jamk.fi

Tietoliikennevoiden monitorointi Software-Defined Networkingissä

Pauli Raitanen

Opinnäytetyö
Joulukuu 2015

Tietotekniikan koulutusohjelma
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Kuvailulehti

Tekijä(t)	Julkaisun laji	Päivämäärä
Raitanen, Pauli	Opinnäytetyö	11.12.2015
	Sivumäärä 49 + 4	Julkaisun kieli Suomi
		Verkkojulkaisulupa myönnetty: (x)

Työn nimi
Tietoliikennevoiden monitorointi Software-Defined Networkingissä

Koulutusohjelma
Tietotekniikan (Tietoverkkotekniikan) koulutusohjelma

Työn ohjaaja(t)
Karo Saharinen
Sampo Kotikoski

Toimeksiantaja(t)
JAMK Cyber Trust
Janne Alatalo

Tiivistelmä

Opinnäytetyön toimeksiantajana toimi Cyber Trust –hanke Jyväskylän ammattikorkeakoulussa. Työn tavoitteena oli vertailla kahta eri monitorointitekniikkaa, jotka voitaisiin implementoida SDN-ympäristöön. Tekniikoiksi valikoituivat sFlow sekä Netflow.

Käytännön osuus toteutettiin virtuaalisessa VMware-ympäristössä, jossa olleille palvelimille sFlow'n monitorointi pystytettiin. sFlow'lla valvottiin Open vSwitchillä luodussa mininet-verkossa olleen kytkimen portteja.

Käytännön toteutuksessa havaittiin puutteita DDoS-mitigaatioskriptissä, joka oli tehty käytettäväksi yhdessä sFlow-RT-ohjelmiston kanssa. SDN-tekniikalle olennaisen piirteen eli nopean kehityksen vuoksi erilaiset skriptit saattavat jäädä nopeasti jälkeen, ja ne vaativat myös päivitystä.

Työ onnistui vaatimusten määrittämällä tavalla. Lopputuloksena työlle oli SDN-ympäristö, jota oli mahdollista monitoroida käyttämällä sFlow'ta.

Tulevaisuudessa monitorointia voidaan kehittää keskustelemaan yhdessä OpenFlow-kontrollerin kanssa, jotta liikenteeseen reagoiminen automaattisesti on mahdollista.

Avainsanat (asiasanat)
SDN, Openflow, sFlow, Netflow, Monitorointi

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

Description

Author(s)	Type of publication	Date
Raitanen, Pauli	Bachelor's thesis	11.12.2015
	Number of pages 49 + 4	Language of publication Finnish
		Permission for web publication: (x)

Title of publication

Software-defined networking with flow monitoring

Degree programme

Information technology

Tutor(s)

Karo Saharinen
Sampo Kotikoski

Assigned by

JAMK Cyber Trust
Janne Alatalo

Abstract

The bachelor's thesis was assigned by Cyber Trust project at JAMK University of Applied Sciences. The objective of this thesis was to compare two monitoring technologies that could be implemented into SDN network. The technologies that were chosen were sFlow and NetFlow.

The practical part was implemented into a virtual WMware environment and the monitoring was setup on the servers located in the virtual environment. sFlow was used to monitor switches created in Open vSwitches mininet.

Shorcomings were found in the practical implementation DDoS mitigation script. It was meant to be used together with sFlow-RT monitoring software. Due to the fast development of SDN-technology, a typical characteristic to SDN, the scripts might fall behind with development and require further attention to be brought up to date.

The work succeeded as was stated in the requirements. The work resulted in the implementation of monitoring of the SDN-network with sFlow setup.

In the future monitoring could be developed to communicate with OpenFlow controller to react to changes in traffic flows automatically.

Keywords/tags (subjects)

SDN, Openflow, sFlow, Netflow, Monitoring

Miscellaneous

Sisältö

KUVIOT

Lyhenteet

A-CPI	Application-Controller Plane Interface
ACL	Access List
ARP	Address Resolution Protocol
AS	Autonomous system
ASIC	Application-Specific Integrated Circuit
ASN.1	Abstract Syntax Notation One
BGP	Border Gateway Protocol
CLI	Command Line Interface
CoS	Class of Service
DDoS	Distributed Denial of Service
D-CPI	Data-Controller Plane Interface
FIB	Forwarding Information Base
LFIB	Label Forwarding Information Base
LIB	Label Information Base
MIB	Management information base
ONF	Open Networking Foundation
OSPF	Open Shortest Path First
RIB	Routing Information Base
SDN	Software-Defined Networking
SNMP	Simple Network Management Protocol
SSH	Secure Shell
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
VRF	Virtual Routing and Forwarding
QoS	Quality of Service
XDR	External Data Representation

1 TYÖN LÄHTÖKOHDAT

1.1 Toimeksiantaja

Tämän työn toimeksiantajana toimi Cyber Trust –hanke Jyväskylän ammattikorkeakoulussa. Sen päätavoitteena on palauttaa käyttäjien luottamus tietoturvaan digitaalisissa ympäristöissä sekä lisätä tietoturvan yleistä tietoisuutta kansallisesti. Hankkeeseen kuuluu useita eri tietoturvalaboratorioita Suomessa, ja se on käynnisttety toukokuussa 2015. (Savola 2014.)

Cyber Trust –hanke tähtää siihen, että Suomi on maailmanlaajuisesti tunnettu ja luotettava toimija. Tämä saavutetaan tarjoamalla korkean tason kyberturvallisuus palveluita ja ratkaisuja sekä kehittämällä ja ylläpitämällä näitä palveluita yhdessä johtavien asiantuntijoiden sekä yrityksien kanssa. (Savola 2014.)

1.2 Tehtävä ja tavoitteet

Tehtävänä oli tutkia sFlow-tekniikkaa, sekä pystyttää monitorointipalvelin SDN-verkkoon. Lisäksi haluttiin selvittää, millä tavalla SDN-kontrollerin ja sFlow-monitorointipalvelimen on mahdollista keskustella keskenään, jotta OpenFlow-pohjainen SDN-kontrolleri pystyisi reagoimaan verkossa tapahtuviin muutoksiin.

Tilaaja määritti kaksi eri tekniikkaa monitorointiin, joita tuli vertailla ja paneutua toiseen tarkemmin. sFlow valittiin tekniikaksi, jolla monitorointi toteutettaisiin ja sen lisäksi NetFlow’n ja sFlow’n eroavaisuuksia tutkitaan hieman teorian pohjalta.

Teoriaosuudessa käsiteltiin SDN-tekniikkaa, OpenFlow’ta sekä sFlow’ta. Sekä tutkittiin, kuinka sFlow ja OpenFlow pystyvät keskustelemaan toistensa kanssa, vaikuttaen reaaliaikaisesti verkon liikenteeseen.

Tavoitteena oli myös pyrkiä todistamaan käytännössä, että sFlow ja OpenFlow pystyvät keskustelemaan toistensa kanssa ja vaikuttamaan liikenteeseen reaaliajassa pienentäen haittoja, jotka mahdollisista katkoksista tai kapasiteetin loppumisista aiheutuvat.

Toissijainen tavoite opinnäytetyön toteutukselle oli implementoida jokin tapa, jolla todistetaan sFlow'n ja OpenFlow'n keskustelun toimivuus. Tarkoituksena oli, että ne pystyisivät reagoimaan verkon liikenteeseen, kun havaitaan muutoksia, jotka vaikuttavat verkon palvelutasoon.

2 SOFTWARE-DEFINED NETWORKING

2.1 Open Networking Foundation

Tällä hetkellä SDN:n (Software-Defined Networking) kehitystä ajaa eteenpäin konsortio, joka tunnetaan nimellä ONF (Open Networking Foundation). Se on täysin voittoa tavoittelematon järjestö, jonka tarkoituksena on edistää SDN:n tunnettavuutta ja johtaa teknologian kehitystä eteenpäin. (Software-Defined Networking: The New Norm for Networks 2012, 2.)

Tällä hetkellä ONF koostuu yli 150 jäsenestä, joihin kuuluu maailmalla laajalti toimivia ja tunnettuja teleoperaattoreita sekä laitevalmistajia kuten Intel, Nokia, Vodafone, China Telecom, Juniper ja Cisco sekä esimerkiksi maailman käytetyin hakukone Google. ONF:n jäsenistöstä useat kuuluvat ns. Global 500 kastiin, eli ovat maailman 500 suurimman yrityksen joukossa. (ONF Member listing. n.d.; ONF Membership. n.d.)

2.2 Tarve muutokselle

2.2.1 Yleistä

Nykypäivänä perinteiset tietoverkot eivät enää sovellu yritysten, teleoperaattorien ja loppukäyttäjien käytettäväksi, vaan ne vaativat kehitystä. Tämän vuoksi oli tarpeellista löytää uudenlainen lähestymistapa tietoverkkojen luomiseen. ONF on toiminut edelläkävijänä muuttaessaan tietoverkkoarkkitehtuuria SDN:n avulla. (Software-Defined Networking: The New Norm for Networks 2012, 2.)

Tilanteeseen on päästy, kun mobiililaitteiden ja niille tuotetun sisällön määrä on kasvanut, palvelimia on alettu virtualisoimaan ja pilvipalvelut ovat alkaneet yleistymään räjähdysmäisesti viimeisen muutaman vuoden aikana. Tämä on antanut tietoliikenneverkkoja toteuttaville tahoille aihetta miettiä uudelleen, ovatko perinteiset verkkoarkkitehtuurit enää relevantteja. Monet perinteiset tietoliikenneverkot ovat rakenteeltaan hie-

rarkisia, joihin on rakennettu kerroksia Ethernet-kytkimiä ja ne on aseteltu puurakenteeseen. (Software-Defined Networking: The New Norm for Networks 2012, 2.)

Aikaisemmin kun verkkojen arkkitehtuuri oli asiakas-palvelinpohjaista, tämä oli järkevää. Nykyisin kun datakeskuksien, kampusverkkojen ja operaattoriympäristöjen määrä on alkanut lisääntymään, on todettu, että vanhasta staattisesta mallista on päästävä eroon ja täytyy löytää dynaamisempia ratkaisuja, joilla tietoverkot toteutetaan. (Software-Defined Networking: The New Norm for Networks 2012, 3.)

2.2.2 Muuttuvat liikennevirrat

Yritysten datakeskuksissa kulkevat tietoliikennevirrat ovat muuttuneet huomattavasti. Nykyisin kun ohjelmistot hakevat tietoa erilaisista tietokannoista ja palvelimista ovat liikennevirrat muuttuneet eräänlaiseksi hälinäksi jolloin liikennettä kulkee useisiin eri suuntiin tietoverkkojen sisällä. Samaan aikaan käyttäjät muuttavat verkon liikennevirtoja, kun he pyrkivät saamaan pääsyä yrityksen tiedostoihin ja ohjelmistoihin kaiken tyyppisiltä laitteilta ja sijainnista riippumatta. (Software-Defined Networking: The New Norm for Networks 2012, 3.)

2.2.3 Pilvipalveluiden nousu

Yritykset ovat innokkaasti ottaneet vastaan molemmat, yksityiset ja julkiset pilvipalvelut, mikä omalta osaltaan on johtanut ennen näkemättömään kasvuun näissä palveluissa. Yritykset haluavat nykyisin joustavan pääsyn ohjelmistoihin, infrastruktuuriin ja muihin IT-resursseihin milloin tahansa ja mistä tahansa. Sen lisäksi, että ohjelmistoihin ja resursseihin on päästävä käsiksi milloin tahansa, on myös huolehdittava nousevista tietoturvan tarkistuksista ja vaatimuksista samalla kun yrityksen organisaatiot muuttuvat, yhdistyvät sekä oletukset voivat muuttua yhden yön aikana. Kun käyttäjille tarjotaan mahdollisuutta käyttää yrityksen resursseja yksityisessä tai julkisessa pilvipalvelussa, vaaditaan verkolta, muistilta ja tietojenkäsittelyltä joustavuutta. (Software-Defined Networking: The New Norm for Networks 2012, 4.)

2.2.4 Suuri määrä tietoa

Nykypäivänä tiedostokoot ovat kasvaneet yhä isommiksi. Tämä vaatii omalta osaltaan massiivista keskustelua palvelinten välillä, joilla kaikilla täytyy olla suora yhteys toisiinsa. Tämä aineiston kasvaminen on omalta osaltaan vaikuttamassa yhä suurempaan vaatimukseen tietoverkkojen kapasiteetissa datakeskuksissa. Nyt operaattorit, jotka hallinnoivat näitä isoja datakeskuksia, ovat pelottavan tehtävän edessä, jotta ne pystyvät säilyttämään kaikki yhteydet datakeskusten välillä ilman, että yritykset ajautuvat konkurssiin. (Software-Defined Networking: The New Norm for Networks 2012, 4.)

2.2.5 Nykyisten teknologioiden rajoittuneisuus

Kun huomioidaan nykyiset markkinoiden vaatimukset tietoverkoille, on käytännössä mahdotonta onnistua toteuttamaan tietoverkot perinteisillä teknologioilla ja menetelmillä. Nykyisellään yritysten budjettia on rajoitettu ja IT-osastojen vastuulle jää saavuttaa maksimoitu hyöty verkosta käyttäen mahdollisimman halpoja laitteistoja, mutta kuitenkin vastaten yrityksien vaatimuksiin. Operaattorit kohtaavat myös vastaavanlaisia ongelmia. (Software-Defined Networking: The New Norm for Networks 2012, 4.)

Tällä hetkellä asiakkaiden vaatimukset yhteyksille kasvavat koko ajan. Kaistaa vaaditaan lisää ja halutaan joustavampia ratkaisuja. Samalla runkoverkon laitteiden kustannukset kasvavat ja voitot operaattoreille pysyvät ennallaan tai laskevat. Niin sanottuja vanhoja tietoverkkoteknologioita ei ole suunniteltu nykypäivän dynaamisiin ja joustaviin ratkaisuihin, joita käyttäjät, yritykset ja operaattorit vaativat, vaan tietoverkkojen suunnittelijat kohtaavat ongelmia ja rajoituksia nykyisissä verkoissa. (Software-Defined Networking: The New Norm for Networks 2012, 4.)

2.2.6 Monimutkaisuus, joka johtaa seisahdukseen

Tietoverkkoteknologiat näihin päiviin asti ovat koostuneet laajalti pelkästään erillisistä protokollista, jotka on suunniteltu yhdistämään kaksi isäntää luotettavasti toisiinsa erilaisten topologioiden, matkojen ja linkkien nopeuksien avulla. Jotta liiketoiminnan teknisiin vaatimuksiin ja tarpeisiin on pystytty vastaamaan, on viimeisten vuosikymmenien aikana kehitetty erilaisia tietoliikenneprotokollia, joiden avulla saavutetaan parempi suorituskyky, yhteydellisyys sekä tiukempi turvallisuus. (Software-Defined Networking: The New Norm for Networks 2012, 4.)

Yleisesti ottaen erilaiset protokollat määritellään erikseen ja jokaisella protokollalla ratkaistaan erikseen jokin tietty ongelma. Tämä monimutkaisuus on johtanut yhtenä suurimpana tekijänä rajoittuneisuuteen nykypäivän tietoverkoissa. Esimerkiksi jos haluaa lisätä tai siirtää laitetta isommassa tietoverkossa, pääkäyttäjän on konfiguroitava useita kytkimiä, reitittimiä ja palomuureja. Näissä laitteissa tulee päivittää ACL:iä (Access List), VLAN:eja (Virtual Local Area Network), QoS:eja (Quality of Service) ja muita protokollapohjaisia mekanismeja käyttäen laitteilla olevia hallintatyökaluja. Tämän lisäksi muutoksia tehdessä tulee ottaa huomioon verkon topologia, laitteiden mallit ja ohjelmistoversiot. Tämän monimutkaisuuden vuoksi nykypäivän verkot ovat hyvinkin paikallaan pysyviä, kun palveluiden estyminen pyritään minimoimaan. (Software-Defined Networking: The New Norm for Networks 2012, 5.)

Nykyinen tietoverkkojen staattinen luonne on jyrkässä ristiriidassa palvelinten vaatimaan dynaamiseen ympäristöön. Palvelinten virtualisointi on saanut aikaan suuren nousun isäntien määrässä ja on samalla perinpohjaisesti muuttanut fyysisten sijaintien oletamuksia. Aiemmin ennen virtualisointia sovellukset sijaitsivat yhdellä palvelimella ja pääasiassa vaihtoivat liikennettä valittujen asiakkaiden kanssa. Nykypäivänä sovellukset on jaettu useille virtuaalisoiduille koneille, jotka vaihtavat liikennevirtoja toistensa välillä. (Software-Defined Networking: The New Norm for Networks 2012, 5.)

Muuttuvien virtualisointiteknologioiden lisäksi monet yritykset operoivat IP-pohjaisilla verkoilla ääni-, data- ja videoliikenteelle. Vaikka nykyisellään verkkoteknologiat pystyvätkin tarjoamaan eriytettyjä QoS-tasoja erilaisille sovelluksille, on näiden resurssien provisiointi kaikesta huolimatta erittäin manuaalista ja työlästä. Pääkäyttäjän on konfiguroitava jokainen laite erikseen sekä säädettävä kaikkien parametrit vastaamaan tahtotilaa. Tämän staattisen olemuksen takia verkko ei voi dynaamisesti muuttua tarpeiden vaatimalla tavalla. (Software-Defined Networking: The New Norm for Networks 2012, 5.)

2.2.7 Epäjohdonmukaiset politiikat

Jotta on mahdollista ottaa käyttöön koko verkon laajuinen politiikka, pääkäyttäjien täytyy mahdollisesti konfiguroida satoja tai jopa tuhansia laitteita ja mekanismeja. Tietoverkkojen monimutkaisuus tekee pääkäyttäjien tehtävästä verkonhallinnassa paljon haastavampaa, ja tähän on löydettävä uusia ratkaisuja. (Software-Defined Networking: The New Norm for Networks 2012, 5.)

2.2.8 Kykenemättömyys skaalautumaan

Kun vaatimukset kasvavat, täytyy myös tietoverkkojen kasvaa. Monimutkaisuus tietoverkoissa kasvaa kuitenkin valtavasti, kun verkkoihin tulee paljon uusia laitteita, joita joudutaan konfiguroimaan ja hallinnoimaan. Nykypäivän liikennerakenne on hyvin ennalta-arvaamatonta, minkä vuoksi linkkejä on jouduttu ylimitoittamaan niiden riittävyyden takaamiseksi. (Software-Defined Networking: The New Norm for Networks 2012, 6.)

Jotta operaattorit pystyvät pysymään kilpailukykyisinä, niiden on tarjottava yhä paremmin eriytettyjä palveluita asiakkailleen. Koska operaattorin verkon tulee palvella erilaisia asiakkaita, tämä tuo oman monimutkaisuutensa verkkojen rakentamiseen sekä suunnitteluun. Tietyt avainominaisuudet, kuten asiakasliikenteen ohjaaminen siten, että se voidaan toimittaa haluttuna aikana asiakkaalle, ovat erittäin monimutkaisia nykyisillä tietoverkoilla etenkin, kun puhutaan operaattorimittakaavan verkoista. Jotta näitä erikoisempia avainominaisuuksia pystytään tarjoamaan, ne vaativat monimutkaisempia laitteita verkkojen reunalle, mikä taas omalta osaltaan nostaa operaattorin kustannuksia

uusien palveluiden implementoimiseen tietoverkkoihin. (Software-Defined Networking: The New Norm for Networks 2012, 6.)

2.2.9 Myyjäriippuvuus

Operaattorit ja yritykset pyrkivät valjastamaan uusia palveluita yritysten ja käyttäjien tarpeen ja vaatimusten mukaan mahdollisimman nopeasti. Kuitenkin tähän alati muuttuvaan kysyntään on vaikea vastata johtuen laitemyyjistä, joiden tuotteiden elämänkaari saattaa olla jopa kolme vuotta tai enemmänkin. Standardien ja avoimien rajapintojen puuttuminen rajoittaa operaattorien mahdollisuuksia räätälöidä tietoverkkoja omiin tarkoituksiinsa. Tämä yhteensopimattomuus verkon kykenevyyden ja markkinoiden vaatimusten kanssa on johtanut omalta osaltaan SDN-arkkitehtuurin kehittämiseen. (Software-Defined Networking: The New Norm for Networks 2012, 6)

2.3 Tekniikka

2.3.1 Arkkitehtuuri

SDN on uusi nouseva tietoverkkojen arkkitehtuuri, jonka ONF määrittelee seuraavasti. Siinä verkon hallinta on eriytetty välitystasosta ja sen hallinta on keskitetty. Hallinnan eriyttämisellä saavutetaan helpompi hallittavuus ja verkon infrastruktuuri on irroitettuna ohjelmista.(Software-Defined Networking: The New Norm for Networks 2012, 7.)

Yksi tärkeä osa SDN-teknologiaa on OpenFlow. Se on avoin standardi kommunikaatioprotokollalle, joka mahdollistaa hallintatason ja välitystason välisen keskustelun. On myös mainittava, että OpenFlow ei ole ainut protokolla, jota kehitetään hallintatason ja välitystason keskustelun välittämiseksi. (What’s Software-Defined Networking (SDN)? n.d.)

SDN-arkkitehtuuri sisältää pääosin kahta erilaista laitetyyppiä: kontrollereita ja verkkoelementtejä. Verkkoelementit ovat esimerkiksi kytkimiä tai reitittimiä. Tämän lisäksi

SDN:ssä on neljä erilaista verkon osaa: ne ovat data-, hallinta-, kontrolli- ja Cluster interconnect -verkot. (Karhinen 2015, 17.)

Kontrolli- ja välitystaso ovat nykyisten tietoverkkojen, ydin kun halutaan liikuttaa pakettaja eri pisteiden välillä. Kolmas tärkeä komponentti on hallintataso, joka on pääosin käyttäjän ja laitteiston välistä vuorovaikutusta. Nämä operointitasot ovat perusta kerroksittain rakennetulle arkkitehtuurille, joihin tietoverkot ovat kehittyneet nykypäivään mennessä. Salisburyn mukaan data- ja välitystasolla ei juurikaan ole eroavaisuuksia josta johtuen tässäkin opinnäytetyössä molempia käsitellään välitystason alla. (Salisbury 2012.)

SDN-verkossa kontrollitason tehtävät hoitaa oma laitteensa, jota kutsutaan kontrolleriksi. Sen tehtävänä on hallinnoida verkkoa siten, että se näkyy loogisesti vain yhtenä kytkimenä, jolloin verkon hallinta helpottuu. Kun kontrolleri on keskitetty, tällöin ei verkkolaitteiden tarvitse ymmärtää useita erilaisia protokollia, vaan ne ovat kontrollerin vastuulla. (Karhinen 2015, 17.)

SDN-kontrolleri ja verkkolaitteet ovat jatkuvasti yhteydessä toisiinsa. Tämä keskustelu hoidetaan erilaisilla protokollilla, joista tässä työssä on käytössä OpenFlow. Muita kehitteillä olevia protokollia ovat NetConf, OVSDB sekä YANG. Protokolla, jota käytetään kontrollerin ja verkkolaitteiden väliseen keskusteluun, voi vaikuttaa verkon arkkitehtuuriin. (Karhinen 2015, 18.)

2.3.2 Verkkojen eri tasot

Kontrollitaso

Kontrollitaso on komponentti, joka keskittyy siihen, miten yksittäinen verkon laite on yhteydessä naapureihinsa. RIB (Routing Information Base) ja LIB (Label Information Base) käsitellään ohjelmallisesti ja niiden käsittelyn tuloksilla täytetään FIB (Forwarding Information Base) sekä LFIB (Label Forwarding Information Base). Toimittajat voivat toteuttaa kannat eri muoteilla, kuinka reititystaulut jaetaan osiin useiden reititystapausten välillä. Esimerkiksi jollain reitittimellä on BGP- (Border Gateway Protocol) ja OSPF-

reititysprotokollien (Open Shortest Path First) naapuruuksia. Näillä reititysprotokollilla on erilaiset algoritmit, joilla ne päättävät, mitä reitittiä käytetään päästäkseen haluttuun verkkoon. (Salisbury 2012.)

RIB on tietoverkoissa käytettävä reititystaulu. Reitittimellä voi myös olla useampi reititystaulu, jos käytössä on VRF:iä (Virtual Routing and Forwarding). VRF:iä on taas käytössä yksi per asiakas, ja jokaisella VRF:llä on oma RIB. FIB on optimoidumpi versio RIB:stä. Tarkemmin sanottuna se on taulu, josta reititin katsoo, mihin liikenne itse asiassa kuuluu välittää. Kuten RIB:ssä reitittimellä voi olla myös useampi FIB. LIB on MPLS-taulukko. Tässä taulukossa pidetään kaikki tunnetut MPLS-leimat. Viimeiseksi on LFIB, joka on myöskin MPLS-taulukko. Tätä taulua reititin käyttää välittämään leimatut paketit verkon lävitse. Kuten RIB käyttää FIB:iä liikenteen välittämiseen, niin käyttää myös LIB LFIB:iä. (O'Connor 2011.)

Nykyisin lähes jokainen keskiverto reititin pystyy toimimaan itsenäisesti. Reitittimen kontrollitaso voi olla erillinen naapuristaan, ikään kuin ne sijaitsisivat omissa hallinnollisissa alueissaan. (Salisbury 2012.)

Kontrollitaso syöttää välitystasolle tietoa, kuinka sen tulee luoda välitystaulunsa, ja myös päivittää topologiamuutokset kun niitä tapahtuu. Päivityksiä tapahtuu hyvin verkkaisella aikavälillä. Tämän vuoksi kontrollitasoa voidaan joskus kutsua hidas polku –nimellä. Perinteisen reititysmoottorin tehtäviin voidaan luokitella seuraavia asioita:

- Resurssien jakaminen välitystasolle
- Reititystila
- ARP (Address Resolution Protocol) kyselyjen käsittely yleiskäyttöisellä prosessorilla
- Tietoturvallisuusfunktiot kontrollitason pääsyn turvaamiseksi käyttäen esimerkiksi Telnettiä tai SSH:ta (Secure Shell)
- Hallintaistuntojen muodostaminen ja säilyttäminen
- Reititystilan välitys naapuriverkkoelementeille

- Valmistaja ja alustakohtaiset toiminnot kuten pinot, klusterit tai pariuttaminen (Salisbury 2012.)

Välitystaso

Välitystaso on tietoverkkojeme työjuhta. Sen vastuulla on pakettien otsakkeiden jäsentäminen, joka tapahtuu ASIC-piireissä (Application-specific integrated circuit). Se hallinnoi mm. palvelun laatua, filtteröintiä, enkapsulointia ja jonotusta. (Salisbury 2012.)

Välitystasolla tehtävät operaatiot on oltava nopeita, jotta pystytään pysymään mukana suorityskyvyn vaatimuksissa datakeskuksissa ja ydinverkoissa. Tämän takia välitystasoa voidaan kutsua nopeaksi poluksi. (Salisbury 2012.)

Hallintataso

Hallintatason vastuulla on hoitaa käyttäjän pääsy laitteisiin käyttäen esimerkiksi telnettiä tai SSH:ta. (Salisbury 2012.)

2.3.3 SDN-kontrolleri

SDN-tekniikassa nopea ja hidas polku tarkoittavat seuraavaa: ensimmäinen paketti joudutaan käsittelemään kontrollerilla, jotta se voi tehdä tarvittavat muutokset kontrollitasolle sekä päivittää verkkoelementeillä sijaitsevat tietokannat välitystasolle. Tätä kutsutaan hitaaksi poluksi. (Salisbury 2012.)

Seuraavia paketteja, jotka kulkevat samaa reittiä ei tarvitse käsitellä kontrollerilla, koska niille on olemassa jo reitti verkossa. Tällöin voidaan käyttää suoraan välitystasoa kontrollitason sijasta ja paketin välittäminen on nopeampaa. Tämä on niin sanottu nopea polku. (Salisbury 2012.)

SDN-ohjelma (SDN application) kommunikoi kontrollitasolla olevan SDN-kontrollerin kanssa. SDN-ohjelmalla voi olla oikeus hallita kontrollerin sille antamia resursseja, tai kaikki päätäntävalta voi olla myös vain kontrollerilla. Kontrollerin ja SDN-ohjelman välinen liikenne on ohjelman ja kontrollerin A-CPI-rajapinnan (Application-Controller Plane Interface) välistä liikennettä. Verkon fyysisten laitteiden ja kontrollerin välinen liikenne

taas tapahtuu D-CPI-rajapinnan (Data-Controller Plane Interface) avulla. (Karhinen 2015, 24)

SDN-tekniikassa on olemassa oma kontrollilogiikka (SDN Control Logic). Sen tehtävä on vastaanottaa SDN-ohjelmien pyynnöt, jotka se sitten muuttaa verkkoelementeille lähetettäviksi ohjeiksi. (Karhinen 2015, 25.)

2.4 OpenFlow

2.4.1 Yleistä

OpenFlow on kontrolli- ja välitystason välille kehitetty rajapinta, joka mahdollistaa välitystason muokkaamisen SDN-verkkolaitteella. Verkkolaitteet OpenFlow'ssa tunnetaan OpenFlow-kytkiminä (OpenFlow-switch), joista jokainen koostuu yhdestä tai useammasta vuotaulusta (flow table) sekä ryhmätaulusta (group table). OpenFlow käyttää vuo- ja ryhmätauluja hoitaakseen pakettien välittämisen. (Karhinen 2015, 37)

Vuotaulut sisältävät vuomerkintöjä. OpenFlow-protokollalla pystytään muokkaamaan, lisäämään tai poistamaan näitä merkintöjä. Vuomerkinnät sisältävät vertailukentän, laskurit ja ohjeet siitä, kuinka kytkimen tulisi pakettia käsitellä. Vuotaulujen vertailua kutsutaan putkistoprosessiksi, kun vertailuun käytetään useampaa taulua. Vertailu aloitetaan ensimmäisestä vuotaulusta ja sitä jatketaan tarvittaessa muihin vuotauluihin. Vuotauluista etsitään paketeille prioriteettijärjestyksessä vastaavuuksia vuomerkinnöistä. Jos vastaavuus löytyy, toteutetaan vuomerkinnän sisältämät toimenpiteet paketille. Jos vastaavuutta ei kuitenkaan löydy, voidaan paketti välittää kontrollerille, pudottaa tai välittää edelleen seuraavalle vuotaululle. Toiminto, joka paketille tehdään vastaavuuden puuttuessa, määritetään kytkimen konfiguraatiossa. (Karhinen 2015, 37.)

Vuotaulujen toimintaohjeet voivat sisältää toimintoja tai muokkaavat putkistoprosessia. Toiminnoilla tarkoitetaan pakettien välitystä, muokkausta ja ryhmäkäsittelyä. Putkistoprosessin ohjeilla sallitaan pakettien välittäminen seuraaville vuotauluille, jotta pakettia voidaan lisäkäsitellä. Putkistoprosessi päättyy, kun vuomerkinnässä ei enää määritellä vuotaulua, johon paketti siirrettäisiin. Kun seuraavaa vuotaulua ei enää löydy, on paketti tällöin muokattu sekä välitetty eteenpäin. (Karhinen 2015, 38.)

Vuomerkkien toiminnoissa paketit voidaan välittää myös ryhmään käsiteltäväksi. Ryhmään välittämisellä paketille määrätään lisää käsittelyä. Tämä käsittely voi esimerkiksi sisältää välitystoimintoja tai tulvittamista. (Karhinen 2015, 38.)

2.4.2 Perusteet

Open-Flow on protokolla, jossa kontrolleri ja kytkin keskustelevat OpenFlow-kanavan avulla. Kanava on tarkoitettu kontrollerille, jotta se saa tietoa kytkimeltä sen tapahtumista ja voi konfiguroida sekä hallinnoida kytkintä. OpenFlow-kytkimen kontrollikanavassa voi olla käytössä useampi kanava, jolloin kytkin on yhteydessä samanaikaisesti useampaan kontrolleriin. Kontrollikanavassa käytetään yleisesti TLS-salausta, vaikka sitä joskus käytetäänkin salaamattomana TCP-yhteydellä. (Karhinen 2015, 41.)

OpenFlow käyttää kolmea erilaista viestityyppiä, jotka kaikki sisältävät erilaisia alityyppejä. Nämä viestityypit ovat kontrollerilta kytkimelle, asynkroninen ja symmetrinen. Kontrollerilta kytkimelle viestit ovat pääasiallisesti kytkimen hallinnointiin sekä tilan tarkistamiseen käytettäviä viestejä. Asynkroniset viestit ovat kytkimeltä kontrollerille lähetettäviä viestejä, joilla kytkin päivittää kontrollerille viimeisimmät tiedot tapahtumista kytkimellä. Symmetriset viestit taas ovat molemminsuuntaisia, ne voi olla lähettänyt kytkin kontrollerille tai kontrolleri kytkimelle. (Karhinen 2015, 41.)

3 MONITOROINTITEKNIIKAT

3.1 sFlow

3.1.1 Yleistä

sFlow on teknologia, jolla pystytään monitoroimaan liikennettä tietoverkoissa, jotka sisältävät reitittimiä ja kytkimiä. sFlow-monitorointijärjestelmä koostuu sFlow-agentista (sulautettu reitittimeen, kytkimeen tai itsenäiseen anturiin) ja keskitetystä sFlow-keräimestä. sFlow'n arkkitehtuuri ja näytteenottotekniikat suunniteltiin toimittamaan jatkuvaa monitorointi tietoja yrityksen ja toimipisteen laajuisesti. Tällä mallilla erityisesti saadaan katettua ongelmat, jotka liittyvät

- tarkkaan monitorointiin suurilla nopeuksilla (Gigatavu tai korkeampi)
- skaalautuvuuteen monitoroida kymmeniä tuhansia agentteja yhdellä sFlow keräimellä
- pienen kustannuksen sFlow agenttien implementointi (Phaal & Lavine 2004, 1.)

sFlow-agentti käyttää näytteenottoteknologiaa saadakseen tietoa liikenteestä laitteelta, jota se monitoroi. sFlow-datagrammeja käytetään välittämään välittömästi näytteitä sFlow-keräimelle analysointia varten. Tällä hetkellä sFlow'sta on käytössä versio viisi. (Phaal & Lavine 2004, 1.)

3.1.2 Terminologia

sFlow'n arkkitehtuurin määrittämiseen käytettäviä termejä:

Verkkolaite (Network Device) on esimerkiksi kytkin tai reititin, joka välittää paketteja. (Phaal & Lavine 2004, 2-3.)

Tietolähde (Data Source) viittaa verkkolaitteessa sijaintiin, joka voi tehdä mittauksia. Mahdolliset tietolähteet sisältävät muun muassa liitännät ja fyysiset yksiköt laitteessa. Jokaisella tietolähteellä on pääsy osaan laitteen läpi kulkevasta liikenteestä. Jos verkkolaite halutaan kokonaisuudessaan valvontaan, tällöin tietolähteiden määrä ja tyyppi vaihtelevat laitteen arkkitehtuurin mukaan. Tyypillisesti jokaiselle liitännälle määritellään oma tietolähde. Tällä varmistetaan jokaisen paketin tarkkailu. (Phaal & Lavine 2004, 2-3.)

Pakettivirta (Packet Flow)määritellään sinä polkuna tai ratana, jonka paketti kulkee verkkolaitteen läpi. Esimerkiksi kun paketti vastaanotetaan liitäntään, täytyy tehdä kytkentä/reitityspäätös, jonka jälkeen paketti lähetetään toiseen liitäntään. (Phaal & Lavine 2004, 2-3.)

Pakettivirran näytteenotto (Packet Flow Sampling) viittaa satunnaisvalintaan osasta pakettivirtoja, joita tarkkaillaan tietolähteessä. (Phaal & Lavine 2004, 2-3.)

Näytteenoton nopeus (Sampling Rate) määrittelee suhteen, jolla havaituista paketeista tehdään näytteitä. Esimerkiksi, jos näytteenoton nopeus on 100, se tarkoittaa, että jokaista 100:aa havaittua pakettia kohden luodaan yksi näyte. (Phaal & Lavine 2004, 2-3.)

Pakettivirran tallenne (Packet Flow Record) kuvaa pakettivirran ominaisuuksia. On olemassa kahdenalaista informaatiota tallenteissa:

- Itse paketin informaatio. Tyypillisesti otsake, paketin pituus tai kapselointi.
- Tieto polusta, jonka paketti on kulkenut laitteen läpi sisältäen tiedon edelleen lähettämisen reitistä. (Phaal & Lavine 2004, 2-3.)

Laskurien näytteenotto (Counter Sampling) tarkoittaa tietolähteen laskurien näytteenottoa tai pollausta. (Phaal & Lavine 2004, 2-3.)

Näytteenoton aikavälillä (Sampling Interval) tarkoitetaan aikaväliä, jolla perättäisiä näytteitä laskureista otetaan. (Phaal & Lavine 2004, 2-3.)

Laskurien tallenne (Counter Record) sisältää tietolähteen laskurin arvoja näytteenoton aikavälin lopusta. (Phaal & Lavine 2004, 2-3.)

sFlow instanssi (sFlow Instance) viittaa mittausprosessiin tietolähteessä. Yhdessä tietolähteessä voi olla yksi tai useampi mittausprosessi samanaikaisesti. Jokainen sFlow-instanssi toimii itsenäisesti muihin nähden. (Phaal & Lavine 2004, 2-3.)

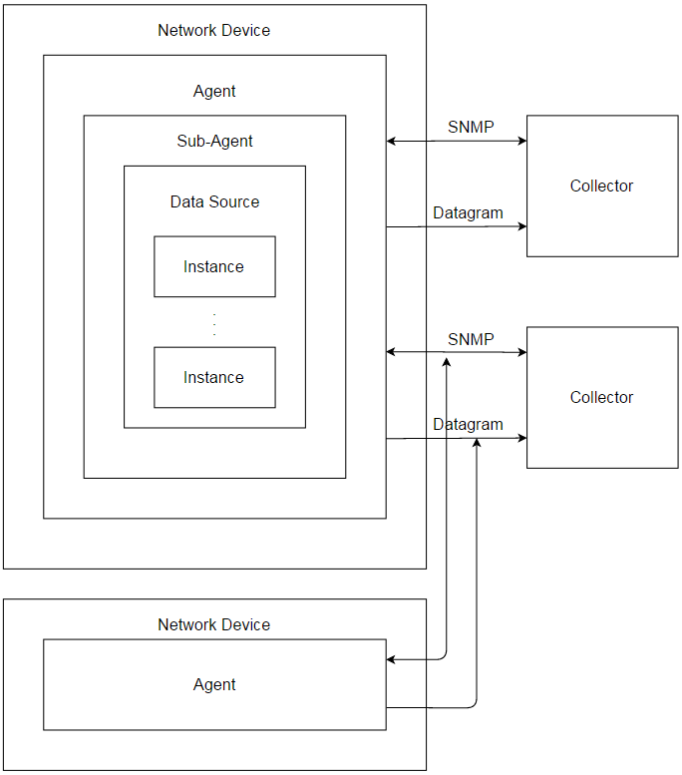
sFlow agentti (sFlow Agent) tarjoaa rajapinnan sFlow-instanssien konfiguroimiseen laitteelle. Agentti voi tukea komentorivi- tai SNMP-pohjaista (Simple Network Management Protocol) konfigurointia. Sen vastuulla on myös ylläpitää mittausistunnot sFlow-kerääjien kanssa. Se sijoittaa tietoa sFlow-datagrammeihin, jotta ne voidaan lähettää sFlow-keräilijöille. sFlow-agentti myös vapauttaa resursseja, kun istunto päättyy. (Phaal & Lavine 2004, 2-3.)

sFlow aliagenttia (sFlow Sub-Agent) voidaan haluta käyttää tilanteissa, joissa sFlow on implementoitu jaettuun laitearkkitehtuuriin. Tällöin sFlow-agenttien toiminnallisuus voidaan haluta jakaa. Jokainen aliagentti on vastuussa omasta osajoukostaan tietolähteissä. (Phaal & Lavine 2004, 2-3.)

sFlow kerääjä (sFlow Collector) vastaanottaa sFlow-datagrammit yhdeltä tai useammalta sFlow-agentilta. Kerääjä voi myös konfiguroida sFlow-tapauksia käyttäen konfiguraatiomekanismeja, jotka sFlow-agentti on tarjonnut. (Phaal & Lavine 2004, 2-3.)

sFlow datagrammi (sFlow Datagram) on UDP-datagrammi (User Datagram Protocol), joka sisältää mittausdatan sekä tietoa mittauskohteesta ja prosessista. (Phaal & Lavine 2004, 2-3.)

Kuviossa 1 on kuvattuna, kuinka edellä mainitut osat sFlow-järjestelmässä asettuvat kokonaisuuteen.



Kuvio 1. sFlow-järjestelmän yhteydet eri kokonaisuuksien kanssa

3.1.3 Arkkitehtuuri

sFlow-kerääjä hyödyntää SNMP:tä kommunikoidessaan sFlow-agentin kanssa konfiguroidakseen valvonnan verkkolaitteeseen. sFlow MIB (Management Information Base) määrittää hallitut objektit, jotta sFlow-agentti voidaan konfiguroida. sFlow-instanssi suorittaa pakettivirran näytteenotot sekä laskurien näytteenotot, jotka liittyvät yksittäisiin tietolähteisiin sFlow-agentissa. Jotta sFlow-instanssi voi kerätä näytteitä pakettivirrasta, sille on konfiguroitava näytteenoton nopeus. Pakettivirran näytteenoton prosessin tuotoksena generoidaan pakettivirran tallenteita. Jotta on mahdollista suorittaa näytteenottoa laskureista, on sFlow-instassille konfiguroitava näytteenoton aikaväli. Tällä prosessilla generoidaan laskurien tallenteet. sFlow-agentit keräävät laskurien tallenteita, sekä pakettivirran tallenteita, jotka lähetetään edelleen sFlow datagrammeina sFlow kerääjille. (Phaal & Lavine 2004, 4-5.)

3.1.4 Näytteenottotekniikat

sFlow-agentti käyttää kahta erilaista näytteenottotapaa statistista pakettipohjaista reititetyistä tai kytketyistä pakettivirroista sekä aikapohjaista näytteenottoa laskureista. Alla esitellään tavat tarkemmin joilla näytteitä voidaan ottaa. (Phaal & Lavine 2004, 5.)

Pakettivirtojen näytteenotto

Jokainen sFlow-instanssi suorittaa pakettvirtojen näytteenottoa. Sen on varmistettava, että jokainen tietolähteellä tarkistettu paketti saa yhtä suuren mahdollisuuden tulla näytteeksi katsomatta pakettivirtaa, johon se kuuluu. (Phaal & Lavine 2004, 5.)

Pakettivirran näytteenotto suoritetaan seuraavasti: Kun paketti saapuu liitäntään, verkkolaite tekee sille suodatuspäätöksen pudotetaanko paketti vai ei. Jos pakettia ei ole suodatettu kohde liitätännän määrää kytkentä tai reititysfunktio. Tämä mekanismi sisältää laskurin, joka pienenee jokaista käsiteltyä pakettia kohden. Kun laskuri saavuttaa arvon nolla näyte otetaan. Vaikka näytettä ei olisikaan otettu, Total_Packets-laskuria kasvatetaan silti. Total_Packets on laskuri, joka kertoo kaikkien mahdollisten pakettien määrän joista näyte olisi voitu ottaa. Näytteen ottaminen sisältää joko paketin otsakkeen kopioimisen tai sen ominaisuuksien talteen ottamisen. (Phaal & Lavine 2004, 5.)

Joka kerta kun näyte otetaan, Total_Samples-laskuria kasvatetaan. Total_Samples laskurilla kerrotaan kaikkien syntyneiden näytteiden määrä. Näytteet lähetetään sFlow-instanssin toimesta sFlow-agentille prosessoitavaksi. Näyte sisältää paketin informaation sekä Total_Packets- ja Total_Samples-laskurien arvot. sFlow-agentti voi käyttää näytteitä saadakseen lisätietoa paketin kulusta laitteen läpi. Esimerkkejä näistä tiedoista ovat lähde ja kohde liitäntä, lähde ja kohde VLAN, seuraavan hypyn aliverkko sekä täydellinen AS-polku (Autonomous system). (Phaal & Lavine 2004, 5-6.)

Kun näyte otetaan laskuri, joka kertoo yli hypättävien pakettien määrän ennen seuraavaa näytettä on resetoitava. Laskurin arvo tulisi asettaa johonkin sattumanvaraiseen kokonaislukuun, joka on riippuvainen Tota_Packets ja Total_Samples arvoista. Kaava on Total_Packets/Total_Samples = Sampling Rate, eli kokonaispakettien määrä jaettuna kaikkien näytteiden määrällä tulisi vastata näytteenoton nopeutta. (Phaal & Lavine 2004, 6.)

Vaihtoehtoinen tapa pakettivirtojen näytteenottoon on luoda satunnainen numero jokaiselle paketille, jonka jälkeen tuota numeroa verrataan valmiiksi määriteltyyn kynnykseen. Jos kynnystä ei ylitetä, tällöin otetaan näyte. Sopivan kynnyksen laskeminen riippuu satunnaislukugeneraattorin ominaisuuksista. Siitä huolimatta näytevirran tulee olla edellisessä kappaleessa mainitun kaavan mukainen. (Phaal & Lavine 2004, 6.)

Laskurien näytteenotto

Laskurien näytteenoton tavoitteena on tehokkaasti tuoda tietoa laskureista tietolähteiden sisällä. Tyypillisesti tietolähde liittyy jokaiseen liitäntään verkkolaitteessa, joka voi olla ulosmeno tai sisääntulo liitäntä pakettivirroille. Näitä liitäntöjen tietolähteitä sitten käytetään viemään liitäntöihin liittyviä laskureita. (Phaal & Lavine 2004, 6.)

Enimmäis näytteenoton aikaväli on sidottu jokaiseen sFlow-instanssiin, joka liittyy liitäntöjen tietolähteeseen. Kuitenkin sFlow-agentilla on vapaus aikatauluttaa pollaamista maksimoidakseen sisäisen tehokkuuden. (Phaal & Lavine 2004, 6.)

Pakettivirtojen- ja laskureiden näytteenotto on suunniteltu osana integroitua järjestelmää. Molemmat näistä näytetyypeistä yhdistetään sFlow-datagrammeissa. Koska pakettivirtojen näytteenotto lähettää näytteitä sFlow-kerääjille tasaiseen tahtiin, mutta kuitenkin satunnaisest, voidaan laskurien näytteitä ottaa opportunistisesti jotta datagrammit saadaan täytettyä. (Phaal & Lavine 2004, 6.)

Yksi tapa laskurien näytteenottoon on sellainen, jossa sFlow-agentti säilyttää listaa kaikista näytteiden lähteistä. Kun pakettivirrasta otetaan näyte, sFlow-agentti tutkii listaa ja lisää laskurin näytteen datagrammiin. Laskurien näyte lisätään datagrammiin vain, jos ne ovat vain pienen ajan, kuten viiden sekunnin välillä näytteenoton aikavälistä. Aina kun laskurin lähteen statistiikkaa lisätään datagrammiin, on viimeisen näytteen aikaa päivitettävä ja lisättävä listan pohjalle. Määräajoin sFlow-agentti tarkistaa listan laskurien lähteistä ja lähettää kaikki laskurit, jotka tarvitsee lähettää, jotta ne vastaavat näytteenoton aikaväliä. (Phaal & Lavine 2004, 6-7.)

Jos sFlow-agentti päättää tehdä säännöllisesti laskurien näytteet, tulee sen silloin määrittää kaikille laskurien lähteille oma aika, jotta näytteiden aika ei ole synkronoitu agentin sisällä tai muiden agenttien kanssa. (Phaal & Lavine 2004, 7-8.)

3.1.5 sFlow-MIB

sFlow-MIB tarjoaa standardin mekanismin, jolla voidaan etänä hallita ja konfiguroida sFlow-agenttia. Pääsy hallittuihin objekteihin tapahtuu virtuaalisen informaatio varaston kautta, jota kutsutaan MIB:ksi. MIB koostuu kolmesta eri ryhmästä objekteja, jotka ovat vastaanotinryhmä, virtauksen näyteryhmä ja laskurien pollaus ryhmä. (Phaal & Lavine 2004, 7-8.)

Vastaanotin ryhmä

Tämä ryhmä määrittää objektit, joilla ylläpidetään sFlow-instuntoa agentin ja kerääjän välillä. Ennenkuin sFlow-kerääjä voi tehdä mitään muutoksia konfiguraatioon, on sen löydettävä tyhjä rivi vastaanotin taulukosta ja vaatia se itselleen kirjoittamalla omistajan jono, sekä varausaika vapaalle riville taulukossa. Istunto katkaistaan automaattisesti ja resurssit vapautetaan mikäli kerääjä ei määräajoin käy päivittämässä taulukon kirjausta. Säännöllisesti vastaanotin taulun merkintää päivittämällä sFlow-kerääjä varmistaa, että sen osoitteen ovat oppineet kaikki sillat polulla, joka kulkee sFlow-agentille. Samalla minimoiden riskit, joissa sFlow-datagrammit tulvittuisivat sillan toimesta. (Phaal & Lavine 2004, 8.)

Merkintöjä ei voi lisätä tai poistaa vastaanotin ryhmän taulusta. sFlow-agentin toteuttajan tulee rajoittaa suurin samanaikaisten istuntojen määrä manuaalisesti siten, että laitteen maksimi kapasiteetti ei ylity. (Phaal & Lavine 2004, 8.)

Kun rivi vastaanotin taulusta on hankittu sFlow-kerääjä määrittää osoitteen ja portin, joita se käyttää sFlow-datagrammien vastaanottamiseen. sFlow-datagrammien maksimi koko voidaan konfiguroida, jotta pystytään estämään pakettien sirpaloitumista. (Phaal & Lavine 2004, 8.)

Virtauksen näyteryhmä

Tämä ryhmä määrittää sijainnit tai tietolähteet verkkolaitteessa jotka kykenevät pakettivirran näytteenottoon. Tietolähteet voivat vastata liitäntöjä, VLAN:eja tai muita kokonaisuuksia verkkolaitteessa. Se ryhmä tietolähteitä, joita sFlow-agentti mainostaa riippuu laitteen arkkitehtuurista. Esimerkiksi laitteella voi olla integroitu pakettivirtojen näytteenotto ASIC:ssä. Tässä tapauksessa se mainostaa tietolähdettä jokaisesta liitännästä. Vaihtoehtoisesti ohjelmistopohjaisella reitittimellä voi olla vain yksi tietolähde reititysmoduuliin. (Phaal & Lavine 2004, 9.)

Jokainen tietolähde voi kyetä tukemaan yhtä tai useampaa itsenäistä näyteprosessia. Tällöin jokaisella prosessilla on useampi tietolähteeseen liittyvä sFlow-instanssi. Jokaisella sFlow-instassilla voi olla oma näytteenoton nopeus. (Phaal & Lavine 2004, 9.)

Vaikka tietolähteen raudalle olisi mahdollistas tuottaa vain yksi virta näytteitä paketeista, on mahdollista että sFlow-agentti käyttää alinäytteenottoa luodakseen useamman sFlow-instanssin tietolähteelle. Kun sallitaan vain toisen potenssin näytteenoton nopeudet, voidaan rautaan konfiguroitava nopeus pitää pienimpänä ja ohjelmallisesti tehdä alinäytteenottoa muilla nopeuksilla. (Phaal & Lavine 2004, 9.)

Laskurin pollaus ryhmä

Tämä ryhmä määrittää sijainnit tai tietolähteet laitteessa, jotka voivat tarjota laskurien informaatiota. Tyypillisesti nämä tietolähteet ovat liitäntöjä verkkolaitteessa. Jokainen tietolähde voi kyetä tukemaan yhtä tai useampaa yksittäistä pollaus prosessia. Tässä tapauksessa on olemassa useampi laskurien pollaus instanssi jokaiselle tietolähteelle.

Jokaisella laskurin pollaus instanssilla voi olla oma itsenäinen pollausaikaväli. (Phaal & Lavine 2004, 9-10.)

3.1.6 sFlow-Datagrammi formaatti

sFlow-datagrammin formaatti määrittää standardin, jolla sFlow-agentti lähettää näytedataa sFlow-kerääjälle. Formaatti, jota sFlow-datagrammit käyttävät on XDR (External Data Representation) standardi. XDR on kompaktimpi ja yksinkertaisempi tapa sFlow-agenttille koodata ja sFlow-kerääjälle purkaa kuin ASN.1 (Abstract Syntax Notation One). (Phaal & Lavine 2004, 22-23.)

Näytteet lähetetään UDP-paketteina isännälle, joka määritellään sFlow-MIB:ssä. sFlow'lle määritelty portti on 6343, sama portti on myös sFlow-MIB:n oletusportti. Standardoitu sFlow-portti auttaa eliminoimaan konfiguraatio ongelmia sFlow-agenttien ja sFlow-kerääjien välillä, sFlow-liikenne on helpompi tunnistaa ja sen tulisi helpottaa liikenteen välittämistä laitteiden, kuten palomuurien, läpi. (Phaal & Lavine 2004, 23.)

UDP:n epäluotettavuus ei merkittävästi vaikuta mittauksien tarkkuuteen, joita sFlow-agentilta saadaan. Jos laskurien näytteitä menetetään, uudet arvot lähetetään seuraavan pollausaikavälin aikana. Nettovaikututs, joka menetetyistä pakettivirran näytteistä tulee, on pieni lasku tehokkaassa näytteenottotaajuudessa. (Phaal & Lavine 2004, 23.)

UDP:n käyttäminen vähentää puskuroitavan datan muistinkäyttöä ja lisäksi tarjoaa lujatekoisen tavan välittää ajastettua liikenneinformaatiota intensiivisen liikenteen (kuten palvelunestohyökkäyksen) aikana. UDP on lujatekoinen mekanismi luotettavaan tiedonsiirtoon, koska ainut vaikutus, joka sillä on kokonais suorituskykyyn on pieni nousu tiedonsiirron viiveessä, sekä suuri määrä hävitettyjä paketteja, joista kummallakaan ei ole suurta vaikutusta sFlow-pohjaisessa monitorointijärjestelmässä. Jos käytössä olisi luotettava siirtotekniikka toisi se mukanaan pitkät siirtoviiveet, sekä agentille olisi vaatimuksena isompi määrä muistia. (Phaal & Lavine 2004, 23.)

sFlow-datagrammin rakenne sallii siis useamman näytteen sisällyttämisen jokaiseen datagrammiin. sFlow-agentti ei voi odottaa puskurin täyttymistä ennenkuin se lähettää

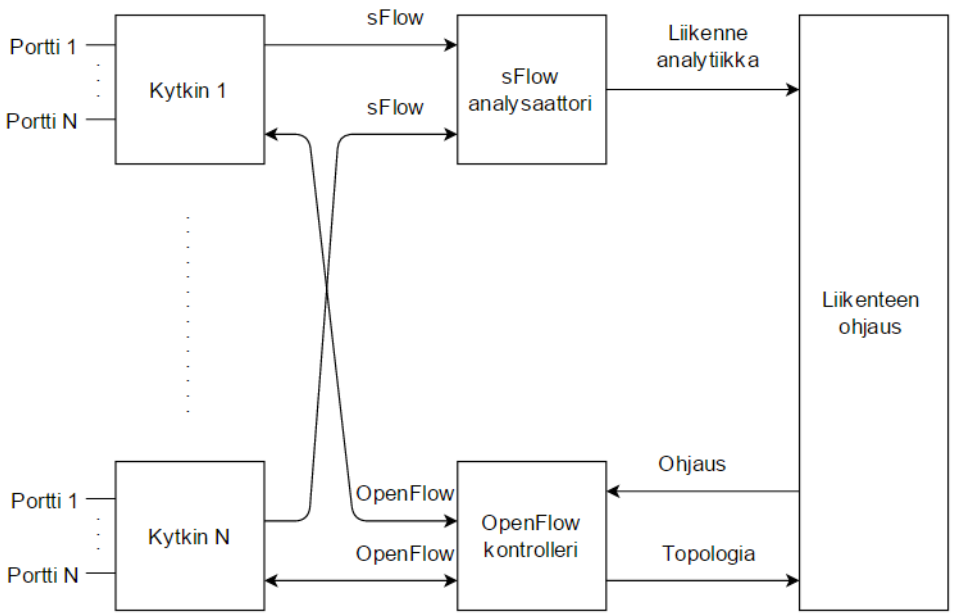
datagrammin. sFlow on suunniteltu tarjoamaan ajankohtaista tietoa liikenteestä. sFlow-agentti voi enimmillään viivästyttää näytteen lähettämistä yhdellä sekunnilla ennenkuin sen lähettäminen pakotetaan. (Phaal & Lavine 2004, 23.)

Ennenkuin agentti lähettää datagrammin, puskurissa jäljellä oleva tila voidaan käyttää laskurien näytteisiin. sFlow-agentti ottaa laskureista näytteitä harkintaa käyttäen siten, että sFlowCounterSamplingInterval eli laskurin näytteen aikaväli on maksimi arvo joka näytteiden ottamisen välillä voi pisimmillään olla. sFlow-agentti voi halutessaan ottaa näytteen siis aiemminkin ennenkuin tuo aika täyttyy, jos sillä on datagrammeissa tilaa, jota käyttää. (Phaal & Lavine 2004, 23-24.)

3.2 sFlow ja OpenFlow

OpenFlow on protokolla, joka sallii erillisen kontrollerin hallita välityspäätöksiä tietoverkon kytkimillä. sFlow taas tarjoaa täydentävää toiminnallisuutta. Kytkimet lähettävät sFlow datagrammeja, jotka sisältävät pakettien näytteitä, välitysinformaatiota sekä liitäntöjen laskureita niitä analysoivalle ohjelmistolle. (Phaal & Lavine 2014.)

Kuviossa 2 näkyy kytkin 1, joka sisältää portit 1 – N. sFlow-analysaattori tunnistaa kytkimen, joka lähettää sFlow-datagrammeja analysoitavaksi käyttäen uniikkia ”agent_address”-osoitetta. Myös kaikki kytkimen portit tunnistetaan SNMP:n ifIndex-tiedolla. Kytkin myös ylläpitää OpenFlow-yhteyttä OpenFlow’n k ontrolleriin. Kytkin kuvailee resurssinsa kontrollerille tunnistamalla yhden tai useamman datapolun, joista jokainen sisältää yhden tai useamman portin, jotka tunnistetaan portin numerolla. (Phaal & Lavine 2014.)



Kuvio 2. Kuinka sFlow ja OpenFlow toimivat yhdessä

Liikenteen ohjauksen ohjelmisto käyttää liikenteen analytiikkaa, joita sFlow-analysaattori sille lähettää. Tämän jälkeen sFlow-analysaattori ohjeistaa OpenFlow-kontrolleria tekemään muutoksia. Kontrolleri lähettää muutokset tämän jälkeen kytkimille, jotta liikennevirrat verkossa muuttuvat. sFlow'n reaaliaikainen monitorointi sekä OpenFlow'n reaaliaikainen liikenteen hallinta mahdollistavat verkon automaattisten mukautumisen muuttuviin liikennevirtoihin. (Phaal & Lavine 2014.)

3.3 NetFlow

NetFlow on alunperin Ciscon kehittämä verkonmonitorointi teknologia Cisco IOS-ohjelmistoihin. NetFlow on työkalumjonka avulla pääkäyttäjät pystyvät ymmärtämään kuka, milloin, mitä ja miten liikenne tietoverkoissa virtaa. Nykyisellään on tietoverkkojen operoijille on tärkeää ymmärtää kuinka verkot käyttäytyvät kuten:

- Verkon tuottavuus ja käyttöaste
- Verkon muutosten vaikutus

- Verkon säännöttömyys ja tietoturva haavoittuvuudet (Introduction to Cisco IOS NetFlow - A Technical Overview 2012.)

NetFlow kerää informaatiota verkosta verkosta erinäisten attribuuttien kautta, joista esimerkkejä ovat lähde IP-osoite ja portti, kohde IP-osoite ja portti, layer 3 (verkkokerros) -protokollan tyyppi, CoS (Class of Service) sekä reitittimen tai kytkimen liitäntä. Jokainen paketti, joka verkossa välitetään tarkastetaan löytyykö siitä näitä attribuutteja. Näillä attribuuteilla myös määritetään onko paketti ainutlaatuinen vai onko se samankaltainen kuin muut paketit, joita verkossa kulkee. Kaikki paketit joilla on sama lähde/kohde IP-osoite tai portti, protokollan tyyppi tai CoS lisätään yhteen vuohon, jonka jälkeen paketit ja tavut sidotaan toisiinsa. (Introduction to Cisco IOS NetFlow - A Technical Overview 2012.)

Tällä informaatiolla, jota voista saadaan pystytään ymmärtämään paremmin verkon käyttäytymistä. Kuten seuraavat esimerkit todistavat:

- Lähdeosoite ja kohdeosoite kertovat kuka lähettää ja kuka vastaanottaa liikennettä
- Portit kertovat minkä sovellus on kyseessä
- CoS tutkii liikenteen prioriteettejä
- Laitteen liitäntä kertoo kuinka korkea laitteen käyttö on
- Toisiinsa sidotut paketit ja tavut kertovat liikenteen määrästä (Introduction to Cisco IOS NetFlow - A Technical Overview 2012.)

NetFlow:n luomaan dataan päästään käsiksi joko CLI:n (Command Line Interface) tai asentamalla jonkin sovelluksen, joka on tarkoitettu datan raportointiin. Jos NetFlow:n dataa halutaan viedä johonkin ulkoiseen sovellukseen on tällöin muutama tärkeä vaihe, joiden avulla NetFlow data raportointi saadaan implementoitua:

- NetFlow konfiguroidaan kaappaamaan vuot NetFlow välimuistiin (NetFlow Cache)
- NetFlow ohjataan lähettämään vuot NetFlow kerääjille (NetFlow Collector)
- Kätköstä etsitään voita, jotka ovat päättyneet ja ne viedään NetFlow kerääjille
- Tyypillisesti noin 30-50 vuota niputetaan yhteen ja lähetetään UDP-formaatissa NetFlow kerääjälle
- NetFlow kerääjän ohjelmisto luo reaaliaikaista tai historiallista raportointia tästä datasta (Introduction to Cisco IOS NetFlow - A Technical Overview 2012.)

Ajastimilla määritetään onko vuo toimeton tai onko se ollut aktiivinen liian pitkään. Oletusarvo vuon toimettomuudelle on 15 sekunttia ja aktiiviaika 30 minuuttia, nämä ajat on myös mahdollista konfiguroida haluamikseen. Vuo on valmis vietäväksi NetFlow kerääjälle, kun se on ollut toimeton tietyn aikaa (esim. jos uusia paketteja ei ole tullut) tai jos vuo on ollut aktiivinen pidempään kuin ajastimeen määritetty aika. Vuo voidaan myös viedä kerääjälle jos TCP lippu (TCP Flag) määrittää, että vuo on päättynyt. (Introduction to Cisco IOS NetFlow - A Technical Overview 2012.)

Tyypillisesti NetFlow implementoidaan päätoimistolla, koska lähes kaikki liikenne etäpaikoilta tunnetaan ja ne ovat NetFlow:n käytettävissä. Paikka johon NetFlow implementoidaan voi olla riippuvainen raportoinnin tavasta ja verkon topologiasta. NetFlow:ta voidaan myös käyttää etäpaikoilla, mutta tällöin se käyttää noin 1-5% kaistasta toimipisteiden välillä. (Introduction to Cisco IOS NetFlow - A Technical Overview 2012.)

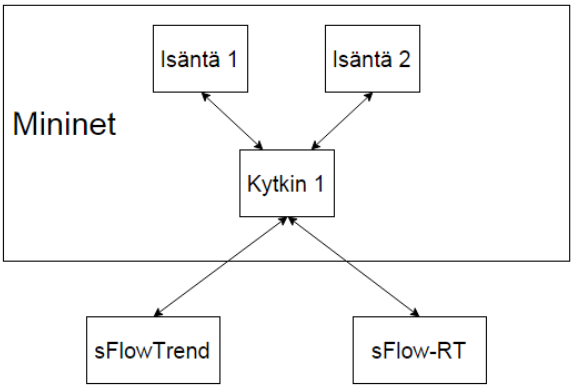
4 TOTEUTUS

4.1 Suunnittelu

Toteutuksessa mainitut palvelinten nimet johtuvat niille asennettujen SDN-kontrollereiden nimistä. sFlow määritettiin käytettäväksi tekniikaksi työn edetessä, joten suunnittelu tätä työtä varten pohjautuu monitoroitaviin liitäntöihin. Ympäristön kytkimessä s1 on kaksi porttia, jotka ovat yhteydessä verkon isäntiin h1 ja h2. Molemmat näistä porteista halutaan saada monitorointiin, jotta verkosta saadaan dataa, joka kertoo linkkien tilanteesta. Normaaliin monitorointiin käytetään sFlowTrend-nimistä ohjelmaa, joka asennetaan Floodlight-palvelimelle. Kun ohjelmisto on asennettu sitä pystytään sitä käyttämään Windows 7 -työasemalta selaimella osoitteesta 192.168.2.104:8087.

Toinen osa käytännön toteutusta on DDoS-mitigaation (Distributed Denial of Service) implementointi, jonka tarkoitukesena on estää DDoS-hyökkäys, kun sellainen havaitaan. Tähän käytetään skriptiä, joka löytyy liitteestä 1. DDoS-hyökkäyksen havaitsemiseen ja estämiseen käytetään sFlow-RT-nimistä ohjelmaa, joka asennetaan Opendaylight-palvelimelle. Ohjelmat asennetaan eri palvelimille sen takia, että yhdellä palvelimella ei voi olla käytössä kahta sFlow-ohjelmistoa, koska ne käyttävät samaa porttia, jolloin toista ohjelmista ei voida käynnistää. Kun sFlow-RT on asennettu voidaan ohjelmistoa käyttää Windows 7 -työasemelta selaimella osoitteessa 192.168.2.102:8008.

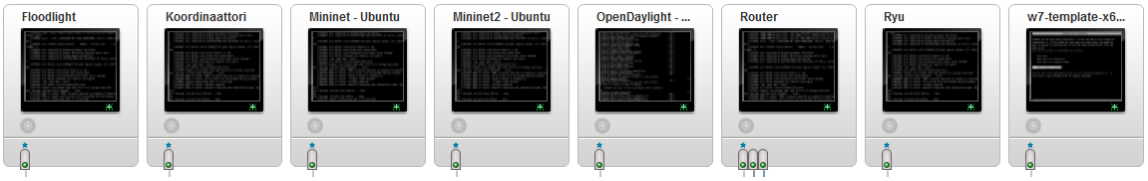
Kuviossa 3 on esitelty suunniteltu verkon topologia ja kuinka monitorointi-palvelimet kytkeytyvät kytkimeen 1, johon monitoroinnit rajapinnoille asetetaan.



Kuvio 3. Ympäristön monitorointi

4.2 Ympäristö

Käytännöntoteutuksen osuus työstä tehtiin VMwaren virtuaalisessa ympäristössä. Kyseisessä ympäristössä oli useampi virtualisoitu palvelin, joilla SDN-verkkoa pystyttiin emuloimaan. Käytettävissä oli useita eri kontrolleri vaihtoehtoja, kuten Floodlight, OpenDaylight ja Ryu. Työn toteuttamisessa käytettiin Floodlight-kontrolleria, Mininet– Ubuntu palvelinta sekä Windows7 tietokonetta. Käytettävien virtuaalikoneiden IP-osoitteet ovat Floodlight (192.168.2.104), Mininet – Ubuntu (192.168.2.101), Opendaylight – Ubuntu (192.168.2.102) sekä w7-template-x64 (192.168.2.108). Kuviossa 4 on ympäristön kaikki virtuaalikoneet VMware-ympäristössä.



Kuvio 4. VMware-ympäristö

Floodlight-kontrolleri voidaan käynnistää Floodlight-palvelimelta. Sen käynnistys tapahtuu kuvion 5 esittämällä tavalla.

```
floodlight@Floodlight:~$ cd ~/floodlight
floodlight@Floodlight:~/floodlight$ java -jar target/floodlight.jar_
```

Kuvio 5. Floodlight-kontrollerin käynnistys

Kun kontrolleri on käynnistetty, alkaa se kuuntelemaan löytyykö verkosta kytkimiä, joita se voisi käyttää. Kun kontrolleri on käynnistetty voidaan seuraavaksi käynnistää mininet Windows 7 koneelta, ensin avaamalla SSH-yhteys Mininet1-palvelimeen. Mininetin voi myös käynnistää suoraan mininet-palvelimelta, mutta sen puolelta ei pystytä avaamaan xterm-istuntoja kytkimiin tai isäntiin. Kun yhteys mininettiin on avattu, voidaan siellä käynnistää yhden kytkimen ja kahden isännän verkko käyttämällä komentoa:

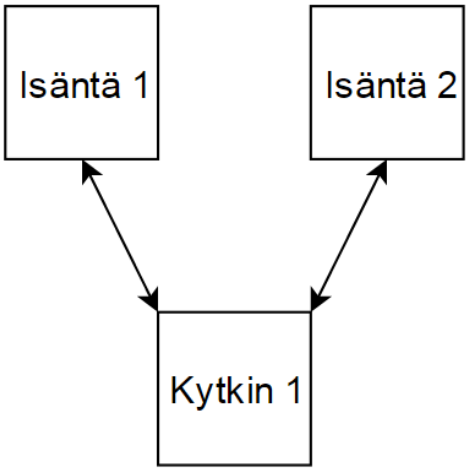
sudo mn --switch=ovs --controller=remote,ip=192.168.2.104,port=6653 --topo=tree,1

Kun komento on syötetty avautuu mininetin CLI, johon voidaan antaa lisää komentoja, joilla mininettiä ohjataan. Mininetin käynnistyksen yhteydessä lisätään automaattisesti linkit isäntien ja kytkimen välille, jonka mininet näyttää käynnistyksessä kuvion 6 esittämällä tavalla.

```
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(s1, h1) (s1, h2)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Kuvio 6. Mininetin käynnistys

Kuviossa 7 on esitelty mininetin topologia, joka edellä mainitulla komennolla käynnistettiin.



Kuvio 7. Mininetin topologia

Mininet ja kontrolleri ovat nyt käynnissä. Tässä vaiheessa on hyvä varmistaa, että reititys verkossa toimii. Tämä voidaan tehdä esimerkiksi käyttämällä ”pingall”-komentoa, jolloin molemmat isännät pingaavat toisiaan. Kuten kuviosta 8 nähdään molempien isäntien lähettämään pakettiin vastattiin, joten ympäristö toimii.

```
mininet> pingall
*** Ping: testing ping reachability
h1 -> h2
h2 -> h1
*** Results: 0% dropped (2/2 received)
```

Kuvio 8. Mininetin ping testi

4.3 sFlowTrend-monitorointiohjelman asentaminen

sFlowTrend-ohjelmiston pystyttäminen on hyvin yksinkertaista. Debian-pohjaiselle käyttöjärjestelmälle on jo valmiiksi käännetty asennuspaketti, joka täytyy ladata ja asentaa palvelimelle. Vaatimuksena sFlowTrendin toimimiselle on Java, koska ohjelmisto on kirjoitettu Javalla. Javan version täytyy myös olla 1.7 tai uudempi. Ohjelmisto asennettiin Floodlight-palvelimelle ja sen osoite tässä ympäristössä on 192.168.2.104. sFlowTrend saadaan ladattua ja asennettua alla näkyvällä komennolla.

wget
http://www.inmon.com/products/sFlowTrend/downloads/sFlowTrend-linux-6_2.deb
sudo dpkg -i sFlowTrend-linux-6_2.deb

Kun ohjelmisto on asennettu, sen toimintaa voi tarkkailla osoitteessa 192.168.2.104:8087. Tässä vaiheessa palvelin ei vielä vastaanota mitään näytteitä, josta johtuen myöskään dataa ei ole luettavissa. Seuraava vaihe on asettaa agentit, jotka lähettävät dataa sFlowTrend-ohjelmistolle.

4.4 Agenttien asettaminen

Agentit on helpoint konfiguroida käyttämällä xterm-ikkunaa, jolla pystytään ottamaan suora hallintayhteys mininetistä löytyvään kytkimeen. Hallintayhteys kytkimeen saadaan avattua kirjoittamalla seuraava komento mininetin CLI:ssä.

xterm s1

Rajapinnat pystyy selvittämään kytkimen rajapintojen listauksesta, koska kyseessä on linux-pohjainen käyttöliittymä on tämä komento ifconfig. Kuviossa 9 esitellään ympäristön kytkimen liitännät, joista s1-eth1 menee isännälle h1 ja s1-eth2 isännälle h2.

```
s1-eth1   Link encap:Ethernet  HWaddr 9a:05:a1:3e:82:82
          inet6 addr: fe80::9805:a1ff:fe3e:8282/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:29 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:648 (648.0 B)  TX bytes:2048 (2.0 KB)

s1-eth2   Link encap:Ethernet  HWaddr 2a:a6:a4:7b:5a:cc
          inet6 addr: fe80::28a6:a4ff:fe7b:5acc/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8 errors:0 dropped:0 overruns:0 frame:0
          TX packets:31 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:648 (648.0 B)  TX bytes:2208 (2.2 KB)
```

Kuvio 9. Kytkimen 1 rajapinnat isäntiin

Kun liitäntöjen nimet, joita halutaan monitoroida on tunnistettu voidaan sFlow-monitorointi asettaa niihin alla näkyvällä komennolla. Alla näkyvällä komennolla asetetaan sFlow'n agentti kytkimen rajapintaan s1-eth1. Komennon aluksi agentiksi määritetään s1-eth1, joka on kytkimen s1 liitäntä isäntään h1. Target -parametri määrittää palvelimen IP-osoitteen 192.168.2.104, joka on Floodlight-kontrollerin IP-osoite sekä portiksi 6343, joka on oletusportti, jota sFlow:ssa käytetään. Sampling=10 -parametri tarkoittaa, että näytteitä paketeista otetaan suhteella 1:10, joka tarkoittaa yhtä näytettä 10

pakettia kohden. Polling -parametri taas määrittää, että laskureita pollataan kerran 20 sekunnissa.

ovs-vsctl -- --id=@sflow create sflow agent=s1-eth1 target=\"192.168.2.104:6343\" sampling=10 polling=20 -- -- set bridge s1 sflow=@sflow

Kun rajapintaan s1-eth1 on asetettu valvonta, voidaan asettaa valvonta myös rajapintaan s1-eth2, joka on kytkimen s1 rajapinta isäntään h2.

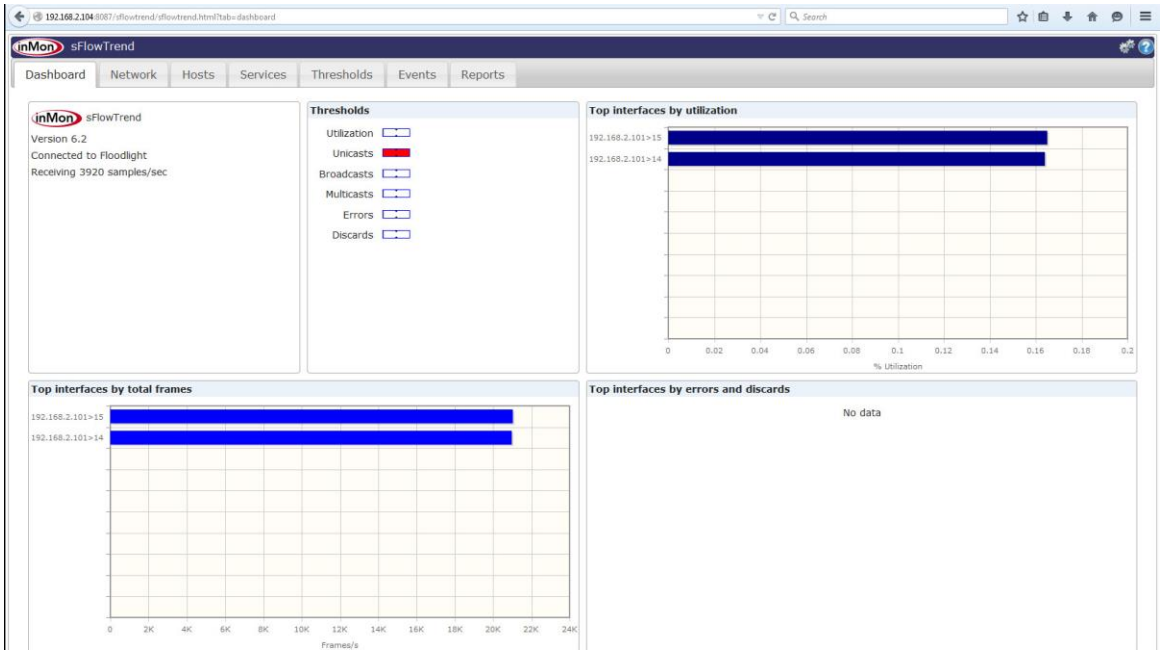
ovs-vsctl -- --id=@sflow create sflow agent=s1-eth2 target=\"192.168.2.104:6343\" sampling=10 polling=20 -- -- set bridge s1 sflow=@sflow

Nyt kun monitorointi kytkimen s1 liitäntöihin isäntiin h1 ja h2 on asetettu, voidaan testate sen toimivuus. Käynnistetään mininettiin avatusta SSH-yhteydestä ping flood isännältä h1 isäntään h2, komento on esitetty kuviossa 10. Käyttämällä tätä komentoa paketteja verkossa saadaan liikkumaan paljon, jotta monitorointiin sadaan näkyviin dataa, jota voidaan analysoida.

```
mininet> h1 ping -f h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
.
```

Kuvio 10. Ping flood h1 - h2

Kun ping flood on käynnissä, alkaa monitoroinnissa näkymään dataa. Kuviossa 11 on esitetty sFlowTrendin yleisnäkymä, josta nähdään sen vastaanottamat näytteet sekä verkon yleiskuvaa, kuten unicast liikennettä. Ping-komennon generoima liikenne on unicast-liikennettä, joten valvonnassa se näkyy Tresholds-valikossa unicast-liikenteenä. Yleisnäkymässä nähdään myös dataa erinäisistä liitännöistä verkossa. Kuviossa 11 oikeassa yläkulmassa näkyy ylimmät liitännät käytön perusteella, vasemmassa alakulmassa ylimmät liitännät kehyksien määrän perusteella ja oikeassa alakulmassa liitännät, joissa on eniten virheitä tai hylkäyksiä. Tällä hetkellä verkossa ei virheitä tai hylkäyksiä kerry, jonka johdosta niissä ei myöskään mitään dataa ole saatavilla. Sen sijaan verkossa liikkuu isäntien välisen ping-komennon ansiosta liikennettä, joka näkyy kahden portin välisenä liikenteenä.

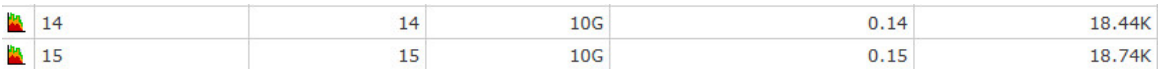


Kuvio 11. sFlowTrend yleisnäkymä

sFlowTrendin yleisnäkymän vasemmassa yläkulmassa nähdään, kuinka paljon monitorointipalvelin vastaanottaa näytteitä, kuten kuviosta 12 nähdään. Tämä johtuu monitorointiin asetetuista arvoista, joiden mukaan näytteitä lähetetään 1:10 suhteella eli yksi näyte kymmentä pakettia kohden.

Kuvio 12. sFlowTrendin vastaanotetut näytteet

sFlowTrend'stä löytyy Network -> Interfaces valikkojen takaa tieto erilaisista liitännöistä, joita on valvonnassa. Kuviossa 13 näkyy, että monitorointiin oikeasti välittyy dataa kahdesta eri liitännästä. Kuvion kentissä nähdään liitäntöjen nimi, indeksi, nopeus, utilisaatio ja unicast-pakettien määrä, joka niissä on liikkunut. Kentistä on myös mahdollista nähdä multicast-pakettien määrä sekä hylätyt ja virheelliset paketit.



	14	14	10G	0.14	18.44K
	15	15	10G	0.15	18.74K

Kuvio 13. sFlowTrendin liitäntöjen esitystapa

Events-välilehdeltä löytyy automaattisesti generoituja varoituksia, joita voidaan generoida esimerkiksi virheilyn perusteella tai tietyn tyypin liikenteen määrän ylittyessä. Kuten kuviosta 14 nähdään, on ping floodin aikana ollut verkossa paljon unicast-liikennettä, josta johtuen unicast-liikenteen rajat ovat alkaneet ylittymään ja monitorointi on generoinut automaattisen varoituksen rajojen ylittymisestä. Varoituksia on näkyvissä kolmen tyyppisiä. Warning, joka ilmaisee rajojen lähestymisestä. Information, joka kertoo jos liikenteen määrät ovat alkaneet pienentymään. Critical on viimeinen, joka syntyy kun raja-arvo on saavutettu.

Critical	Threshold	Sun, Nov 15, 2015 1:38 PM EET	Threshold Unicasts for interface 192.168.2.101>15 is becoming more critical
Critical	Threshold	Sun, Nov 15, 2015 1:38 PM EET	Threshold Unicasts for interface 192.168.2.101>14 is becoming more critical
Informational	Threshold	Sun, Nov 15, 2015 1:37 PM EET	Threshold Unicasts for interface 192.168.2.101>14 is becoming less critical
Informational	Threshold	Sun, Nov 15, 2015 1:37 PM EET	Threshold Unicasts for interface 192.168.2.101>15 is becoming less critical
Warning	Threshold	Sun, Nov 15, 2015 1:35 PM EET	Threshold Unicasts for interface 192.168.2.101>15 is becoming more critical
Warning	Threshold	Sun, Nov 15, 2015 1:35 PM EET	Threshold Unicasts for interface 192.168.2.101>14 is becoming more critical
Informational	System	Sun, Nov 15, 2015 1:10 PM EET	Client 192.168.2.107 connected to server

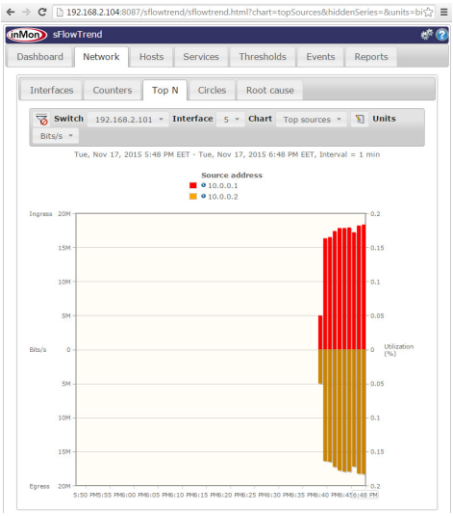
Kuvio 14. Reports-välilehti

4.5 sFlowTrend'n ominaisuuksia

sFlowTrend'llä on paljon erilaisia ominanisuuksia, joita voidaan hyödyntää verkon monitoroinnissa. Kuviossa 15 näkyy laskuri, joka kertoo liittännässä kulkevien bittien määrästä per sekuntti.

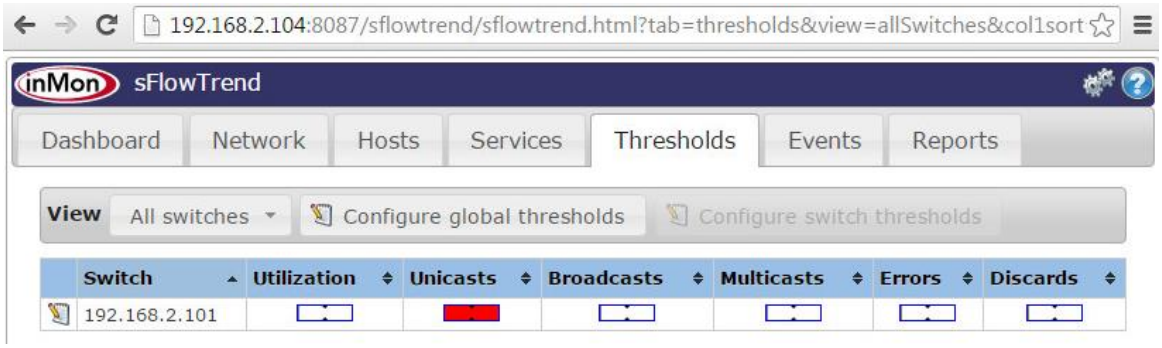
Kuvio 15. sFlowTrend liitännän sisään/ulos bittilaskuri

Kuviossa 16 näkyy, mitkä lähdeosoitteet käyttävät liitännän kapasiteettia. Palkkien alapuolella aika kulkee vasemmalta oikealle ja pystysuunnassa palkit on jaettu kahteen osaan. Puolivälistä ylöspäin on sisääntuleva liikenne 10.0.0.1 lähdeosoitteesta, joka on kytkimeltä s1 isännälle h1 menevä liitäntä ja puolivälistä alaspäin on ulosmenevä liikenne 10.0.0.2 lähdeosoitteesta, joka taas on kytkimeltä s1 isännälle h2 menevä liitäntä.



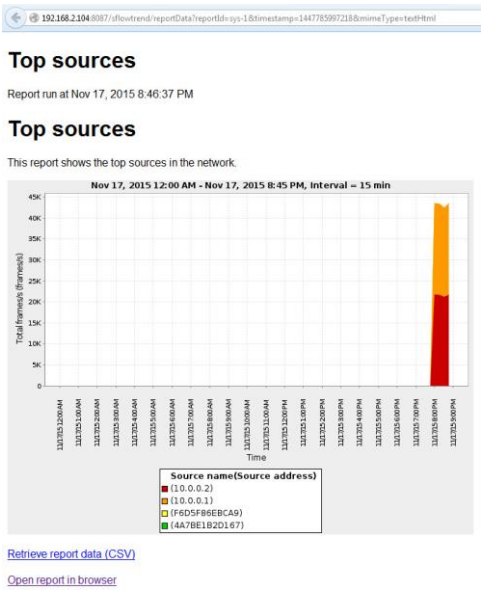
Kuvio 16. Verkon utilisoijat

sFlowTrend'stä on myös mahdollista katsoa Thresholds-välilehden alta kaikki valvottavat kytkimet. Välilehden alta pystyy näkemään normaalia statistiikkaa tietoliikenne valvontaan liittyen, kuten utilisaation, unicastin, broadcastin sekä multicastin määrät ja virheilyt sekä hylkäykset. Kaikki nämä raja-arvot ovat konfiguroitavissa sFlowTrend'n "Configure global thresholds"-painikkeen alta. Thresholds-välilehti on esitetty kuviossa 17.



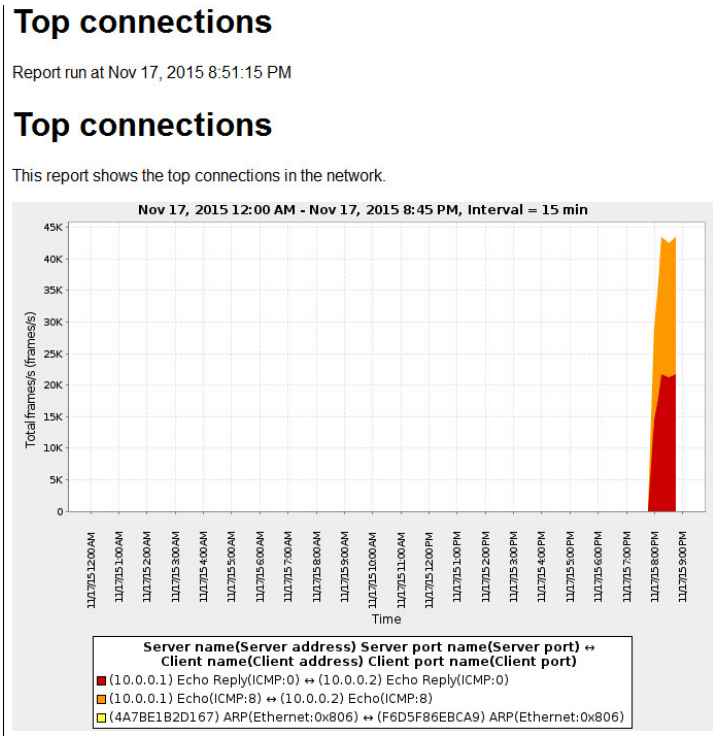
Kuvio 17. Thresholds-välilehti

sFlowTrend'llä on myös mahdollista ajaa erilaisia raportteja verkosta. Kuviossa 18 on annettu esimerkki raportista, jossa näkyy lähdeosoitteet, jotka käyttävät verkkoa



Kuvio 18. Suurimmat verkon käyttäjät –raportti

Kuvion 19 raportista näkyy hyvin, kuinka verkon monitorointia on testattu. Testaaminen on käytännössä ollut ping-komennon käyttämistä, josta johtuen verkko on täynnä ICMP-viestejä ”Echo request”- ja ”Echo reply”. Palkin punainen osio sisältää ”Echo reply”-viestit ja keltainen osa ”Echo request”-viestit.



Kuvio 19. Top Connections-raportti

4.6 DDoS-mitigaatio

sFlow-RT asennettiin Opendaylight-palvelimelle, koska Floodlight palvelimelle sitä ei voitu asentaa johtuen toisesta monitorointi ohjelmistosta. Useampi ohjelma ei pysty samanaikaisesti käyttämään sFlow'lle määritettyä porttia 6343. sFlow-RT'n asentamista varten täytyy ensin ladata ohjelmiston tiedostot palvelimelle. Tämän jälkeen ne täytyy purkaa ja käynnistää sFlow-RT-palvelin sen omasta kansiosta "./start.sh"-komennolla.

```
wget http://www.inmon.com/products/sFlow-RT/sflow-rt.tar.gz
tar -xvzf sflow-rt.tar.gz
cd sflow-rt
./start.sh
```

DDoS-mitigaatio skripti, joka löytyy liitteestä 1 on kirjoitettu node.js ohjelmointikielellä. Tästä johtuen palvelimelle täytyy asentaa nodejs-ohjelmisto komennolla:

```
sudo apt-get install nodejs
```

Kun ohjelmisto on asennettu aloitetaan ympäristön pystyttäminen. Floodlight-palvelimelle syötetään komennot, jolla kontrolleri käynnistetään:

```
cd floodlight
./floodlight.sh
```

Mininet ympäristö voidaan käynnistää mininetin CLI:llä komennolla:

```
sudo mn --switch=ovs --controller=remote,ip=192.168.2.104,port=6653 --topo=tree,1
```

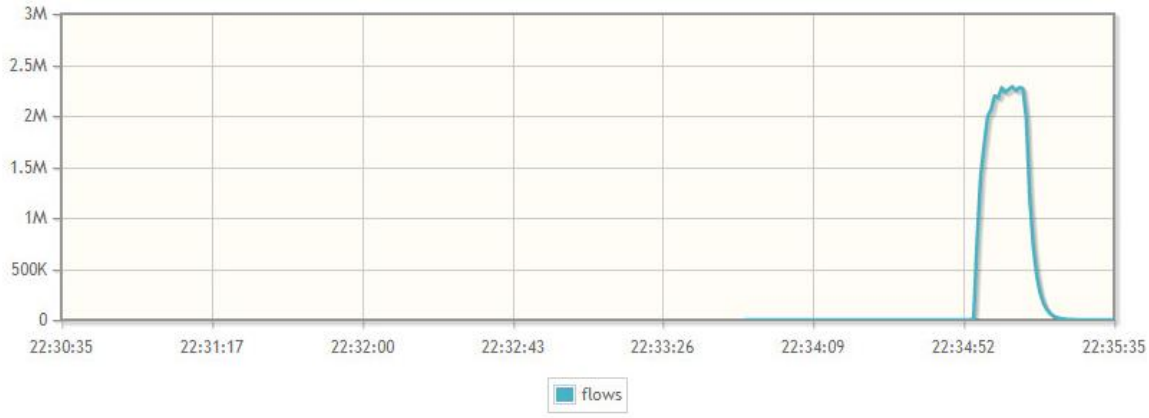
Seuraavaksi asetetaan valvonta kytkimen s1 rajapintoihin s1-eth1 ja s1-eth2 kommennoilla:

```
ovs-vsctl -- --id=@sflow create sflow agent=s1-eth1 target=\"192.168.2.102:6343\" sampling=10 polling=20 -- -- set bridge s1 sflow=@sflow
ovs-vsctl -- --id=@sflow create sflow agent=s1-eth1 target=\"192.168.2.102:6343\" sampling=10 polling=20 -- -- set bridge s1 sflow=@sflow
```

Komennoilla viitataan eri palvelimeen kuin viimeksi. Tällä kertaa kyseessä on Opendaylight-palvelin osoitteessa 192.168.2.102, joka käyttää myös samaa sFlow-porttia. Seuraavaksi Opendaylight-palvelimella käynnistetään sFlow-RT komennoilla:

cd sflow-rt
./start.sh

Kun on ympäristö on pystytetty voidaan verkon toiminta testata käyttämällä ping-komentoa. Kuten kuviosta 20 näkyy verkossa liikkuu dataa ja se näkyy valvonnassa. DDoS-mitigaatio skriptin pitäisi vaikuttaa tähän käyrään siten, että se ei enää nouse kohtuuttomasti vaan liikenteen määrä rajoittuum, jonka seurauksena palvelussa ei tapahdu estoja.



Metric	Top Key	Agent	Value
flows	10.0.0.1,10.0.0.2,eth.ip.icmp	192.168.2.101	9.29

Kuvio 20. sFlow-RT-monitorointi

Seuraavaksi liitteessä 1 näkyvä skripti on lisättävä Floodlight-palvelimelle. Kun skripti on lisätty tiedostoon mininet.js voidaan se ajaa käyttämällä komentoa:

nodejs mininet.js

Kuten kuviosta 21 nähdään se ei kuitenkaan onnistu. Tämä johtuu sFlow-RT'lle tehdyistä päivityksistä sekä Floodlight-kontrollerin kehityksestä, jonka johdosta skriptin syntaksi on vanhentunut ja sitä ei enää pystytä ajamaan päivitetyillä versioilla ohjelmistoista.

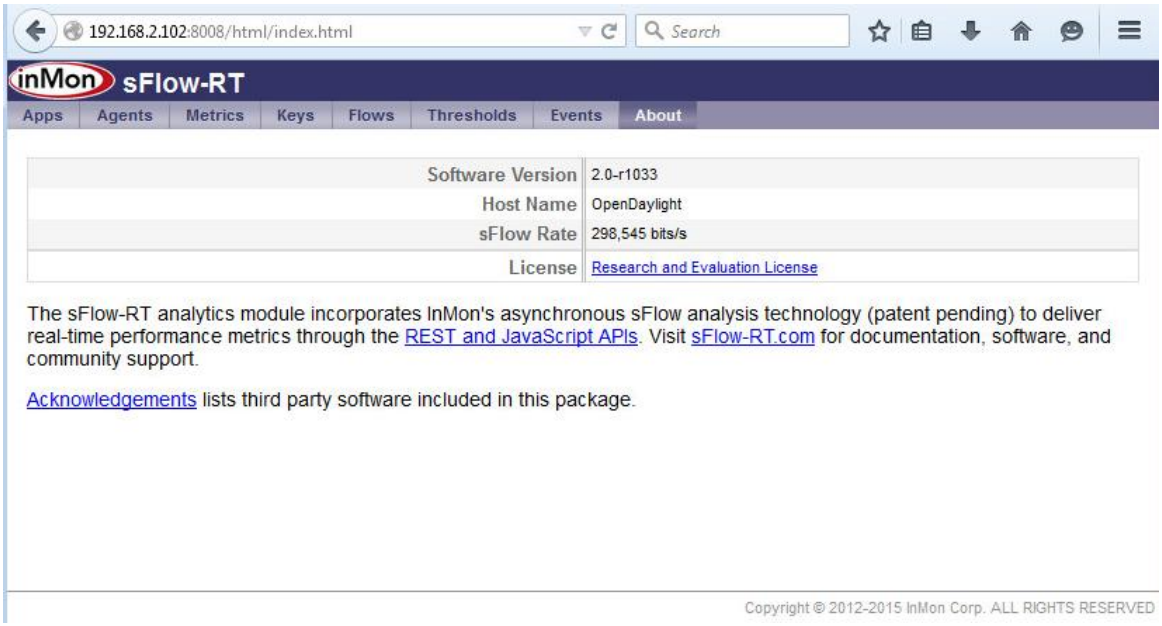
```
floodlight@Floodlight:~$ nodejs mininet.js

/home/floodlight/mininet.js:140
        for(var j = 0; j < ports.length; j++) {
                           ^

TypeError: Cannot read property 'length' of undefined
    at /home/floodlight/mininet.js:140:33
    at IncomingMessage.<anonymous> (/home/floodlight/mininet.js:47:33)
    at IncomingMessage.EventEmitter.emit (events.js:117:20)
    at _stream_readable.js:920:16
    at process._tickCallback (node.js:415:13)
```

Kuvio 21. Mininet.js skriptin käynnistys yritys

Liitteestä 1 löytyvä skripti on poimittu blog.sflow.org verkkosivustolta. Ohjeistusta DDoS-mitigaation toteuttamiseen on myös tutkittu samalta sivulta. Kuten sivulta näkyvistä kommenteista käy ilmi, on DDoS-mitigaatioskriptin syntaksi vanhentunut, jonka johdosta se ei enää ole käyttökelpoinen. Muuttujia yritettiin muuttaa koodista manuaalisesti kiinteisiin, jotta virheet pystyttäisiin ohittamaan, mutta se johti aina uusiin ongelmiin, jonka jälkeen päätettiin jättää DDoS-mitigaatioskripti implementoimatta. (Controlling large flows with OpenFlow 2013.)

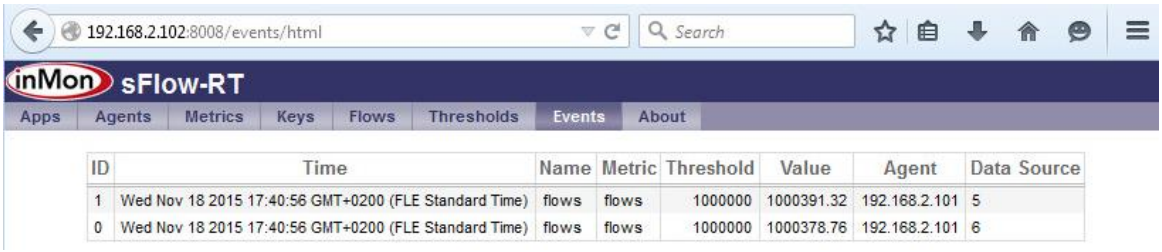
4.7 Verkon näkymän laajeneminen sFlow-RT'n myötä

Jos verrataan sFlow-RT-ohjelmistoa sekä sFlowTrend'iä voidaan havaita, että ne antavat erilaista dataa pääkäyttäjän käytettäväksi. Kokonaisuuten sFlow-RT vaikuttaisi hieman reaktiivisemmalta ohjelmistolta verkon monitorointiin, kuin sFlowTrend. Kuviossa 22 on sFlow-RT'n etusivu, joka aukeaa automaattisesti kun ohjelmistoon otetaan yhteys. Kuten sFlowTrend'ssäkin nähdään myös sFlow-RT'n etusivulla samankaltaista tieto. Esimerkiksi vastaanotettujen näytteiden määrä sekuntia kohden. Lisäksi nähtävillä on ohjelmistoversio sekä palvelimen nimi, jolle ohjelmisto on asennettu.



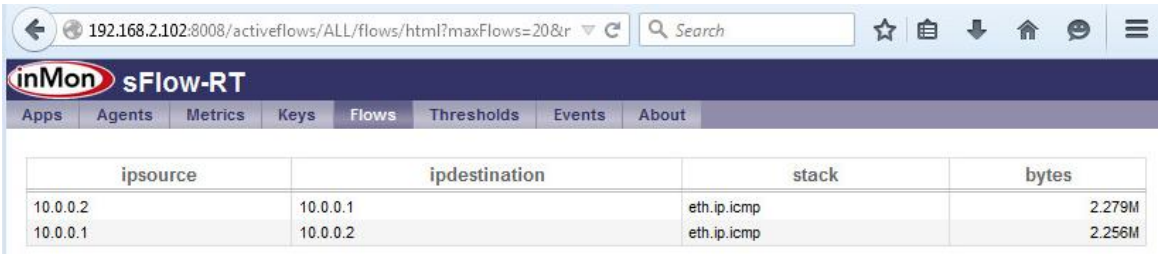
Kuvio 22. sFlow-RT'n etusivu

sFlow-RT'n Events-välilehdeltä nähdään aktiivisit tietoliikennevuot kuvion 23 osoittemalla tavalla.



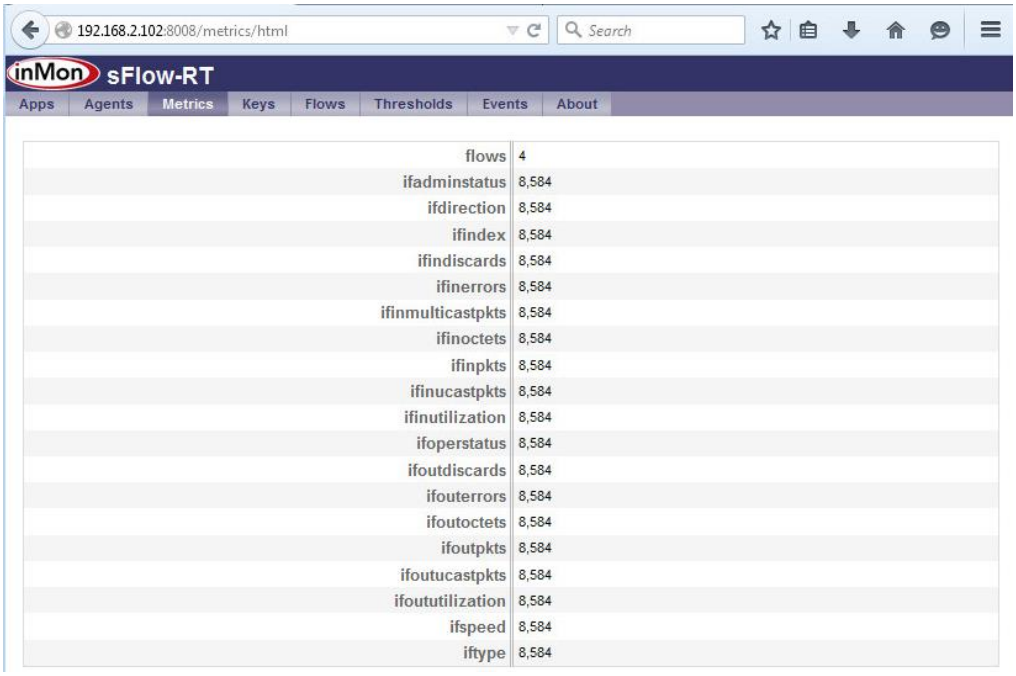
Kuvio 23. sFlow-RT'n aktiiviset vuot

Flows-välilehdeltä on mahdollista katsoa kunkin vuon sen hetkisen liikenteen määrä. Kuviossa 24 näkyy, että tällä hetkellä kulkee ping-komento molempien hostien välillä ja voiden suuruus on noin 2,2 Mb/s.



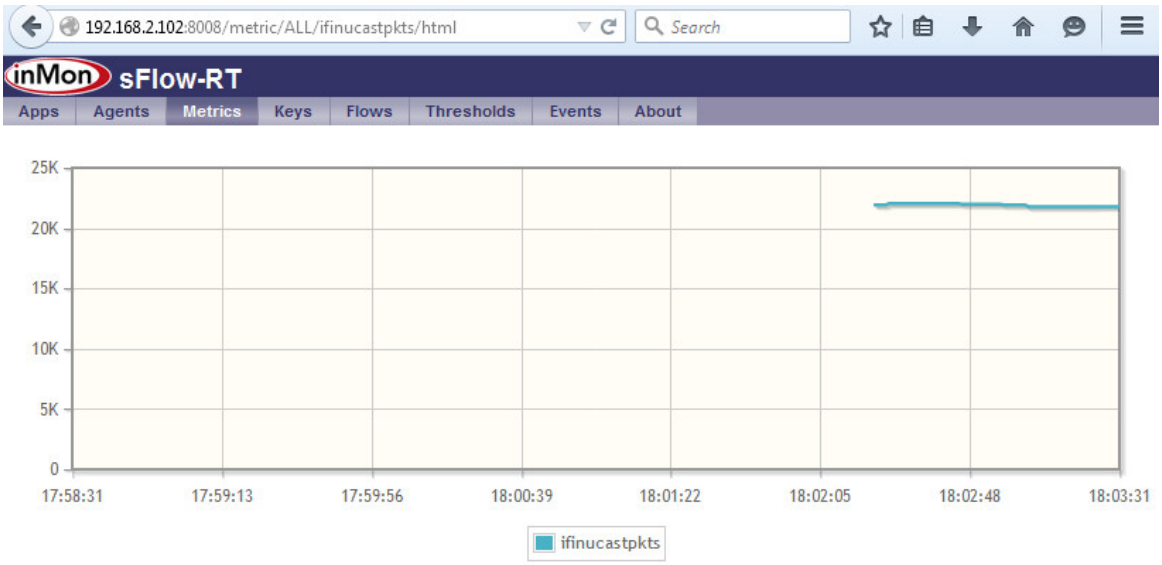
Kuvio 24. Flows-välilehti

Metrics-välilehdeltä nähdään erilaista tietoa verkosta. Eri osioita klikkaamalla pystytään siirtymään sivulle, josta nähdään yksityiskohtaisempaa tietoa, kuten unicast-pakettien määriä tai ulko- tai sisäsuunnassa olevaa virheilyä. Kuviossa 25 on esitetty Metrics-välilehti.



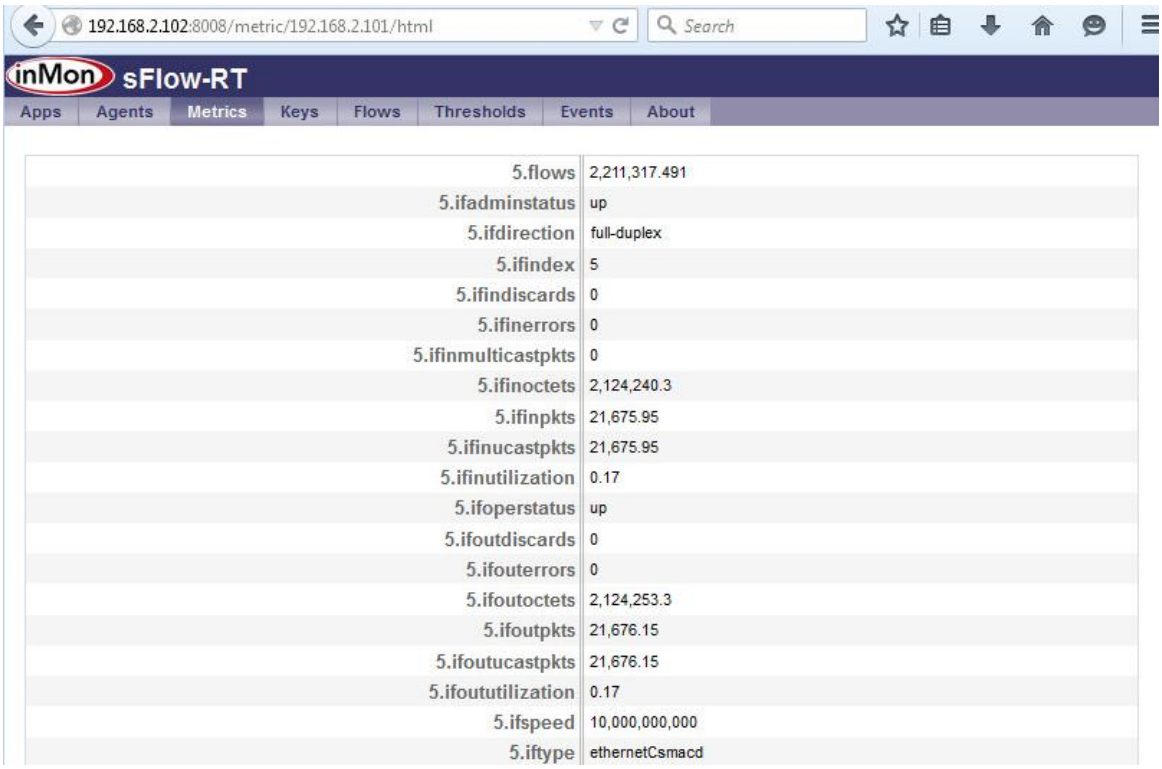
Kuvio 25. Flows-välilehti

Kuviossa 26 näkyy esimerkkinä sisään tulevien unicast-pakettien monitorointi.



Kuvio 26. Unicast-pakettien monitorointi

Kuviossa 27 näkyy kytkimen liitäntöjen statistiikkaa. Statistiikkaa päästään katsomaan Agents-välilehden kautta, jossa näkyy yksi agentti IP-osoitteella, joka on 192.168.2.101. Kuviossa näkyy esimerkkinä liitännän 5 statistiikka.



Kuvio 27. Kytkimen liitäntöjen statistiika

5 YHTEENVETO

5.1 Pohdinta

Opinnätetyö sai alkunsa tarpeesta monitoroida SDN-ympäristöä, joka Cyber Trust -hankkeella oli jo valmiiksi pystytettynä. Ympäristö koostu VMware-alustalle asennetuista palvelimista ja isännästä, joiden ympärille SDN-verkko voitiin pystyttää.

Teoriaosuudessa kirjoitettiin SDN-teknologiasta, OpenFlow-protokollosta ja sFlow- sekä Netflow-tekniikoista. Teorian työstämisen aikana havaittiin, että oli jo valmiita malleja olemassa, joilla pystyttäisiin toteuttamaan SDN-kontrollerin ja sFlow-analysaattorin välistä keskustelua ja reaaliaikaisesti vaikuttamaan verkon liikenteeseen.

Käytännön osuuden dokumentaatio koostuu sFlowTrend-palvelimen pystytyksestä sekä DDoS-mitigaation implementoimisesta SDN-verkkoon. Työn käytännön osuuden aikana havaittiin, että työhön löydetty metodi implementoida DDoS-mitigaatio verkkoon ei ollut enää ajantasalla. SDN-tekniikan nopeasta kehityksestä johtuen mitigaatio skriptin syntaksi oli vanhentunut, ja näin ollen sitä ei enää pystytty käyttämään toteutuksessa.

sFlowTrend'iä käytettäessä tehtiin havainto, jonka perusteella paras selainohjelmisto sovelluksen käyttämiseen olisi Mozilla Firefox, kun verrataan Googlen Chromeen tai Internet Exploreriin, jotka Windows 7 –työasemalta löytyivät. Chrome ei näyttänyt kaavioita oikein, eikä Internet Explorerilla saatu valvontaa avattua lainkaan.

Lopputuloksena monitorointiohjelmista voidaan sanoa, että vaikka sFlow-RT onkin reaktiivisempi ohjelmisto monitorointiin on sFlowTrend silti parempi sen tarjoaman raportoinnin vuoksi. sFlow-RT-ohjelmistossa ei ollut minkäänlaista mahdollisuutta ajaa raportteja verkosta.

Opinnäytetyö onnnistui ilman suurempia haasteita sekä toteutuksen, että teorian kirjoittamisen osalta. Lähteitä löytyi hyvin, joskaan ei kovin monipuolisesti. Monipuolisuuden puute lähteissä ei ole haitaksi sillä lähteet, joita käytettiin olivat teknologioiden kehittäjien omilta kotisivuilta, joten luotettavuus niissä pitäis olla kohdallaan.

5.2 Kehitysideat

Koska sFlow'n ja OpenFlow'n on mahdollista keskustella toistensa kanssa ja vaikuttaa reaaliaikaisesti tietoverkon liikenteeseen on loogista, että tämä opinnäytetyö oli hyvä aihio alkaa rakentamaan ympäristöä, joka reagoi automaattisesti ja reaaliaikaisesti verkossa tapahtumiin muutoksiin, kuten kapasiteetin loppumiseen tietystä linkistä, virheilyyn tai katkeamiseen.

LÄHTEET

Controlling large flows with OpenFlow. 2013. Artikkeli sFlow'n blogisivustolla. Julkaistu 27.5.2013 Viitattu 28.11.2015. http://blog.sflow.com/2013/05/controlling-large-flows-with-openflow.html

Introduction to Cisco IOS NetFlow - A Technical Overview. 2012. Artikkeli laitevalmistaja Ciscon verkkosivustot. 1.5.2012. Viitattu 9.11.2015 http://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html.

Karhinen, H. 2015. Avoimen lähdekoodin SDN-kontrollerien evaluointi. Opinnäytetyö. Jyväskylän ammattikorkeakoulu, Tekniikan ja liikenteen ala, tietotekniikan koulutusohjelma. Viitattu 26.10.2015. https://www.theseus.fi/bitstream/handle/10024/89561/Karhinen_Hiski.pdf?sequence.

ONF Member listing. N.d. Open networking järjestön verkkosivusto. Viitattu 14.10.2015. https://www.opennetworking.org/our-members.

ONF Membership. N.d. Open networking järjestön verkkosivusto. Viitattu. 14.10.2015. https://www.opennetworking.org/about/membership-inquiry-en.

ONF Overview. N.d. Open networking järjeston verkkosivusto. Viitattu 14.10.2015. https://www.opennetworking.org/about/onf-overview.

O'Connor, D. 2011. RIB, FIB, LFIB, LIB etc. Blogikirjoitus Darren's Blog sivustolla. 30.3.2011. Viitattu 26.10.2015. https://mellowd.co.uk/ccie/?p=788.

Phaal, P. & Lavine, M. 2004. sFlow version 5. Dokumentti sFlow.org verkkosivustolla. Julkaistu 1.6.2004. Viitatu 26.10.2015. http://www.sflow.org/sflow_version_5.txt.

Phaal, P. & Lavine, M. 2014. sFlow OpenFlow structures. Dokumentti sFlow.org verkkosivustolla. Julkaistu 1.11.2014. Viitattu 4.11.2015. http://www.sflow.org/sflow_openflow.txt.

Salisbury, B. 2012. The Control Plane, Data Plane and Forwarding Plane in Networks. NetworkStatic verkkosivusto. Julkaistu 27.9.2012. Viitattu 25.10.2015. http://networkstatic.net/the-control-plane-data-plane-and-forwarding-plane-in-networks/

Savola, R. 2014. DIGILE Cyber Trust and Related Security Projects. PDF-tiedosto Teknologian tutkimuskeskus VTT:n verkkosivustoilla. Julkaistu 18.11.2014. Viitattu 19.11.2015. http://www.vtt.fi/files/sites/eemeli18/12_Reijo_Savola_security_calls.pdf

SDN architecture. 2014. Dokumentti Open networking foundation verkkosivustolla. Julkaistu 5.6.2014. Viitattu 26.10.2015.

https://www.opennetworking.org/images/stories/downloads/sdn-resources/technical-reports/TR_SDN_ARCH_1.0_06062014.pdf.

Software-Defined Networking: The New Norm for Networks. Julkaistu 13.4.2012. Viitattu 14.10.2015. https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf.

What’s Software-Defined Networking (SDN)?. N.d. SDxCentral verkkosivusto. Viitattu 25.10.2015. https://www.sdxcentral.com/resources/sdn/what-the-definition-of-software-defined-networking-sdn/.

LIITTEET

Liite 1. DDoS-mitigaatioskripti

```
var fs = require("fs");
var http = require('http');

var keys = 'inputi-
findex,ethernetprotocol,macsource,macdestination,ipprotocol,ipsource,i
pdestination';
var value = 'frames';
var filter = 'source-
group=external&destinationgroup=internal&outputifindex!=discard';
var thresholdValue = 100;
var metricName = 'ddos';
// mininet mapping between sFlow ifIndex numbers and switch/port
names
var ifindexToPort = {};
var nameToPort = {};
var path = '/sys/devices/virtual/net/';
var devs = fs.readdirSync(path);
for(var i = 0; i < devs.length; i++) {
  var dev = devs[i];
  var parts = dev.match(/(.*)-(.*)/);
  if(!parts) continue;
  var ifindex = fs.readFileSync(path + dev + '/ifindex');
  var port = {"switch":parts[1],"port":dev};
  ifindexToPort[parseInt(ifindex).toString()] = port;
  nameToPort[dev] = port;
}
var fl = { hostname: 'localhost', port: 8080 };
var groups = {'external':['0.0.0.0/0'],'internal':['10.0.0.2/32']};
var rt = { hostname: 'localhost', port: 8008 };
var flows = {'keys':keys,'value':value,'filter':filter};
var threshold = {'metric':metricName,'value':thresholdValue};

function extend(destination, source) {
  for (var property in source) {
    if (source.hasOwnProperty(property)) {
      destination[property] = source[property];
    }
  }
  return destination;
}
```

```
function jsonGet(target,path,callback) {
  var options = extend({method:'GET',path:path},target);
  var req = http.request(options,function(resp) {
    var chunks = [];
    resp.on('data', function(chunk) { chunks.push(chunk); });
    resp.on('end', function() { callback(JSON.parse(chunks.join(''))); });
  });
  req.end();
};

function jsonPut(target,path,value,callback) {
  var options = extend({method:'PUT',headers:{'content-
type':'application/json'}
,path:path},target);
  var req = http.request(options,function(resp) {
    var chunks = [];
    resp.on('data', function(chunk) { chunks.push(chunk); });
    resp.on('end', function() { callback(chunks.join('')); });
  });
  req.write(JSON.stringify(value));
  req.end();
};

function jsonPost(target,path,value,callback) {
  var options = extend({method:'POST',headers:{'content-
type':'application/json'},"path":path},target);
  var req = http.request(options,function(resp) {
    var chunks = [];
    resp.on('data', function(chunk) { chunks.push(chunk); });
    resp.on('end', function() { callback(chunks.join('')); });
  });
  req.write(JSON.stringify(value));
  req.end();
}

function lookupOpenFlowPort(agent,ifIndex) {
  return ifindexToPort[ifIndex];
}

function blockFlow(agent,dataSource,topKey) {
  var parts = topKey.split(',');
  var port = lookupOpenFlowPort(agent,parts[0]);
  if(!port || !port.dpid) return;

  var message = {"switch":port.dpid,
                 "name":"dos-1",
                 "ingress-port":port.portNumber.toString,
```

```
                "ether-type":parts[1],
                "protocol":parts[4],
                "src-ip":parts[5],
                "dst-ip":parts[6],
                "priority":"32767",
                "active":"true"};

  console.log("message=" + JSON.stringify(message));
  jsonPost(fl,'/wm/staticflowentrypusher/json',message,
     function(response) {
       console.log("result=" + JSON.stringify(response));
     });
}

function getTopFlows(event) {
  jsonGet(rt,'/metric/' + event.agent + '/' + event.dataSource + '.' +
event.metric + '/json',
    function(metrics) {
      if(metrics && metrics.length == 1) {
        var metric = metrics[0];
        if(metric.metricValue > thresholdValue
          && metric.topKeys
          && metric.topKeys.length > 0) {
          var topKey = metric.topKeys[0].key;
          blockFlow(event.agent,event.dataSource,topKey);
        }
      }
    }
  );
}

function getEvents(id) {
  jsonGet(rt,'/events/json?maxEvents=10&timeout=60&eventID='+ id,
    function(events) {
      var nextID = id;
      if(events.length > 0) {
        nextID = events[0].eventID;
        events.reverse();
        for(var i = 0; i < events.length; i++) {
          if(metricName == events[i].thresholdID) getTopFlows(events[i]);
        }
      }
      getEvents(nextID);
    }
  );
}
```

```
// use port names to link dpid and port numbers from Floodlight
function getSwitches() {
  jsonGet(fl,'/wm/core/controller/switches/json',
    function(switches) {
      for(var i = 0; i < switches.length; i++) {
        var sw = switches[i];
        var ports = sw.ports;
        for(var j = 0; j < ports.length; j++) {
          var port = nameToPort[ports[j].name];
          if(port) {
            port.dpid = sw.dpid;
            port.portNumber = ports[j].portNumber;
          }
        }
      }
      setGroup();
    }
  );
}

function setGroup() {
  jsonPut(rt,'/group/json',
    groups,
    function() { setFlows(); }
  );
}

function setFlows() {
  jsonPut(rt,'/flow/' + metricName + '/json',
    flows,
    function() { setThreshold(); }
  );
}

function setThreshold() {
  jsonPut(rt,'/threshold/' + metricName + '/json',
    threshold,
    function() { getEvents(-1); }
  );
}

function initialize() {
  getSwitches();
}

initialize();
```