehon käyttö (%)	30,85	195,03	196,16
RAM-muistin käyttö (%)	5,89	32,43	32,8
Säikeet (kpl)	42	52,27	49
Vastaanotetut paketit (kpl/s)	3,33	65,03	22,15
Lähetetyt paketit (kpl/s)	5,33	14,28	7,41
Vastaanotettu liikenne (Mt/s)	0,00	0,06	0,00
Lähetetty liikenne (Mt/s)	0,00	0,01	0,00
Kaistanleveys (Gb/s)	15,63	8,06	-
Latenssin vaihtelu (ms)	24,60	48,02	-
Pakettihäviö (%)	0,16	23,34	-

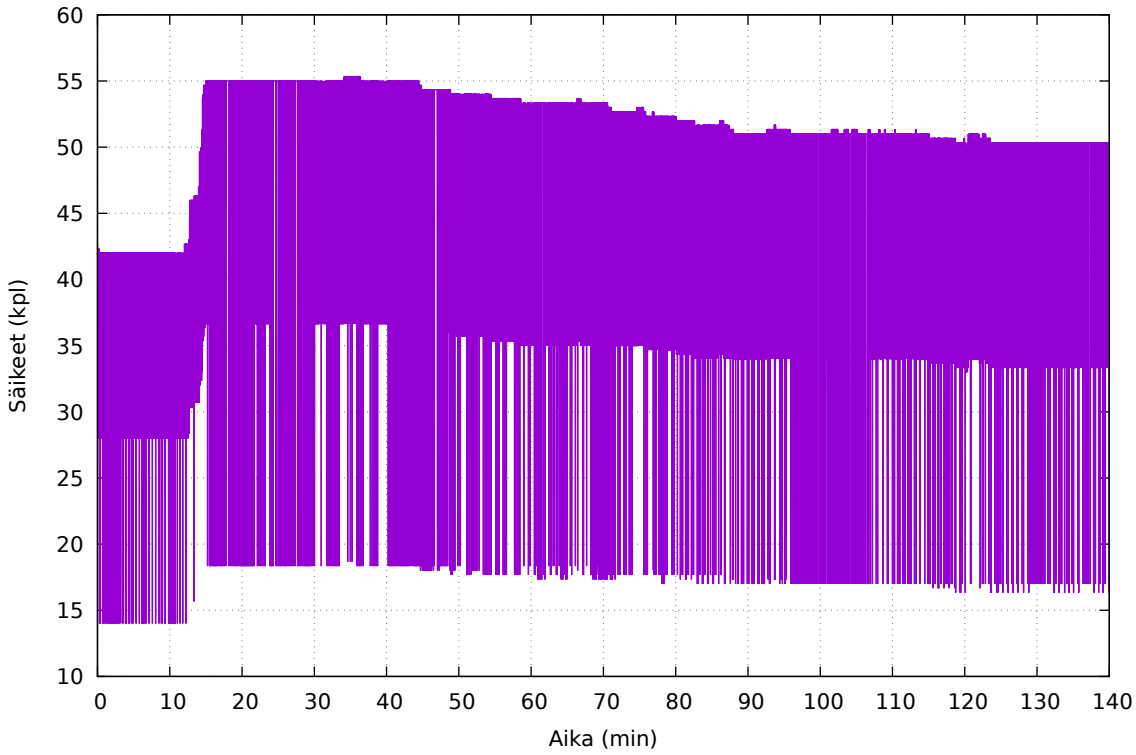
Taulukko 7: Floodlight-ohjaimen keskiarvoiset tulokset.



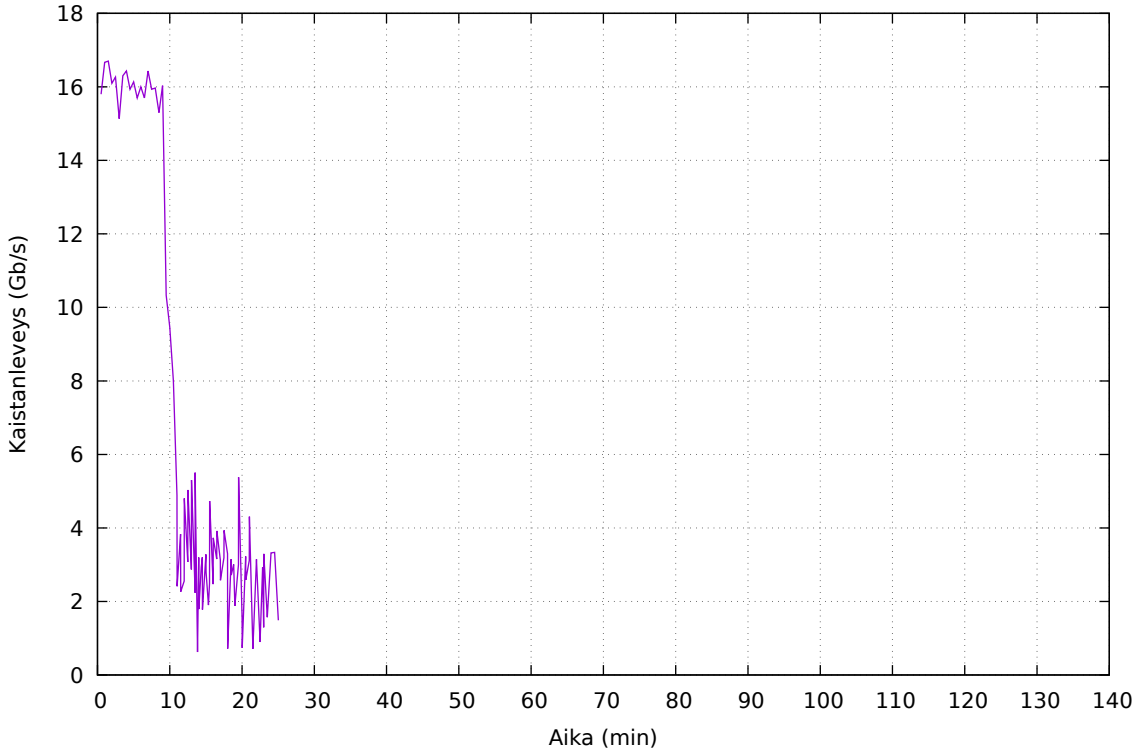
Kuvio 5: Floodlight-ohjaimen suoritintehon käyttö.



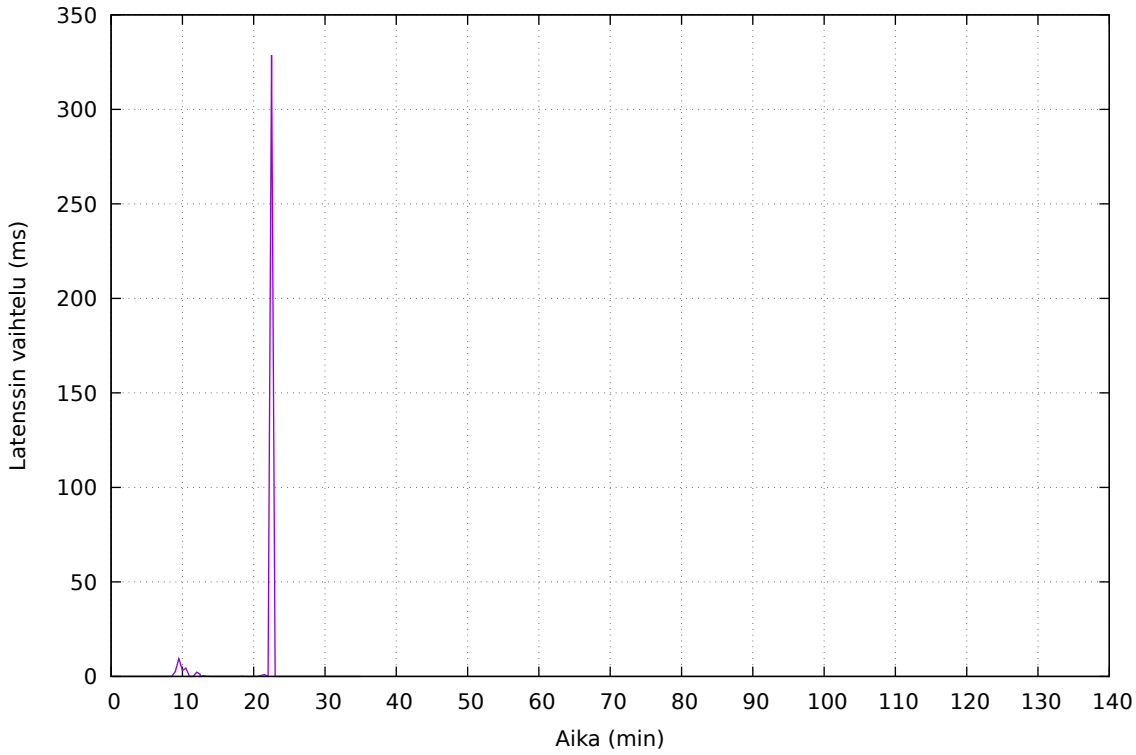
Kuvio 6: Floodlight-ohjaimen RAM-muistin käyttö.



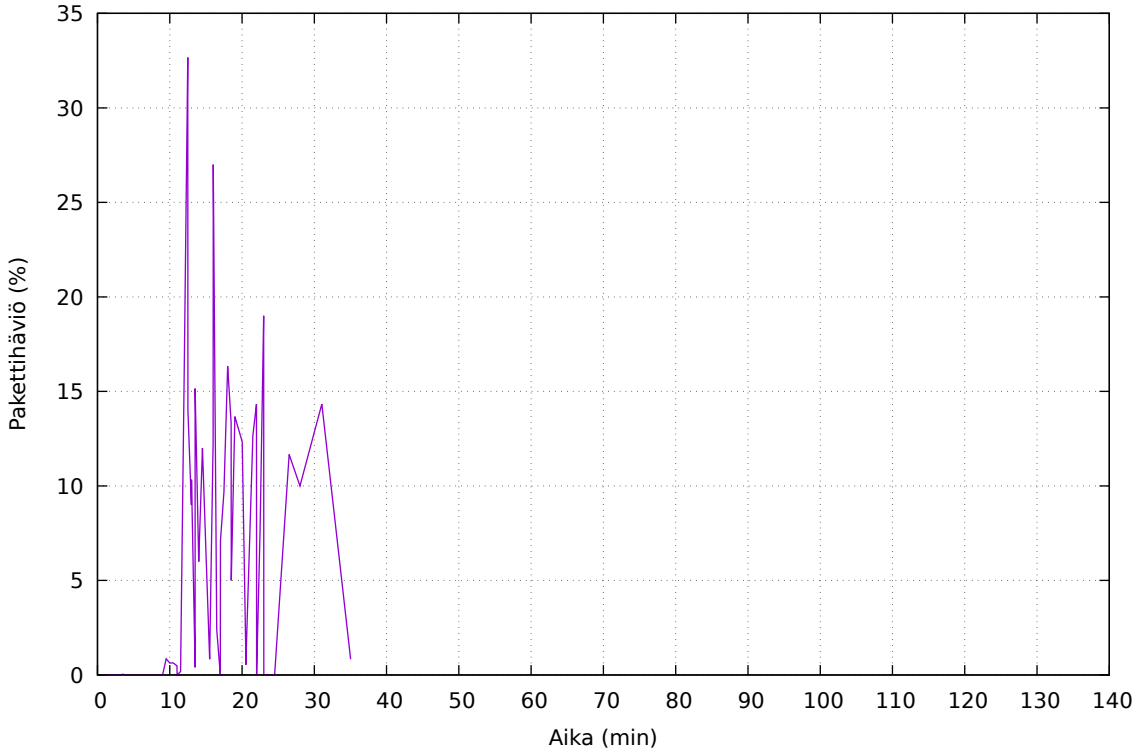
Kuvio 7: Floodlight-ohjaimen säikeitten määrä.



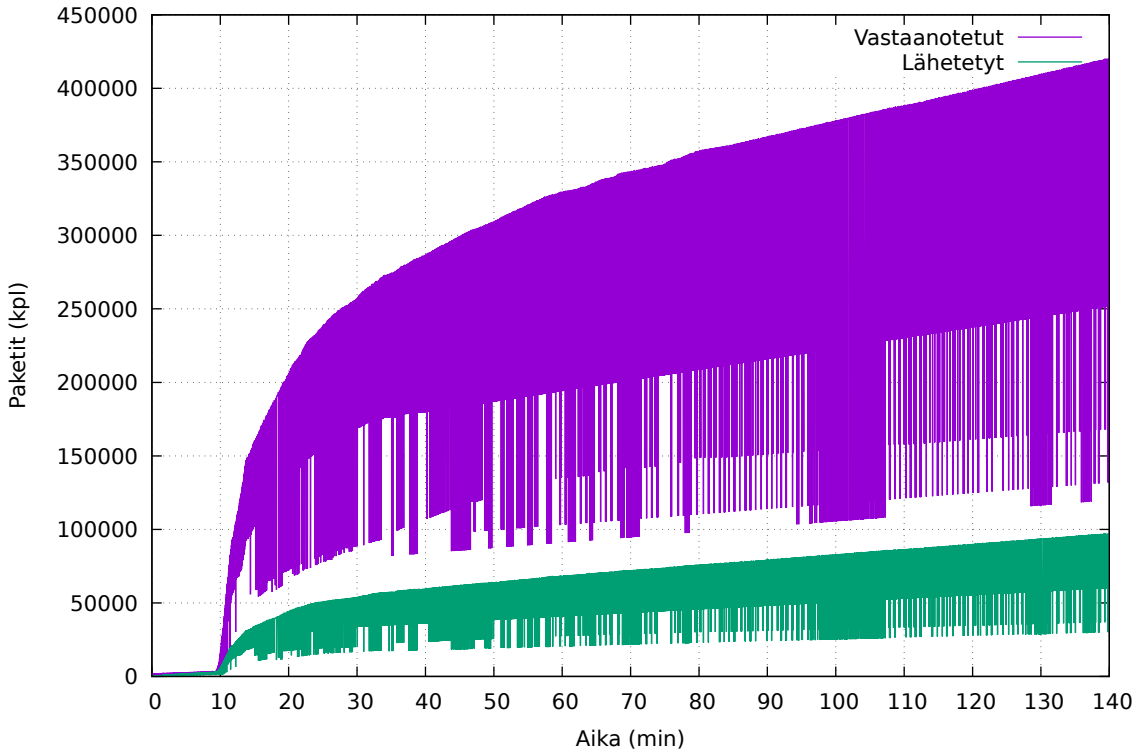
Kuvio 8: Floodlight-ohjaimen verkon kaistanleveys.



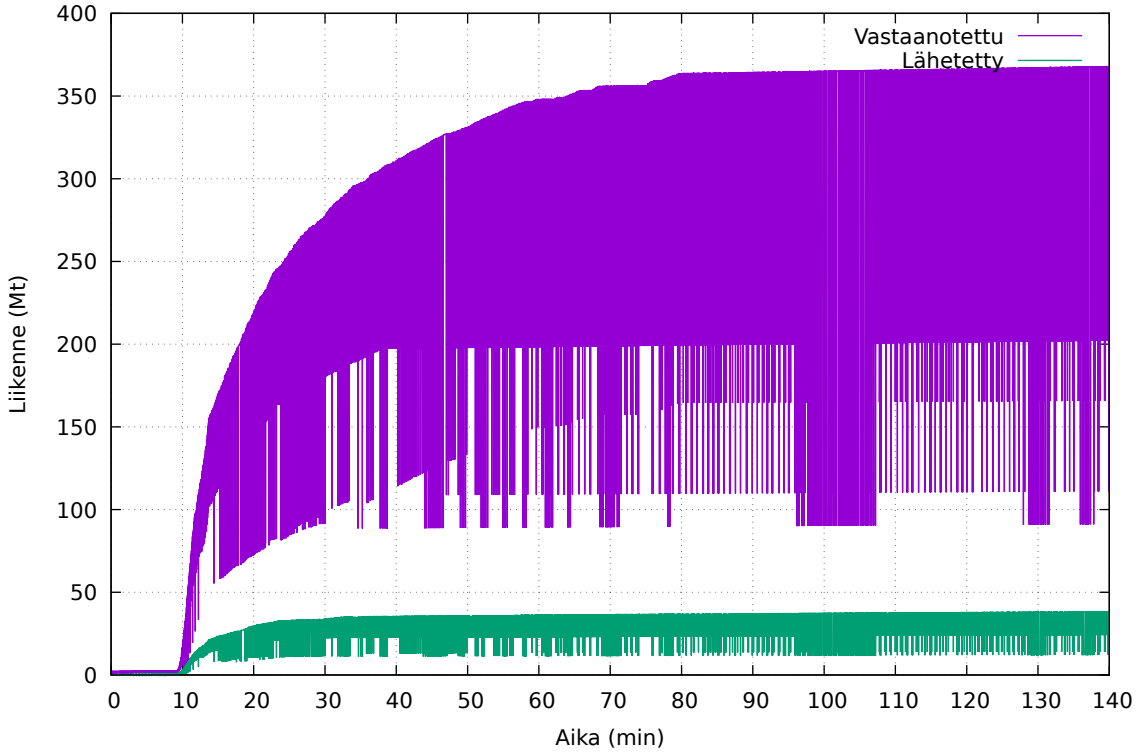
Kuvio 9: Floodlight-ohjaimen verkon latenssin vaihtelu.



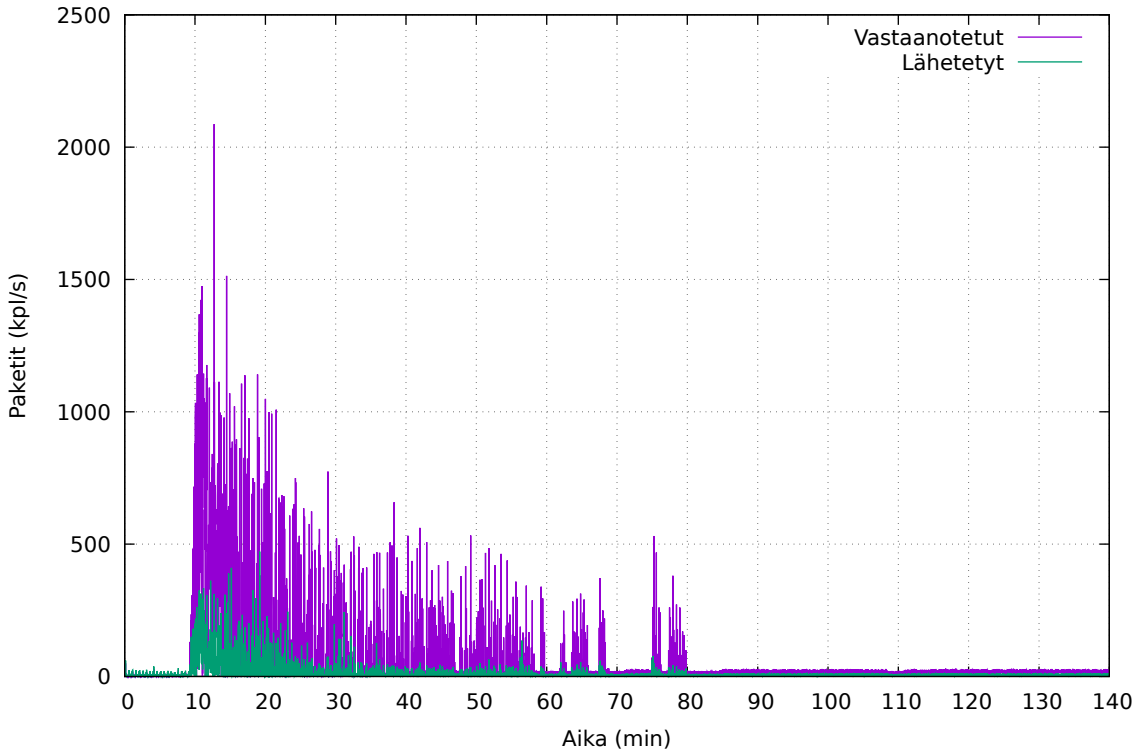
Kuvio 10: Floodlight-ohjaimen verkon pakettihäviö.



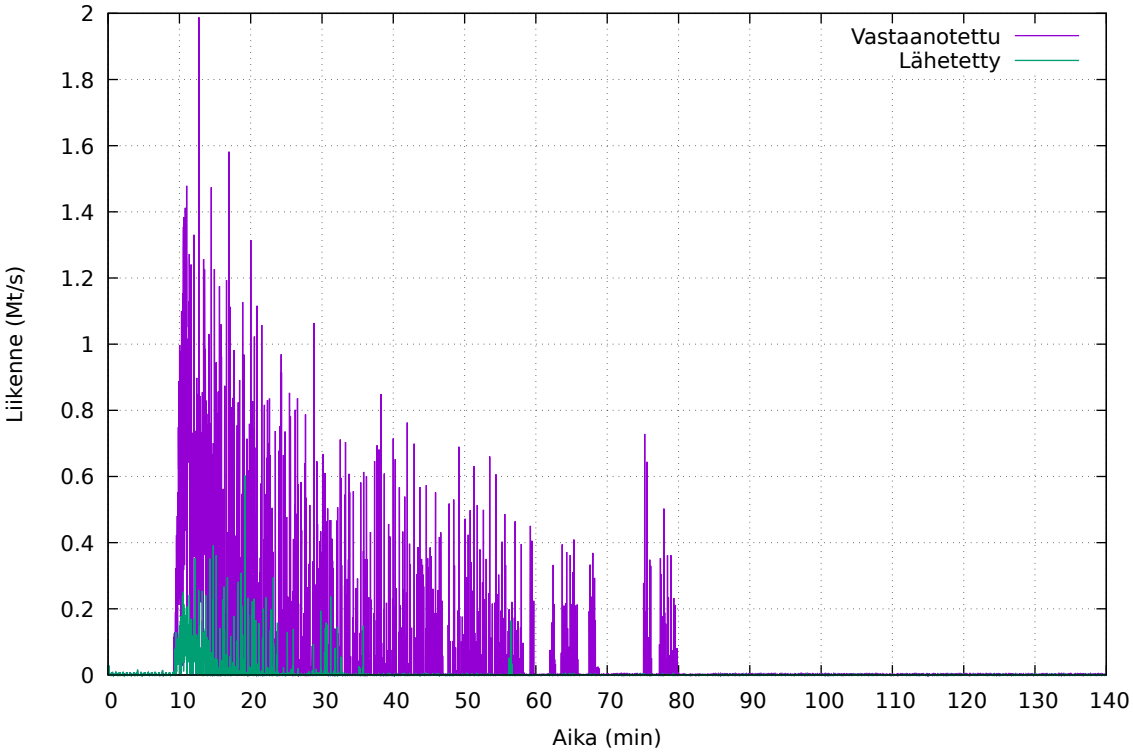
Kuvio 11: Floodlight-ohjaimen vastaanottamat ja lähettämät paketit.



Kuvio 12: Floodlight-ohjaimen vastaanottama ja lähettämä liikenne.



Kuvio 13: Floodlight-ohjaimen vastaanottamat ja lähettämät paketit sekunnissa.



Kuvio 14: Floodlight-ohjaimen vastaanottama ja lähettämä liikenne sekunnissa.

7.2 OpenDaylight

OpenDaylightin kokeissa pakettikaappaukset päättyvät noin 11, 6 ja 1 minuutin kohdalla. Taulukkoon 8 koottujen pakettien määrät eivät siis ole vertailukelpoisia muiden kokeiden kanssa. Kaappausten päättymisen syy lienee eri kuin Ryun tapauksessa (ks. luku 7.3), sillä liikennemäärän pitäisi olla maltillinen ainakin ennen 10:tä minuuttia. Ensimmäisessä kaappauksessa saatiin kuitenkin ohjaimen vastaanottamia paketteja miltei 224 000 kappaletta, joka ei ole linjassa muiden kokeiden kanssa edes suhteessa kaappauksen kestoon. Syy kaappausten katkeamiseen ei selvinnyt.

	Koe 1	**Koe 2**	**Koe 3**	**Keskiarvo**
Vastaanotetut paketit	223 608	997	357	75 087,3
Lähetetyt paketit	3929	1917	227	2044,3
PacketIn-viestit	221 907	87	197	74 063,7
PacketOut-viestit	440	241	22	234,4
PacketIn-viestit hyökkäyksessä	-	-	-	-
PacketOut-viestit hyökkäyksessä	-	-	-	-
Viimeinen PacketIn	11,34 min	4,49 min	0,56 min	5,47 min
Viimeinen PacketOut	11,27 min	5,88 min	0,74 min	5,97 min

Taulukko 8: OpenDaylight-ohjaimen vastaanottamat ja lähettämät paketit kokeittain.

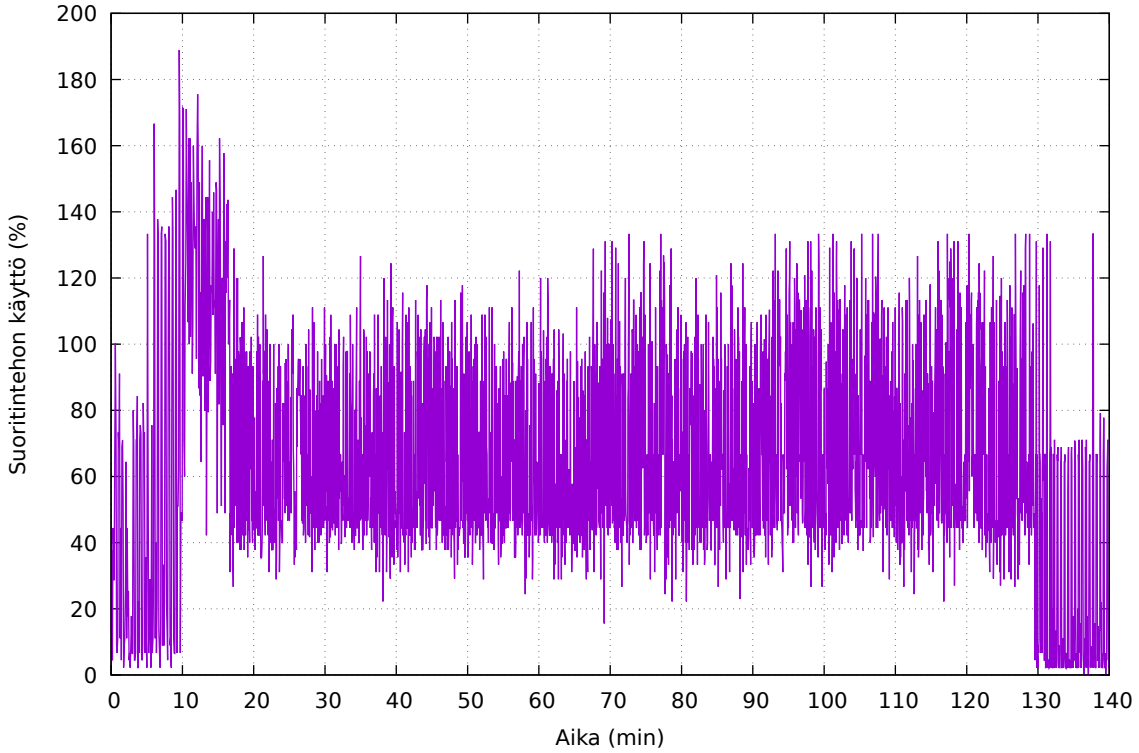
OpenDaylightin säikeiden määrä oli heti kokeen alussa keskimäärin 100,69 kappaletta, joka on miltei kaksinkertainen määrä verrattuna hyökkäykseen ja kokeen loppuun. Säikeitten määrä putoaa noin 16 minuutin kohdalla. Myös suoritintehon ja muistin käyttö olivat samaan aikaan korkeat ja laskevat sitten. Tämä saattaa selittyä ohjaimen käynnistämisessä tarvittavilla resursseilla.

Kontrollikoneiden välinen yhteys säilyi koko kokeen ajan. Latenssin vaihtelulle ja pakettihäviölle ei mitattu nollasta poikkeavia arvoja, eikä tälle löytynyt selitystä. Kokeen lopussa suoritintehon käyttö putosi noin puoleen ja kaistanleveys kasvoi. Muissa mitatuissa arvoissa hyökkäyksen ja kokeen lopun välinen ero jäi pieneksi. Ohjaimen tulokset on koottu taulukkoon 9 ja kuvioihin 15–24.

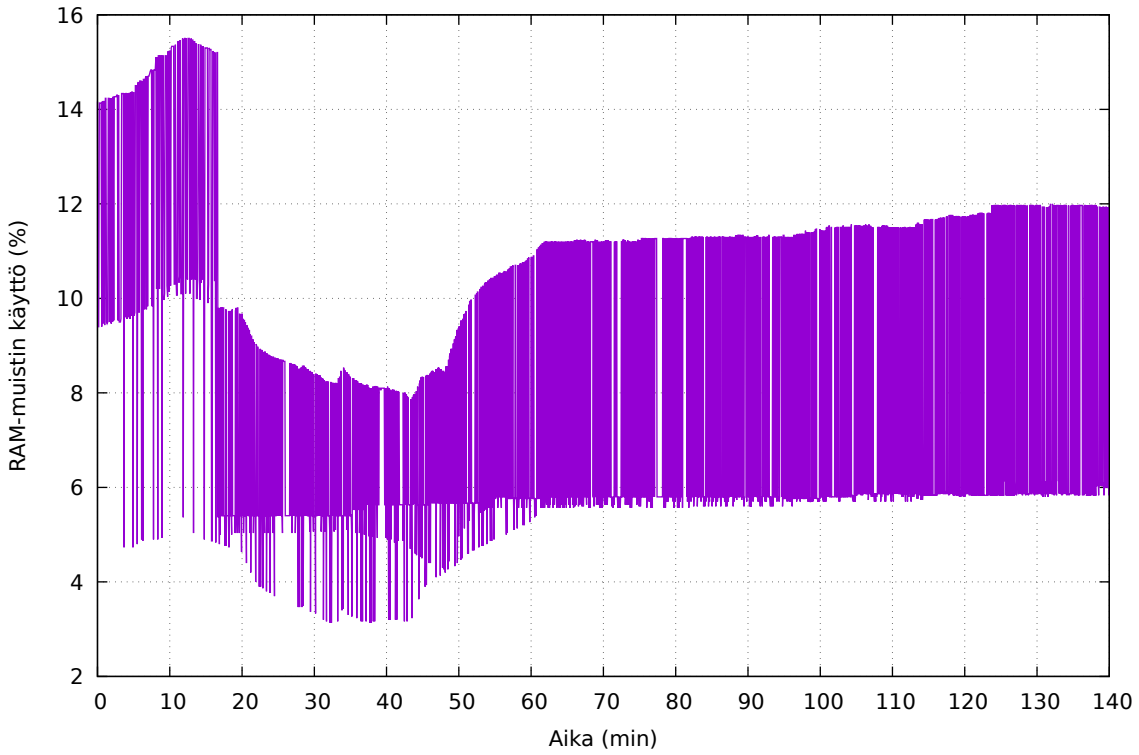
Myös Polat ja Polat (2017) ovat mitanneet OpenDaylightin kaistanleveyttä palvelunesto-hyökkäyksen aikana. Heidän kokeessaan kaistanleveys putosi nollaan noin minuutissa hyökkäyksen alusta, kun taas tässä tutkimuksessa kaistanleveys oli alimmillaankin yli 1 gigabittiä sekunnissa. Myös heidän hyökkäyksensä toteutettiin Mininet-verkossa ja perustui satunnaisesti generoidun lähettäjän aiheuttamiin PACKET_IN-viesteihin. Koejärjestelyt eroavat siinä, että tukijoiden Polat ja Polat (2017) kontrollikoneet sijaitsivat eri kytkinten takana, mikä kenties selittää tulosten eron.

	Kokeen alku	**Hyökkäys**	**Kokeen loppu**
Suoritintehon käyttö (%)	44,37	68,64	29,14
RAM-muistin käyttö (%)	7,66	7,61	8,24
Säikeet (kpl)	100,69	56,24	55,03
Vastaanotetut paketit (kpl/s)	2601,87	2277,46	2405,57
Lähetetyt paketit (kpl/s)	195,30	578,10	155,42
Vastaanotettu liikenne (Mt/s)	3,52	1,60	3,25
Lähetetty liikenne (Mt/s)	0,01	0,04	0,01
Kaistanleveys (Gb/s)	5,98	2,91	4,64
Latenssin vaihtelu (ms)	0,00	0,00	0,00
Pakettihäviö (%)	0,00	0,00	0,00

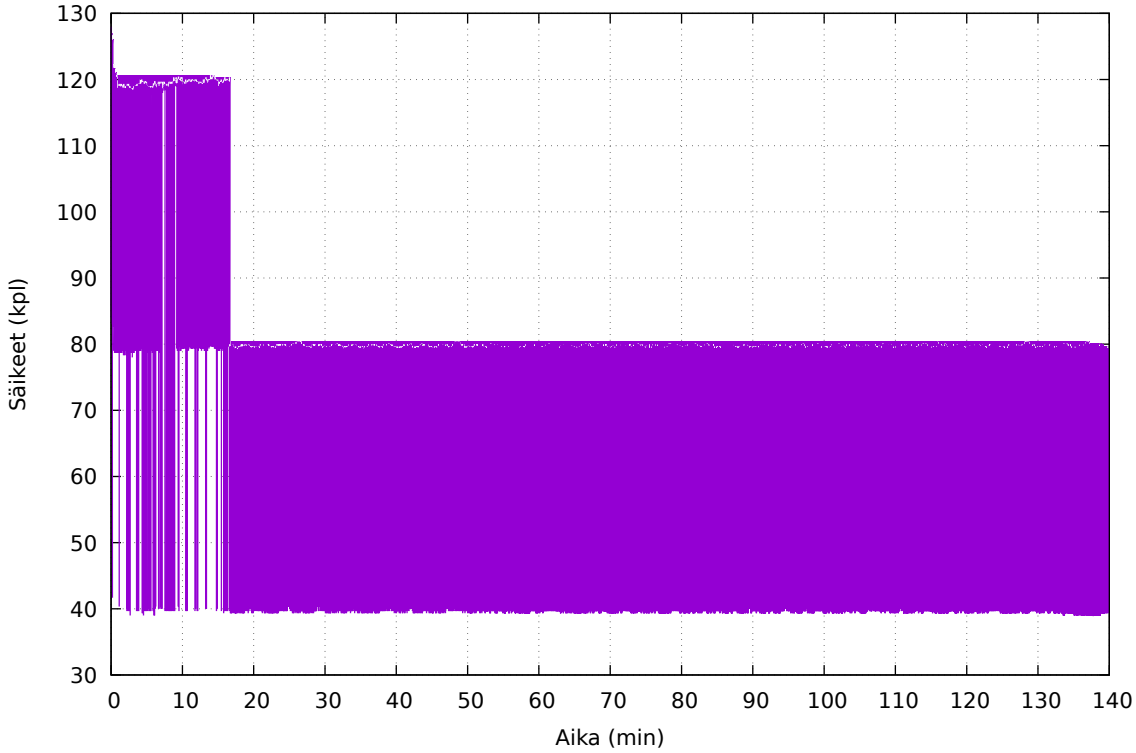
Taulukko 9: OpenDaylight-ohjaimen keskiarvoiset tulokset.



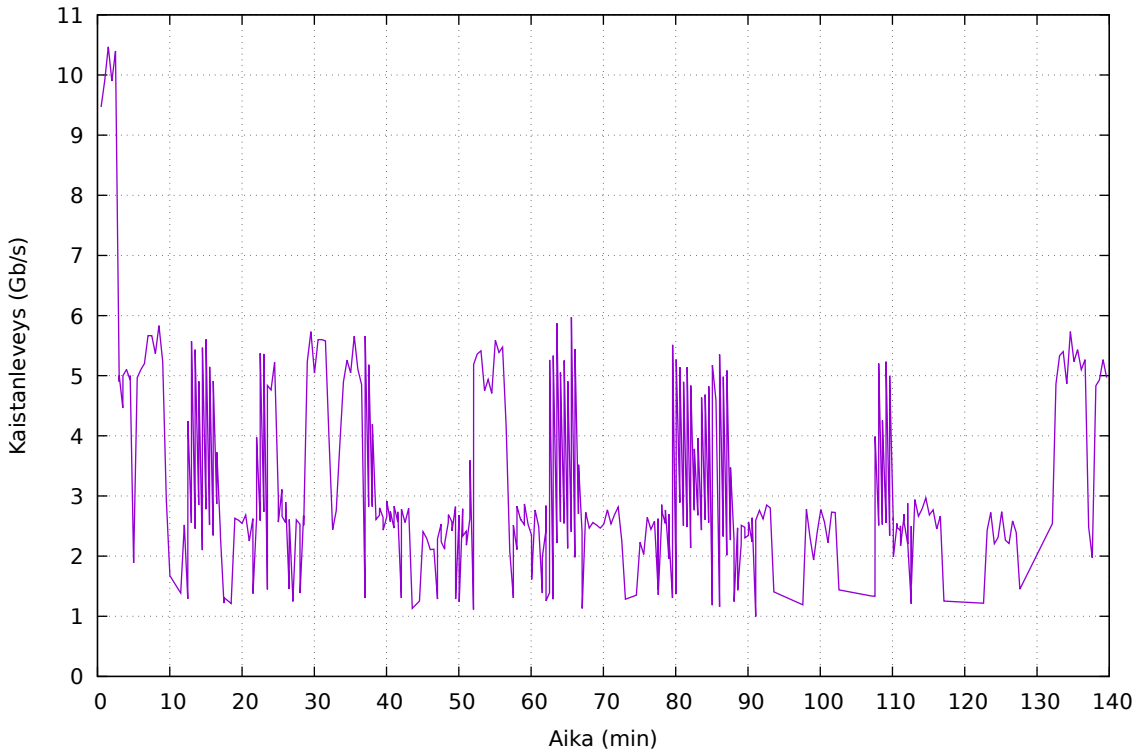
Kuvio 15: OpenDaylight-ohjaimen suoritintehon käyttö.



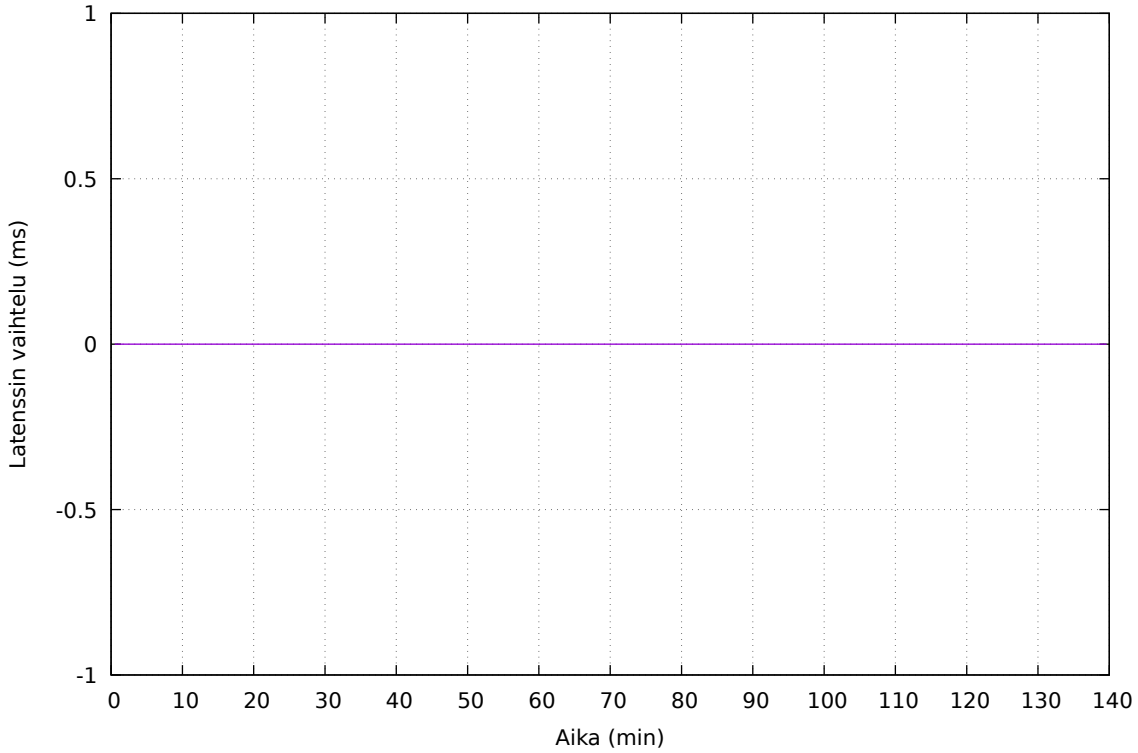
Kuvio 16: OpenDaylight-ohjaimen RAM-muistin käyttö.



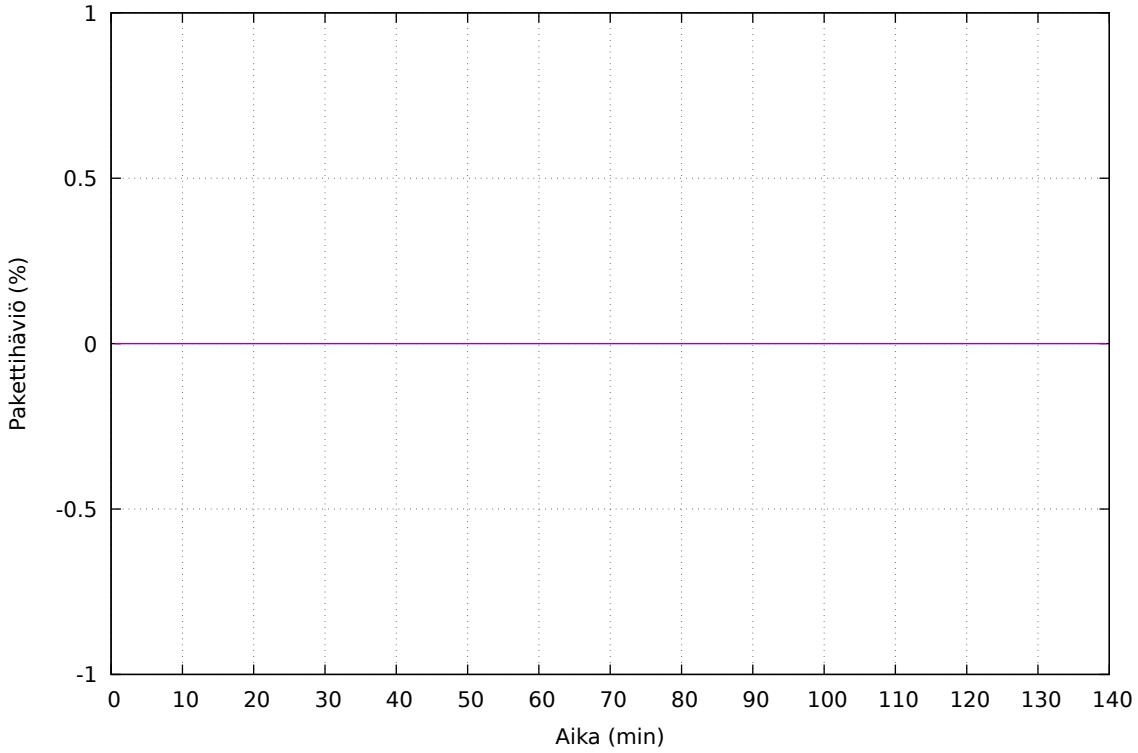
Kuvio 17: OpenDaylight-ohjaimen säikeitten määrä.



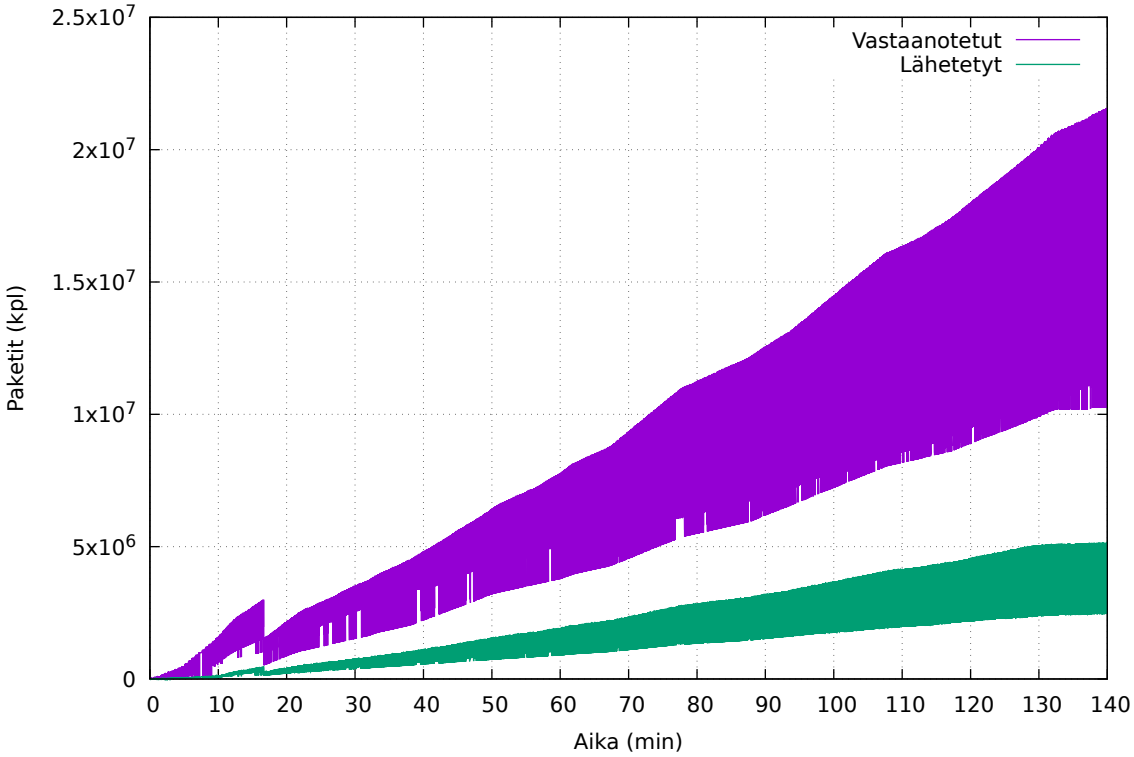
Kuvio 18: OpenDaylight-ohjaimen verkon kaistanleveys.



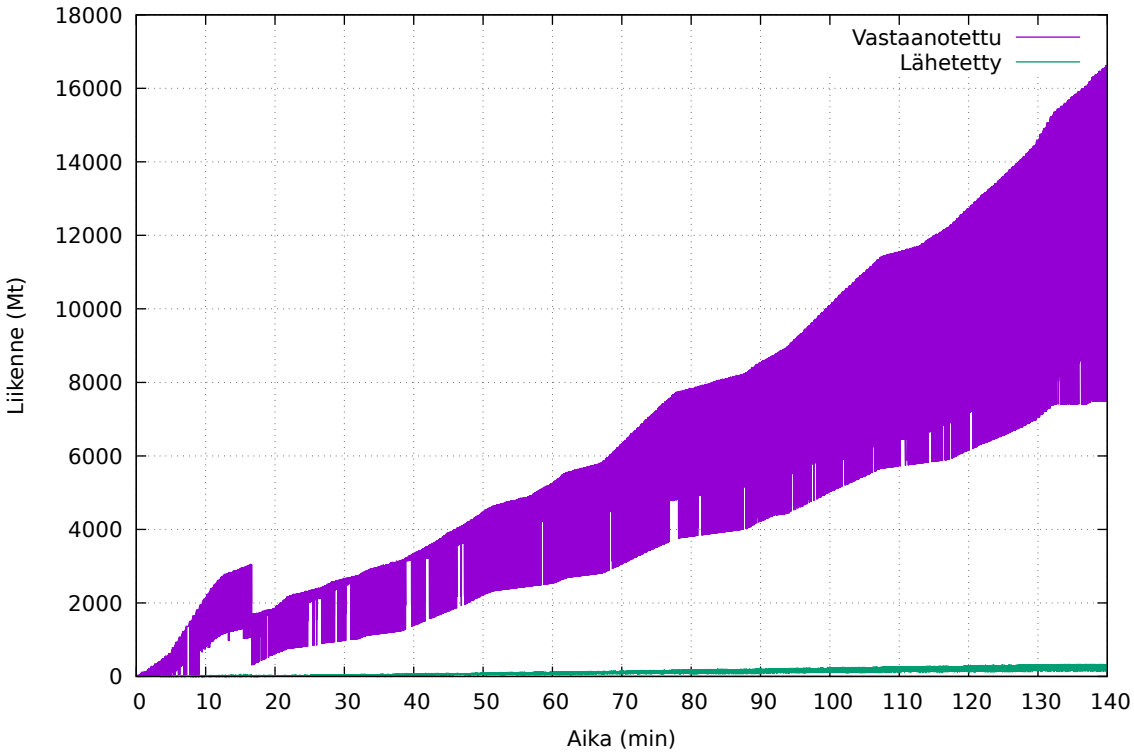
Kuvio 19: OpenDaylight-ohjaimen verkon latenssin vaihtelu.



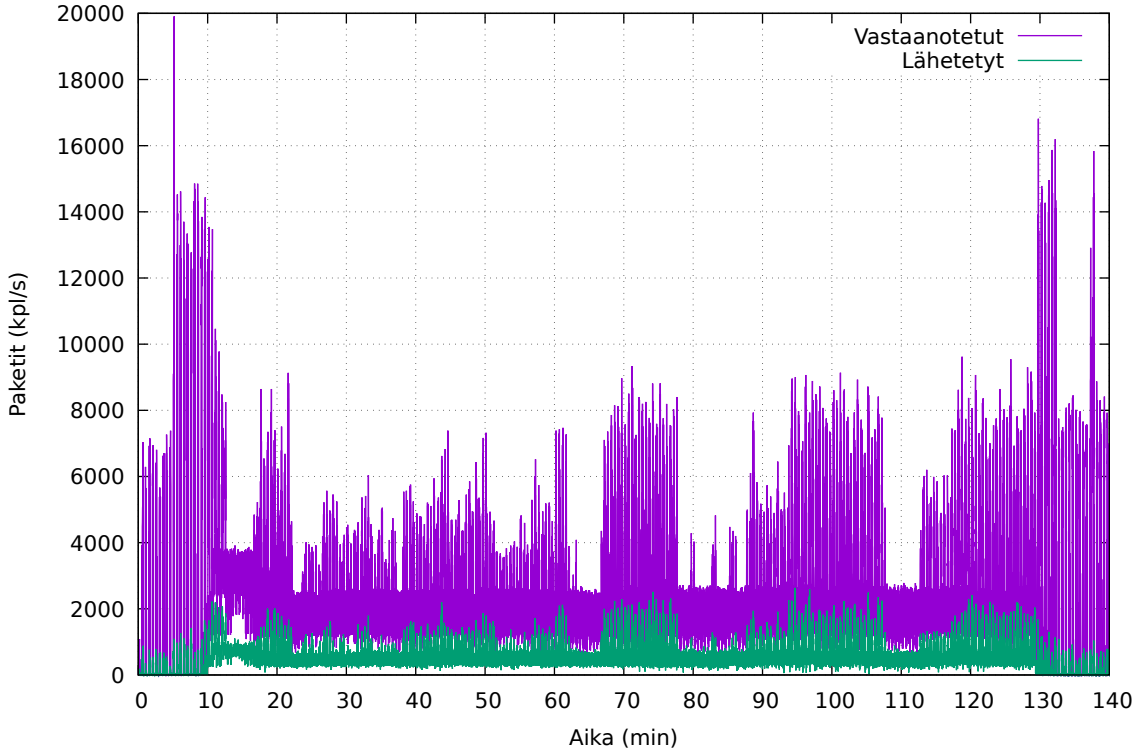
Kuvio 20: OpenDaylight-ohjaimen verkon pakettihäviö.



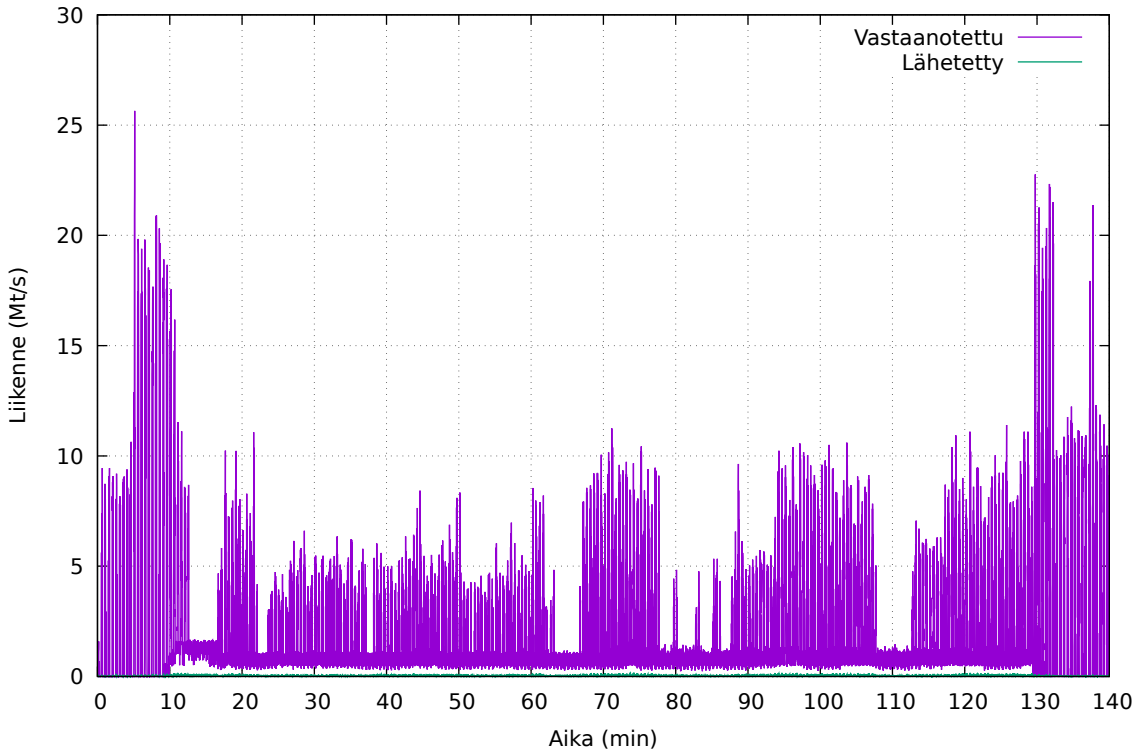
Kuvio 21: OpenDaylight-ohjaimen vastaanottamat ja lähettämät paketit.



Kuvio 22: OpenDaylight-ohjaimen vastaanottama ja lähettämä liikenne.



Kuvio 23: OpenDaylight-ohjaimen vastaanottamat ja lähettämät paketit sekunnissa.



Kuvio 24: OpenDaylight-ohjaimen vastaanottama ja lähettämä liikenne sekunnissa.

7.3 Ryu

Ryu-ohjain käytti kokeen aikana melko vähän suoritintehoa, kenties yksisäikeisyytensä vuoksi. Muistin käyttö kasvoi hyökkäyksen aikana 50 prosenttiin. Kaistanleveys jäi hyökkäyksen päätyttyä vain noin puoleen siitä, mitä se oli kokeen alussa. Ohjaimen keskiarvoiset tulokset on koottu taulukkoon 10 ja kuvioihin 25–33.

	Kokeen alku	**Hyökkäys**	**Kokeen loppu**
Suoritintehon käyttö (%)	0,32	24,57	2,45
RAM-muistin käyttö (%)	0,16	51,08	0,44
Vastaanotetut paketit (kpl/s)	1,18	85,22	0,90
Lähetetyt paketit (kpl/s)	0,22	132,77	0,92
Vastaanotettu liikenne (Mt/s)	0,00	0,04	0,00
Lähetetty liikenne (Mt/s)	0,00	0,08	0,00
Kaistanleveys (Gb/s)	11,48	2,93	5,60
Latenssin vaihtelu (ms)	0,00	0,00	0,00
Pakettihäviö (%)	0,00	0,47	0,43

Taulukko 10: Ryu-ohjaimen keskiarvoiset tulokset.

Ryu lähetti enemmän liikennettä kuin vastaanotti kokeen aikana sekä tavuissa että paketeissa mitattuna. Ilmiö selittynee sillä, että oletusasetuksilla Ryu kaiuttaa tuntemattoman vastaanottajan paketin kaikille (*Ryu, versio 4.24* 2018). Kokeen aikana vastaanotetut ja lähetetyt paketit on koottu taulukkoon 11. Kontrollikoneiden välinen yhteys säilyi koko kokeen ajan. Pakettihäviössä n. 66 minuutin kohdalla mitatulle 300 prosentin piikille ei löytynyt selitystä. Luultavasti arvo on täysin virheellinen, koska se on suurempi kuin 100 prosenttia.

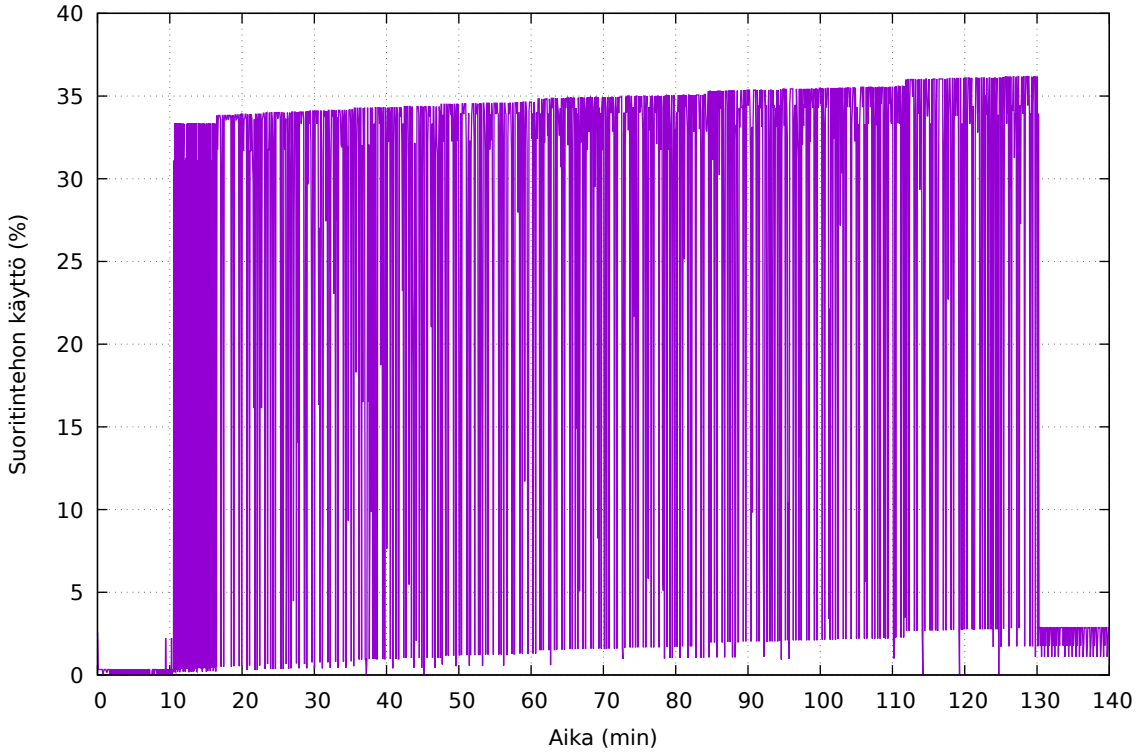
Myös Alharbi, Layeghy ja Portmann (2017) mittasivat Ryun pakettihäviötä ja suoritintehon käyttöä samankaltaisessa palvelunestohyökkäyksessä. He mittasivat Ryun suoritintehon käytöksi noin 20 prosenttia, kun hyökkäyksen pakettien määrä oli 1000 pakettia sekunnissa. Tässä tutkimuksessa Ryu vastaanotti enimmillään 300 pakettia sekunnissa, jona aikana suoritintehon käyttö oli noin 36–37 prosenttia. Tutkijoitten Alharbi, Layeghy ja Portmann (2017) kokeessa pakettihäviö alkoi nousta vasta hyökkäysnopeuden ylittäessä 2000 pakettia

sekunnissa; tässä tutkimuksessa Ryun pakettihäviö pysyi nollassa. Koe suoritettiin Mininet-verkossa ja kaikki isäntäkoneet olivat yhden kytkimen takana, mikä saattaa selittää eroja tämän tutkimuksen tuloksiin.

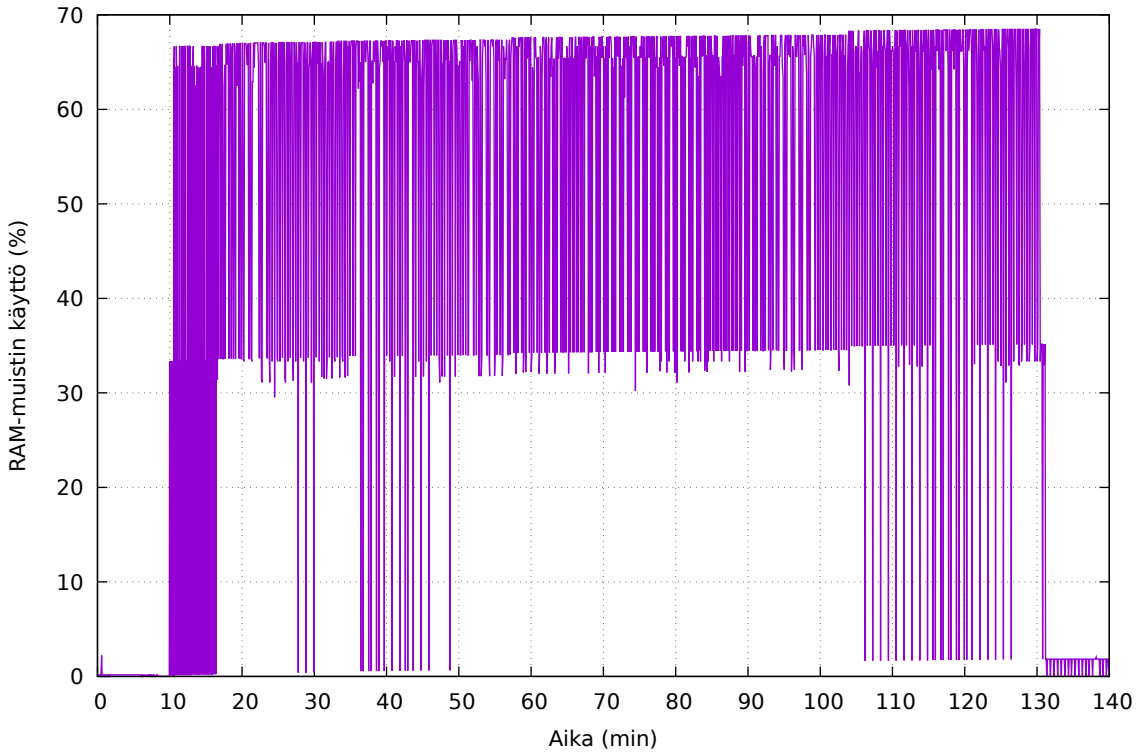
	Koe 1	**Koe 2**	**Koe 3**	**Keskiarvo**
Vastaanotetut paketit	2814	18 582	19 293	13 563
Lähetetyt paketit	120 042	1 119 341	912 876	717 419
PacketIn-viestit	19 407	189 030	159 792	122 743
PacketOut-viestit	68 200	576 742	478 062	374 668
PacketIn-viestit hyökkäyksessä	19 376	170 302	159 792	116 490
PacketOut-viestit hyökkäyksessä	68 170	517 130	478 062	354 454
Viimeinen PacketIn	19,80 min	99,52 min	131,10 min	83,47 min
Viimeinen PacketOut	53,54 min	99,52 min	131,10 min	94,72 min

Taulukko 11: Ryu-ohjaimen vastaanottamat ja lähettämät paketit kokeittain.

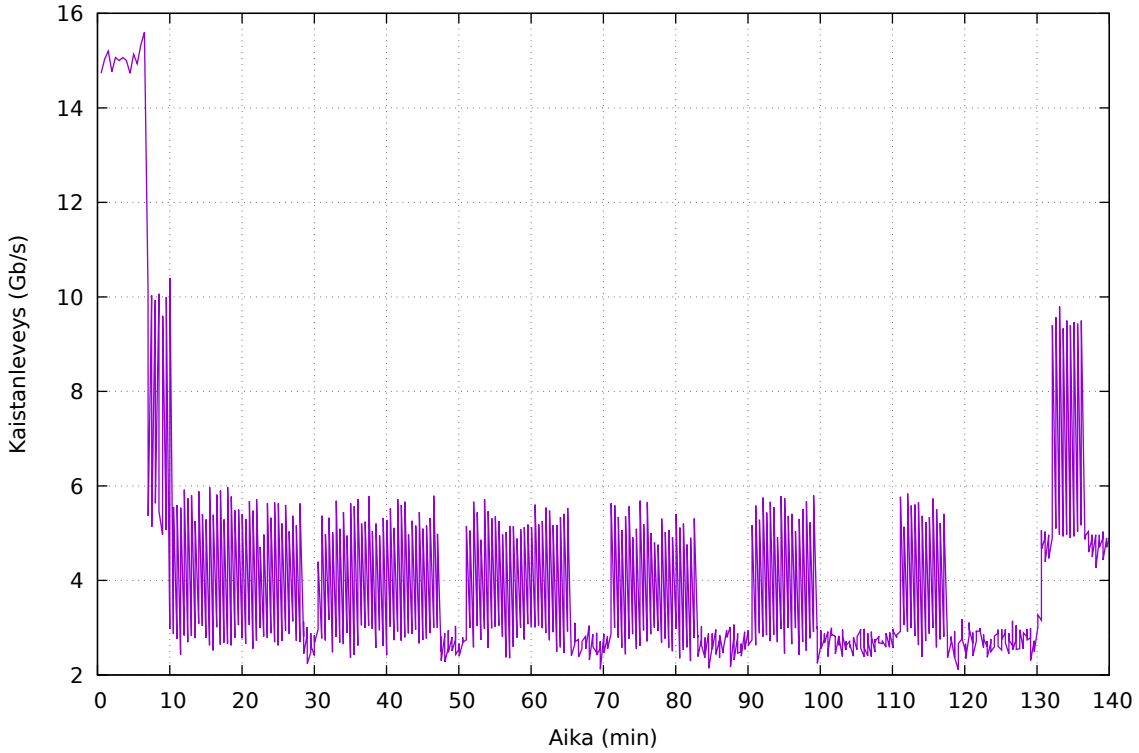
Pakettikaappaus päättyi ensimmäisesssä kokeessa noin 60 minuutin kohdalla ja toisessa 100 minuutin kohdalla. Kolmannessa kokeessa saatiin kaapattua koko kokeen paketit. Käytetyssä *TShark*-työkalussa on tunnettu ohjelmointivirhe, jonka vuoksi ohjelma saattaa kaatua kesken kaappauksen (*KnownBugs/OutOfMemory* 2013). Ohjelmointivirhe ei ollut tiedossa työkalua valittaessa. Työkalun kaatuminen kokeessa johtunee pakettien suuresta määrästä ja verkon kuormittamisesta. Erityisesti ensimmäisen kokeen pakettien määriä ei voida pitää vertailukelpoisina.



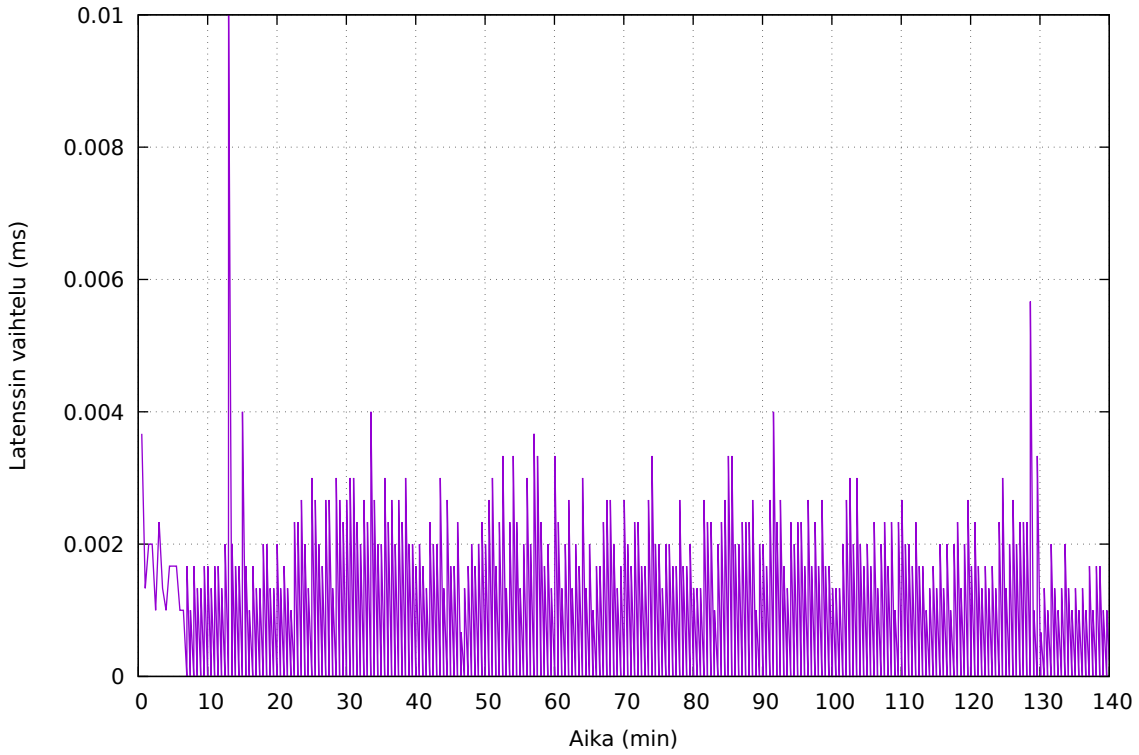
Kuvio 25: Ryu-ohjaimen suoritintehon käyttö.



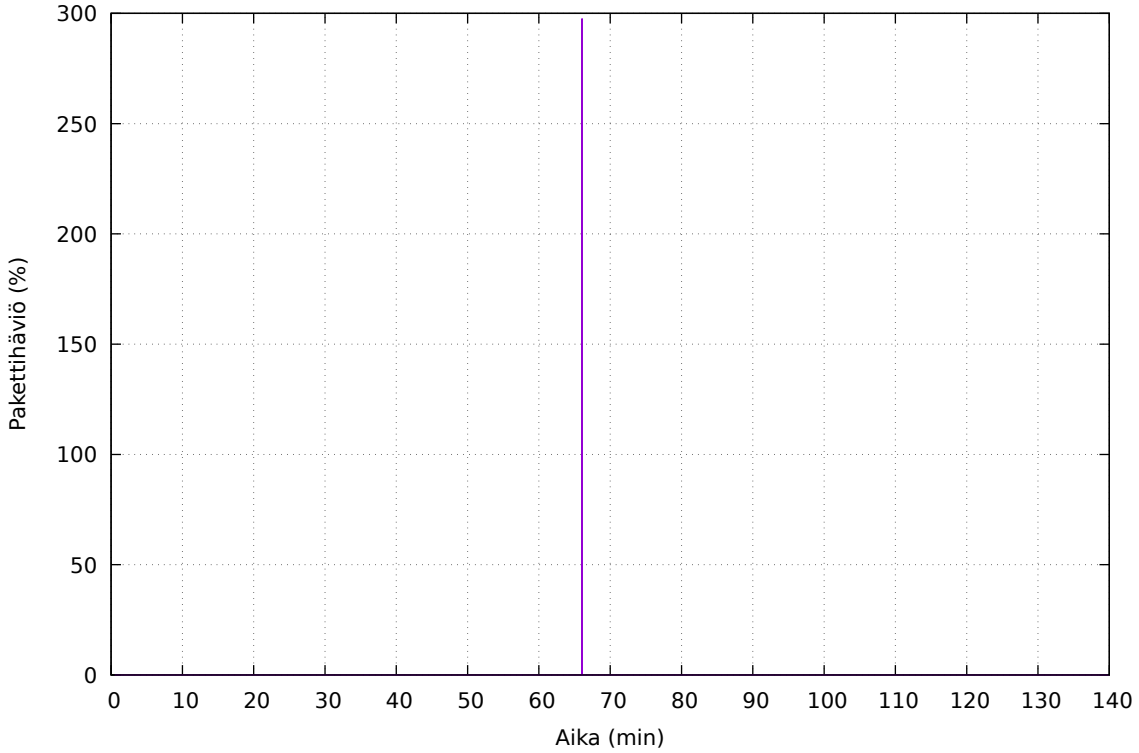
Kuvio 26: Ryu-ohjaimen RAM-muistin käyttö.



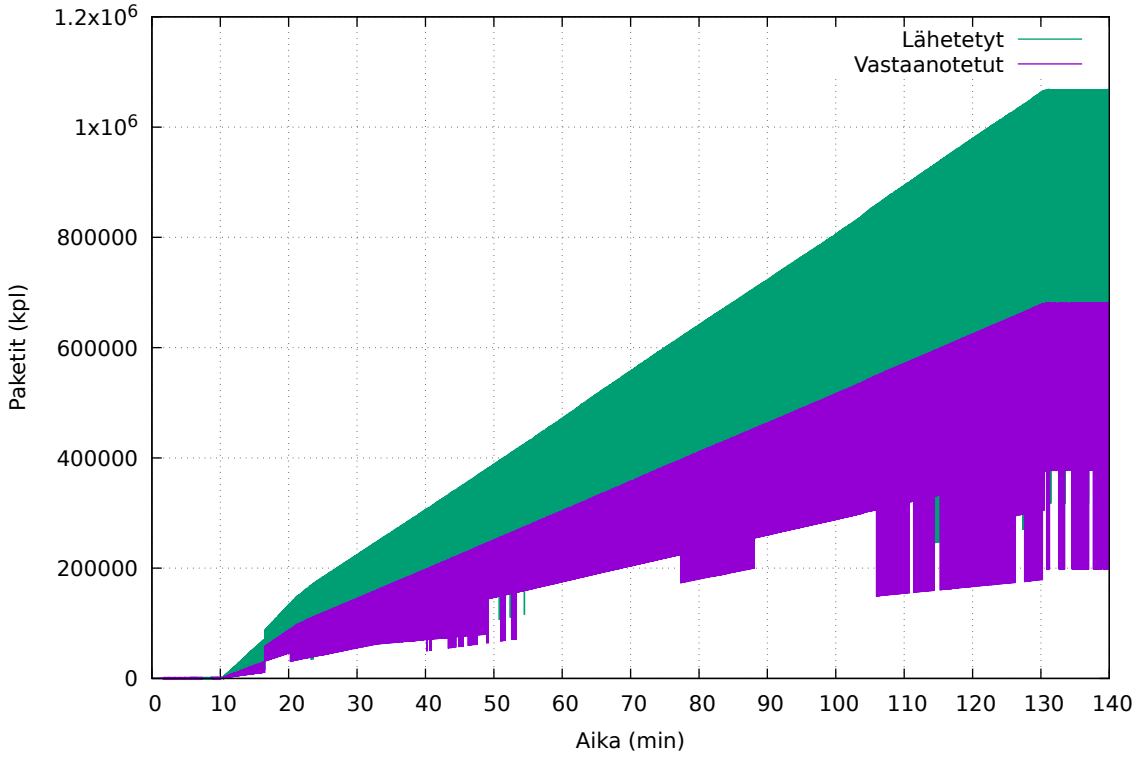
Kuvio 27: Ryu-ohjaimen verkon kaistanleveys.



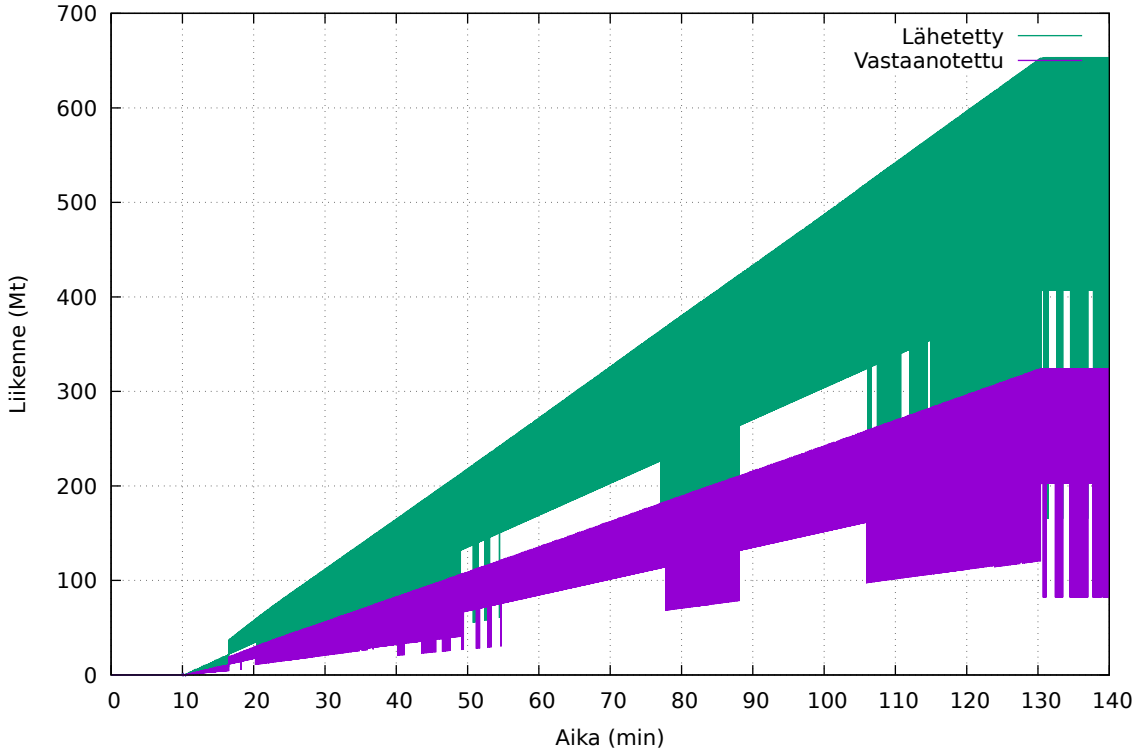
Kuvio 28: Ryu-ohjaimen verkon latenssin vaihtelu.



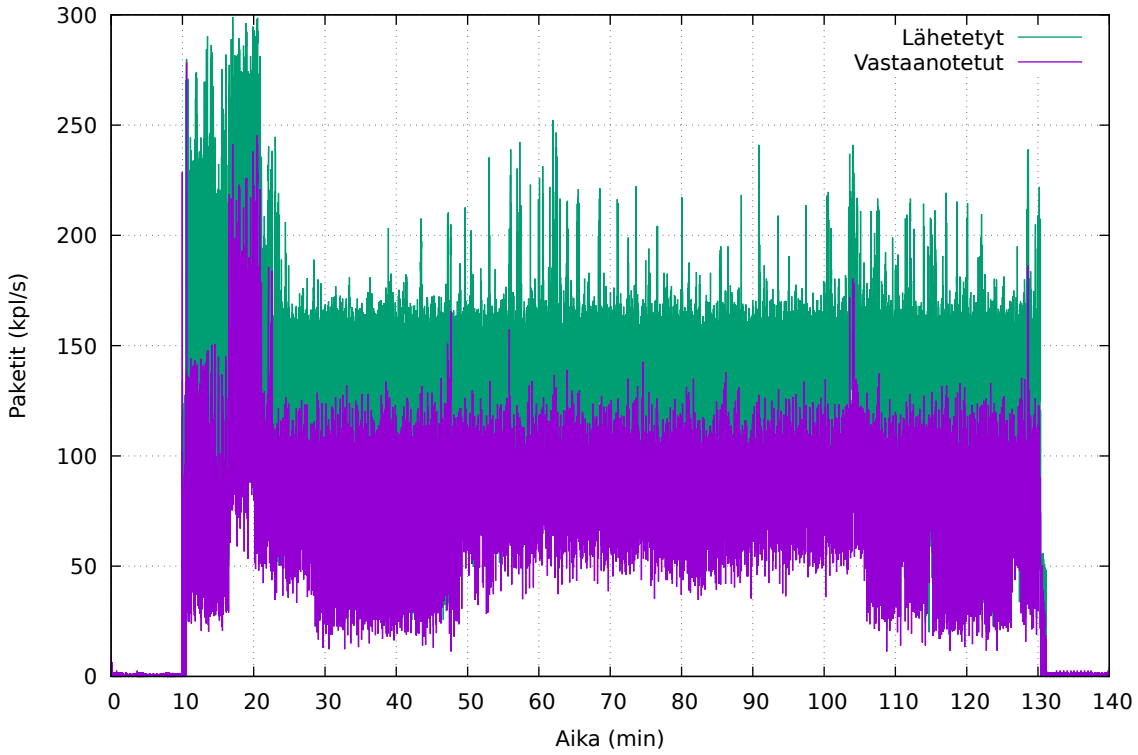
Kuvio 29: Ryu-ohjaimen verkon pakettihäviö.



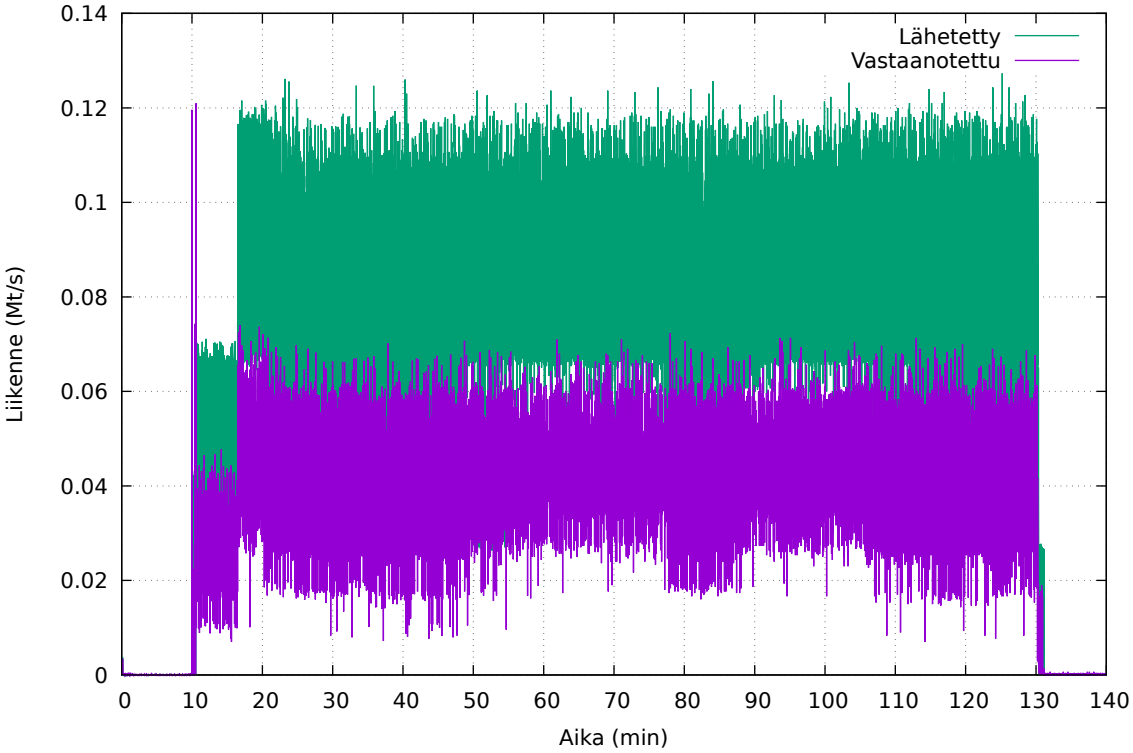
Kuvio 30: Ryu-ohjaimen vastaanottamat ja lähettämät paketit.



Kuvio 31: Ryu-ohjaimen vastaanottama ja lähettämä liikenne.



Kuvio 32: Ryu-ohjaimen vastaanottamat ja lähettämät paketit sekunnissa.



Kuvio 33: Ryu-ohjaimen vastaanottama ja lähettämä liikenne sekunnissa.

7.4 Vertailu

Kolmesta testatusta ohjaimesta OpenDaylight ja Ryu selvisivät hyökkäyksestä toimintakuntoisina kaikissa kolmessa toistossa. Floodlight kaatui joka kerta kesken hyökkäyksen eikä palannut toimintakykyiseksi hyökkäyksen päätyttyä. Floodlightin kaatuminen liittynee ohjelman muistinkäyttöön, mihin viittaa virheilmoitus Java-keon täyttymisestä. Kaaduttuaankin Floodlight käytti merkittävän määrän suoritintehoa, muistia ja säikeitä. Myös Javalla toteutetussa OpenDaylightissä ei ilmennyt samaa ongelmaa, joten ongelma ei selity pelkällä toteutuskielellä. Floodlightin suoriutumiseen olisi luultavasti voitu vaikuttaa antamalla Java-keolle enemmän tilaa kasvaa.

Ohjainten väliset erot ovat miltei joka mitatussa arvossa melko suuria. Floodlight-ohjaimen suoritintehon käyttö oli selvästi suurinta (n. 136 % kahdella ytimellä), OpenDaylightin noin puolet siitä ja Ryun erittäin pieni, vain 13 prosenttia (yhdellä ytimellä). OpenDaylight käytti enemmän säikeitä kuin Floodlight, mikä saattaa selittää myös suoritintehon käyttöä. Ryu käytti vähiten suoritintehoa, mutta miltei yhtä paljon muistia kuin Floodlight. OpenDaylight käytti paljon suoritintehoa, muistia ja säikeitä heti kokeen alussa, joten ohjaimen käynnistyminen vaatinee enemmän resursseja kuin tasainen kuormitus. Selvästi nopeiten käynnistyvä ohjain oli Ryu.

Vastaanotettujen pakettien määristä voidaan päätellä, että OpenDaylight pystyi käsittelemään selvästi eniten paketteja kokeen aikana, noin 2300 kappaletta. Floodlight ja Ryu jäivät merkittävästi vähäisempiin noin 52 ja 41 pakettiin. OpenDaylight ja Floodlight suoriutuivat tasaisesti, jos verrataan lähetettyjen ja vastaanotettujen pakettien määriä: molemmat lähettivät jokaista vastaanotettua pakettia kohti 0,23 pakettia. Ryu lähetti suhteellisesti enemmän paketteja, 0,65 kappaletta, mikä johtunee tuntemattoman vastaanottajan paketin kaiuttamisesta (ks. luku 7.3). Kaikkiaan liikenteen määriä verrattaessa nähdään, että Floodlightin käsittelemät liikennemäärät hiipuivat 20–30 minuutin kohdalla, kun taas OpenDaylightin ja Ryun liikennemäärät pysyivät tasaisina koko hyökkäyksen ajan.

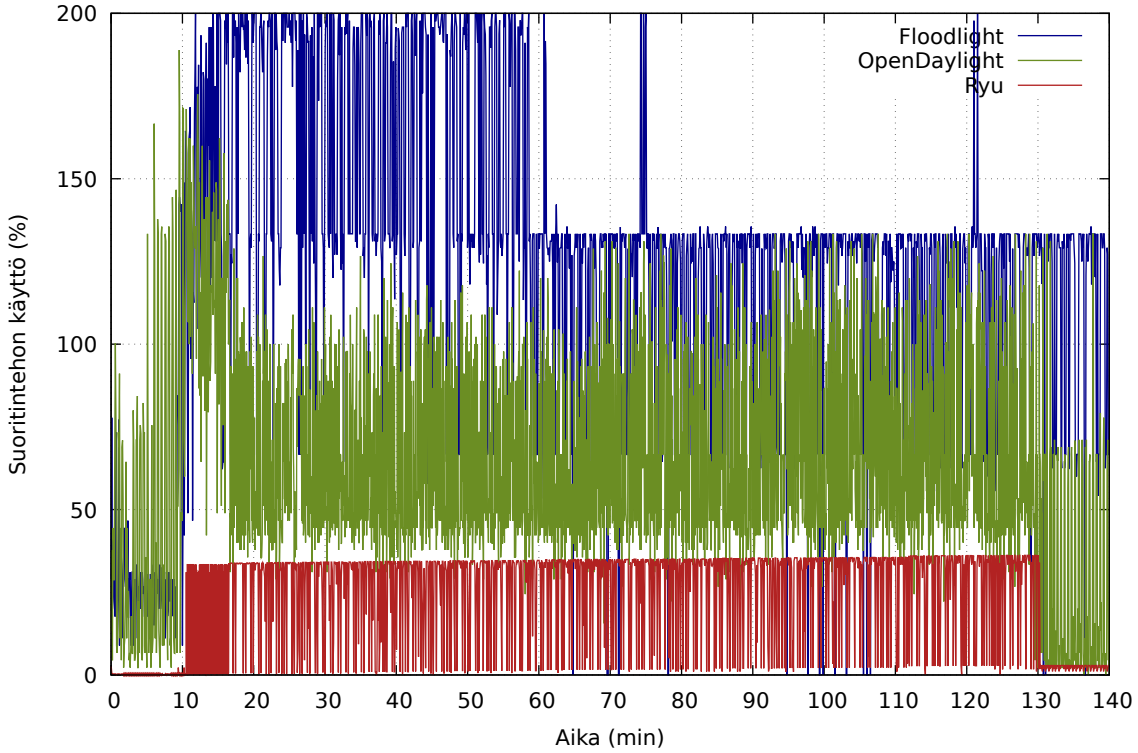
OpenDaylight vastaanotti enemmän liikennettä kokeen alussa ja lopussa kuin hyökkäyksen aikana. Tämä saattaa johtua siitä, että ohjain lähettää paljon OpenFlow-protokollan ohjausviestejä, mutta käytti hyökkäyksen aikana enemmän prosessointitehoa kytkinten neuvomi-

seen. Puutteellisten pakettikaappausten takia asiaa ei voitu varmistaa.

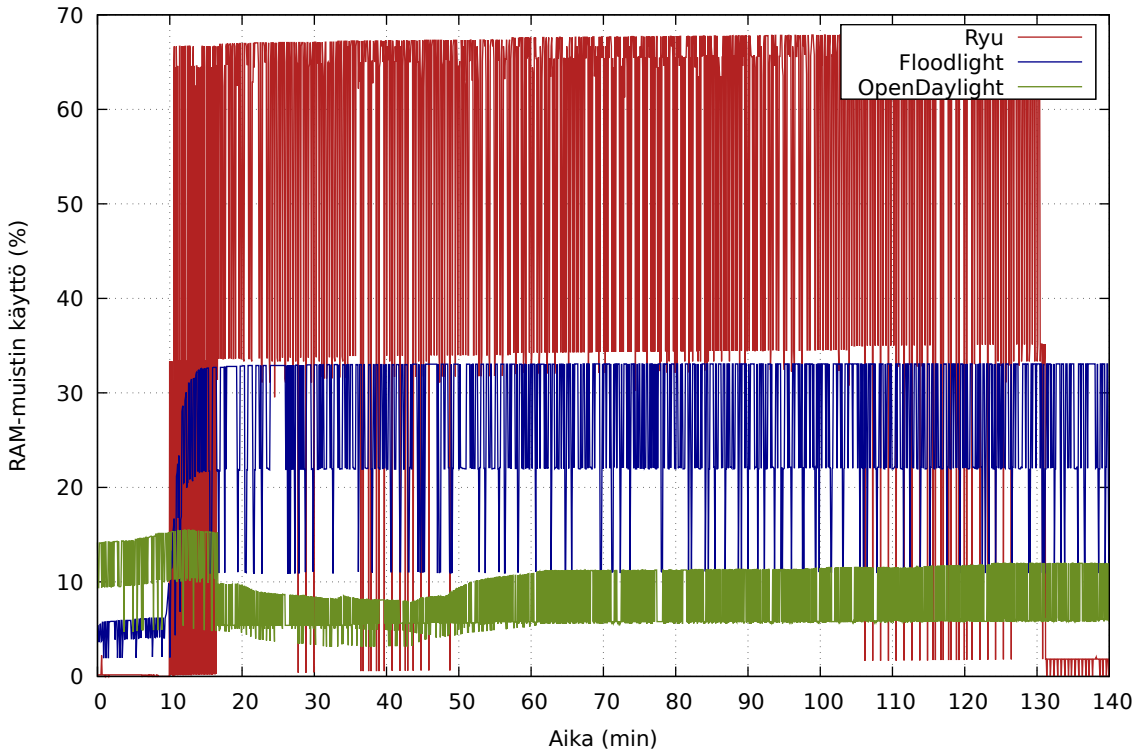
Hyökkäyksen vaikutuksia verkon toimintaan voidaan vertailla kaistanleveyden, latenssin vaihtelun ja pakettihäviön avulla. OpenDaylightin kaistanleveys oli muihin verrattuna pienin ensimmäisten 10 minuutin aikana, mutta hyökkäyksen päätyttyä kaistanleveys palautui ennalleen toisin kuin Ryun. Floodlightin tuloksissa näkyy selvä kasvu latenssin vaihtelussa ja pakettihäviössä ennen ohjaimen kaatumista, mikä kertoo verkon ylikuormittumisesta. Floodlightin tapauksessa kontrollikoneiden välisen yhteyden mittaus päättyi 25-35 minuutin välillä, joten kokeen lopun arvoja ei saatu mitattua. Kaikkien ohjainten keskiarvoiset tulokset on koottu taulukkoon 12 ja kuvioihin 34–47.

	Floodlight	**OpenDaylight**	**Ryu**
Suoritintehon käyttö (%)	136,28	64,66	13,03
RAM-muistin käyttö (%)	31,07	7,85	24,81
Säikeet (kpl)	51,13	57,99	1
Vastaanotetut paketit (kpl/s)	52,11	2300,64	41,50
Lähetetyt paketit (kpl/s)	12,50	530,20	64,31
Vastaanotettu liikenne (Mt/s)	0,04	1,80	0,02
Lähetetty liikenne (Mt/s)	0,00	0,04	0,04
Kaistanleveys (Gb/s)	10,91	3,19	5,06
Latenssin vaihtelu (ms)	8,68	0,00	0,00
Pakettihäviö (%)	9,18	0,00	0,31

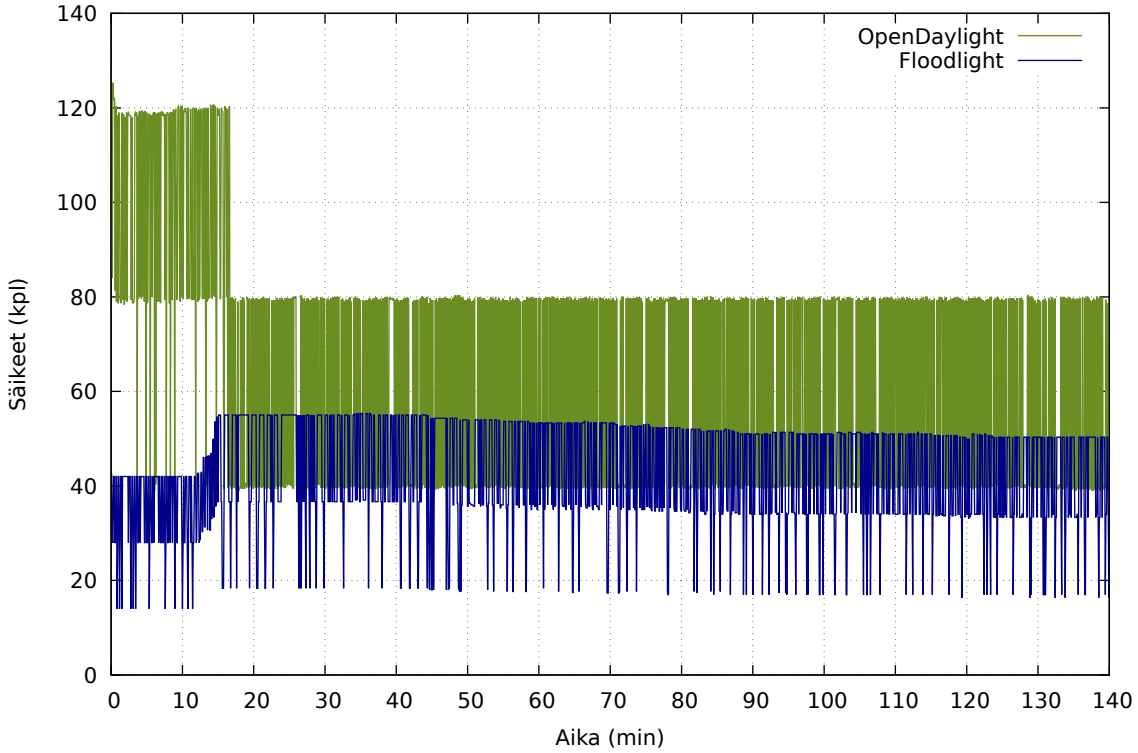
Taulukko 12: Ohjainten tulokset keskiarvoina.



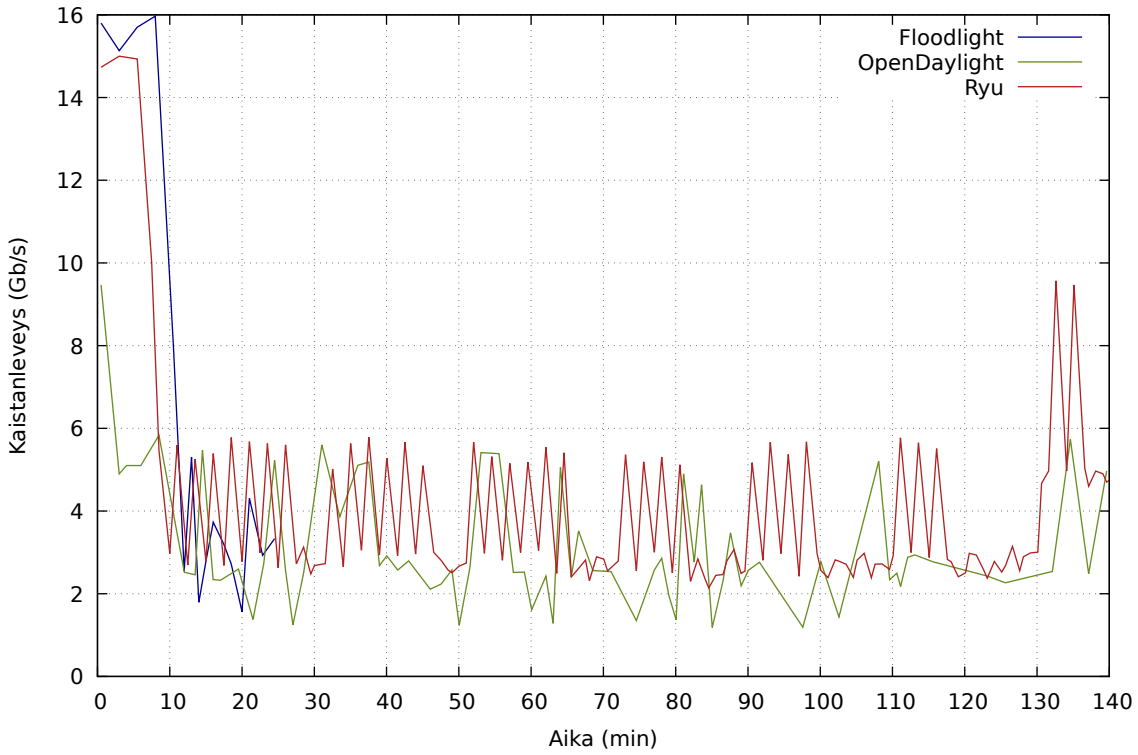
Kuvio 34: Ohjainten suoritintehon käyttö.



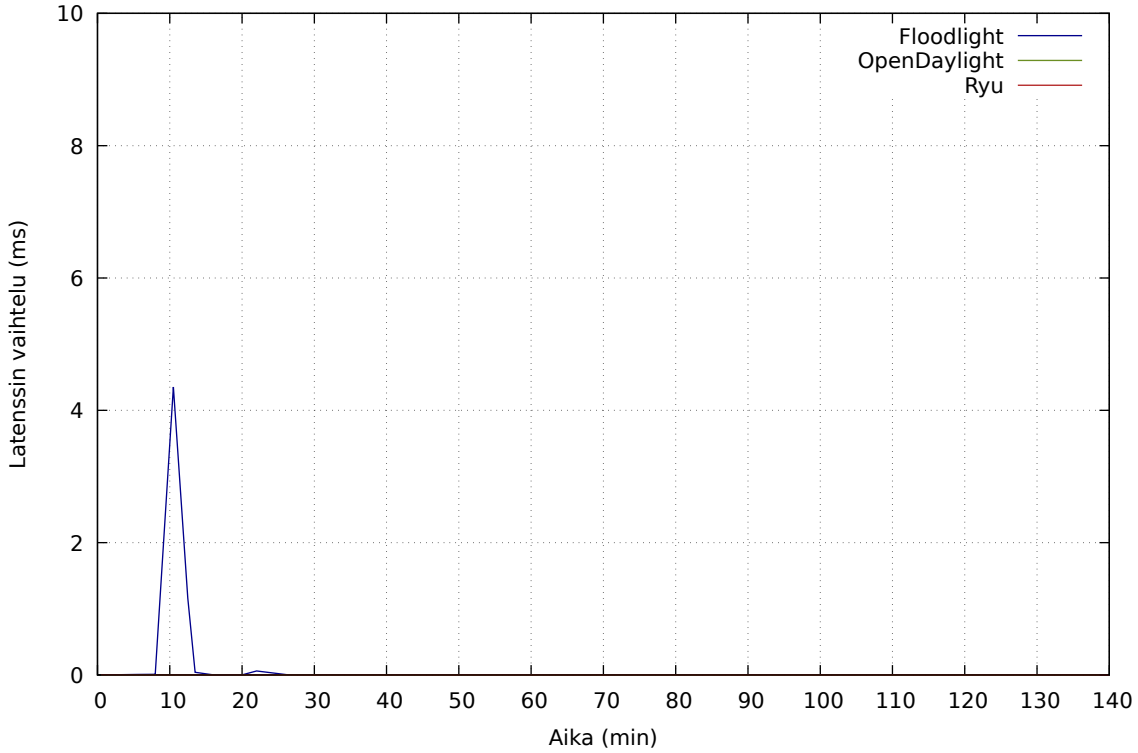
Kuvio 35: Ohjainten RAM-muistin käyttö.



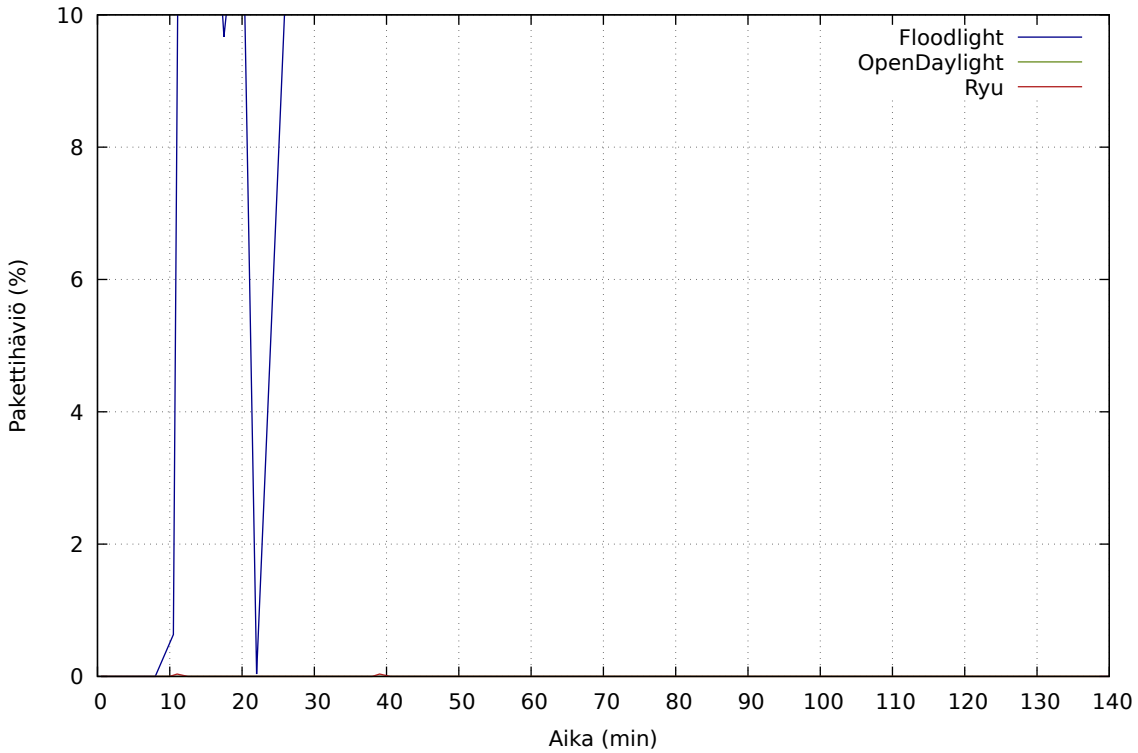
Kuvio 36: Ohjainten säikeitten määrä.



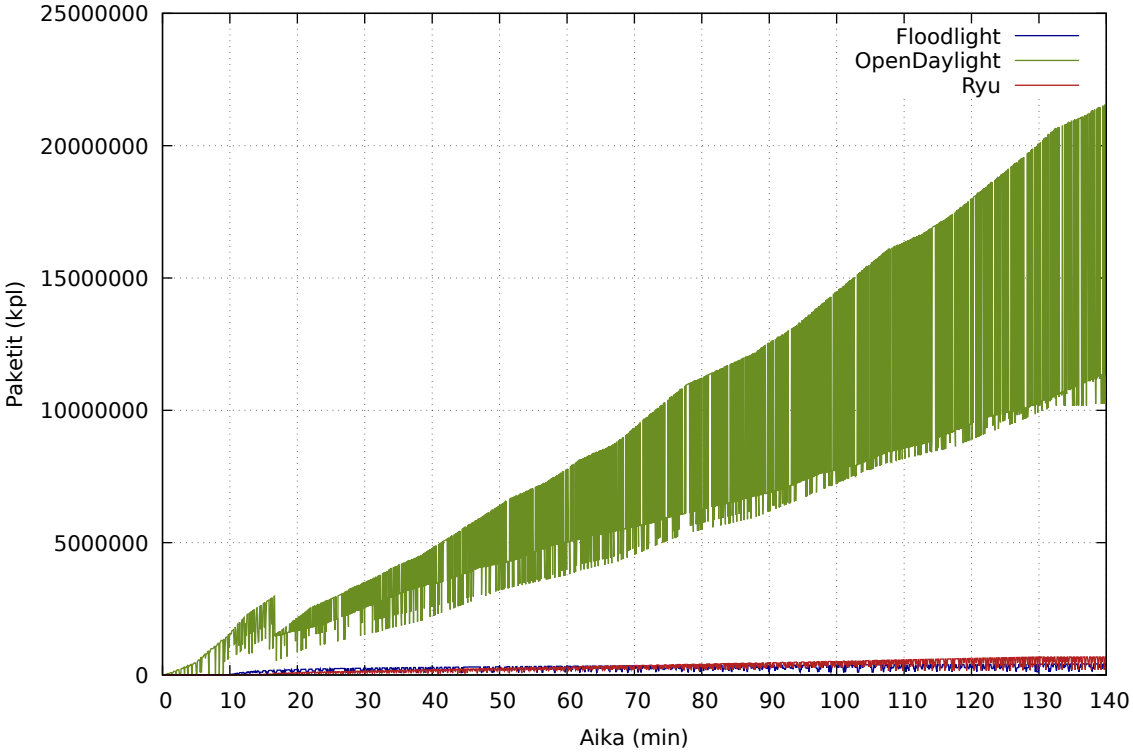
Kuvio 37: Ohjainten verkon kaistanleveys.



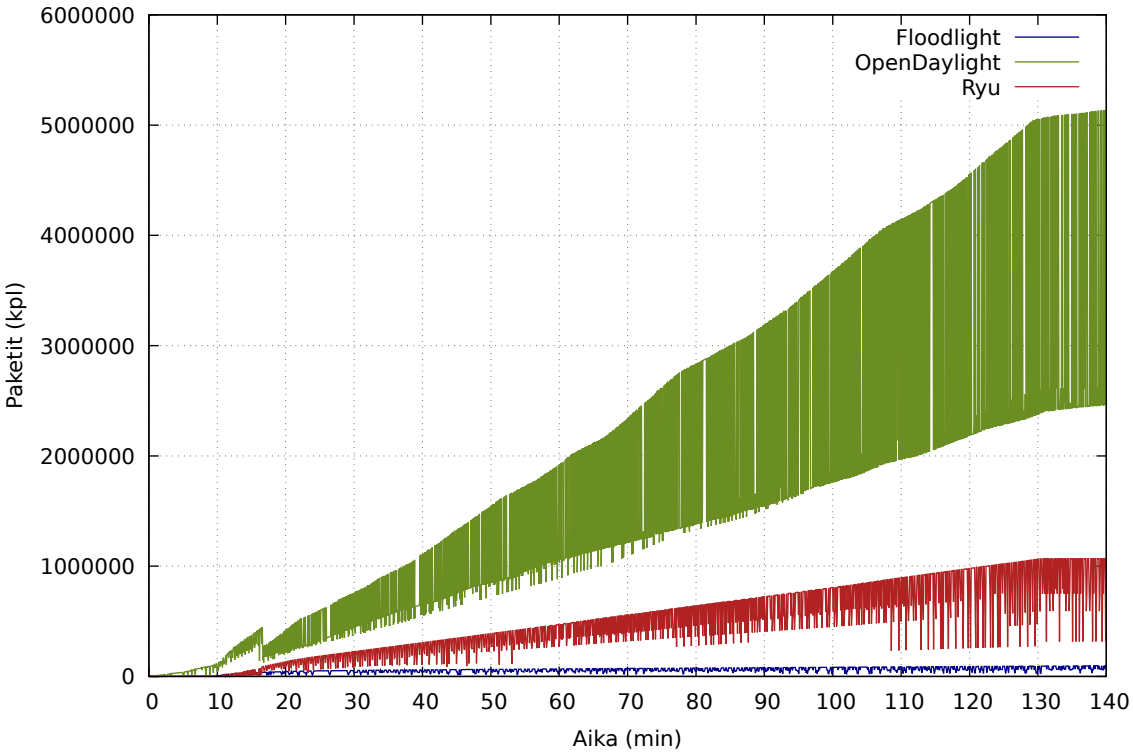
Kuvio 38: Ohjainten verkon latenssin vaihtelu.



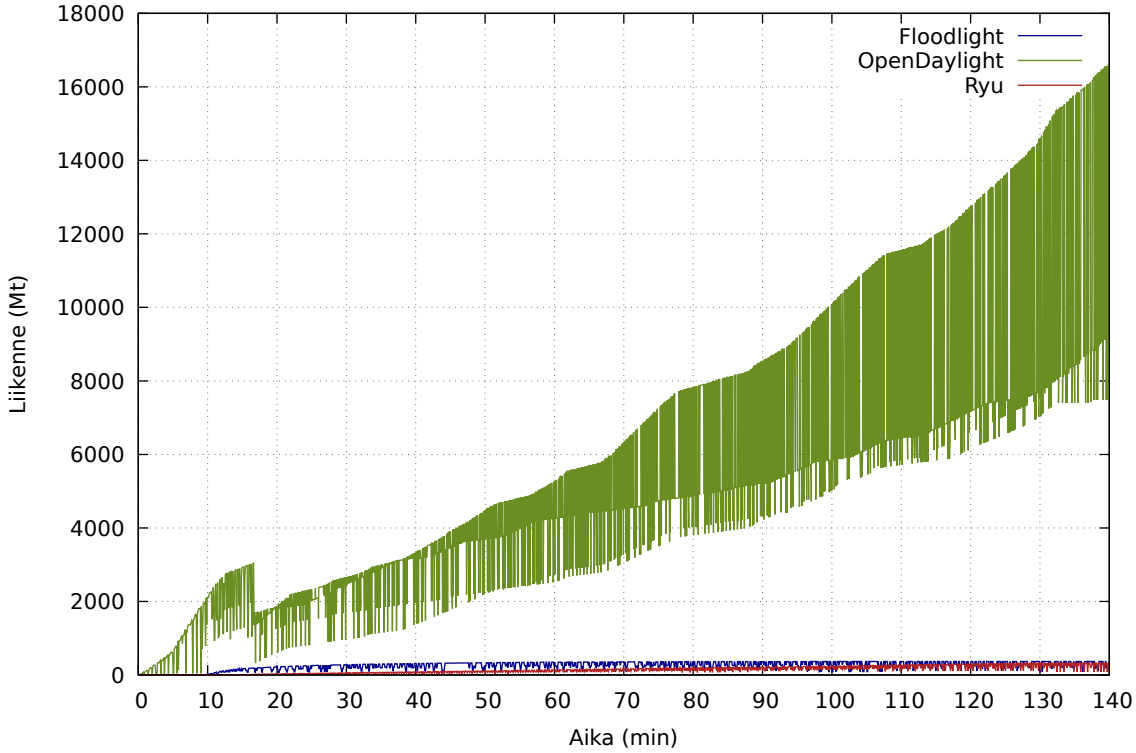
Kuvio 39: Ohjainten verkon pakettihäviö.



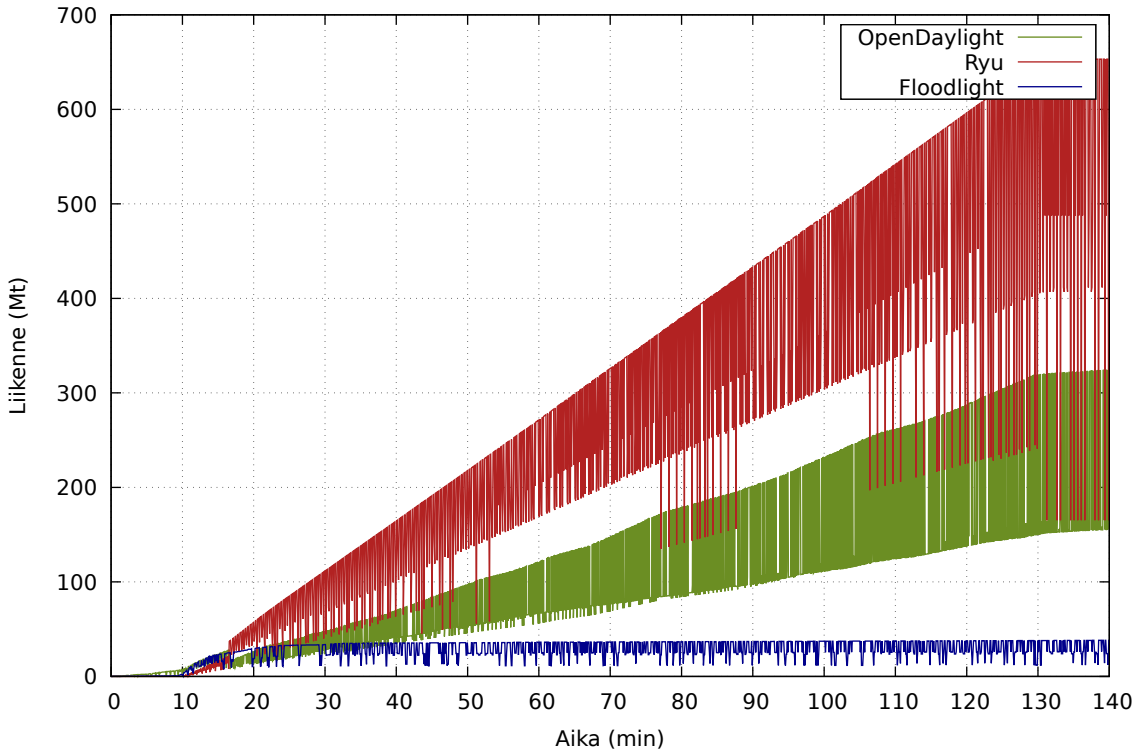
Kuvio 40: Ohjainten vastaanottamat paketit.



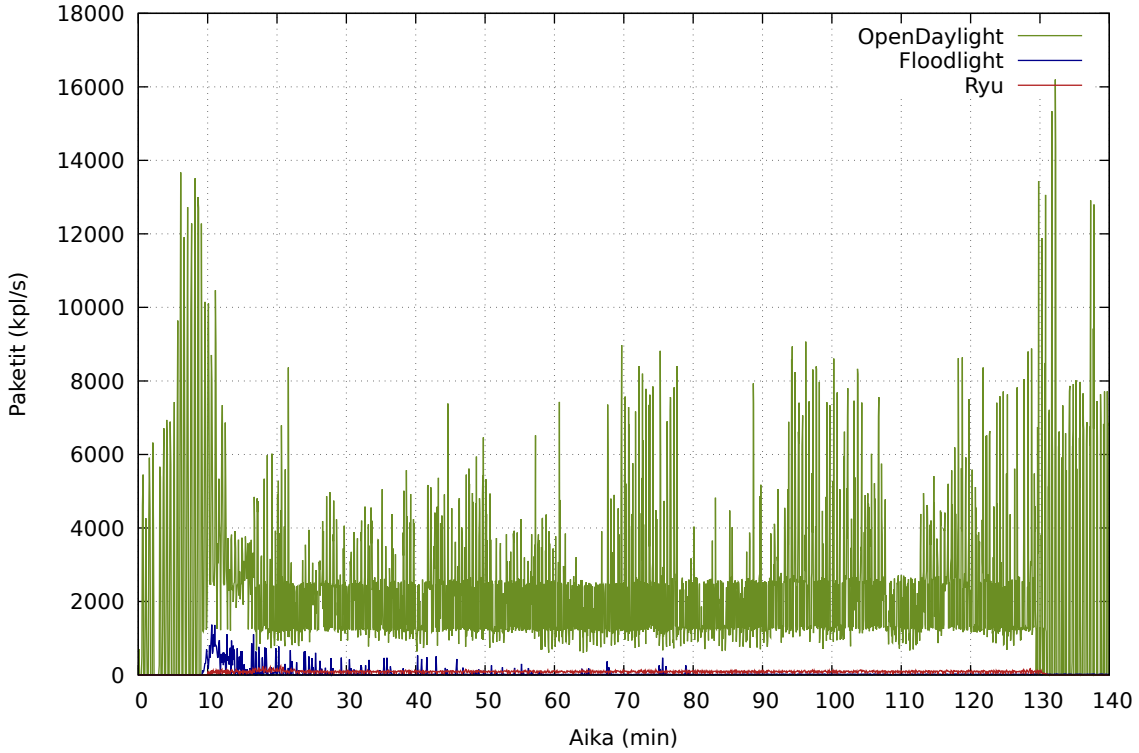
Kuvio 41: Ohjainten lähettämät paketit.



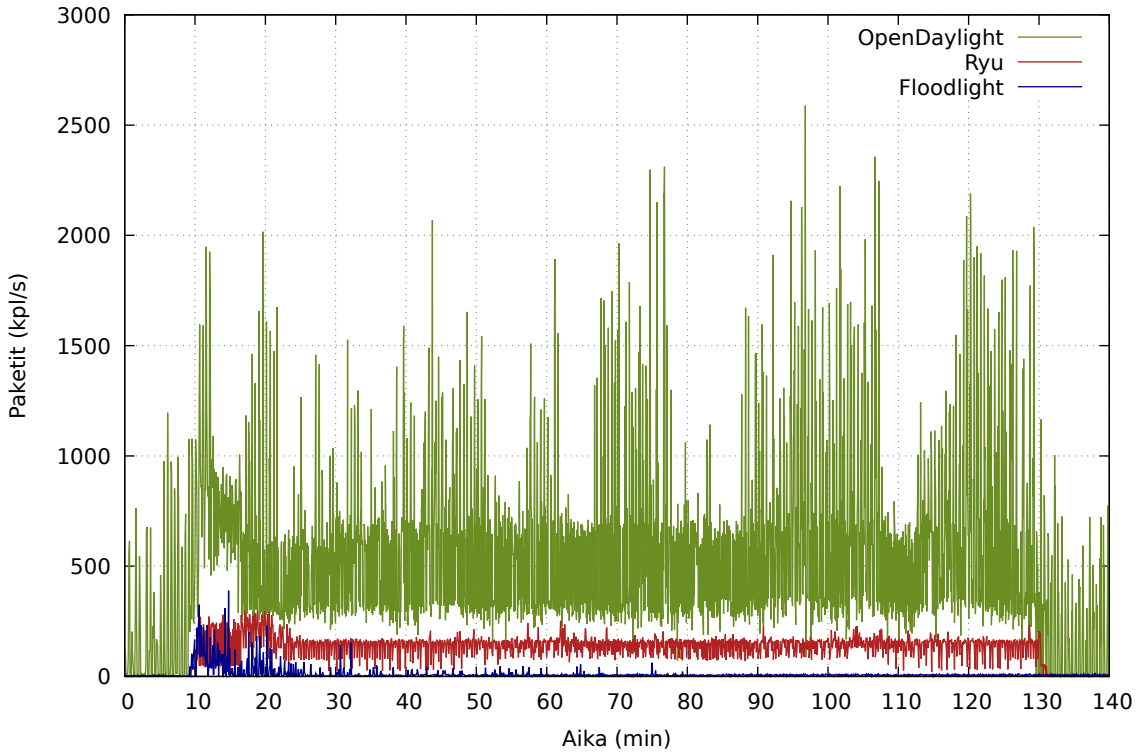
Kuvio 42: Ohjainten vastaanottama liikenne.



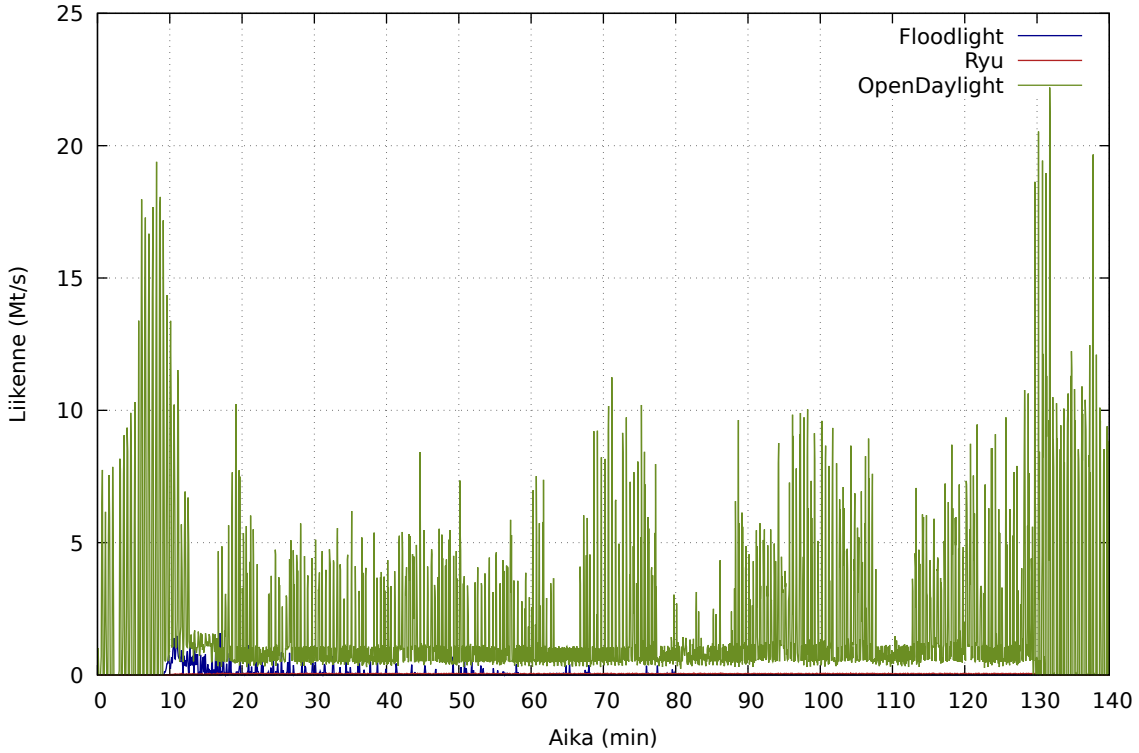
Kuvio 43: Ohjainten lähettämä liikenne.



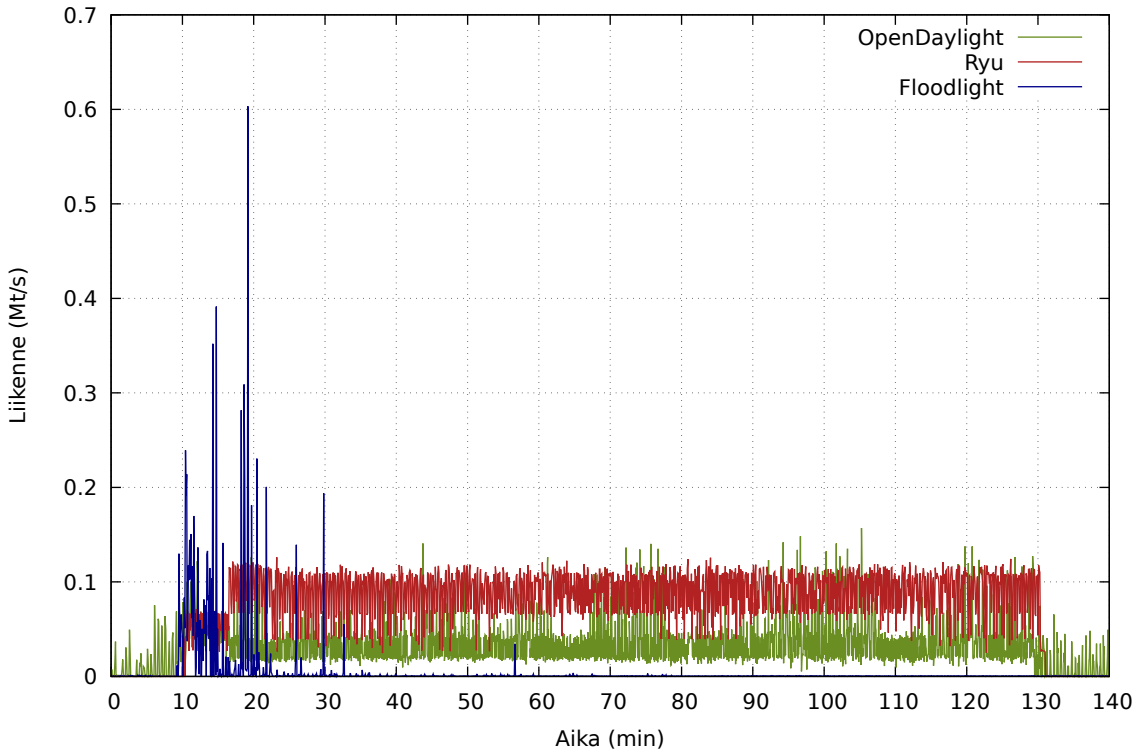
Kuvio 44: Ohjainten vastaanottamat paketit sekunnissa.



Kuvio 45: Ohjainten lähettämät paketit sekunnissa.



Kuvio 46: Ohjainten vastaanottama liikenne sekunnissa.



Kuvio 47: Ohjainten lähettämä liikenne sekunnissa.

8 Johtopäätökset ja pohdinta

Tässä tutkimuksessa selvitettiin kirjallisuuskartoituksen avulla, mitä haavoittuvuuksia ohjelmisto-ohjatun tietoverkkoarkkitehtuurin ohjaimessa on. Haavoittuvuudet liittyivät verkon tietojen keräämiseen, verkon topologiatietojen väärentämiseen, ohjaimen käyttämiin verkkosovelluksiin ja arkkitehtuurin rajapintoihin ohjaimen, sovellusten ja verkkolaitteiden välillä. Verkon tietojen keräämistä hyökkääjä voi käyttää muiden hyökkäysten valmistelussa. Topologiatietojen väärentämisellä hyökkääjä voi haitata niitä käyttävien sovellusten toimintaa tai kaapata liikenteen. Verkkosovellusten autentikoinnin ja oikeuksien rajoittamisen puute mahdollistaa haitallisten sovellusten toiminnan verkossa. Arkkitehtuurin kerroksellisuus synnyttää haavoittuvuuksia myös rajapinnoissa, kuten esimerkiksi yhteyden salauksen tai rajapintakutsujen valvonnan puute.

Tutkimuksen toisena tavoitteena oli testata käytännössä, miten ohjelmisto-ohjatun tietoverkkoarkkitehtuurin ohjainohjelmistot selviytyvät palvelunestohyökkäyksestä. Tutkimus osoitti, että pahimmassa tapauksessa hyökkäys voi kaataa ohjainohjelmiston. Hyökkäyksen aikana ohjain kuluttaa enemmän suoritintehoa ja muistia, eikä se voi palvella oikeita verkon käyttäjiä kunnolla. Hyökkäysliikenne kuormittaa myös verkon käyttäjien välisiä yhteyksiä. Tutkimuksessa käytetyistä ohjaimista Floodlight suoriutui kokeista heikoiten, koska se kaatui eikä palautunut toimintakykyiseksi. OpenDaylight käytti melko paljon resursseja, mutta myös toimi tehokkaasti. Ryu käytti ohjaimista vähiten suoritintehoa, mutta kuormitti verkkoa kaiuttamalla tuntemattoman vastaanottajan paketteja.

Tutkimusta suunniteltaessa haluttiin, että ohjainten tulokset olisivat vertailukelpoisia. Mahdollisuutta ohjainten testaamiseen oikeassa palvelinympäristössä ei ollut, joten mittausten absoluuttiset arvot eivät ole merkityksellisiä. Koska koeympäristö pysyi kaikissa kokeissa samana, tuloksia voidaan kuitenkin pitää keskenään vertailukelpoisina. Vertailua haittaa kuitenkin työkaluissa olleet puutteet, jotka joissain kokeissa vaikuttivat mittausten onnistumiseen. Puutteellisten mittausten arvoja ei voida siis verrata toisiinsa. Tutkimusta suunniteltaessa olisi pitänyt perehtyä työkalujen suorituskykyyn ja tunnettuihin puutteisiin tarkemmin ja valita työkalut paremmin. Ohjaimen toimintakykyä oli tarkoitus seurata myös ohjaimen ja kytkimen vaihtamien Echo-viestien avulla. Näitä viestejä ei kuitenkaan lähettänyt

kuin Floodlight, joten niistä ei saatu vertailtavaa tietoa. Koetta suunniteltaessa olisi pitänyt selvittää eri ohjainten Echo-viestien lähettämistä ja generoida niitä tarvittaessa kytkimissä, jolloin ohjaimen tilaa olisi voitu seurata.

Tarkoitus oli myös selvittää, vaikuttaako joku ohjainohjelmiston ominaisuus saatuihin tuloksiin. Ohjelmistojen dokumentaatio ei antanut tähän riittävän kattavia tietoja eikä niitä ollut mahdollista tämän tutkimuksen aikana itse selvittää. Tästä syystä ohjelmistojen erojen syyt jäivät hämäriksi.

Tutkimus eroaa aiemmin tehdystä tutkimuksesta. Ohjainohjelmistoja on testattu yksittäin ja tietyillä työkaluilla, mutta tässä tutkimuksessa uutta on testattujen ohjainten vertailu ja mitattujen ominaisuuksien määrä. Koejärjestely ja sillä saadut tulokset voivat toimia pohjana seuraaville tutkimuksille. Koejärjestelyssä havaittuja puutteita korjaamalla ja ohjainten määrää kasvattamalla vertailevista kokeista voitaisiin saada vielä kattavampia tuloksia. Tämän tutkimuksen tulokset eivät liene itsessään käyttökelpoisia, vaan tutkimuksen arvo on vertailevan kokeen suunnittelussa ja sen onnistumisten ja kehityskohtien hyödyntämisessä jatkotutkimuksessa.

Lähteet

Ahmad, Ijaz, Suneth Namal, Mika Ylianttila ja Andrei Gurtov. 2015. “Security in Software Defined Networks: A Survey”. *IEEE Communications Surveys Tutorials* 17 (4): 2317–2346. ISSN: 1553-877X. doi:10.1109/COMST.2015.2474118.

Akhunzada, Adnan, Ejaz Ahmed, Abdullah Gani, Muhammad Khurram Khan, Muhammad Imran ja Sghaier Guizani. 2015. “Securing Software Defined Networks: Taxonomy, Requirements, and Open Issues”. *IEEE Communications Magazine* 53, numero 4 (huhtikuu): 36–44. ISSN: 0163-6804. doi:10.1109/MCOM.2015.7081073.

Alcoy, Philippe, Darren Anstee, Steinthor Bjarnason, Paul Bowen, C. F. Chui, Kirill Kasavchenko ja Gary Sockrider. 2018. *NETSCOUT Arbor’s 13th Annual Worldwide Infrastructure Security Report.* Kyselytutkimus, Arbor Networks Special Report 13.

Alharbi, Talal, Siamak Layeghy ja Marius Portmann. 2017. “Experimental Evaluation of the Impact of DoS Attacks in SDN”. Teoksessa *2017 27th International Telecommunication Networks and Applications Conference (ITNAC),* 1–6. Marraskuu. doi:10.1109/ATNAC.2017.8215424.

Alharbi, Talal, Marius Portmann ja Farzaneh Pakzad. 2015. “The (In)security of Topology Discovery in Software Defined Networks”. Teoksessa *2015 IEEE 40th Conference on Local Computer Networks (LCN),* 502–505. Lokakuu. doi:10.1109/LCN.2015.7366363.

Alsmadi, Izzat, ja Dianxiang Xu. 2015. “Security of Software Defined Networks: A Survey”. *Computers & Security* 53 (Supplement C): 79–108. ISSN: 0167-4048. doi:https://doi.org/10.1016/j.cose.2015.05.006.

Azzouni, Abdelhadi, Othmen Braham, Thi Mai Trang Nguyen, Guy Pujolle ja Raouf Boutaba. 2016. “Fingerprinting OpenFlow Controllers: The First Step to Attack an SDN Control Plane”. Teoksessa *2016 IEEE Global Communications Conference (GLOBECOM),* 1–6. Joulukuu. doi:10.1109/GLOCOM.2016.7841843.

Berde, Pankaj, Matteo Gerola, Jonathan Hart, Yuta Higuchi, Masayoshi Kobayashi, Toshio Koide, Bob Lantz ym. 2014. “ONOS: Towards an Open, Distributed SDN OS”. Teoksessa *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking,* 1–6. HotSDN ’14. Chicago, Illinois, USA: ACM. ISBN: 978-1-4503-2989-7. doi:10.1145/2620728.2620744.

Betgé-Brezetz, Stéphane, Guy-Bertrand Kamga ja Monsef Tazi. 2015. “Trust Support for SDN Controllers and Cirtualized Network Applications”. Teoksessa *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft),* 1–5. Huhtikuu. doi:10.1109/NETSOFT.2015.7116153.

Bifulco, Roberto, Heng Cui, Ghassan O. Karame ja Felix Klaedtke. 2015. “Fingerprinting Software-Defined Networks”. Teoksessa *2015 IEEE 23rd International Conference on Network Protocols (ICNP),* 453–459. Marraskuu. doi:10.1109/ICNP.2015.26.

Brooks, Michael, ja Baijian Yang. 2015. “A Man-in-the-Middle Attack Against OpenDaylight SDN Controller”. Teoksessa *Proceedings of the 4th Annual ACM Conference on Research in Information Technology,* 45–49. RIIT ’15. Chicago, Illinois, USA: ACM. ISBN: 978-1-4503-3836-3. doi:10.1145/2808062.2808073.

Cai, Zheng, Alan L. Cox ja Eugene T. S. Ng. 2010. *Maestro: A System for Scalable OpenFlow Control.* Tekninen raportti. Rice University, Houston, TX, USA. http://hdl.handle.net/1911/96391.

Casado, Martín, Michael J. Freedman, Justin Pettit, Jianying Luo, Nick McKeown ja Scott Shenker. 2007. “Ethane: Taking Control of the Enterprise”. Teoksessa *Proceedings of the 2007 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications,* 1–12. SIGCOMM ’07. Kyoto, Japan: ACM. ISBN: 978-1-59593-713-1. doi:10.1145/1282380.1282382.

Erickson, David. 2013. “The Beacon Openflow Controller”. Teoksessa *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking,* 13–18. HotSDN ’13. Hong Kong, China: ACM. ISBN: 978-1-4503-2178-5. doi:10.1145/2491185.2491189.

Feamster, Nick, Jennifer Rexford ja Ellen Zegura. 2013. “The Road to SDN”. *Queue* (New York, NY, USA) 11, numero 12 (joulukuu): 20:20–20:40. ISSN: 1542-7730. doi:10.1145/2559899.2560327.

Floodlight. 2017. Viitattu 26. syyskuuta 2017. http://www.projectfloodlight.org/floodlight.

Floodlight Installation Guide. 2016. Viitattu 13. huhtikuuta 2018. https://floodlight.atlassian.net/wiki/spaces/floodlightcontroller/pages/1343544/Installation+Guide.

Framework of software-defined networking. 2014. Suositus, Series Y: Global Information Infrastructure, Internet Protocol Aspects and Next-Generation Networks Y.3300. Telecommunication Standardization Sector of ITU. http://www.itu.int/rec/T-REC-Y.3300-201406-I/en.

Gude, Natasha, Teemu Koponen, Justin Pettit, Ben Pfaff, Martín Casado, Nick McKeown ja Scott Shenker. 2008. “NOX: Towards an Operating System for Networks”. *SIGCOMM Comput. Commun. Rev.* (New York, NY, USA) 38, numero 3 (heinäkuu): 105–110. ISSN: 0146-4833. doi:10.1145/1384609.1384625.

Hoque, N., Monowar H. Bhuyan, R. C. Baishya, D. K. Bhattacharyya ja J. K. Kalita. 2014. “Network attacks: Taxonomy, tools and systems”. *Journal of Network and Computer Applications* 40 (Supplement C): 307–324. ISSN: 1084-8045. doi:https://doi.org/10.1016/j.jnca.2013.08.001.

Hyenae. 2016. Viitattu 13. huhtikuuta 2018. https://sourceforge.net/projects/hyenae/.

Installing OpenDaylight. 2018. Viitattu 13. huhtikuuta 2018. http://docs.opendaylight.org/en/stable-nitrogen/getting-started-guide/installing_opendaylight.html.

iPerf - The ultimate speed test tool for TCP, UDP and SCTP. 2018. Viitattu 13. huhtikuuta. https://iperf.fr.

Jain, Sushant, Alok Kumar, Subhasree Mandal, Joon Ong, Leon Poutievski, Arjun Singh, Subbaiah Venkata ym. 2013. “B4: Experience with a Globally-deployed Software Defined Wan”. Teoksessa *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM,* 3–14. SIGCOMM ’13. Hong Kong, China: ACM. ISBN: 978-1-4503-2056-6. doi:10.1145/2486001.2486019.

Jarschel, Michael, Thomas Zinner, Tobias Hossfeld, Phuoc Tran-Gia ja Wolfgang Kellerer. 2014. “Interfaces, attributes, and use cases: A compass for SDN”. *IEEE Communications Magazine* 52, numero 6 (kesäkuu): 210–217. ISSN: 0163-6804. doi:10.1109/MCOM.2014.6829966.

Java Platform, Standard Edition Troubleshooting Guide: Understand the OutOfMemoryError Exception. 2018. Viitattu 13. huhtikuuta 2018. https://docs.oracle.com/javase/8/docs/technotes/guides/troubleshoot/memleaks002.html.

Jeong, Jaehoon, Jihyeok Seo, Geumhwan Cho, Hyoungshick Kim ja Jung-Soo Park. 2015. “A Framework for Security Services Based on Software-Defined Networking”. Teoksessa *2015 IEEE 29th International Conference on Advanced Information Networking and Applications Workshops,* 150–153. Maaliskuu. doi:10.1109/WAINA.2015.102.

Khan, Suleman, Abdullah Gani, Ainuddin Wahab Abdul Wahab, Mohsen Guizani ja Muhammad Khurran Khan. 2017. “Topology Discovery in Software Defined Networks: Threats, Taxonomy, and State-of-the-Art”. *IEEE Communications Surveys & Tutorials* 19 (1): 303–324. ISSN: 1553-877X. doi:10.1109/COMST.2016.2597193.

Kim, Hyojoon, ja Nick Feamster. 2013. “Improving Network Management with Software Defined Networking”. *IEEE Communications Magazine* 51, numero 2 (helmikuu): 114–119. ISSN: 0163-6804. doi:10.1109/MCOM.2013.6461195.

Klöti, Rowan, Vasileios Kotronis ja Paul Smith. 2013. “OpenFlow: A Security Analysis”. Teoksessa *Proceedings of the 21st IEEE International Conference on Network Protocols (ICNP),* 1–6. Göttingen, Germany: IEEE. ISBN: 978-1-4799-1270-4. doi:10.1109/ICNP.2013.6733671.

KnownBugs/OutOfMemory. 2013. Viitattu 13. huhtikuuta 2018. https://wiki.wireshark.org/KnownBugs/OutOfMemory.

Koponen, Teemu, Martín Casado, Natasha Gude, Jeremy Stribling, Leon Poutievski, Min Zhu, Rajiv Ramanathan, Yuichiro Iwata, Hiroaki Inoue, Takayuki Hama ym. 2010. “Onix: A Distributed Control Platform for Large-scale Production Networks”. Teoksessa *9th USENIX Symposium on Operating Systems Design and Implementation,* 351–364. OSDI ’10. Vancouver, BC, Canada. ISBN: 978-1-931971-79-9. https://dl.acm.org/citation.cfm?id=1924968.

Kotzanikolaou, Panayiotis, ja Christos Douligeris. 2007. “Computer Network Security: Basic Background and Current Issues”. Luku 1 teoksessa *Network Security: Current Status and Future Directions,* toimittanut Christos Douligeris ja Dimitrios N. Serpanos, 1–12. Wiley-IEEE Press. ISBN: 978-0-4700-9974-2. doi:10.1002/9780470099742.ch1.

Kreutz, Diego, Fernando M. V. Ramos, Paulo Esteves Veríssimo, Christian Esteve Rothenberg, Siamak Azodolmolky ja Steve Uhlig. 2015. “Software-Defined Networking: A Comprehensive Survey”. *Proceedings of the IEEE* 103, numero 1 (tammikuu): 14–76. ISSN: 0018-9219. doi:10.1109/JPROC.2014.2371999.

Kurose, James F., ja Keith W. Ross. 2013. “Computer Networks and the Internet”. Luku 1 teoksessa *Computer Networking: A Top-Down Approach,* 6. painos, 55–56. Pearson. ISBN: 978-0-13-285620-1.

Lantz, Bob, Brandon Heller ja Nick McKeown. 2010. “A Network in a Laptop: Rapid Prototyping for Software-defined Networks”. Teoksessa *Proceedings of the 9th ACM SIGCOMM Workshop on Hot Topics in Networks,* 19:1–19:6. Hotnets-IX. Monterey, California: ACM. ISBN: 978-1-4503-0409-2. doi:10.1145/1868447.1868466.

Lee, Seungsoo, Changhoon Yoon ja Seungwon Shin. 2016. “The Smaller, the Shrewder: A Simple Malicious Application Can Kill an Entire SDN Environment”. Teoksessa *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization,* 23–28. SDN-NFV Security ’16. New Orleans, Louisiana, USA: ACM. ISBN: 978-1-4503-4078-6. doi:10.1145/2876019.2876024.

Li, Li, Wu Chou, Wei Zhou ja Min Luo. 2016. “Design Patterns and Extensibility of REST API for Networking Applications”. *IEEE Transactions on Network and Service Management* 13, numero 1 (maaliskuu): 154–167. ISSN: 1932-4537. doi:10.1109/TNSM.2016.2516946.

Lukka, Kari. 2001. *Konstruktiivinen tutkimusote.* Metodix. Viitattu 18. tammikuuta 2018. https://metodix.fi/2014/05/19/lukka-konstruktiivinen-tutkimusote.

Matsumoto, Stephanos, Samuel Hitz ja Adrian Perrig. 2014. “Fleet: Defending SDNs from Malicious Administrators”. Teoksessa *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking,* 103–108. HotSDN ’14. Chicago, Illinois, USA: ACM. ISBN: 978-1-4503-2989-7. doi:10.1145/2620728.2620750.

McKeown, Nick, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker ja Jonathan Turner. 2008. “OpenFlow: Enabling Innovation in Campus Networks”. *SIGCOMM Comput. Commun. Rev.* (New York, NY, USA) 38, numero 2 (maaliskuu): 69–74. ISSN: 0146-4833. doi:10.1145/1355734.1355746.

Medved, Jan, Robert Varga, Anton Tkacik ja Ken Gray. 2014. “OpenDaylight: Towards a Model-Driven SDN Controller architecture”. Teoksessa *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014,* 1–6. Kesäkuu. doi:10.1109/WoWMoM.2014.6918985.

Mitrokotsa, Aikaterini, ja Christos Douligeris. 2007. “Denial-of-Service Attacks”. Luku 8 teoksessa *Network Security: Current Status and Future Directions,* toimittanut Christos Douligeris ja Dimitrios N. Serpanos, 117–134. Wiley-IEEE Press. ISBN: 978-0-4700-9974-2. doi:10.1002/9780470099742.ch8.

Mochel, Patrick, ja Mike Murphy. 2011. *sysfs – _The_ filesystem for exporting kernel objects.* Viitattu 13. huhtikuuta 2018. https://www.kernel.org/doc/Documentation/filesystems/sysfs.txt.

Morzhov, Sergey, Igor Alekseev ja Mikhail Nikitinskiy. 2016. “Firewall Application for Floodlight SDN Controller”. Teoksessa *2016 International Siberian Conference on Control and Communications (SIBCON),* 1–5. Toukokuu. doi:10.1109/SIBCON.2016.7491821.

Natarajan, Sriram, Anantha Ramaiah ja Mayan Mathen. 2013. “A Software Defined Cloud-Gateway Automation System Using OpenFlow”. Teoksessa *2013 IEEE 2nd International Conference on Cloud Networking (CloudNet),* 219–226. Marraskuu. doi:10.1109/CloudNet.2013.6710582.

Network World. 2017. *State of the Network Survey.* Viitattu 1. marraskuuta 2017. https://cdn2.hubspot.net/hubfs/1624046/State%20of%20the%20Network%20Executive%20Summary.pdf?t=1500293732547.

Nguyen, Tri-Hai, ja Myungsik Yoo. 2016. “Attacks on Host Tracker in SDN Controller: Investigation and Prevention”. Teoksessa *2016 International Conference on Information and Communication Technology Convergence (ICTC),* 610–612. Lokakuu. doi:10.1109/ICTC.2016.7763545.

Open Networking Foundation. 2017. Viitattu 12. syyskuuta 2017. https://www.opennetworking.org.

Open vSwitch. 2016. Viitattu 13. huhtikuuta 2018. http://openvswitch.org.

OpenFlow Switch Specification Version 1.3.0 (Wire Protocol 0x04). 2012. OpenFlow Switch Specification. Open Networking Foundation. Viitattu 9. maaliskuuta 2018.

Padekar, Hitesh, Younghee Park, Hongxin Hu ja Sang-Yoon Chang. 2016. “Enabling Dynamic Access Control for Controller Applications in Software-Defined Networks”. Teoksessa *Proceedings of the 21st ACM on Symposium on Access Control Models and Technologies,* 51–61. SACMAT ’16. Shanghai, China: ACM. ISBN: 978-1-4503-3802-8. doi:10.1145/2914642.2914647.

Patel, Parveen, Deepak Bansal, Lihua Yuan, Ashwin Murthy, Albert Greenberg, David A. Maltz, Randy Kern ym. 2013. “Ananta: Cloud Scale Load Balancing”. Teoksessa *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM,* 207–218. SIGCOMM ’13. Hong Kong, China: ACM. ISBN: 978-1-4503-2056-6. doi:10.1145/2486001.2486026.

Polat, Huseyin, ja Onur Polat. 2017. “The Effects of DoS Attacks on ODL and POX SDN controllers”. Teoksessa *2017 8th International Conference on Information Technology (ICIT),* 554–558. Toukokuu. doi:10.1109/ICITECH.2017.8080058.

POX. 2017. Viitattu 27. syyskuuta 2017. https://github.com/noxrepo/pox.

procps. 2018. Viitattu 13. huhtikuuta 2018. https://gitlab.com/procps-ng/procps.

Ryu. 2017. Viitattu 26. syyskuuta 2017. https://osrg.github.io/ryu/index.html.

Ryu, versio 4.24. 2018. Lähdekoodi. Viitattu 13. huhtikuuta 2018. https://github.com/osrg/ryu/blob/master/ryu/app/simple_switch_13.py.

Röpke, Christian, ja Thorsten Holz. 2015. “SDN Rootkits: Subverting Network Operating Systems of Software-Defined Networks”. Teoksessa *Research in Attacks, Intrusions, and Defenses,* toimittanut Herbert Bos, Fabian Monrose ja Gregory Blanc, 339–356. Cham: Springer International Publishing. ISBN: 978-3-319-26362-5.

Scott-Hayward, Sandra. 2015. “Design and Deployment of Secure, Robust, and Resilient SDN Controllers”. Teoksessa *Proceedings of the 2015 1st IEEE Conference on Network Softwarization (NetSoft),* 1–5. Huhtikuu. doi:10.1109/NETSOFT.2015.7258233.

Scott-Hayward, Sandra, Sriram Natarajan ja Sakir Sezer. 2016. “A Survey of Security in Software Defined Networks”. *IEEE Communications Surveys Tutorials* 18 (1): 623–654. ISSN: 1553-877X. doi:10.1109/COMST.2015.2453114.

Security requirements and reference architecture for software-defined networking. 2016. Suositus, Series X: Data Networks, Open System Communications and Security X.1038. Telecommunication Standardization Sector of ITU. https://www.itu.int/rec/T-REC-X.1038-201610-I/en.

Shin, Seungwon, ja Guofei Gu. 2013. "Attacking Software-defined Networks: A First Feasibility Study". Teoksessa *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking,* 165–166. HotSDN '13. Hong Kong, China: ACM. ISBN: 978-1-4503-2178-5. doi:10.1145/2491185.2491220.

Shin, Seungwon, Yongjoo Song, Taekyung Lee, Sangho Lee, Jaewoong Chung, Phillip Porras, Vinod Yegneswaran, Jiseong Noh ja Brent Byunghoon Kang. 2014. "Rosemary: A Robust, Secure, and High-performance Network Operating System". Teoksessa *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security,* 78–89. CCS '14. Scottsdale, Arizona, USA: ACM. ISBN: 978-1-4503-2957-6. doi:10.1145/2660267.2660353.

Smyth, Dylan, Victor Cionca, Sean McSweeney ja Donna O'Shea. 2016. "Exploiting Pitfalls in Software-Defined Networking Implementation". Teoksessa *2016 International Conference On Cyber Security And Protection Of Digital Services (Cyber Security),* 1–8. Kesäkuu. doi:10.1109/CyberSecPODS.2016.7502354.

Sonchack, John, Adam J. Aviv ja Eric Keller. 2016. "Timing SDN Control Planes to Infer Network Configurations". Teoksessa *Proceedings of the 2016 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization,* 19–22. SDN-NFV Security '16. New Orleans, Louisiana, USA: ACM. ISBN: 978-1-4503-4078-6. doi:10.1145/2876019.2876030.

Standard for Software Defined Networking and Network Function Virtualization Security. 2017. IEEE-projekti 1915.1. IEEE Standards Association. Viitattu 3. marraskuuta. http://standards.ieee.org/develop/project/1915.1.html.

Taylor, C. R., T. Guo, C. A. Shue ja M. E. Najd. 2017. “On the Feasibility of Cloud-Based SDN Controllers for Residential Networks”. Teoksessa *2017 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN),* 1–6. Marraskuu. doi:10.1109/NFV-SDN.2017.8169827.

Tootoonchian, Amin, Sergey Gorbunov, Yashar Ganjali, Martín Casado ja Rob Sherwood. 2012. “On Controller Performance in Software-defined Networks”. Teoksessa *Proceedings of the 2Nd USENIX Conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services,* 10–10. Hot-ICE’12. San Jose, CA: USENIX Association. http://dl.acm.org/citation.cfm?id=2228283.2228297.

Tri, Hiep T. Nguyen, ja Kyungbaek Kim. 2015. “Assessing the Impact of Resource Attack in Software Defined Network”. Teoksessa *2015 International Conference on Information Networking (ICOIN),* 420–425. Tammikuu. doi:10.1109/ICOIN.2015.7057934.

tshark – The Wireshark Network Analyzer 2.4.6. 2018. Viitattu 13. huhtikuuta. https://www.wireshark.org/docs/man-pages/tshark.html.

Wang, Haopei, Abhinav Srivastava, Lei Xu, Sungmin Hong ja Guofei Gu. 2017. “Bring Your Own Controller: Enabling Tenant-Defined SDN Apps in IaaS Clouds”. Teoksessa *IEEE INFOCOM 2017 – IEEE Conference on Computer Communications,* 1–9. Toukokuu. doi:10.1109/INFOCOM.2017.8057137.

Vasconcelos, Cesar Rocha, Reinaldo Cézar M. Gomes, Anderson F. B. F. Costa ja Daniella Dias C. da Silva. 2017. “Enabling High-Level Network Programming: A Northbound API for Software-Defined Networks”. Teoksessa *2017 International Conference on Information Networking (ICOIN),* 662–667. Tammikuu. doi:10.1109/ICOIN.2017.7899569.

Wen, Xitao, Yan Chen, Chengchen Hu, Chao Shi ja Yi Wang. 2013. “Towards a Secure Controller Platform for OpenFlow Applications”. Teoksessa *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking,* 171–172. HotSDN ’13. Hong Kong, China: ACM. ISBN: 978-1-4503-2178-5. doi:10.1145/2491185.2491212.

Vengainathan, Bhuvaneswaran, Anton Basil, Mark Tassinari, Vishwas Manral ja Sarah Banks. 2018. *Benchmarking Methodology for SDN Controller Performance.* Internet-Draft. Internet Engineering Task Force, tammikuu.

Wohlin, Claes, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell ja Anders Wesslén. 2012. *Experimentation in Software Engineering.* 1. painos. Berliini/Heidelberg, Saksa: Springer. doi:10.1007/978-3-642-29044-2.

Yan, Qiao, F. Richard Yu, Qingxiang Gong ja Jianqiang Li. 2016. "Software-Defined Networking (SDN) and Distributed Denial of Service (DDoS) Attacks in Cloud Computing Environments: A Survey, Some Research Issues, and Challenges". *IEEE Communications Surveys & Tutorials* 18 (1): 602–622. ISSN: 1553-877X. doi:10.1109/COMST.2015.2487361.

Yanbing, Liu, Lu Xingyu, Jian Yi ja Xiao Yunpeng. 2016. "SDSA: A Framework of a Software-Defined Security Architecture". *China Communications* 13, numero 2 (helmikuu): 178–188. ISSN: 1673-5447. doi:10.1109/CC.2016.7405735.

Al-Zewairi, Malek, Dima Suleiman ja Sufyan Almajali. 2017. "An Experimental Software Defined Security Controller for Software Defined Network". Teoksessa *Fourth International Conference on Software Defined Systems (SDS),* 32–36. Toukokuu. doi:10.1109/SDS.2017.7939137.

Zhang, Minjian, Jianwei Hou, Ziqi Zhang, Wenchang Shi, Bo Qin ja Bin Liang. 2017. "Fine-Grained Fingerprinting Threats to Software-Defined Networks". Teoksessa *2017 IEEE Trustcom/BigDataSE/ICESS,* 128–135. Elokuu. doi:10.1109/Trustcom/BigDataSE/ICESS.2017.229.

Zhang, Peng, Huanzhao Wang, Chengchen Hu ja Chuang Lin. 2016. "On Denial of Service Attacks in Software Defined Networks". *IEEE Network* 30, numero 6 (marraskuu): 28–33. ISSN: 0890-8044. doi:10.1109/MNET.2016.1600109NM.