Paul Nurminen

Keskitetty Segment Routing -reititys

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

28.2.2017

Metropolia

Tekijä Otsikko Sivumäärä Aika	Paul Nurminen Keskitetty Segment Routing -reititys 54 sivua + 5 liitettä 28.2.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Tietoverkot
Ohjaaja	Lehtori Erik Pätynen

Insinöörityön tarkoituksena oli tutkia Segment Routing -reititystä ja miten sitä voidaan hallita keskitetyltä kontrollerilta käsin. Segment Routing on uusi lähdereititykseen perustuva teknologia, jossa reititin voi määrittää, mitä polkua pitkin verkkoliikenne ohjataan kohteeseen. Verkkoliikenteen ohjaus kohdeosoitteeseen pohjautuu tyypillisesti lyhimmän polun reititykseen (shortest path), joka ei välttämättä aina ole optimaalisin reitti erityyppiselle liikenteelle.

Tavoitteena oli selvittää, miten keskitetyltä kontrollerilta käsin voidaan ohjata verkkoliikennettä Segment Routing -verkossa ja muodostaa topologia operoitavasta verkosta kontrollerille. Työssä muodostettiin testiverkko virtuaaliympäristöön, jossa suoritettiin kolme erilaista käytännön testausta, joissa verkkoliikennettä optimoitiin hyödyntäen Segment Routing -teknologiaa. Tuloksia tarkasteltiin suhteessa tilanteeseen, jossa verkkoliikennettä ei ollut optimoitu. Lisäksi testattiin, miten operoitavasta verkosta voidaan muodostaa topologia ohjelmisto-ohjatulle kontrollerille useasta autonomisesta alueesta ja ohjata verkkoliikenne niiden läpi.

Tutkimus ja tulokset osoittivat, että Segment Routing yhdessä keskitetyn kontrollerin kanssa on potentiaalinen yhdistelmä verkkoliikenteen ohjaamiseen eri sovellusten vaatimusten mukaisesti.

Avainsanat	Segment Routing, PCEP, BGP-LS, SDN, MPLS

Author Title Number of Pages Date	Paul Nurminen Centralized segment routing 54 pages + 5 appendices 28 February 2017
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Network Technologies
Instructor	Erik Pätynen, Senior Lecturer

This thesis presents a study of segment routing and how it can be implemented in a centralized manner. Segment routing is a new routing technology based on source routing where a router can choose a path which a packet must traverse to the destination. Network traffic is typically forwarded based on the shortest path routing which is not always the most optimal path for different types of traffic.

The aim of this study was to find out how the network traffic can be controlled from a centralized controller in a segment routing enabled network and how a topology can be constructed from the network to the controller. A test network was built in a virtual environment where three different tests were performed exploiting the segment routing technology. The results were examined in relation to a situation where the network traffic was not optimized. Additionally, it was tested how the network topology can be constructed to the Software-Defined Networking controller from multiple autonomous systems and how the network traffic can be forwarded through these.

The study and the results showed that segment routing and a centralized controller are a potential combination in terms of network traffic forwarding based on different application needs.

Keywords	Segment Routing, PCEP, BGP-LS, SDN, MPLS

Sisällys

Liitteet

Liite 1. Verkkotopologia 1

Liite 2. Verkkotopologia 2

Liite 3. Reitittimien IP-osoitteet

Liite 4. XRv-A:n konfiguraatio

Liite 5. XRv-D:n konfiguraatio

1 Johdanto

Verkkoliikenteen ohjaus kohdeosoitteeseen pohjautuu tyypillisesti lyhimmän polun reititykseen (shortest path), joka ei välttämättä aina ole optimaalisin reitti erityyppiselle liikenteelle. Esimerkiksi reaaliaikasovellukset ovat herkkiä latenssille ja viiveen vaihtelulle, kun taas suuria datamääriä liikuttelevat sovellukset tarvitsevat enemmän kaistanleveyttä. Liikenteen erottelu eri sovellusten vaatimusten mukaisesti voi olla työlästä toteuttaa käyttäen käytäntöperusteista reititystä (policy-based routing), joka vaatii staattisia konfiguraatiomuutoksia reitittimille.

Segment Routing -arkkitehtuuri (SR) perustuu lähdereitityksen paradigmaan. Reititin päättää, mitä polkua pitkin paketti ohjataan kohteeseen lisäämällä paketin otsikkoon segmenttilistan, joka ohjeistaa paketin vastaanottavaa reititintä siitä, miten paketti tulee ohjata eteenpäin. Keskitetyssä mallissa Segment Routing tarvitsee kuitenkin rinnalle komponentin, joka laskee ja ohjeistaa sisääntuloreitittimelle (ingress router) polun, jota pitkin määrätynlainen liikenne tulee ohjata. Tämä tehtävä voidaan toteuttaa tilallisella Path Computation Element -palvelimella (PCE), joka voidaan sisällyttää SDN-kontrolleriin.

Tässä työssä tutkitaan SR-teknologiaa ja sitä, miten SR-reititystä voidaan toteuttaa keskitetysti SDN-kontrollerilta. Työssä käsitellään SR-reititys MPLS-välitystasolla ja käydään läpi erilaisia tekniikoita ja protokollia, joita käyttäen voidaan muodostaa kuva operoitavasta verkosta sekä toteuttaa SR-reititystä verkon ulkopuolisesta komponentista käsin, tässä työssä OpenDaylight-kontrollerista (ODL).

Virtuaaliympäristöön muodostetaan SR-teknologiaa tukeva verkko käyttäen Ciscon IOS XRv -virtuaalireitittimiä ja asennetaan ODL-kontrolleri ja Linux-pääteasemat, joilla testataan ja mitataan verkkoliikennettä erilaisissa tilanteissa.

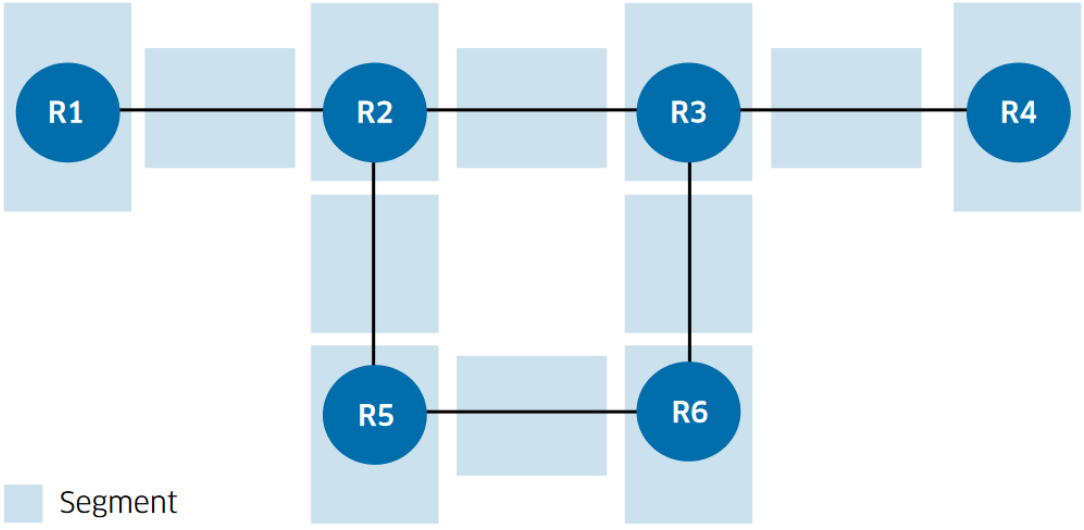
Käytännön testeissä suoritetaan kolme erilaista skenaariota, joissa SR-teknologiaa hyödynnetään verkkoliikenteen optimoimiseen, sekä kerätään ja vertaillaan tuloksia tilanteeseen, jossa liikennettä ei ole optimoitu. Tavoitteena on selvittää, miten SR-reititystä voidaan ohjata keskitetyltä kontrollerilta eri tarpeiden mukaan.

2 Segment Routing -teknologia

Segment Routing (SR) on lähdereititykseen (source routing) perustuva verkkoteknologia, jossa reititin voi määritellä, mitä polkua pitkin paketti ohjataan verkossa kohteeseen. Lähdereititin määrittää polun lisäämällä paketin otsikkoon yksittäisen segmenttitunnisteen tai segmenttilistan, joka opastaa paketin vastaanottavaa reititintä, minne paketti tulee ohjata. [1.]

Segmentillä viitataan yksittäiseen verkkolaitteeseen, kuten reitittimeen, laiteryhmään tai laitteiden välisiin linkkeihin. Jokaiselle segmentille määritellään oma segmenttitunniste (SID), jota mainostetaan SR-alueen sisällä käyttäen IGP (Interior Gateway Protocol) -protokollaa, kuten OSPF (Open Shortest Path First) ja IS-IS (Intermediate System-to-Intermediate System) tai BGP (Border Gateway Protocol) -protokollaa, mikäli segmenttiä halutaan mainostaa autonomisen alueen (AS) ulkopuolelle. [2.]

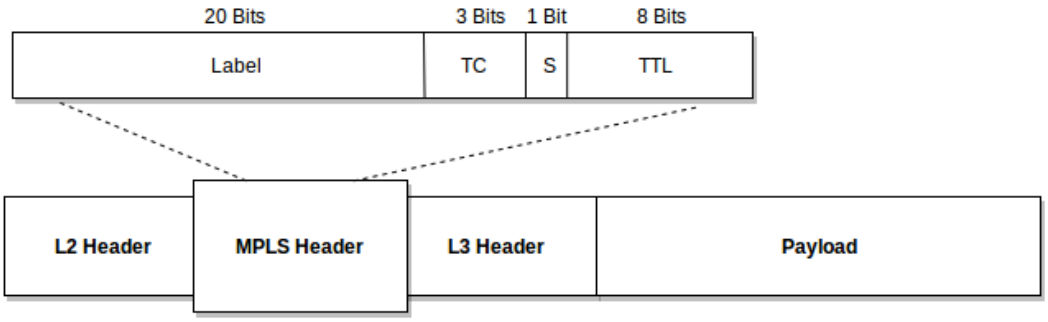
Verkkolaitteet, jotka tukevat SR-ominaisuutta, muodostavat yhdessä alueen (SR domain), jossa jokainen verkkolaite ja niiden väliset linkit ovat oma segmenttinsä ja jokaiselle segmentille on määritelty segmenttitunniste. Kuva 1 havainnollistaa, miten SR-alue jakaantuu segmentteihin.



Kuva 1. Segment Routing -alue [3].

SR-arkkitehtuuri voidaan ottaa käyttöön MPLS-verkoissa ilman, että välityskerrokseen (data plane) tarvitsee tehdä muutoksia. Tyypillisesti reitittimiin tarvitaan vain ohjelmistopäivitys, jotta SR-toiminnallisuus voidaan ottaa käyttöön. SR MPLS-toteutuksessa

yksittäinen segmenttitunniste koodataan MPLS-leimana ja segmenttilista MPLS-leimapinona. Kuvassa 2 on MPLS-otsikko, josta nähdään, miten se sijoittuu L2- ja L3-otsikoiden väliin. Segmenttitunniste on 32-bittinen kokonaisuus, jonka 20 ensimmäistä bittiä yksilöi segmenttitunnisteen. Segmenttilista muodostuu useasta peräkkäisestä MPLS-otsikosta, jotka sijoittuvat L2- ja L3- otsikoiden väliin. Leimapinon aktiivinen segmentti on lähimpänä L2-otsikkoa. MPLS-otsikon S-bitti määrittelee leimapinon viimeisen leiman, jolloin sen arvo on 1. [4.]



Kuva 2. MPLS-otsikko [8].

SR-arkkitehtuuri voidaan ottaa käyttöön myös IPv6-välitystason kanssa, jolloin IPv6-osoitteeseen lisätään erillinen otsikko, Segment Routing Header (SRH), johon segmentit koodataan IPv6-osoitteina. Segmenttilistat koodataan IPv6-osoitelistana, joka sisällytetään SRH-otsikkoon. Osoitin SRH-otsikon sisällä osoittaa aktiiviseen segmenttiin, ja kun reititin on käsitellyt aktiivisen segmentin, sitä ei poisteta listalta, vaan osoitin vaihtaa seuraavaan segmenttiin listassa. [5.]

Tässä työssä keskitytään SR MPLS -toteutukseen, joten SR IPv6 -toteutusta ei käsitellä tämän syvällisemmin.

2.1 Segment Routing -arkkitehtuuri

Segmentit

Segmentit voidaan jakaa kahteen luokkaan, globaaleihin ja lokaaleihin, sen mukaan, mikä niiden merkitys SR-alueen sisällä on. Jokainen SR-alueen reititin osaa tulkita globaaleja segmenttejä ja lisää ne omaan FIB (Forwarding Information Base) -tauluunsa. Globaalit segmentit ovat uniikkeja SR-alueen sisällä, joten niiden arvoja ei voida käyt-

tää useaan segmenttiin. Globaaleja segmenttejä käytetään tunnistamaan SR-alueen reitittimiä ja IP-prefiksejä. [2.]

Lokaalit segmentit ovat tulkittavissa vain sen laitteen osalta, josta segmentti on peräisin. Toiset SR-alueen laitteet eivät osaa tulkita niitä eivätkä lisää niitä omaan FIB-tauluunsa. Reitittimien linkit ovat lokaaleja segmenttejä, joiden arvoja voidaan käyttää usealle eri reitittimen segmentille. [2.]

Segment Routing Global Block

SRGB on MPLS-leimojen arvoalue, joka on varattu globaalien segmenttitunnisteiden käyttöön. Jokainen SR-alueeseen kuuluva reititin allokoi määrätyn arvoalueen sen lokaalien leimojen alueelta SRGB:n käyttöön. SRGB voi käyttää absoluuttisia arvoja tai indeksiarvoja, jolloin SRGB-arvoalue voi vaihdella eri reitittimillä. Mikäli absoluuttisia arvoja käytetään, tulee SR-alueen kaikilla reitittimillä olla määritelty sama SRGB-alue. Indeksiarvoja käytettäessä kukin reititin voi itsenäisesti määritellä SRGB-arvoalueen haluamalleen välille. Segmenttitunnisteen arvo saadaan lisäämällä indeksiarvo SRGB:n ensimmäiseen lukuun. [6.]

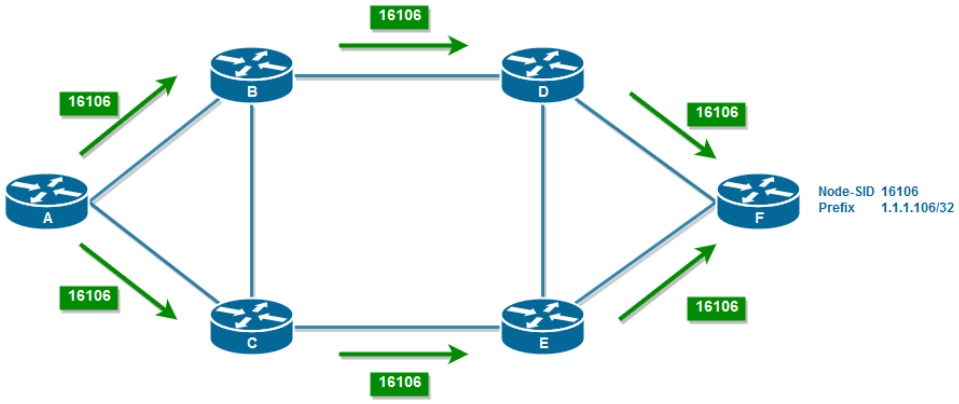
Segmenttitunnisteet

Jokaisella SR-alueen segmentillä on oma tunniste (SID). Segmenttitunnisteet voidaan jakaa kahteen pääryhmään, jotka ovat Prefix-SID ja Adjacency-SID. Prefix-SID-ryhmän tunnisteet ovat globaaleja tunnisteita, jotka löytyvät jokaisen SR-alueen laitteen FIB-taulusta. Adjacency-SID tunnisteet ovat lokaaleja, ja ne lisätään vain paikallisesti sen laitteen FIB-tauluun, josta segmentti on lähtöisin. [2.]

Prefix-SID

Prefix-SID on globaali segmenttitunniste, joka toimii tunnisteena tietylle verkkoprefiksille. Kun reititin vastaanottaa paketin, jonka aktiivisena segmenttitunnisteena on Prefix-SID, ohjaa reititin paketin kohteeseen käyttäen lyhimmän polun reititystä sekä suorittaa ECMP (Equal Cost Multi Path) -kuormantasausta, mikäli verkkoprefiksille löytyy useita samanarvoisia polkuja. Prefix-SID-tunnisteilla voidaan tunnistaa joko yksittäisiä reitittimiä tai reititinryhmiä. Prefix-SID-segmenttitunnisteelle on kaksi alityyppiä, Node-SID ja Anycast-SID. [1.]

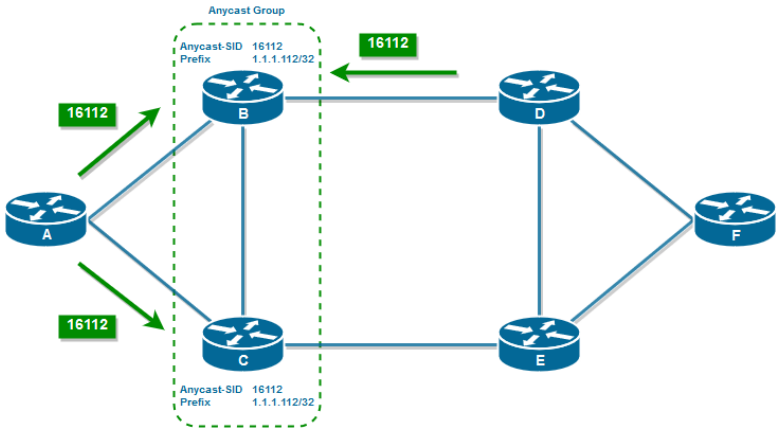
Node-SID-tunnisteella viitataan tiettyyn reitittimeen. Se voidaan liittää ainoastaan IP-osoitteeseen, jolla tunnistetaan kyseinen reititin. Tyypillisesti tämä osoite on reitittimen loopback-osoite. Kuvan 3 esimerkissä esitetään, miten reititin A ohjaa paketit reitittimelle F, jonka Node-SID on 16106. Koska kohdereitittimelle on kaksi samanarvoista polkua, A-reititin suorittaa ECMP-kuormantasausta ohjatessaan paketteja F-reitittimelle.



Kuva 3. Node-SID-tunnisteen toimintaperiaate.

Anycast-SID on toinen Prefix-SID:n alityyppi, ja se toimii tunnisteena kaikille anycast-ryhmän reitittimille, johon kyseinen verkkoprefiksi on liitetty. Anycast-ryhmä muodostuu kahdesta tai useammasta reitittimestä, jotka mainostavat samaa anycast-prefiksiä liitettynä samaan Anycast-SID-tunnisteeseen. Kun reititin vastaanottaa paketin, jonka segmenttilistan aktiivisena tunnisteena on Anycast-SID, paketti ohjataan lähimpään reitittimeen, joka mainostaa kyseistä anycast-prefiksiä. Jos samaa prefiksiä mainostaa useampi reititin, joihin on samanarvoinen etäisyys, ohjataan paketit käyttäen kuormantasausta reitittimien välillä. [1.]

Kuvassa 4 esitetään anycast-segmentin toimintaa.

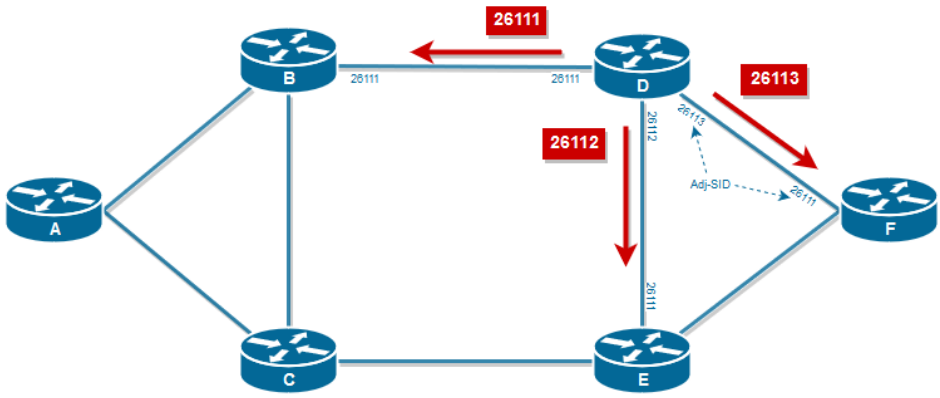


Kuva 4. Anycast-SID-tunnisteen toimintaperiaate.

Reitittimet B ja C muodostavat anycast-ryhmän, jotka molemmat mainostavat samaa anycast-prefiksiä, johon on liitetty sama Anycast-SID. Reititin A:lla on kaksi samanarvoista polkua lähimpään anycast-segmenttiin, joten se ohjaa paketit molemmille reitittimille suorittamalla kuormantasausta. Reititin D ohjaa paketit vain B-reitittimelle, koska samalla etäisyydellä ei ole muita kyseisen anycast-ryhmän reitittimiä.

Adjacency-SID

Adjacency-SID (Adj-SID) on segmenttitunniste, joka viittaa reitittimen tiettyyn linkkiin. Se on lokaali tunniste, joten vain reititin, josta tunnistetta mainostetaan, lisää sen omaan FIB-tauluunsa. Tämän vuoksi samaa Adj-SID leiman arvoa voidaan käyttää usealla reitittimellä. Reititin allokoi Adj-SID-tunnisteet SRGB-alueen ulkopuolelta. Adj-SID-segmenttiä käytetään, kun halutaan ohjata paketti tiettyyn linkkiin, vaikka tämä ei olisi lyhin polku kohdeprefiksiin. [1.]



Kuva 5. Adjacency-SID-tunnisteen toimintaperiaate.

Kuvasta 5 nähdään, miten reititin D allokoi jokaiselle linkilleen Adj-SID-tunnisteen. Sama arvo voi olla käytössä myös toisten reitittimien Adj-SID-tunnisteissa, koska se on vain lokaalisti merkittävä. Kun paketti saapuu reitittimelle ja paketin aktiivisena segmenttinä on Adj-SID, ohjataan paketti tunnistetta vastaavaan linkkiin riippumatta siitä, onko se lyhyin polku kohdeprefiksiin.

2.2 Segmenttilistan operaatiot

SR-alueella paketit kuljettavat segmenttilistaa niiden otsikossa. Segmenttilistan aktiivinen segmentti määrittää, miten kyseinen paketti ohjataan eteenpäin. Segmenttilistan ylläpitoon on määritelty kolme operaatiota, joita reitittimet voivat soveltaa segmenttilistaan:

- PUSH – segmentti tai segmenttilista lisätään paketin otsikkoon ja listan päällimmäinen segmentti asetetaan aktiiviseksi
- CONTINUE – aktiivista segmenttiä ei ole vielä suoritettu loppuun, joten se säilyy edelleen aktiivisena
- NEXT – aktiivinen segmentti on suoritettu loppuun, joten se poistetaan segmenttilistasta ja seuraava segmentti listassa tulee aktiiviseksi.

SR-arkkitehtuuri voidaan ottaa käyttöön MPLS-verkoissa ilman, että MPLS-välitystasoon tarvitsee tehdä muutoksia. Segmentit koodataan MPLS-leimoina ja segmenttilistat leimapinoina. Aktiivinen segmentti on leimapinon ylimmäisenä, ja se poistetaan pinosta, kun se on prosessoitu. [2.]

MPLS-arkkitehtuurissa [7] on määritelty kolme operaatiota, joita reitittimet suorittavat leimapinoille:

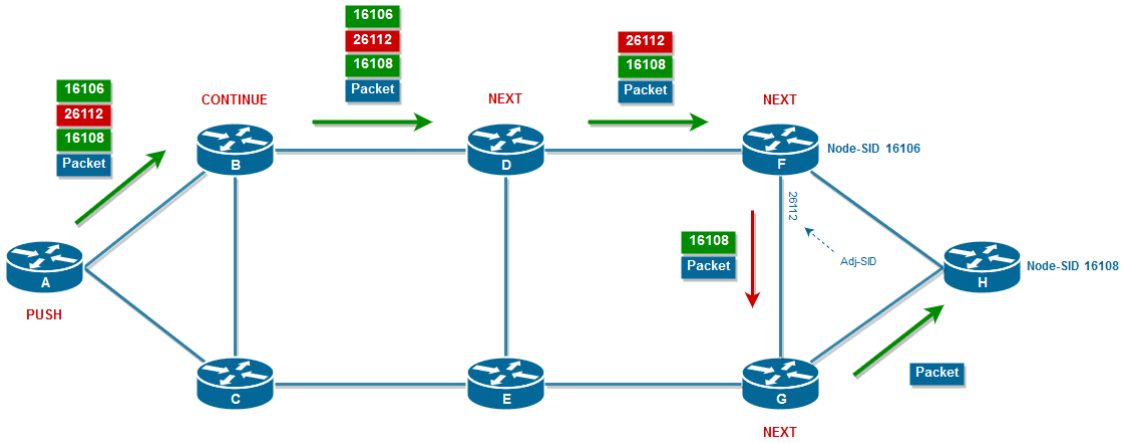
- PUSH – leima lisätään leimapinon ylimmäksi
- SWAP – leimapinon ylin leima korvataan uudella leimalla
- POP – leimapinon ylin leima poistetaan pinosta.

SR käyttää näitä samoja välitystason operaatioita MPLS-verkoissa, ja niiden vastaavuudet nähdään taulukossa 1.

Taulukko 1. Segmenttilistan operaatiot verrattuna MPLS-leimapinon operaatioihin [9].

Segmenttilistan operaatio	MPLS-leimapinon operaatio
PUSH	PUSH
CONTINUE	SWAP
NEXT	POP

Kuvassa 6 on havainnollistettu segmenttilistan operaatioita MPLS-verkossa. Kuvassa IP-paketti saapuu A-reitittimelle, jonka kohde on H-reitittimen takana. Yhteydelle on määritelty Label Switched Path -polku (LSP), joka kulkee segmenttien 16106, 26112 ja 16108 kautta.



Kuva 6. Segmenttilistan operaatiot MPLS-välitystasolla.

Reititin A lisää paketin otsikkoon segmenttilistan ja määrittää segmentin 16106 aktiiviseksi suorittamalla PUSH-operaation. Tämän jälkeen paketti ohjataan B-reitittimelle, koska lyhin polku segmenttiin 16106 kulkee sen kautta.

Reititin B suorittaa paketille CONTINUE-operaation, joten segmenttilista pysyy muuttumattomana ja paketti ohjataan lyhintä reittiä kohti aktiivista segmenttiä 16106, tässä tapauksessa D-reititintä kohti.

Reititin D suorittaa paketille NEXT-operaation, joten aktiivinen segmentti 16106 poistetaan listasta ja paketti ohjataan reitittimelle F, jonka Node-SID on 16106. Segmenttilistan aktiiviseksi segmentiksi tulee 26112, joka on reitittimen F lokaali segmenttitunniste.

Reititin F suorittaa paketille NEXT-operaation, koska segmenttilistan aktiivisena segmenttinä on Adj-SID, joille suoritetaan aina NEXT-operaatio. Tämä jälkeen aktiiviseksi

segmentiksi tulee 16108 ja paketti ohjataan Adj-SID-segmenttiä vastaavaan linkkiin kohti G-reititintä.

Reititin G suorittaa paketille NEXT-operaation, ja poistaa aktiivisen segmentin 16108 paketin otsikosta, minkä jälkeen paketti ohjataan ilman SR-otsikkoa reitittimelle H, joka ohjaa paketin edelleen kohteeseen tavallisena IP-pakettina.

2.3 Segmenttien mainostus SR-alueella

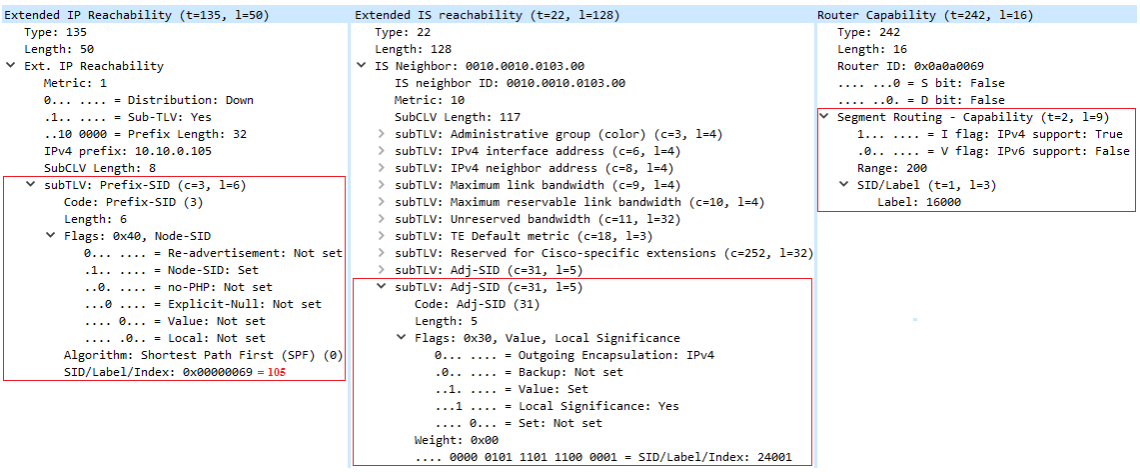
SR-arkkitehtuuri ei määrittele mitään tiettyä ohjaustason (control plane) toteutustapaa, miten segmenttejä tulee mainostaa SR-alueella. Vaikka segmentit voitaisiin teoriassa konfiguroida staattisesti reitittimiin, käytetään niiden mainostamiseen yleensä jotakin reititysprotokollaa. SR-ohjaustaso tukee IGP-protokollia IS-IS ja OSPF sekä BGP-protokollaa. Segmentit voidaan siis jakaa IGP-segmentteihin ja BGP-segmentteihin. [1.]

IGP-segmenttien mainostus

Segmenttejä voidaan mainostaa IGP-protokollilla, joihin on lisätty laajennusosia ja jotka sisällytetään IGP-mainostuksiin.

IS-IS on helposti laajennettavissa oleva protokolla, joka ei käytä kiinteästi määriteltyä formaattia linkkitilojen mainostuksessa vaan sisältää joukon Type-Length-Value (TLV) -triplettejä, joihin informaatio koodataan. TLV:hen voidaan sisällyttää sub-TLV-triplettejä, joihin voidaan enkapsuloida tarvittavia lisätietoja. Uuden toiminnallisuuden lisääminen voidaan tehdä helposti määrittelemällä uusi TLV tai laajentamalla jo määriteltyä TLV:tä uusilla sub-TLV-tripleteillä. [10.]

IS-IS-protokollaan on lisätty sub-TLV-laajennukset Prefix-SID- ja Adjacency-SID-tunnisteille sekä SRGB-alueelle, joita käyttäen voidaan mainostaa tarvittavat tiedot reitittimen SR-ominaisuuksista. Kuva 7 esittää pakettikaappausta IS-IS-protokollan linkkitilapaketista (link-state packet), jonka vasemmanpuoleisesta osasta nähdään, kuinka Prefix-SID sub-TLV on sisällytetty IP Reachability TLV:n alle. Keskimmäinen osa näyttää, kuinka Adj-SID sub-TLV:n sisältyy IS Reachability TLV:n alle. Oikeanpuolimmaisessa osassa näkyy SR-Capability sub-TLV ja se, miten se on sisällytetty Router Capability TLV:n sisään.



Kuva 7. Sub-TLV-laajennukset IS-IS-protokollan linkkitilapaketissa.

Prefix-SID sub-TLV sisältää tiedon Prefix-SID-indeksiarvosta ja siitä, mitä algoritmia käyttäen reitti kohdeprefiksiin on laskettu. Liput-osiosta (Flags) nähdään, mikäli segmenttitunniste on Node-SID. Jos N-lipun arvo on 1, se tarkoittaa, että segmenttitunniste on Node-SID. [10.]

Adj-SID sub-TLV liitetään IS-IS-naapurin IS Reachability TLV:n alle, ja se määrittää Adj-SID-tunnisteen arvon, joka on absoluuttinen, mikäli V-lipun arvo on 1. L-lippu määrittää segmenttitunnisteen merkittävyyden SR-alueella. Mikäli tunniste on lokaalisti merkittävä, sen arvo on 1.

SR-Capabilities sub-TLV on sisällytetty Router Capability TLV:n alle ja sisältää tiedon SRGB-alueesta. SID/Label sub-TLV-osio määrittää SRGB:n ensimmäisen luvun ja Range SRGB:n alueen suuruuden. Lisäksi I- ja V-liput määrittävät, mitkä IP-versiot on tuettu.

Myös OSPF-protokollaan on lisätty laajennuksia, joilla voidaan mainostaa segmenttejä SR-alueella. Tässä työssä ei kuitenkaan käsitellä SR-arkkitehtuuria OSPF:n osalta.

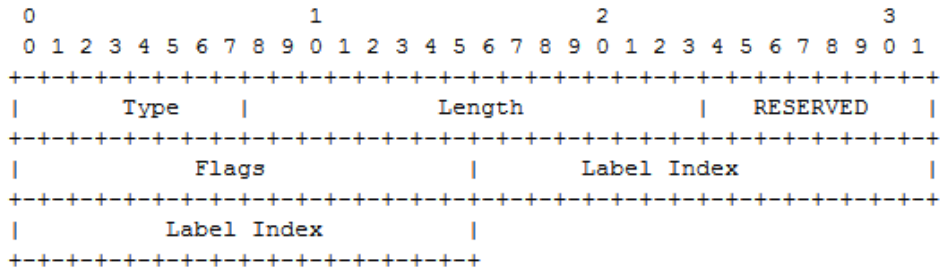
BGP-segmenttien mainostus

BGP Prefix-SID -segmenttitunnisteiden mainostukseen käytetään valinnaista (optional) ja transitiivista (transitive) BGP Prefix-SID path-attribuuttia (tyyppi 40), joka sisällytetään BGP UPDATE-viestiin. Mikäli vastaanottava reititin ei tunnista sitä, se ohjaa kuitenkin tiedon eteenpäin toisille reitittimille. BGP Prefix-SID-attribuutti voidaan liittää IP-prefiksiin käyttäen BGP Labeled Unicast (AFI 1(2) / SAFI 4) -osoiteperhettä. BGP Pre-

fix-SID -attribuutti voi sisältää yhden tai useita TLV triplettejä, joita ovat MPLS-välityskerrosta käytettäessä

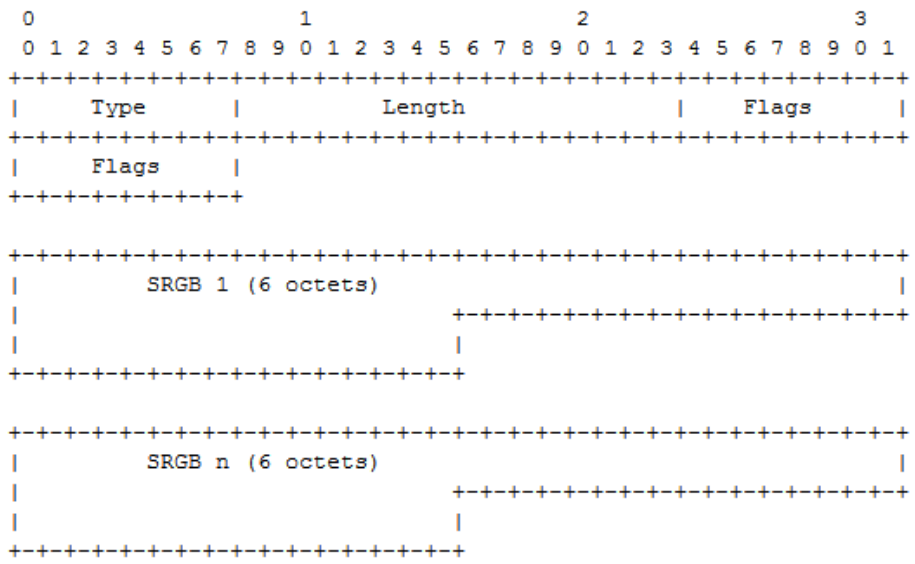
- Label-Index TLV
- Originator SRGB TLV.

Label-Index TLV sisältää BGP Prefix-SID -tunnisteen indeksiarvon, joka nähdään kuvassa 8. Indeksiarvo lisätään reitittimen SRGB-alueen ensimmäiseen lukuun. Label-Index TLV on pakollinen arvo, joka tulee sisällyttää BGP Prefix-SID path-attribuuttiin. [11.]



Kuva 8. Label-Index TLV [11].

Originator SRGB TLV on valinnainen ja voidaan käyttää yhdessä Label-Index TLV:n kanssa. Originator SRGB TLV sisältää prefiksiä mainostavan reitittimen SRGB-arvoalueen, ja niitä voi olla myös useita, mikäli reitittimen SRGB-alue muodostuu useasta eri alueesta. Kuvasta 9 nähdään Originator SRGB TLV:n rakenne. [11.]



Kuva 9. Originator SRGB TLV [11].

3 Segment Routing ohjelmisto-ohjatuissa verkoissa

SR-reititys voidaan toteuttaa hajautetusti tai keskitetysti. Hajautetussa mallissa jokainen reititin päättää itse SR-polkujen käyttöönotosta, ja ne konfiguroidaan manuaalisesti reitittimeen. Tätä toimintamallia ei kuitenkaan suositella kuin testaukseen ja vianetsintätarkoituksiin, koska hajautettu malli on huonommin skaalautuva ja sen hallittavuus rajoitteisempaa kuin keskitetyssä mallissa [8]. Keskitetyssä mallissa ohjaustaso voidaan erottaa reitittimestä ulkopuoliselle komponentille, kuten SDN-kontrollerille, jolla on näkyvyys koko verkon alueelle ja joka tekee päätökset SR-polkujen luomisesta ja päivittämisestä reitittimien puolesta.

Tässä luvussa käydään läpi komponentteja ja protokollia, joita käytetään keskitetyssä SR-mallissa. Niitä ovat muun muassa SDN-kontrolleri sekä BGP-LS- ja PCEP-protokollat.

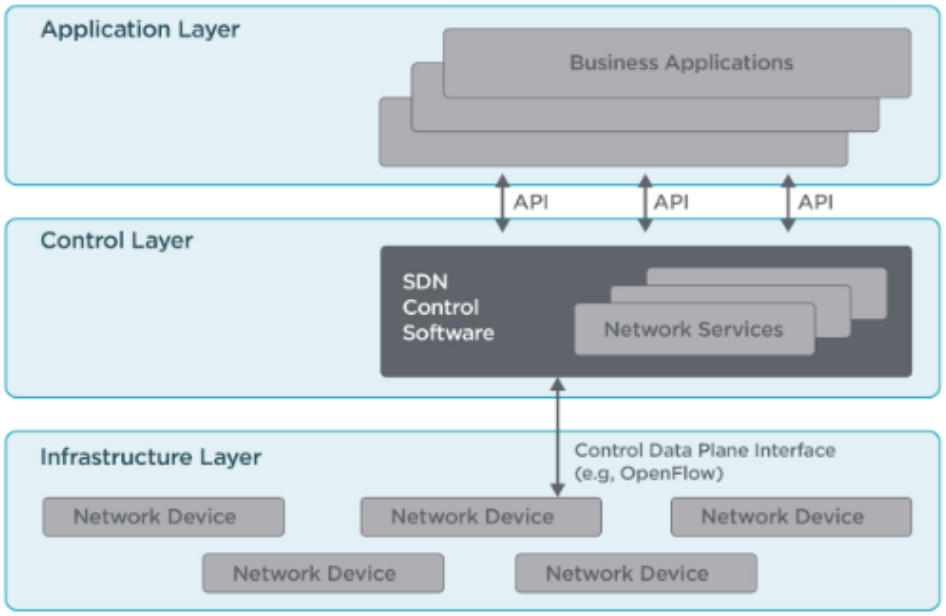
3.1 Software-Defined Networking (SDN)

3.1.1 SDN-arkkitehtuuri

Software-Defined Networking on verkkoarkkitehtuuri, jossa ohjaustaso (control plane) voidaan siirtää verkkolaitteilta loogisesti keskitetylle kontrollerille erottamalla ohjaustaso ja välitystaso (data plane) toisistaan.

Perinteisissä verkoissa molemmat tasot sijaitsevat samassa laitteessa, jolloin jokainen verkon laite tekee verkkoon kohdistuvat päätökset (esim. reitityspäätös) itsenäisesti. Tämän hajautetun toimintamallin on perinteisesti katsottu olevan paras tapa varmistaa verkon toimintavarmuus. Hajautettu verkko on kuitenkin kompleksinen ja suhteellisen staattinen ympäristö, jonka hallinta ja kontrollointi voi olla kankeaa. Jokainen muutos täytyy tehdä manuaalisesti kullekin verkon laitteelle. [14.]

Kuvassa 10 nähdään SDN-arkkitehtuuri, joka voidaan jakaa kolmeen loogiseen osaan: sovellustasoon, ohjaustasoon ja välitystasoon.



Kuva 10. Looginen SDN-arkkitehtuuri [15].

Sovellustaso on SDN-arkkitehtuurin ylin taso, jonka kautta verkkoa hallitaan. SDN-sovellukset ovat ohjelmia, jotka kommunikoivat ohjelmallisesti SDN-kontrollerille niiden vaatimuksista verkolle. Sovelluksella voi olla näkymä verkkotopologiasta, jota se käyttää tehdessään verkkoa koskevia päätöksiä. Esimerkkinä voi olla sovellus, joka varaa tietyn määrän kaistanleveyttä verkosta itselleen ennalta määrättynä ajankohtana, jonka SDN-kontrolleri laskee ja varaa sovellukselle. [16.]

Ohjaustaso niputtaa sovellustasolta tulevat palvelupyynnöt komennoiksi ja direktiiveiksi, jotka se välittää välitystason laitteille ja toimittaa SDN-sovelluksille informaatiota välitystason topologiasta ja toiminnoista. SDN-kontrolleri operoi tässä kerroksessa. [17.]

Northbound-rajapinta mahdollistaa SDN-sovellusten pääsyn ohjaustason toimintoihin ja palveluihin ilman, että niiden tarvitsee tietää yksityiskohtia välitystason laitteista. Northbound-rajapinta mahdollistaa verkon ohjelmoinnin sovellustasosta käsin ja sen on tyypillisesti katsottu olevan enemmän ohjelmointirajapinta (application programming interface) kuin protokolla. [16.]

Southbound-rajapinta tarjoaa loogisen yhteyden välitystason laitteiden ja SDN-kontrollerin välille. Tunnetuin protokolla Southbound-rajapinnalle on OpenFlow, mutta myös muita protokollia on kehitetty, kuten NETCONF, PCE tai BGP-LS. [16.]

SDN pyrkii myös korvaamaan patentoidut rajapinnat kontrolli- ja välitystason välillä avoimilla kommunikointiprotokollilla. Tämän vuoksi SDN mahdollistaa laitevalmistajasta riippumattoman (vendor agnostic) verkon toiminnan, jossa ohjaustaso kommunikoi eri laitevalmistajien kehittämien välitystasojen kanssa. [16.]

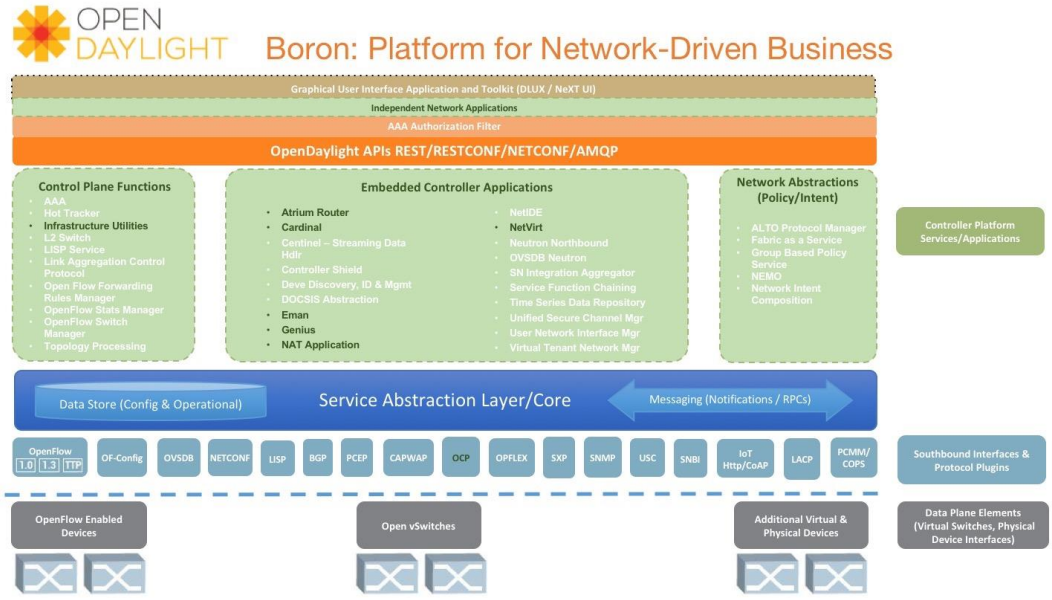
Välitystaso on SDN-arkkitehtuurin alin taso, joka muodostuu verkon fyysisistä tai virtuaalisista verkkolaitteista. Välitystason verkkolaitteet huolehtivat verkkoliikenteen kuljettamisesta ja prosessoimisesta sen mukaan, miten SDN-kontrolleri on niiden ohjeistanut tekevän. Välitystason laitteet suorittavat liikenteen edelleen lähettämistä ilman ohjaustason komponenttia, joten ne eivät tee itsenäisiä päätöksiä liikenteen ohjauksen suhteen. [17.]

3.1.2 OpenDaylight (ODL)

OpenDaylight on avoimen lähdekoodin projekti, joka on perustettu johtavien teknologiayhtiöiden (esim. Cisco, IBM, Juniper, Microsoft, VMWare) toimesta tavoitteenaan edistää SDN-teknologian omaksumista ja innovaatioita kehittämällä yhteisesti tuettua SDN-alustaa. OpenDaylight-projekti käynnistettiin vuonna 2013, ja sen ensimmäinen versio julkaistiin helmikuussa 2014. Se kantoi koodinimeä ”Hydrogen”. Uusin ja samalla viides versio julkaistiin syyskuussa 2016 koodinimellä ”Boron”. [18.]

ODL-kontrolleri on täysin ohjelmistopohjainen ja toimii Java-virtuaalikoneen (JVM) sisällä, joten se voidaan ottaa käyttöön missä tahansa laitteistossa tai käyttöjärjestelmässä, joka tukee Javaa. [18.]

ODL-arkkitehtuuri voidaan jakaa kolmeen kerrokseen, jonka ylin kerros sisältää Northbound-ohjelmointirajapinnan (API) ja sovellukset. Keskimmäinen kerros sisältää kontrollerialustan, ja alin kerros koostuu Southbound-liitännäisistä ja protokollista. ODL-arkkitehtuuri nähdään kuvassa 11.



Kuva 11. ODL-arkkitehtuuri [18].

ODL tukee REST-ohjelmointirajapintoja, joita käytetään integroimaan ulkopuolisia sovelluksia ODL-alustaan. REST API:a käytetään myös ODL:n graafisen käyttöliittymän (GUI) rakentamisessa. Ohjelmoitavuus on tärkeä osa SDN-alustoja ja ODL:n REST API mahdollistaa tarkoin määriteltyjen verkkosovellusten kehittämisen ODL-alustan ulkopuolella. Ne voivat hyödyntää ODL:n ominaisuuksia ja informaatiota REST API:n kautta. [16.]

Kontrollerialusta suorittaa SDN- ja verkkotoimintoja, kuten toimittaa verkon topologian, monitoroi suorituskykyä tai pitää yllä siihen liitettyjen verkkolaitteiden hallintaa. Tämä kerros toimii liimana Northbound- ja Southbound-rajapintojen välissä. [16.]

Service Abstraction Layer on ODL:n kriittisin osa, jonka pääasiallisena tarkoituksena on kartoittaa joukko sekalaisia verkkoteknologioita yhtenäiseen tietomalliin (data model). Kaikki ODL:n operoimat palvelut toimivat käyttäen tätä abstraktia tietomallia, mikä tekee siitä laitevalmistajasta riippumattoman SDN-kontrollerin. [16.]

ODL tukee useita eri protokollia, joita käyttäen se voi kommunikoida verkkolaitteiden kanssa. Koska ODL tukee liitännäistä arkkitehtuuria Southbound-rajapinnassa, siihen voidaan helposti lisätä tuki uusille protokollille. [16.]

3.2 Path Computation Element (PCE)

PCE on verkkoelementti, joka laskee pyydetyn verkkopolun tai reitin perustuen verkkotopologiaan ja soveltaa erilaisia rajoitteita ja ehtoja sen laskennassa. PCE kerää linkkitilatiedot verkon reitittimistä, joista se muodostaa verkon topologian Traffic Engineering Database (TED) -kantaan, jota se hyödyntää polkujen laskennassa. PCE voi sijaita reitittimessä tai tyypillisemmin erillisellä palvelimella. [19.]

Path Computation Client (PCC) on reititin, joka lähettää PCE:lle pyynnön laskea verkkopolku tarvitsemaansa kohteeseen. PCC on tyypillisesti MPLS-verkon reunalla sijaitseva reititin. [20.]

PCE-palvelimien käyttö polkujen laskennassa ei ole uusi asia. PCE:tä on tähän asti käytetty RSVP-TE:hen pohjautuvien LSP-polkujen laskentaan. RSVP-TE ei ole kuitenkaan saavuttanut suurta suosiota sen skaalautuvuus ongelmien takia [20], minkä vuoksi PCE-palvelimia ei ole tuotantoympäristöissä juurikaan nähty. [3.]

SR mahdollistaa saman lähdereitityksen periaatteen kuin RSVP-TE, mutta se skaalautuu paremmin, koska tilaa ei tarvitse ylläpitää polun varrella olevissa reitittimissä, vaan tila on paketissa itsessään. Tämän vuoksi reunareititin ei kuitenkaan pysty laskemaan polkua verkon läpi vapaana olevaan kaistanleveyteen perustuen. PCE on varteen otettava vaihtoehto, kun halutaan yhdistää SR:n skaalautuvuus ja verkkopolun laskenta perustuen vapaaseen kaistanleveyteen. [3.]

3.2.1 PCE-arkkitehtuuri

PCE-arkkitehtuuri muodostuu kahdesta toiminnallisesta komponentista, joita ovat PCE ja PCC. PCE on komponentti, joka suorittaa monimutkaisia polun laskentoja PCC:n puolesta. PCC on mikä tahansa verkkoelementti, joka pyytää polun laskentaa PCE:ltä. Tyypillisesti PCC on reititin, mutta voi olla myös toinen PCE, kuten tapauksessa, jossa PCE pyytää polun laskentaa toisen toimialueen PCE:ltä. PCE ja PCC käyttävät kommunikointiin PCEP-protokollaa. [22.]

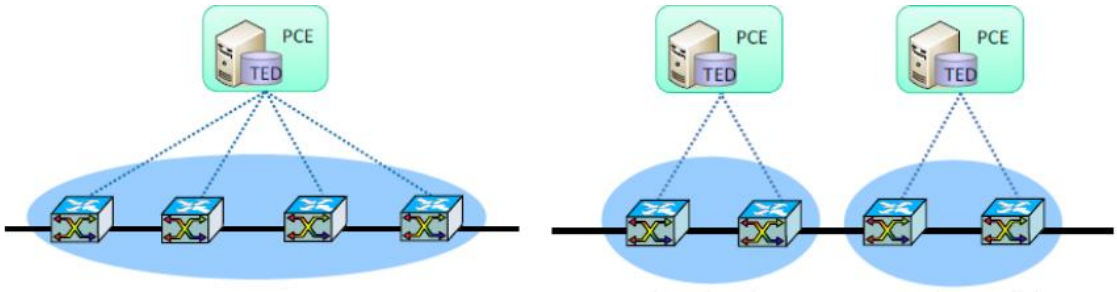
PCE on riippuvainen TED-kannasta suorittaessaan polkujen laskentaa. TED sisältää kaikki olennaiset tiedot verkon resursseista, kuten verkon topologian sekä linkkien kaistanleveydet ja metriikat, joita PCE tarvitsee suorittaessaan laskentoja. PCE lisää tiedot

verkon resursseista TED-kantaan joko IGP-protokollien linkkitilapäivityksistä (OSPF LSA tai IS-IS LSP), mikäli PCE osallistuu IGP-alueeseen tai BGP-LS-protokollan päivityksistä, jos PCE on muodostanut BGP naapuruuden jonkin IGP-alueen reitittimen kanssa. Lisäksi, jos PCE toimii tilallisena (stateful), se pitää yllä myös Label Switched Path-Database (LSP-DB) -kantaa, josta löytyy tiedot kullakin PCC:llä käyttöönotetuista LSP-poluista. PCE ylläpitää LSP-DB-kannan tietoja muodostamalla PCEP session jokaisen PCC:n kanssa ja synkronoimalla kannan tietoja säännöllisesti. [22.]

Keskitetyssä laskentamallissa (centralized computation model) yksittäinen PCE suorittaa kaikki laskennalliset toimenpiteet. PCE voi toimia erillisenä palvelimena (External PCE), tai vaihtoehtoisesti se voi toimia reitittimessä, joka tukee PCE toiminnallisuutta (Composite PCE). Keskitetyssä mallissa jokainen PCC lähettää laskentapyynnön samalle PCE:lle. Keskitetylle PCE:lle voidaan ottaa käyttöön varmistava PCE (backup PCE), jotta vältytään yksittäiseltä vikaantumispisteeltä (single point of failure). Kuitenkin vain yksi PCE toimii aktiivisesti samaan aikaan. [19.]

Hajautetussa laskentamallissa (distributed computation model) verkossa tai toimialueella voi toimia useita PCE:itä yhtä aikaa, jolloin ne jakavat laskentatoimitukset keskenään. Yksittäinen PCE voi laskea pyydetyn polun IGP-alueella ilman yhteistyötä toisen PCE:n kanssa, tai saman polun laskentaan voidaan käyttää useampaa PCE:tä, jolloin ne kommunikoivat keskenään, kukin sen IGP-alueen osalta, johon kuuluu. [19.]

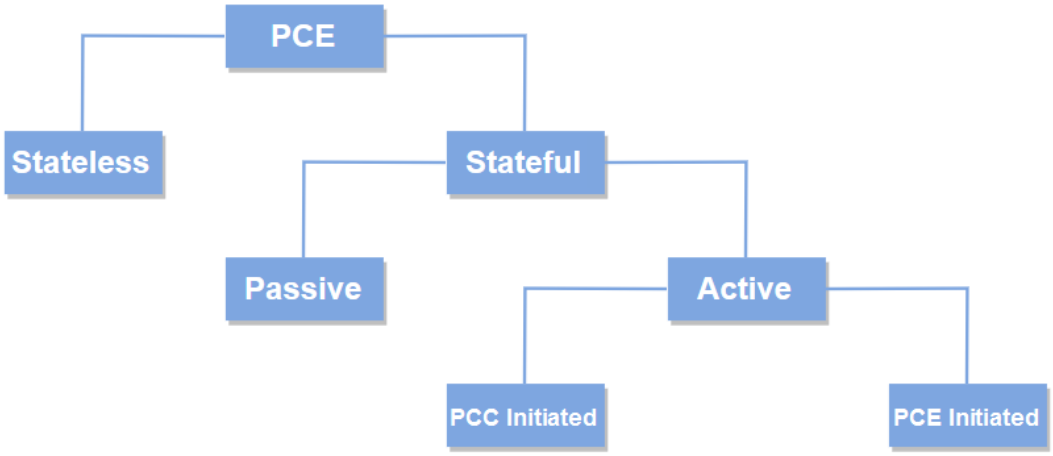
Kuvassa 12 näkyy keskitetyn ja hajautetun laskentamallin periaate. Kuvassa vasemmalla nähdään keskitetyn laskentamallin toiminta, jolloin jokainen PCC kommunikoi saman PCE:n kanssa. Kuvan oikealla puolella näkyy hajautetun laskentamallin periaate, jossa useampi PCE toimii kukin omalla toimialueellaan ja ne voivat tehdä yhteistyötä keskenään, mikäli halutaan laskea polku molempien toimialueiden läpi.



Kuva 12. Keskitetty ja hajautettu laskentamalli [22].

3.2.2 PCE-tyypit

PCE voi toimia joko tilattomana (stateless) tai tilallisena (stateful). Sen lisäksi tilallinen PCE voi olla joko passiivinen tai aktiivinen [19]. Kuvassa 13 näkyy, miten PCE-tyypit jakautuvat alikategorioihin.



Kuva 13. Tilaton ja tilallinen PCE [23].

Tilaton PCE ei ylläpidä tietoa verkossa käyttöönotetuista LSP-poluista, mikä rajoittaa sen kykyä optimoida verkon resursseja. Tilaton PCE suorittaa polun laskennan perustuen TED-kannan senhetkiseen tilaan ja käsittelee jokaisen pyynnön itsenäisesti toisistaan välittämättä. Tämä voi johtaa virheellisiin tuloksiin, mikäli TED-kanta on epäsynkronissa verkon senhetkisen tilan kanssa. Tilaton PCE ei voi myöskään uudelleenoptimoida verkossa käyttöönotettuja LSP-polkuja. [3.]

Tilallisella PCE:llä on tieto verkon tilan lisäksi myös verkossa muodostetuista LSP-poluista, joista se ylläpitää tietoja LSP-DB-kannassa. PCE hyödyntää polun laskennassa TED-kannan lisäksi tietoja LSP-DB-kannasta, mikä johtaa optimaalisempiin päätöksiin polkujen laskennassa, mutta tarvitsee luotettavan synkronointimekanismin PCE:n ja verkon välillä. [3.]

Tilallinen PCE voi olla passiivinen tai aktiivinen. Passiivinen PCE synkronoi TED- ja LSP-DB-kantojen tietoja, joita se käyttää polun laskennassa, mutta sillä ei ole oikeutta muokata verkossa muodostettuja polkuja. PCC on vastuussa polkujen muodostamisista ja niiden päivityksistä. Passiivinen PCE suorittaa laskentatoimituksia vain, kun PCC pyytää sitä. [24.]

Aktiivisella PCE:llä on enemmän oikeuksia toimia verkossa. PCC voi delegoida polkujen hallinnan PCE:lle, jolloin PCE voi muokata LSP:n ominaisuuksia milloin tahansa. Lisäksi aktiivinen PCE voi muodostaa täysin uusia LSP-polkuja verkkoon ja pitää niiden hallinnan itsellään. [24.]

3.2.3 Path Computation Element Protocol (PCEP) -protokolla

PCEP on protokolla, jota käytetään PCE:n ja PCC:n tai kahden PCE:n välisessä kommunikaatiossa. PCE-arkkitehtuuri edellyttää luotettavaa kommunikointia ja vuonhallintaa, joten PCEP-protokolla toimii TCP-protokollan päällä. PCEP-protokolla toimii TCP-portissa 4189. [25.]

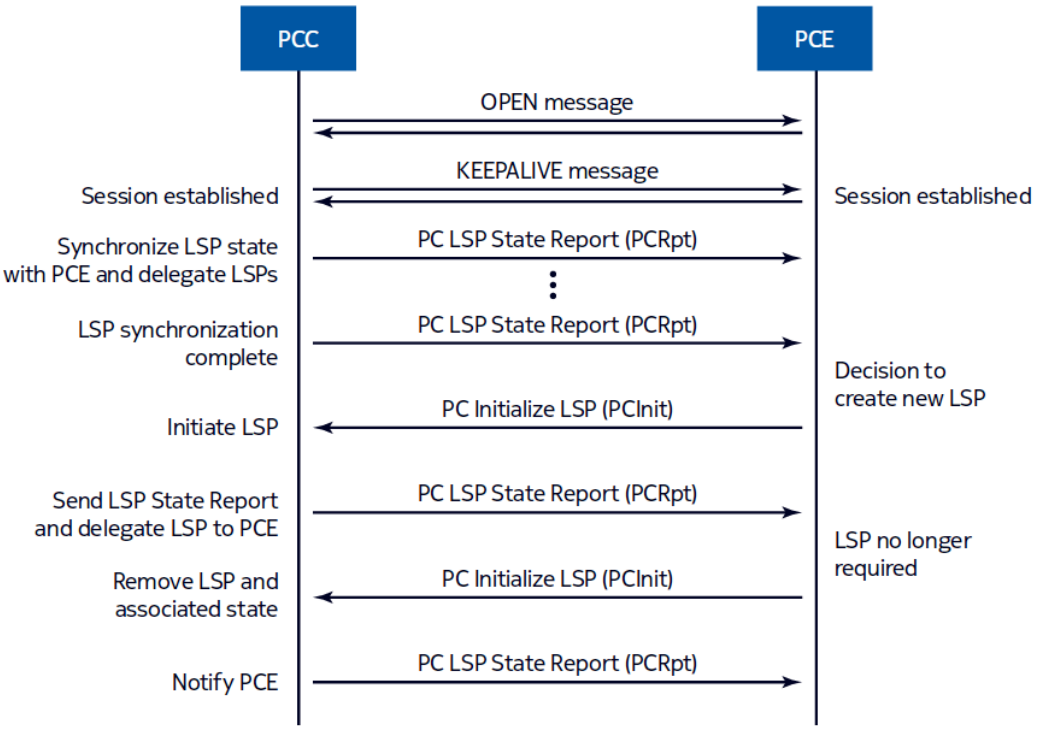
PCEP-protokolla toimii asiakas-palvelinperiaatteella ja käyttää erityyppisiä viestejä PCEP-session alustukseen, ylläpitoon ja PCE-operaatioihin. Taulukossa 2 on listattu PCEP-viestityypit, joita käytetään tilattomissa ja tilallisissa PCEP-sessioissa. Listan ensimmäiset seitsemän viestityyppiä on määritelty RFC5440:ssä, jonka julkaisuaikaan ei vielä ollut määrittelyä tilalliselle PCE:lle. PCE-arkkitehtuuriin on tämän jälkeen ehdotettu laajennus tilalliselle PCE:lle, jossa on määritelty uusia viestityyppejä PCEP-protokollalle.

Taulukko 2. PCEP-protokollan viestityypit [21].

Arvo	Viestityyppi	PCE-tyyppi	Lähde
1	Open	Stateless/Stateful PCE	RFC5440
2	Keepalive	Stateless/Stateful PCE	RFC5440
3	Path Computation Request, PCReq	Stateless/Stateful PCE	RFC5440
4	Path Computation Reply, PCRep	Stateless/Stateful PCE	RFC5440
5	Notification, PCNtf	Stateless/Stateful PCE	RFC5440
6	Error, PCErr	Stateless/Stateful PCE	RFC5440
7	Close	Stateless/Stateful PCE	RFC5440
10	Report, PCRpt	Stateful PCE	draft-ietf-pce-stateful-pce
11	Update, PCUpd	Stateful PCE	draft-ietf-pce-stateful-pce
12	Initiate, PCInit	Stateful PCE	draft-ietf-pce-pce-initiated-lsp

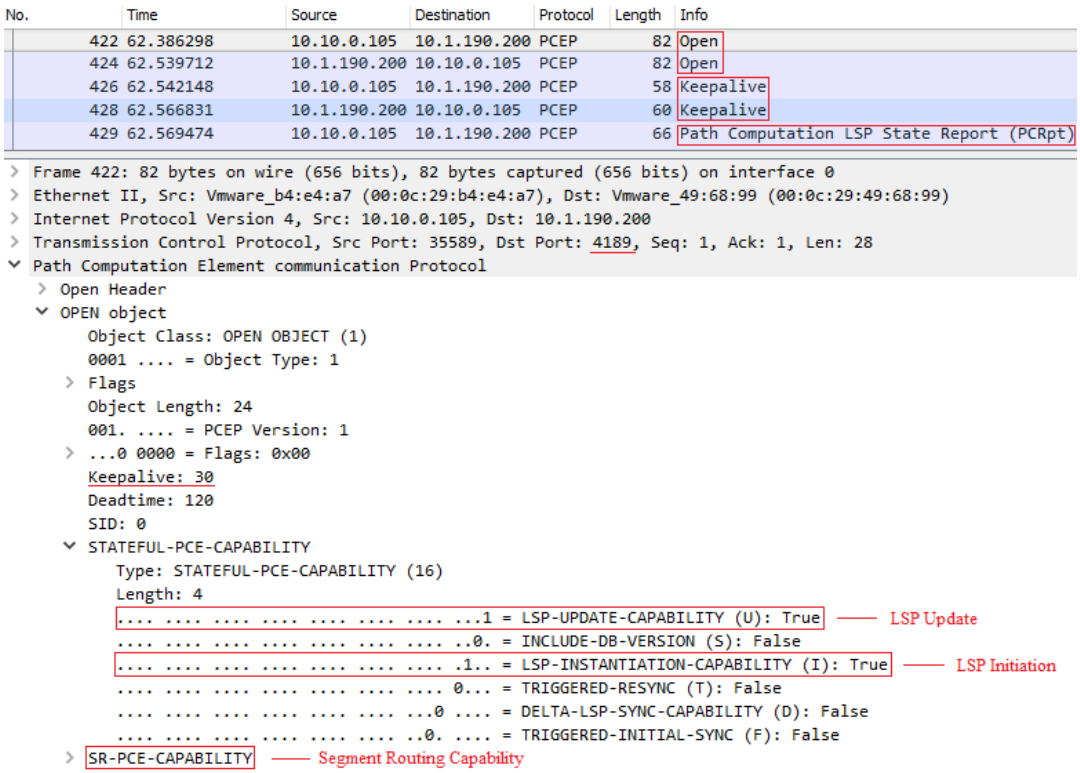
PCC voi muodostaa useita PCEP-sessioita usean eri PCE:n kanssa samoin kuin PCE voi muodostaa usean PCC:n kanssa, mutta saman PCE:n ja PCC:n välillä voi olla vain yksi PCEP-sessio yhtä aikaa. [24.]

Kuvassa 14 esitetään, kuinka PCC muodostaa PCEP-session PCE:n kanssa, jolle se delegoi LSP-polkujen hallinnan ja joka voi tehdä aloitteen LSP-polun muodostamisesta.



Kuva 14. Active Stateful PCEP -sessio [3].

Jotta PCEP-sessio voidaan muodostaa, tulee PCC:n ja PCE:n ensin muodostaa TCP-yhteys suorittamalla kolmivaiheisen kättelyn (three-way handshake). PCEP-session muodostus tapahtuu TCP-yhteyden päällä ja alkaa OPEN-viestien vaihdolla, joilla PCC ja PCE sopivat PCEP-session parametreista [24]. Kuvan 15 pakettikaappauksesta nähdään, kuinka PCC:n lähettämässä OPEN-viestissä on määritelty LSP-UPDATE-CAPABILITY- ja LSP-INSTANTIATION-CAPABILITY-lippujen arvoksi 1, joka antaa PCE:lle oikeuden muokata ja muodostaa täysin uusia LSP-polkuja PCC:lle. Lisäksi PCC ilmoittaa tukevansa SR LSP -polkujen toiminnallisuutta sisällyttämällä SR-PCE-CAPABILITY TLV:n OPEN-viestiin.



Kuva 15. PCEP-session alustus OPEN-viestillä.

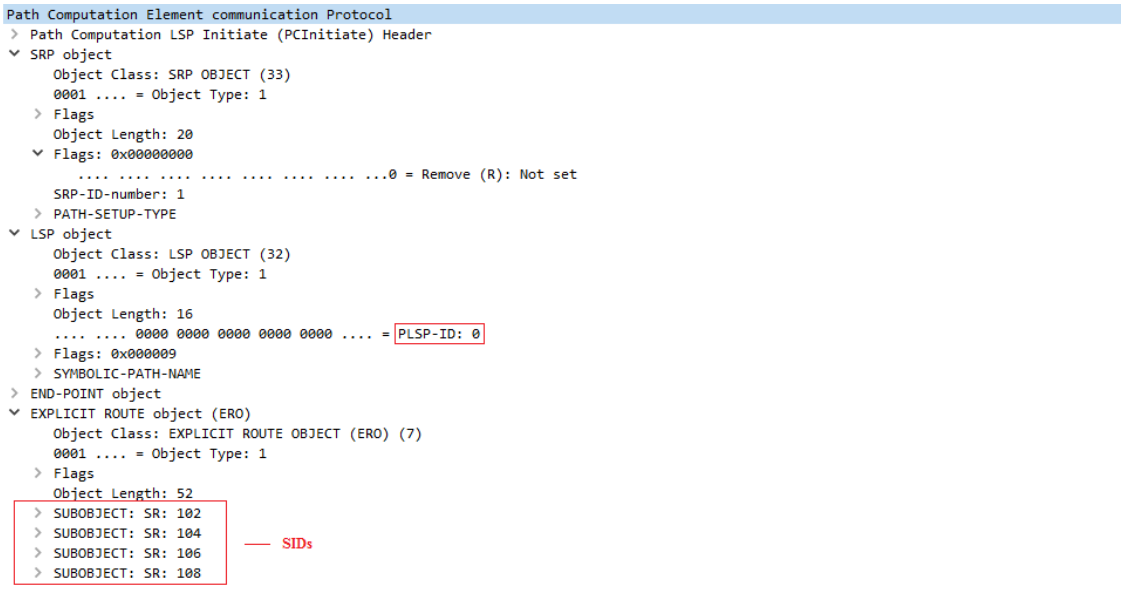
Kun PCC ja PCE ovat saaneet sovittua session parametreista ja muodostaneet PCEP-session, ne kuittaavat OPEN-viestit lähettämällä KEEPALIVE-viestin, joiden lähettäminen jatkuu tämän jälkeen koko PCEP-session ajan [24]. Kuvasta 15 nähdään, että KEEPALIVE-arvoksi on määritelty 30 sekuntia. PCC lähettää 30 sekunnin välein KEEPALIVE viestin PCE:lle tarkastaakseen, että PCEP-sessio on ylhäällä.

PCC synkronoi LSP-polun tai -polkujen tilan PCE:n kanssa PCRpt-viestillä, jonka se lähettää, kun PCEP-sessio on muodostettu. PCC lähettää PCRpt-viestin myös aina, kun uusi LSP-polku lisätään, sen ominaisuuksia päivitetään tai se poistetaan. PCE päivittää nämä tiedot LSP-DB-kantaansa. [26.]

PCE muodostaa uuden LSP:n lähettämällä PCInit-viestin PCC:lle ja sisällyttää viestiin vähintään seuraavat PCEP-objektit:

- LSP
- SRP (Stateful PCE Request Parameters)
- END-POINT
- ERO (Explicit Route Object).

Kuvassa 16 näkyy pakettikaappaus PCInit-viestistä, johon on sisällytetty edellä esitetyn listan objektit.



Kuva 16. PCInit-viesti.

LSP-objektin PLSP-ID:n arvon tulee olla 0, ja Symbolic-Path-Name TLV:n tulee olla määritelty. SRP-objektissa määritellään SRP-ID, jonka arvo yhdistetään kyseiseen PCInit-pyyntöön. END-POINT-objektissa määritellään LSP-polun lähde- ja kohdeosoitteet. ERO-objektissa määritellään LSP-polun reitti, joka sisältää reitittimien SID-tunnisteet. [27.]

Jos PCInit-viestissä olevat parametrit ovat hyväksyttäviä, PCC muodostaa LSP:n ja vastaa PCE:lle PCRpt-viestillä. Mikäli PCInit-viestissä on puutteita tai virheellisiä arvoja, vastaa PCC PCErr-viestillä, johon on liitetty virhettä vastaava tunniste. [27.]

LSP:n poisto tehdään myös PCInit-viestillä, jonka LSP-objektiin on määritelty polun PLSP-ID ja SRP-objektin R-lipun arvoksi määritelty 1, joka viestii PCC:lle, että LSP-polku halutaan poistaa. PCC kuittaa LSP:n poistetuksi PCRpt-viestillä. [27.]

3.3 BGP Link-State -protokollaperhe

BGP Link-State on BGP-protokollaan lisätty laajennus, jota käyttäen voidaan jakaa TE- ja linkkitilatiedot yhdestä tai useista eri IGP-alueista verkon ulkopuoliselle komponentille, kuten keskitetylle PCE:lle tai SDN-kontrollerille käyttäen BGP-protokollaa. BGP-LS

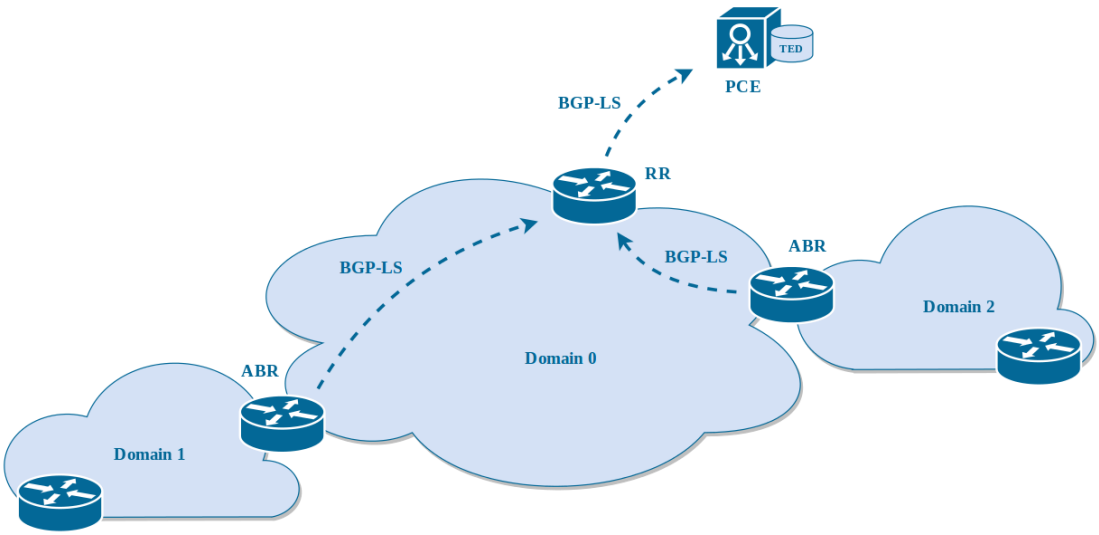
mahdollistaa yhtenäisen topologiakuvauksen muodostamisen koko verkon alueelta. Verkko voi koostua useasta IGP alueesta tai autonomisesta alueesta. [28.]

BGP-LS on uusi BGP-protokollan osoiteperhe (AFI 16388 / SAFI 71), joka on määritelty RFC:ssä RFC7752. Se koostuu kahdesta laajennuksesta, jotka on lisätty BGP-protokollaan:

- BGP-LS NLRI (BGP-LS Network Layer Reachability Information)
- BGP-LS Path Attribute.

BGP-LS määrittelee neljä NLRI-objektia: node, link ja prefiksi (IPv4 ja IPv6). Yhdistämällä tiedot NLRI-objekteista voidaan muodostaa verkon topologia. BGP-LS-attribuuttia käyttäen voidaan jakaa objektien ominaisuuksia, kuten esimerkiksi reitittimen nimi, IGP- ja TE-metriikka tai linkin kaistanleveys. [28.]

Kuvassa 17 on esitetty periaatekuva BGP-LS:n toiminnasta. Kuvan topologiassa on kolme eri toimialuetta, joista keskitetty PCE kerää tiedot TED-kantaansa BGP-LS:n avulla. Toimialueiden reunareitittimet (ABR) muodostavat iBGP-naapuruuden RR (Route Reflector) reitittimen kanssa, jolle ne välittävät tiedot LSDB-kannastaan käyttäen BGP-LS-protokollaa. RR-reititin välittää nämä tiedot edelleen PCE:lle, joka muodostaa verkkotopologian tiedoista, jotka RR on sille välittänyt.



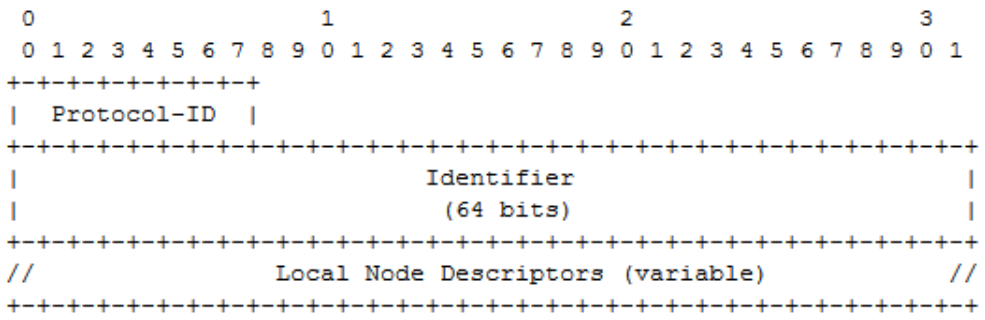
Kuva 17. Topologian muodostus useasta toimialueesta BGP-LS-protokollalla.

3.3.1 BGP-LS NLRI

BGP-LS NLRI sisällytetään BGP UPDATE-viestiin, joka lähetetään, kun BGP-naapuruus muodostetaan reitittimien välille tai verkon topologiassa tapahtuu muutos. BGP-LS määrittelee neljä eri NLRI-tyyppiä:

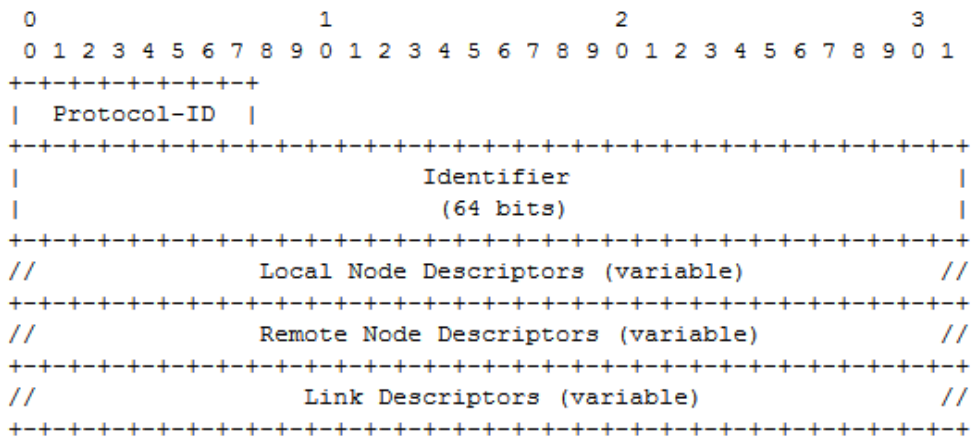
- Node NLRI
- Link NLRI
- IPv4 Topology Prefix NLRI
- IPv6 Topology Prefix NLRI.

Node NLRI sisältää tiedot verkkolaitteesta, tyypillisesti reitittimestä. Local Node Descriptors -kenttä muodostuu useasta TLV-kentästä, joissa on määritelty tiedot, kuten reitittimen AS-numero, BGP-LS-tunniste ja IGP Router-ID [28]. Kuvasta 18 nähdään Node NLRI:n rakenne.



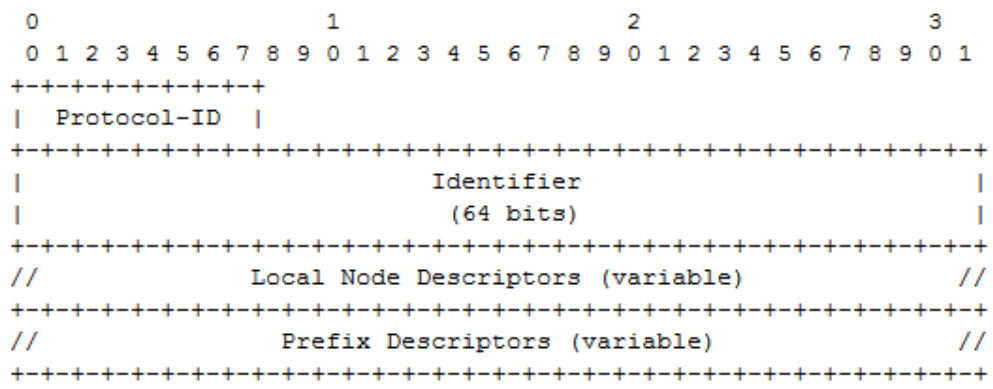
Kuva 18. Node NLRI [28].

Link NLRI edustaa verkon linkkiä. Link NLRI muodostuu Local- ja Remote Node Descriptors -kentistä, jotka sisältävät tiedot linkin molempien päiden reitittimistä ja Link Descriptors -kentästä, joka sisältää linkin molempien päiden IPv4/IPv6-osoitteet. Kuvassa 19 nähdään Link NLRI:n rakenne.



Kuva 19. Link NLRI [28].

Prefix NLRI, sekä IPv4 että IPv6, sisältää Local Node Descriptors -kentän lisäksi Prefix Descriptors -kentän, johon sisällytetään ne IP-prefiksit, joita reititin mainostaa. Kuvasta 20 nähdään Prefix NLRI:n rakenne.



Kuva 20. Prefix NLRI [28].

3.3.2 BGP-LS-attribuutti

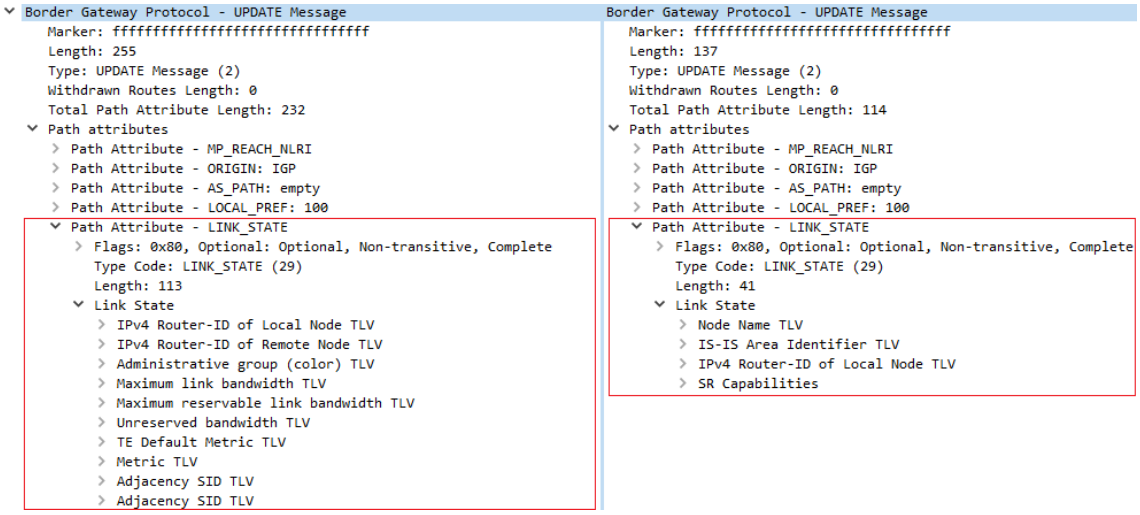
BGP-LS-attribuutti on valinnainen ei-transitiivinen attribuutti, jota ei välttämättä tarvitse sisällyttää BGP UPDATE-viestiin. Sitä käyttäen voidaan jakaa parametreja reitittimistä, linkeistä tai prefikseistä ja käyttää kaikkien BGP-LS NLRI -tyyppien kanssa. [28.]

BGP-LS-attribuutteja ovat

- Node Attribute
- Link Attribute
- Prefix Attribute.

Kukin attribuutti sisältää joukon TLV-triplettejä, joita voidaan käyttää attribuuttia vastaavan NLRI:n kanssa [28]. Kuvassa 21 on esitetty pakettikaappaukset BGP UPDATE -viesteistä, jossa vasemmalla puolella näkyy joukko Link Attribute -TLV-triplettejä ja oikealla puolella joukko Node Attribute -TLV-triplettejä.

BGP-LS-attribuutit voivat kuljettaa lisäksi myös Segment Routing -parametreja, kuten Node-SID tai Adjacency-SID, sekä SRGB-alueen, joka mahdollistaa sen, että keskitetty kontrolleri voi yhdistää segmenttitiedot verkon topologiaan.



Kuva 21. BGP-LS-attribuutit BGP UPDATE -viestissä.

4 Virtuaaliympäristön asennus

4.1 Toteutusympäristö

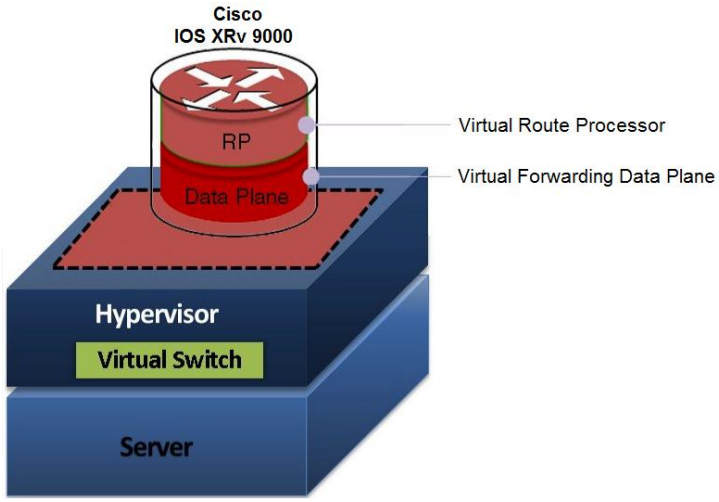
Tässä luvussa käydään läpi, mitä teknologioita ja komponentteja virtuaaliympäristössä käytettiin. Virtuaaliverkko asennettiin VMware Workstationin päälle, johon IOS XRv -virtuaalikoneet laitettiin toimimaan. Verkkoon lisättiin myös OpenDaylight-kontrolleri ja Pathman-SR-sovellus, jota käyttäen LSP-polkuja voitiin lisätä ja muokata verkossa. OpenDaylight-kontrolleri käsiteltiin luvussa 3.2.1, joten sitä ei käsitellä uudelleen.

VMware Workstation

VMware Workstation on virtualisointiohjelmisto, jolla voidaan luoda ja pyörittää useita eri virtuaalikoneita yhden fyysisen koneen päällä. Virtuaalikoneiden käyttöjärjestelmä voi olla eri kuin isäntäkoneessa, ja se tukee useita eri Windows- ja Linux-käyttöjärjestelmiä. VMware WS mahdollistaa virtuaalikoneiden pyörittämisen eristetyssä ympäristössä, josta voidaan sallia pääsy myös samaan fyysiseen verkkoon, kuin missä isäntäkone on. WMware WS sopii erilaisten testiympäristöjen pyörittämiseen. Tässä työssä käytettävä versio on VMware Workstation 12 Pro.

Cisco IOS XRv

Cisco IOS XRv on pilvipohjainen reititin, jonka 64-bittinen IOS XR -ohjelmisto toimii virtuaalikoneessa, jonka laitteisto tukee x86-arkkitehtuuria. Koska IOS XRv pohjautuu IOS XR -ohjelmistoon, siinä on runsaasti samoja reititystoimintoja kuin fyysisissä IOS XR -alustoissa. IOS XRv -reititin yhdistää reititinprosessorin, linjakortin ja virtualisoidun välitystason yhdeksi keskitetyksi välitysinstanssiksi. Kuvassa 22 nähdään IOS XRv -reitittimen looginen arkkitehtuuri. IOS XRv voidaan ottaa käyttöön VMWare- tai KVM-hypervisorien päällä. [29.]



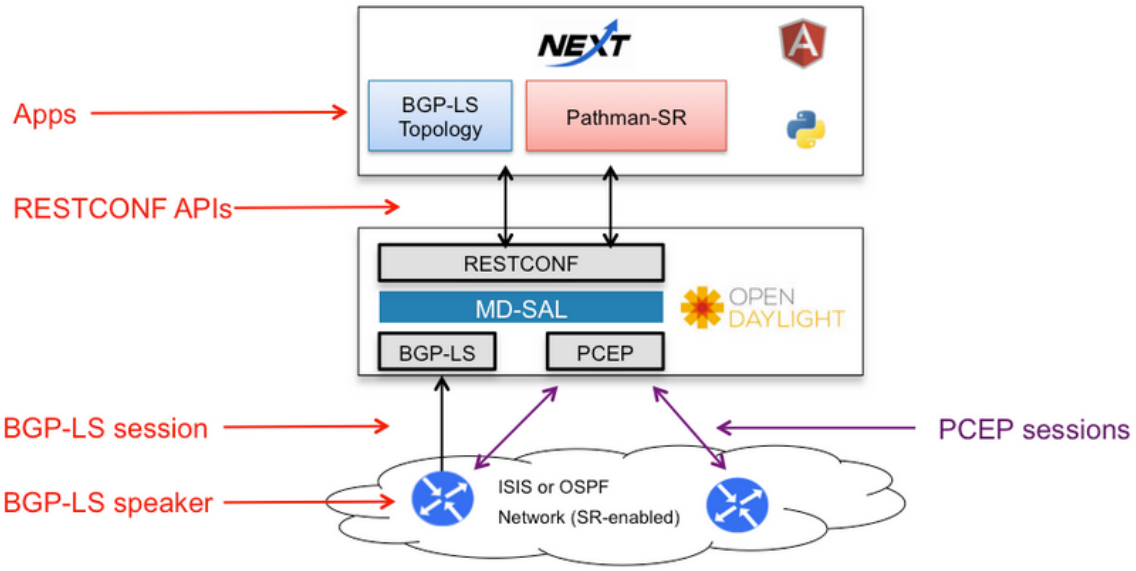
Kuva 22. IOS XRv -arkkitehtuuri [29].

IOS XRv -reitittimestä on saatavilla tuotantoon tarkoitettu versio ja demoversio, joka sisältää suurimmaksi osaksi samat ominaisuudet kuin tuotantoversio, mutta sen kaistankäyttö (rate limit) on rajoitettu arvoon 2 Mbps [30]. Segment Routing -ominaisuus on lisätty IOS XR -ohjelmistoon versiossa 5.2.0 [31].

Pathman-SR

Pathman-SR on avoimen lähdekoodin sovellus, joka on kehitetty toimimaan OpenDaylight-kontrollerin päällä. Sovellusta käyttäen voidaan ohjelmoida kontrolleriin liitettyihin reitittimiin LSP-polkuja. Pathman-SR kerää tiedot kontrolleriin liitetyistä laitteista ja niiden muodostamasta topologiasta suorittamalla RESTCONF-kutsun kontrollerille, joka palauttaa verkon topologian sovellukselle. Kontrolleri muodostaa verkkotopologian BGP-LS-protokollaa käyttäen. Kun uusi LSP-polku halutaan muodostaa verkkoon, lähetetään kontrollerille RESTCONF-käsky, jossa on määritelty polun ominaisuudet, jolloin kontrolleri muodostaa uuden polun halutulle reitittimelle PCEP-protokollaa käyttäen [32].

Pathman-SR voi sijaita samassa palvelimessa kuin OpenDaylight-kontrolleri, mutta sitä voidaan käyttää myös ulkopuolisesta palvelimesta käsin. Kuvassa 23 nähdään Pathman-SR-sovelluksen arkkitehtuuri ja periaate, miten se toimii yhdessä OpenDaylight-kontrollerin kanssa.

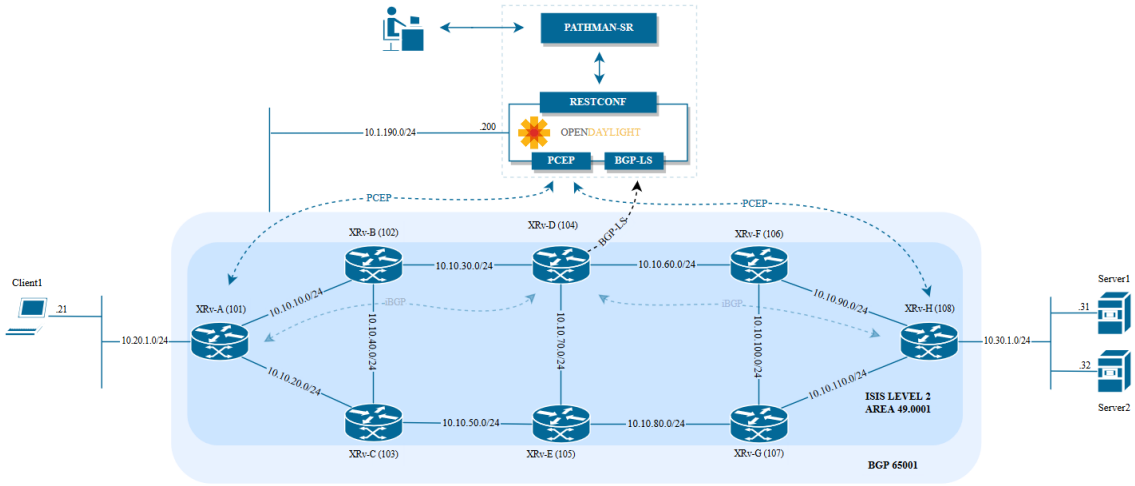


Kuva 23. Pathman-SR-arkkitehtuuri [32].

Pathman-SR on enemmän testitarkoitukseen kehitetty sovellus kuin tuotantoon sellaisenaan kelpaava, eikä se tue kaikkia SR-ominaisuuksia. Esimerkiksi sovellus ei voi ohjelmoida LSP:tä käyttäen Adj-SID-segmenttitunnisteita, vaan se tukee ainoastaan Node-SID-segmenttitunnisteita.

s

Tässä työssä luotiin virtuaaliverkko VMware WS:n päälle, ja siihen lisättiin kahdeksan reititintä, ODL-kontrolleri ja päätelaitteet, joilla mitattiin verkon toimintaa eri tilanteissa. Kuvasta 24 nähdään verkon topologia ensimmäisessä ja toisessa testausvaiheessa. Kolmatta testausta varten topologiaa muutettiin, koska siinä havainnollistettiin kahden autonomisen alueen yli tapahtuvaa SR-reititystä BGP-Prefix-SID-tunnistetta käyttäen. Inter-AS-topologia on liitteessä 2.



Kuva 24. Virtuaaliverkon topologia.

Runkoverkko koostui kahdeksasta reitittimestä, jotka muodostivat yhdessä MPLS-runkoverkon. Runkoverkon IGP-protokollana toimi IS-IS, jota käyttäen reitittimet mainostivat verkkosegmenttinsä ja Node-SID-segmenttitunnisteen, joka liitettiin reitittimen loopback-osoitteen prefiksiin.

Reititin XRv-D muodosti BGP-session ODL-kontrollerin kanssa, jolle se välitti linkkitilatiedot IGP-alueelta käyttäen BGP-LS-osoiteperhettä. Lisäksi se toimi Route Reflector -palvelimena ja muodosti XRv-A- ja XRv-H-reitittimien kanssa iBGP-naapuruuden, joiden IS-IS-instanssin ulkopuolisten verkkosegmenttien prefiksit se välitti näiden reitittimien kesken ipv4 unicast -protokollaperheellä.

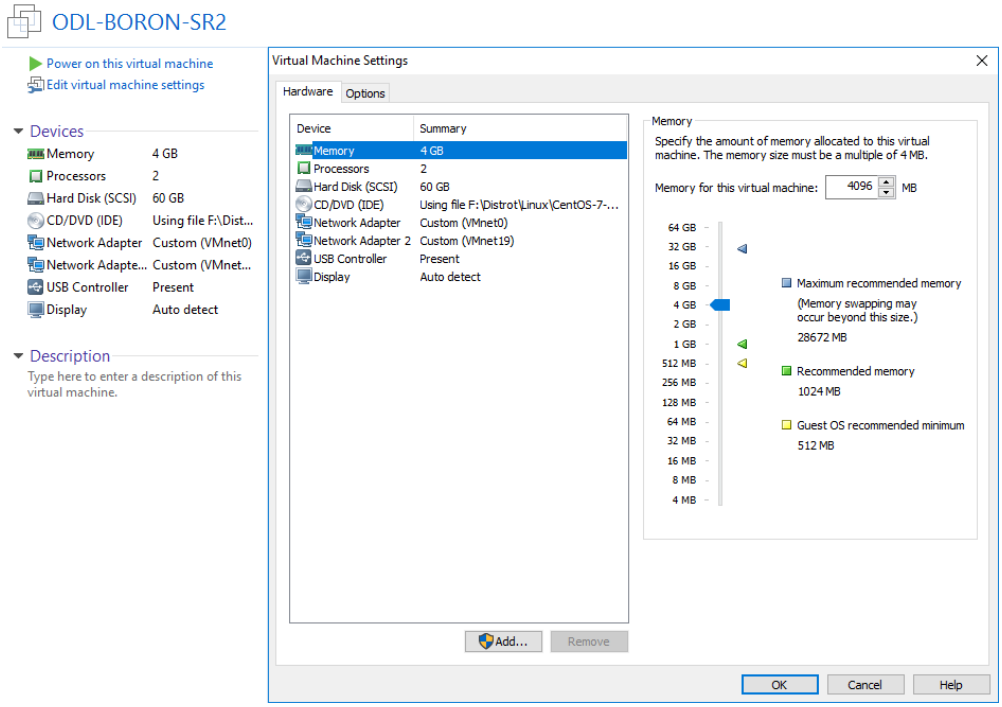
ODL-kontrolleri ja reitittimien hallintaportit liitettiin samaan verkkoon 10.1.190.0/24, jonka kautta kaikki kommunikaatio ODL:n ja reitittimien välillä tapahtui. ODL muodosti PCEP-session reitittimien kanssa ja loi ja päivitti LSP-polkuja PCEP-protokollaa käyttä-

en PCC-laitteille eli reitittimille. Pathman-SR-sovellus asennettiin samaan virtuaalikoneeseen ODL:n kanssa.

4.2 Verkon konfigurointi

4.2.1 OpenDaylight-kontrolleri

ODL-virtuaalikoneelle allokoitiin 4 gigatavua muistia ja 2 virtuaaliprosessoria, jotka riittivät ODL:n toimimiseen testiympäristössä. ODL pystyy periaatteessa toimimaan Linuxin, Macin tai Windowsin päällä, mutta koska Linux on suositeltu käyttöjärjestelmä, asennettiin virtuaalikoneeseen CentOS 7.3-1611 (minimal) -käyttöjärjestelmä, joka pohjautuu Red Hatin 7.3-jakeluversioon. Kuvasta 25 nähdään virtuaalikoneelle allokoidut resurssit.



Kuva 25. ODL-virtuaalikoneelle allokoidut resurssit.

Käyttöjärjestelmän asennuksen jälkeen virtuaalikoneen ensimmäinen verkkoadapteri `ens33` liitettiin NAT-verkkoon (VMnet0), jonka kautta kone sai yhteyden julkiseen verkkoon. Verkkoasetukset adapterille haettiin DHCP:llä:

```
[root@odl ~]# dhclient ens33
```

Toinen verkkoadapteri `ens34` konfiguroitiin staattisesti virtuaalireitittimien hallintaverkkoon (VMnet19):

```
[root@odl ~]# nano /etc/sysconfig/network-scripts/ifcfg-ens34

TYPE=Ethernet
BOOTPROTO=static
IPADDR=10.1.190.200
NETMASK=255.255.255.0
DEFROUTE=yes
PEERDNS=yes
PEERROUTES=yes
IPV4_FAILURE_FATAL=no
NAME=ens34
UUID=f4a17478-5b9c-4e8e-be03-0d39b6976fa0
DEVICE=ens34
ONBOOT=yes
```

ODL vaatii toimiakseen Javan (7 tai 8), joten ennen kontrolleriohjelmiston asennusta virtuaalikoneeseen asennettiin Java SE Development Kit 8u121. Ensin Javan rpm-tiedosto ladattiin Oraclen sivuilta, minkä jälkeen asennus suoritettiin rpm-komennolla:

```
[root@odl opt]# wget --no-cookies --no-check-certificate --
header "Cookie: gpw_e24=http%3A%2F%2Fwww.oracle.com%2F; ora-
clelicense=accept-securebackup-cookie"
"http://download.oracle.com/otn-pub/java/jdk/8u121-
b13/e9e7ea248e2c4826b92b3f075a80e441/jdk-8u121-linux-x64.rpm"

[root@odl opt]# rpm -ivh jdk-8u121-linux-x64.rpm
```

Tämän jälkeen varmistettiin, että asennus oli onnistunut ja koneessa on oikea Java-versio:

```
[root@odl ~]# java -version
java version "1.8.0_121"
Java(TM) SE Runtime Environment (build 1.8.0_121-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.121-b13, mixed mode)
```

Javalle tuli vielä määrittää ympäristömuuttujat, jotta ODL tietää, missä Java sijaitsee:

```
[root@odl ~]# cat /etc/environment
JAVA_HOME="/opt/jdk1.8.0_121"
JRE_HOME="/opt/jdk1.8.0_121/jre"
PATH="/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/opt/jdk1.8.0_121/bin:/opt/jdk1.8.0_121/jre/bin"
```

Seuraavaksi virtuaalikoneeseen asennetaan ODL Boron-SR2 -ohjelmisto, joka ladattiin ODL:n sivuilta [34] ja purettiin `/opt`-kansioon. Tässä työssä käytettiin uusinta ODL-vesiota Boron-SR2:

```
[root@odl opt]# wget
https://nexus.opendaylight.org/content/repositories/opendaylight.release/org/opendaylight/integration/distribution-karaf/0.5.2-Boron-SR2/distribution-karaf-0.5.2-Boron-SR2.tar.gz

[root@odl opt]# tar zxf distribution-karaf-0.5.2-Boron-SR2.tar.gz &&
mv distribution-karaf-0.5.2-Boron-SR2 odl-boron-sr2
```

ODL-ohjelmisto oli pakattu karaf-konttiin (container), jonka sisällä voidaan ottaa käyttöön modulaarisesti eri kontrollerisovelluksia tarpeen mukaan. ODL käynnistettiin suorittamalla `karaf`-komento ODL-kansion sisällä:

```
[root@odl odl-boron-sr2]# ./bin/karaf
```

Tämän jälkeen ODL käynnistyi karaf-konsoliin, josta käsin voitiin lisätä ja poistaa toimintoja sekä tarkkailla kontrollerin lokeja ja statistiikkoja. Tässä työssä tarvittiin seuraavia ominaisuuksia, jotka asennettiin karaf-konsolissa:

- odl-restconf-all
- odl-bgpcep-bgp-all
- odl-bgpcep-pcep-all
- odl-netconf-connector-all

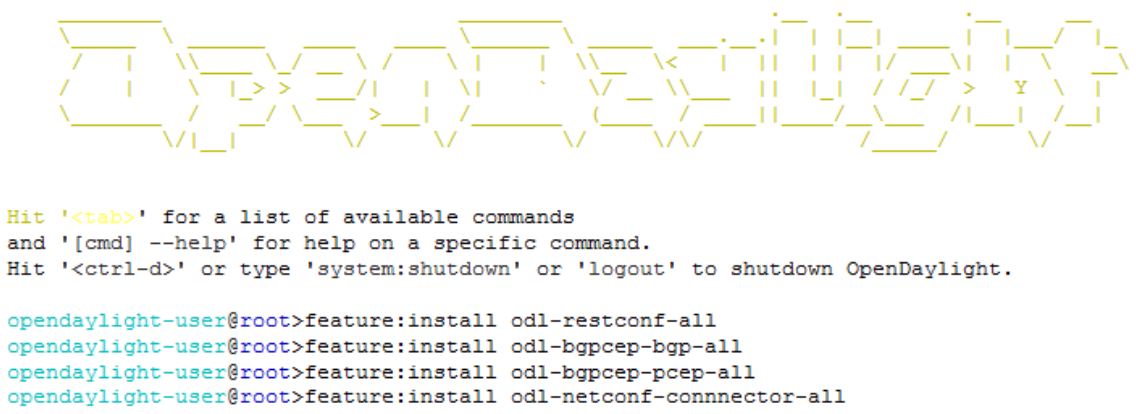
Odl-restconf-all-ominaisuuden asennus loi REST API:n ODL:n Northbound-rajapintaan, jota käyttäen sovellukset voivat tehdä OPTIONS-, GET-, PUT-, POST-, DELETE- ja PATCH-operaatioita käyttäen HTTP-protokollaa.

Odl-bgpcep-bgp-all lisäsi tuen BGP-protokollalle Southbound-rajapinnassa, sisältäen tuen BGP-LS:lle, jota käyttäen ODL vastaanotti linkkitilatiedot siihen kytketystä verkosta.

Odl-bgpcep-pcep-all mahdollisti PCEP-operaatioiden suorittamisen verkkolaitteille Southbound-rajapinnan kautta.

Odl-netconf-connector-all-asennus lisäsi tuen NETCONF-laitteiden hallintaan SSH-protokollalla. Tätä ominaisuutta tarvittiin noutamaan Node-SID-tunniste reitittimeltä.

Ominaisuuksien asennus karaf-konsolissa tehtiin suorittamalla `feature:install odl-<feature>` -komento, kuten kuvasta 26 nähdään.



Kuva 26. ODL-ominaisuuksien asennus.

Jotta asennetut ominaisuudet saatiin toimimaan, tuli ODL käynnistää uudelleen. Lisäksi BGP-konfiguraatiotiedostoa täytyi muokata, jotta kontrolleri muodosti BGP-naapuruuden halutun reitittimen kanssa:

```
[root@odl odl-boron-sr2]# nano etc/opendaylight/karaf/41-bgp-
example.xml
```

Kuvasta 27 nähdään BGP-konfiguraatiotiedoston osia, joita muokattiin. Ylimmässä osassa määriteltiin missä portissa ODL kuuntelee BGP-viestejä. RFC:ssä RFC4271

[33] on määritelty, että BGP-protokolla kuuntelee TCP-porttia 179, joka myös asetettiin ODL:n konfiguraatioon. Keskimmäisessä osassa määriteltiin sen reitittimen osoite, jonka kanssa ODL muodosti naapuruuden ja valittiin BGP-session tyyppi, tässä tapauksessa iBGP, koska ODL ja reititin toimivat saman autonomisen alueen sisällä. Alimmassa osassa määriteltiin ODL:n osoite sekä AS-numero.

```
- - - Snippet - - -
<!--Default binding-port -->
<binding-port>179</binding-port>

- - - Snippet - - -
<module>
    <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">prefix:bgp-peer</type>
    <name>example-bgp-peer</name>
    <host>10.1.190.104</host>
    <holdtimer>180</holdtimer>
    <retrytimer>10</retrytimer>
    <peer-role>ibgp</peer-role>
    ...
    ...
</module>

- - - Snippet - - -
<module>
    <type xmlns:prefix="urn:opendaylight:params:xml:ns:yang:controller:bgp:rib:impl">prefix:rib-impl</type>
    <name>example-bgp-rib</name>
    <rib-id>example-bgp-rib</rib-id>
    <local-as>65001</local-as>
    <bgp-rib-id>10.1.190.200</bgp-rib-id>
    ...
    ...
</module>
```

Kuva 27. BGP-konfiguraatiotiedosto.

ODL:n käynnistyttyä varmistettiin, että tarvittavat sovellukset ovat käynnissä `netstat`-komennolla, kuten kuvasta 28 nähdään.

```
[root@odl ~]# netstat -plunt | grep java
tcp6       0      0 :::1099                 :::*                    LISTEN      6735/java
tcp6       0      0 127.0.0.1:46251         :::*                    LISTEN      6735/java
tcp6       0      0 :::8080                 :::*                    LISTEN      6735/java
tcp6       0      0 :::179 —BGP             :::*                    LISTEN      6735/java
tcp6       0      0 :::8181—RESTCONF        :::*                    LISTEN      6735/java
tcp6       0      0 127.0.0.1:2550          :::*                    LISTEN      6735/java
tcp6       0      0 :::8185                 :::*                    LISTEN      6735/java
tcp6       0      0 :::44444                :::*                    LISTEN      6735/java
tcp6       0      0 :::4189—PCEP            :::*                    LISTEN      6735/java
tcp6       0      0 127.0.0.1:8383          :::*                    LISTEN      6735/java
tcp6       0      0 :::8101                 :::*                    LISTEN      6735/java
tcp6       0      0 :::1830                 :::*                    LISTEN      6735/java
tcp6       0      0 :::38310                :::*                    LISTEN      6735/java
```

Kuva 28. ODL-prosessin (6735) kuunneltavat portit.

Tässä vaiheessa ODL-ohjelmisto oli asennettu valmiiksi. Samaan virtuaalikoneeseen laitettiin toimimaan Pathman-SR-sovellus, joka ladattiin GitHubista:

```
[root@odl ~]# git clone https://github.com/CiscoDevNet/pathman-sr.git
```

Pathman-SR-sovellus tarvitsee toimiakseen kaksi Python-moduulia, jotka asennettiin ennen sovelluksen käynnistystä `pip`-työkalulla:

```
[root@odl opt]# pip install tornado
[root@odl opt]# pip install requests
```

Lisäksi sovelluksen `pathman_ini.py`-tiedostoon määriteltiin, missä osoitteessa ODL toimii, tässä tapauksessa `127.0.0.1`, koska ODL oli asennettu samaan koneeseen. Tiedostossa määriteltiin myös, mihin porttiin sovellus tekee RESTCONF-kutsuja sekä ODL:n käyttäjätunnus ja salasana:

```
[root@odl pathman-sr]# nano pathman_ini.py

odl_ip = '127.0.0.1'
odl_port = '8181'
log_file = '/tmp/pathman.log'
log_size = 2000000
log_count = 3
log_level = 'INFO'
odl_user = '<user>'
odl_password = '<password>'
```

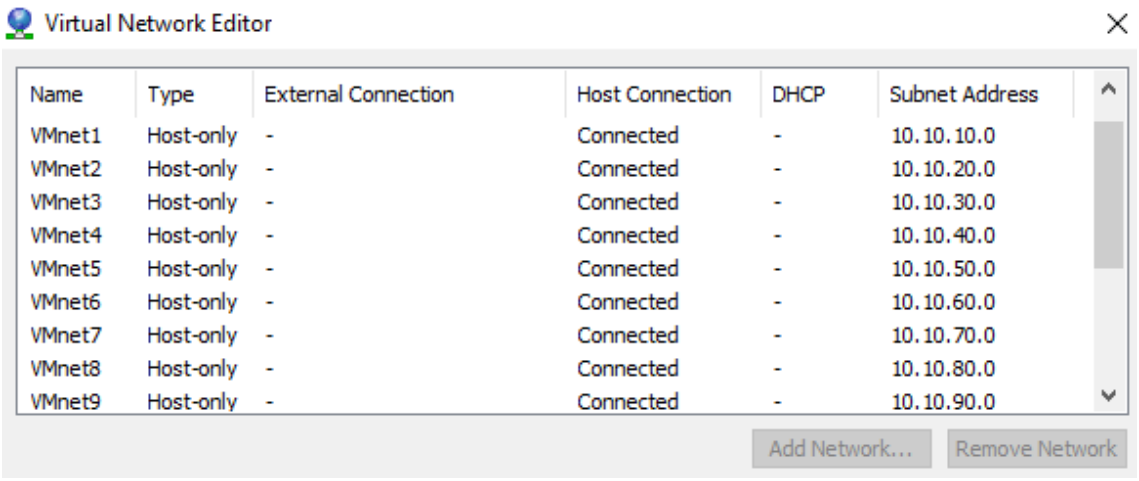
ODL-virtuaalikone oli tämän jälkeen valmis käytettäväksi ja kontrolleriohjelmisto ja Pathman-SR-sovellus voitiin käynnistää.

4.2.2 Reitittimien konfigurointi

IOS XRv -reitittimien virtuaalikone ladattiin Ciscon portaalista [35]. Segment Routing -ominaisuus on lisätty IOS XRv -reitittimiin versiossa 5.2.0, mutta tässä työssä käytettiin uusinta saatavilla olevaa versiota 6.1.2. IOS XRv -virtuaalikone oli valmiiksi asennettu, joten se oli valmis käyttöön sellaisenaan.

IOS XRv -virtuaalikone avattiin VMware Workstationissa ja kloonattiin VMware WS:n kloonityökalulla kahdeksaksi kappaleeksi kuvan 24 topologian mukaisesti. IOS XRv -virtuaalikoneille allokoitiin 1 virtuaaliprosessori ja 3 gigatavua muistia, mikä on minimivaatimus demoversion toimimiseen [34]. Lisäksi reitittimille lisättiin virtuaalinen sarja-adapteri, jonka kautta saatiin yhteys laitteen CLI-hallintaan. Reitittimien verkkoadapterit

asetettiin verkkotopologian mukaisiin verkkosegmentteihin, jotka määriteltiin VMware Workstationin `virtual network editor` -työkalulla, joka nähdään kuvassa 29.



Kuva 29. Virtual Network Editor -työkalu.

IOS XRv -virtuaalikoneen konfiguraatiotiedostoon määriteltiin, missä portissa VMware WS toimii `telnet`-palvelimena, jonka kautta yhteys virtuaalikoneen sarja-adapteriin ohjattiin. Jokaisen IOS XRv -virtuaalikoneen konfigurointitiedostoon lisättiin seuraavat asetukset, joista esimerkkinä XRv-A-reitittimen asetukset:

```
serial0.present = "TRUE"
serial0.yieldOnMsrRead = "TRUE"
serial0.fileType = "network"
serial0.fileName = telnet://127.0.0.1:6101
```

IOS XRv -virtuaalikoneet käynnistettiin ja isäntäkoneen komentoriviltä varmistettiin, että VMware WS toimii telnet-palvelimena jokaisen reitittimen osalta seuraavalla komennolla:

```
PS C:\Users\xxxx> netstat -a | sls LIST | sls 61
TCP    127.0.0.1:6101         DESKTOP-HC2LRKL:0      LISTENING
TCP    127.0.0.1:6102         DESKTOP-HC2LRKL:0      LISTENING
TCP    127.0.0.1:6103         DESKTOP-HC2LRKL:0      LISTENING
TCP    127.0.0.1:6104         DESKTOP-HC2LRKL:0      LISTENING
TCP    127.0.0.1:6105         DESKTOP-HC2LRKL:0      LISTENING
TCP    127.0.0.1:6106         DESKTOP-HC2LRKL:0      LISTENING
TCP    127.0.0.1:6107         DESKTOP-HC2LRKL:0      LISTENING
TCP    127.0.0.1:6108         DESKTOP-HC2LRKL:0      LISTENING
```

Reitittimien hallintakonsoleihin päästiin ottamalla `telnet`-yhteys isäntäkoneen yllä listattuihin portteihin komentorivisovelluksella.

Reitittimien konfigurointi jaettiin seuraaviin osiin, jotka suoritettiin jokaisella reitittimellä, sillä poikkeuksella, että BGP-konfiguroitiin vain A-, D- ja H-reitittimiin:

- porttikohtaiset asetukset
- IGP-protokollan konfigurointi (IS-IS)
- MPLS TE -kohtaiset asetukset
- BGP-protokollan konfigurointi.

Ensimmäinen vaihe reitittimien konfiguroinnissa oli porttikohtaisten määritysten konfiguroiminen. Reitittimien portteihin määriteltiin niiden IP-osoitteet ja porttikuvaukset. Liitteessä 3 on listattu reitittimiin konfiguroidut IP-osoitteet.

Toisessa vaiheessa reitittimiin konfiguroitiin IS-IS-instanssi, jossa myös määriteltiin Node-SID-segmenttitunniste, joka liitettiin reitittimien loopback-osoitteeseen. Kuvassa 30 on XRv-A-reitittimen IS-IS-instanssin konfiguraatio. XRv-A:n konfiguraatio nähdään kokonaisuudessaan liitteessä 4.

```
1   router isis srlab
2    is-type level-2-only
3    net 49.0001.0100.0000.0101.00
4    address-family ipv4 unicast
5     metric-style wide
6     mpls traffic-eng level-2-only
7     mpls traffic-eng router-id Loopback0
8     segment-routing mpls
9    !
10   interface Loopback0
11    address-family ipv4 unicast
12     prefix-sid index 101
13    !
14   !
15   interface GigabitEthernet0/0/0/0
16    point-to-point
17    address-family ipv4 unicast
18     metric 10
19    !
20   !
21   interface GigabitEthernet0/0/0/1
22    point-to-point
23    address-family ipv4 unicast
24     metric 10
```

Kuva 30. IS-IS-instanssin konfiguraatio.

Kuvassa 30 rivillä 2 määriteltiin IS-IS-instanssi osallistumaan vain Level 2 -tason reititykseen ja muodostamaan IS-IS-naapuruuden vain toisen Level 2- tai Level 1/2 -reitittimien kanssa.

Rivillä 3 määriteltiin reitittimen NET (Network Entity Title) -osoite. Numero 49 määritti, että osoite on privaattiluokan osoite, ja yhdessä seuraavan 0001 kanssa ne määrittivät IS-IS-alueen. Kolme seuraavaa, 0100.0000.0101, yksilöi reitittimen ja kaksi viimeistä lukua on aina 00 NET-osoitteessa.

Riveillä 5-7 määriteltiin reititin generoimaan ja hyväksymään vain uudenlaisia TLV-objekteja sekä määriteltiin MPLS TE toimimaan vain Level 2 -tasolla ja käyttämään loopback-osoitetta reitittimen tunnisteena.

Rivillä 8 Segment Routing ominaisuus otettiin käyttöön IS-IS-instanssissa käyttäen MPLS-välitystasoa, ja Node-SID-segmenttitunnisteen indeksiarvo määriteltiin rivillä 12.

Kuvan 30 riveillä 15-24 määriteltiin ne portit, jotka osallistuivat IS-IS instanssiin ja määriteltiin niille metriikat.

SRGB olisi voitu konfiguroida IS-IS-instanssin sisällä, mutta koska sitä käytettiin myös BGP-instanssissa, se konfiguroitiin globaalissa moodissa, jolloin se periytyi kaikkiin instansseihin, joissa Segment Routing on käytössä. Kuvassa 31 nähdään SRGB:n konfiguraatio.

```
1   segment-routing
2    global-block 16000 16199
```

Kuva 31. SRGB:n globaali konfiguraatio.

Tämän lisäksi reitittimellä XRv-D täytyi määritellä `distribute bgp-ls` -parametri, jotta reititin kopioi linkkitilatiedot IS-IS-topologiasta BGP-LS-instanssiin. Tämä parametri konfiguroitiin IS-IS-instanssin sisälle, kuvan 30 rivin 3 jälkeen.

Kun IS-IS-naapuruudet oli muodostettu ja topologia oli valmis, voitiin segmenttitunnisteiden mainostus tarkistaa `show mpls forwarding` –komennolla, kuten kuvasta 32 nähdään. Globaalien segmenttien lisäksi reititin allokoi jokaiselle linkilleen lokaalin segmentin, jotka näkyvät kuvassa `SR Adj` –nimellä. Reitittimeltä XRv-A löytyi kaksi

ECMP-polkua kohti XRv-H-reititintä, joten kuvassa 32 näkyy kaksi polkua segmenttitunnisteelle 16108.

```
RP/0/0/CPU0:xrva#sh mpls forwarding
Mon Feb  6 23:33:04.479 UTC
Local  Outgoing    Prefix             Outgoing     Next Hop        Bytes
Label  Label       or ID              Interface                    Switched
------ ----------- ------------------ ------------ --------------- ------------
16102  Pop         SR Pfx (idx 102)   Gi0/0/0/0    10.10.10.102    0
16103  Pop         SR Pfx (idx 103)   Gi0/0/0/1    10.10.20.103    0
16104  16104       SR Pfx (idx 104)   Gi0/0/0/0    10.10.10.102    703
16105  16105       SR Pfx (idx 105)   Gi0/0/0/1    10.10.20.103    0
16106  16106       SR Pfx (idx 106)   Gi0/0/0/0    10.10.10.102    0
16107  16107       SR Pfx (idx 107)   Gi0/0/0/1    10.10.20.103    0
16108  16108       SR Pfx (idx 108)   Gi0/0/0/0    10.10.10.102    0
       16108       SR Pfx (idx 108)   Gi0/0/0/1    10.10.20.103    0
24000  Pop         SR Adj (idx 1)     Gi0/0/0/0    10.10.10.102    0
24001  Pop         SR Adj (idx 3)     Gi0/0/0/0    10.10.10.102    0
24002  Pop         SR Adj (idx 1)     Gi0/0/0/1    10.10.20.103    0
24003  Pop         SR Adj (idx 3)     Gi0/0/0/1    10.10.20.103    0
```

Kuva 32. Segmenttitunnisteiden mainostus.

Kolmannessa vaiheessa reitittimiin konfiguroitiin MPLS TE -ominaisuus ja asetettiin PCE-määritykset. Kuvan 33 rivillä 3 määriteltiin, mistä osoitteesta löytyy PCE, tässä tapauksessa ODL:n osoite. Riveillä 5-7 määriteltiin SR-parametrit ja annettiin PCE:lle valtuudet tehdä aloite LSP:n asettamisesta reitittimeen rivin 7 `instantiation`-parametrilla. Riveillä 10-11 määriteltiin, minkä arvon PCE:n luomat LSP-tunnelit saavat. Rivillä 13 määriteltiin, kuinka kauan odotetaan LSP:n päivittämisen jälkeen, jotta muutokset tulevat voimaan.

```
1   mpls traffic-eng
2    pce
3     peer ipv4 10.1.190.200
4     !
5     segment-routing
6     stateful-client
7      instantiation
8     !
9    !
10   auto-tunnel pcc
11    tunnel-id min 1 max 199
12   !
13   reoptimize timers delay installation 0
```

Kuva 33. MPLS TE:n konfiguraatio.

Viimeisessä vaiheessa määriteltiin BGP-parametrit reitittimille A, D ja H. Reititin D toimi Route Reflector -palvelimena, johon reitittimet A ja H sekä ODL muodostivat naapuruuden. Lisäksi reitittimen D ja ODL:n välissä otettiin käyttöön BGP-LS, jotta ODL saa

kerättyä linkkitilatiedot IGP-alueesta. Kuvassa 34 näkyy XRv-D-reitittimen BGP-konfiguraatio.

```
1   router bgp 65001
2    bgp router-id 10.0.0.104
3    address-family ipv4 unicast
4    !
5    address-family link-state link-state
6    !
7    neighbor 10.0.0.101
8     remote-as 65001
9     update-source Loopback0
10    description iBGP to xrva
11    address-family ipv4 unicast
12     route-reflector-client
13    !
14   !
15   neighbor 10.0.0.108
16    remote-as 65001
17    update-source Loopback0
18    description iBGP to xrvh
19    address-family ipv4 unicast
20     route-reflector-client
21    !
22   !
23   neighbor 10.1.190.200
24    remote-as 65001
25    update-source MgmtEth0/0/CPU0/0
26    description BGP-LS to ODL
27    address-family ipv4 unicast
28     route-reflector-client
29    !
30    address-family link-state link-state
31     route-reflector-client
```

Kuva 34. Reitittimen XRv-D BGP-konfiguraatio.

Kaikki reitittimet toimivat saman autonomisen alueen (65001) sisällä, joten ne muodostivat iBGP-naapuruuden. Rivillä 5 otettiin käyttöön BGP-LS-osoiteperhe ja rivillä 30-31 määriteltiin ODL Route Reflector -asiakkaaksi BGP-LS-osoiteperheen osalta. Riveillä 7-20 määriteltiin A- ja H-reitittimet ipv4 unicast- osoiteperheen RR-asiakkaiksi.

Kuvassa 35 nähdään, miten reititin XRv-D mainostaa LSDB-kannan tiedot ODL-kontrollerille BGP-LS-instanssissa.

```
RP/0/0/CPU0:xrvd#sh bgp link-state link-state
Mon Feb  6 23:15:49.631 UTC
BGP router identifier 10.0.0.104, local AS number 65001
BGP generic scan interval 60 secs
Non-stop routing is enabled
BGP table state: Active
Table ID: 0x0   RD version: 61
BGP main routing table version 61
BGP NSR Initial initsync version 61 (Reached)
BGP NSR/ISSU Sync-Group versions 0/0
BGP scan interval 60 secs

Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
Prefix codes: E link, V node, T IP reacheable route, u/U unknown
              I Identifier, N local node, R remote node, L link, P prefix
              L1/L2 ISIS level-1/level-2, O OSPF, D direct, S static/peer-node
              a area-ID, l link-ID, t topology-ID, s ISO-ID,
              c confed-ID/ASN, b bgp-identifier, r router-ID,
              i if-address, n nbr-address, o OSPF Route-type, p IP-prefix
              d designated router address
   Network          Next Hop            Metric LocPrf Weight Path
*> [V][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0101.00]]/328    0.0.0.0      0 i
*> [V][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0102.00]]/328    0.0.0.0      0 i
*> [V][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0103.00]]/328    0.0.0.0      0 i
*> [V][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0104.00]]/328    0.0.0.0      0 i
*> [V][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0105.00]]/328    0.0.0.0      0 i
*> [V][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0106.00]]/328    0.0.0.0      0 i
*> [V][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0107.00]]/328    0.0.0.0      0 i
*> [V][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0108.00]]/328    0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0101.00]][R[c65001][b0.0.0.0][s0100.0000.0102.00]][L[i10.10.10.101][n10.10.10.102]]/696 0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0101.00]][R[c65001][b0.0.0.0][s0100.0000.0103.00]][L[i10.10.20.101][n10.10.20.103]]/696 0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0102.00]][R[c65001][b0.0.0.0][s0100.0000.0101.00]][L[i10.10.10.102][n10.10.10.101]]/696 0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0102.00]][R[c65001][b0.0.0.0][s0100.0000.0103.00]][L[i10.10.40.102][n10.10.40.103]]/696 0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0102.00]][R[c65001][b0.0.0.0][s0100.0000.0104.00]][L[i10.10.30.102][n10.10.30.104]]/696 0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0103.00]][R[c65001][b0.0.0.0][s0100.0000.0101.00]][L[i10.10.20.103][n10.10.20.101]]/696 0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0103.00]][R[c65001][b0.0.0.0][s0100.0000.0102.00]][L[i10.10.40.103][n10.10.40.102]]/696 0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0103.00]][R[c65001][b0.0.0.0][s0100.0000.0105.00]][L[i10.10.50.103][n10.10.50.105]]/696 0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0104.00]][R[c65001][b0.0.0.0][s0100.0000.0102.00]][L[i10.10.30.104][n10.10.30.102]]/696 0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0104.00]][R[c65001][b0.0.0.0][s0100.0000.0105.00]][L[i10.10.70.104][n10.10.70.105]]/696 0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0104.00]][R[c65001][b0.0.0.0][s0100.0000.0106.00]][L[i10.10.60.104][n10.10.60.106]]/696 0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0105.00]][R[c65001][b0.0.0.0][s0100.0000.0103.00]][L[i10.10.50.105][n10.10.50.103]]/696 0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0105.00]][R[c65001][b0.0.0.0][s0100.0000.0104.00]][L[i10.10.70.105][n10.10.70.104]]/696 0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0105.00]][R[c65001][b0.0.0.0][s0100.0000.0107.00]][L[i10.10.80.105][n10.10.80.107]]/696 0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0106.00]][R[c65001][b0.0.0.0][s0100.0000.0104.00]][L[i10.10.60.106][n10.10.60.104]]/696 0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0106.00]][R[c65001][b0.0.0.0][s0100.0000.0107.00]][L[i10.10.100.106][n10.10.100.107]]/696   0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0106.00]][R[c65001][b0.0.0.0][s0100.0000.0108.00]][L[i10.10.90.106][n10.10.90.108]]/696 0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0107.00]][R[c65001][b0.0.0.0][s0100.0000.0105.00]][L[i10.10.80.107][n10.10.80.105]]/696 0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0107.00]][R[c65001][b0.0.0.0][s0100.0000.0106.00]][L[i10.10.100.107][n10.10.100.106]]/696   0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0107.00]][R[c65001][b0.0.0.0][s0100.0000.0108.00]][L[i10.10.110.107][n10.10.110.108]]/696   0.0.0.0      0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0108.00]][R[c65001][b0.0.0.0][s0100.0000.0106.00]][L[i10.10.90.108][n10.10.90.106]]/696 0.0.0.0 0 i
*> [E][L2][I0x0][N[c65001][b0.0.0.0][s0100.0000.0108.00]][R[c65001][b0.0.0.0][s0100.0000.0107.00]][L[i10.10.110.108][n10.10.110.107]]/696   0.0.0.0      0 i
```

NODES

LINKS

Kuva 35. LSDB:n tiedot BGP-LS-instanssissa.

4.2.3 Päätelaitteet

Verkkotopologiaan lisättiin kolme päätelaitetta, joilla mitattiin verkon liikennettä ja testattiin SR-polkuja. Yksi asiakastyöasema sijoitettiin XRv-A-reitittimen takana olevaan verkkosegmenttiin, jonka verkko-osoite on 10.20.1.0/24, ja kaksi palvelinta lisättiin XRv-H-reitittimen takana olevaan verkkoon 10.30.1.0/24.

Päätelaitteissa oli Ubuntu Server 16.04.01 LTS -käyttöjärjestelmä, johon asennettiin erilaisia verkkotyökaluja kaistanleveyden, viiveen ja verkkopolkujen testaamiseen. Näitä työkaluja olivat:

- Iperf3 – kaistanleveyden mittaus
- Curl – HTTP GET pyynnön kesto palvelimelta
- Traceroute – verkkopolun tarkastaminen

Testeistä saadut tulokset analysoitiin Jupyter Notebook -ohjelmalla, johon asennetiin Plotly-lisäosa, jolla luotiin graafeja saaduista tuloksista.

5 Segment Routing -käyttötapojen testaus

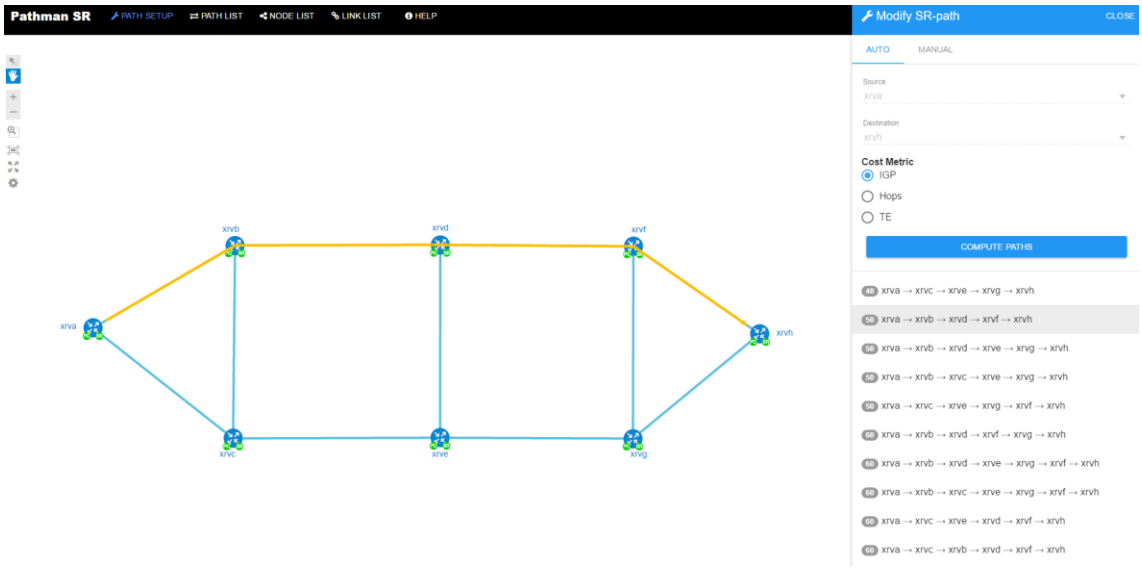
5.1 Kaistanleveyden optimointi

Kaistanleveyden optimoinnissa testattiin, miten SR-reititystä käyttämällä voidaan jakaa verkon kuormitusta kaikkien polkujen kesken, vaikka ne eivät olisikaan SPF-algoritmin laskema lyhin reitti kohteeseen. Verkossa voi helposti syntyä tilanne, että lyhimmälle reitille keskittyy hetkellisesti tavallista suurempi kuormitus, jolloin se voi nousta liian suureksi ja lyhintä polkua käyttävät sovellukset toimivat siksi hitaammin.

Tässä testauksessa käytettiin Iperf3-työkalua, jolla mitattiin, millä kapasiteetilla liikennettä voidaan toimittaa verkon läpi. Kuvasta 24 ja liitteestä 1 nähdään verkon topologia ja verkkoon lisätyt pääteasemat, joilla mitattiin verkon kapasiteettia.

Verkon topologiaan tehtiin muutos reitittimien XRv-B ja XRv-D välisen linkin metriikan osalta, joka nostettiin 10:stä 20:een, jotta reitittimeltä XRv-A olisi vain yksi lyhin polku reitittimelle XRv-H, jota pitkin liikenne ohjataan. Koska IOS XRv -reitittimet tukevat maksimissaan 2 Mbps, täytyi linkkien kaistanleveys laskea 1 Mbps, jotta reitittimien lisenssirajoitus ei vaikuta testaustilanteeseen.

Kuvassa 36 nähdään Pathman-SR-sovelluksen ikkuna, jossa näkyy verkon topologia ja oikeassa reunassa polkujen hallintapaneeli. Sovelluksessa voitiin graafisesti määrittää, minkä reitittimien kautta määrätty liikenne haluttiin reitittää. Kuvassa 36 Pathman-SR-sovellus on laskenut kaikki mahdolliset polut reitittimien XRv-A ja XRv-H välillä, ja niiden sivussa näkyy kullekin polulle yhteenlaskettu metriikka. Keltainen viiva näyttää, mikä laskettu polku valittiin vietäväksi reitittimeen XRv-A.

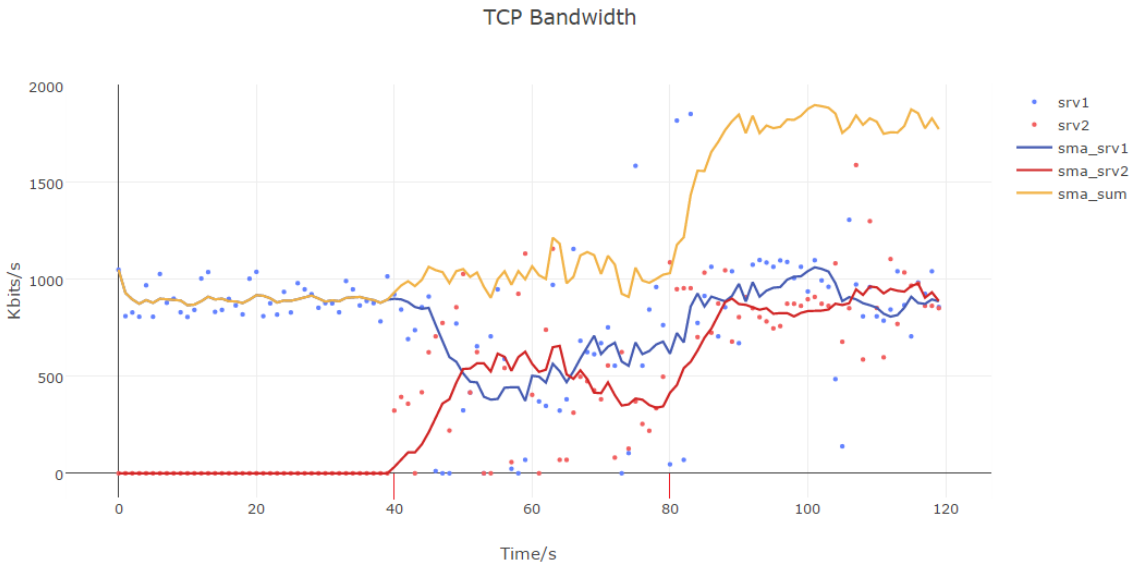


Kuva 36. LSP-polun lisäys Pathman-SR-sovelluksella.

Testaus suoritettiin kolmivaiheisena, ja se kesti 120 sekuntia. Ensimmäiset 40 sekuntia asiakaspääteasema lähetti liikennettä vain palvelimelle 1, minkä jälkeen se alkoi lähettää liikennettä myös palvelimelle 2. Toinen vaihe kesti myös 40 sekuntia, minkä jälkeen reitittimelle XRv-A lisättiin uusi polku, joka kulki suuremman metriikan omaavaa polkua pitkin. Kolmas vaihe kesti 40 sekuntia, jonka jälkeen mittaus päättyi. Sama testi suoritettiin sekä TCP- että UDP-liikenteelle.

Testin alussa liikenne kulki lyhintä polkua pitkin reitittimien XRv-A → XRv-C → XRv-E → XRv-G → XRv-H kautta, jolloin kokonaiskapasiteettia oli käytössä 1 Mbps. 40 sekunnin jälkeen myös toiselle palvelimelle lähetettiin liikennettä, joten kaksi liikennevuota jakoi saman 1 Mbps -kapasiteetin. Kolmannessa vaiheessa palvelimelle 1 lähetettävä liikenne ohjattiin suuremman metriikan reitille, jolloin kokonaiskapasiteetti kasvoi lähelle 2 Mbps. Vaihtoehtoinen reitti kulki reitittimien XRv-A → XRv-B → XRv-D → XRv-F → XRv-H kautta.

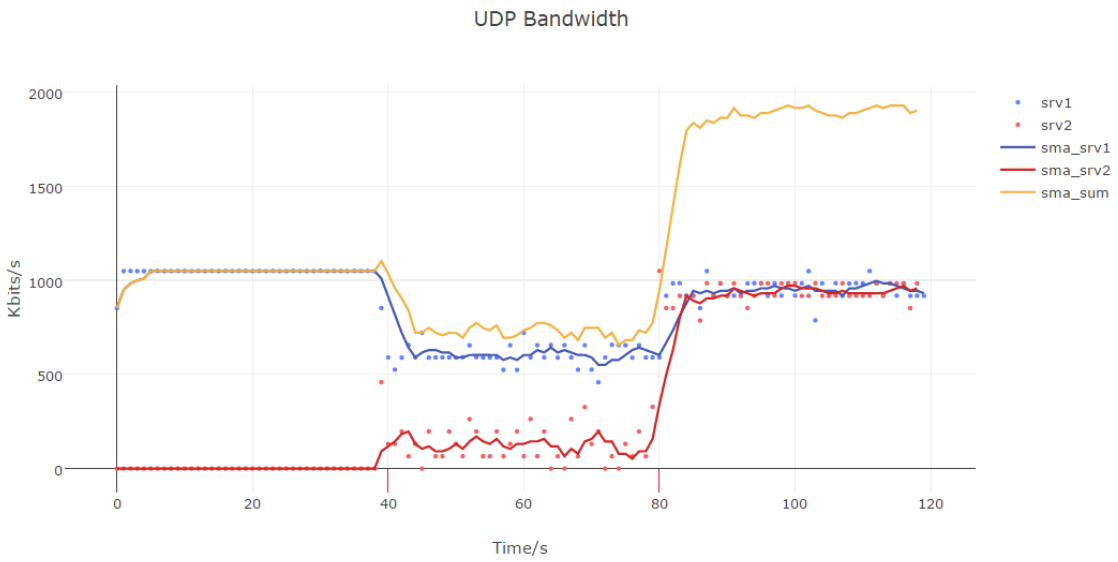
Kuvassa 37 nähdään tulokset TCP-liikenteen kapasiteetille eri vaiheissa. Todelliset mitatut arvot näkyvät graafissa pisteinä, ja niistä on johdettu liukuva keskiarvo palvelimille 1 ja 2 sekä kokonaiskapasiteetti, joka näkyy graafissa oranssina viivana.



Kuva 37. TCP-testin tulokset.

Kuvan 37 graafista nähdään, että kokonaiskapasiteetti pysyi samana ensimmäisen 80 sekunnin ajan, mutta palvelinkohtainen kapasiteetti laski keskijaksolla, koska kahteen palvelimeen kohdistuva liikenne jakoi saman kokonaiskapasiteetin. Kokonaiskapasiteetti kaksinkertaistui, kun reitittimelle XRv-A lisättiin uusi polku, jota pitkin palvelimelle 1 suunnattu liikenne reititettiin.

Kuvassa 38 näkyy graafi UDP-protokollan testistä. Testin keskijaksosta nähdään, että kokonaiskapasiteetti laski verrattuna TCP-protokollan testiin. Tämä johtui pakettihävikistä, mitä syntyi, kun liikennettä lähetettiin enemmän kuin linkin kokonaiskapasiteetti pystyi käsittelemään. UDP-protokolla ei tee uudelleenlähetystä, joten paketit häviävät toisin, kuin TCP-protokolla, joka suorittaa tarkastuksen paketeille ja lähettää ne tarvittaessa uudelleen. Kokonaiskapasiteetti kaksinkertaistui UDP-testissä samalla tavalla kuin TCP-testissäkin, mutta kasvu oli hieman suurempaa suhteessa keskijakson tuloksiin.



Kuva 38. UDP-testin tulokset.

Testauksista voitiin päätellä, että verkon kaistanleveyden hyötyastetta voidaan parantaa helposti jakamalla verkon kokonaiskapasiteetti kaikkien polkujen kesken, ettei kuormitus kohdistuisi liiaksi lyhyimmän polun reitille. Käyttämällä verkon ulkopuolista komponenttia saavutetaan kokonaiskuva verkon tilasta, ja siihen voidaan tehdä muutoksia nopeasti ilman, että reitittimiin tarvitsee koskea. Verkon optimointi voitaisiin myös automatisoida keräämällä telemetriaa verkkolaitteista ja reagoimalla ruuhkatilanteisiin reitittämällä osa liikenteestä automaattisesti vaihtoehtoisia polkuja pitkin. Näin voitaisiin välttää pahimmat pullonkaulat, ennen kuin ne ehtivät muodostua.

5.2 Latenssioptimoitu polku

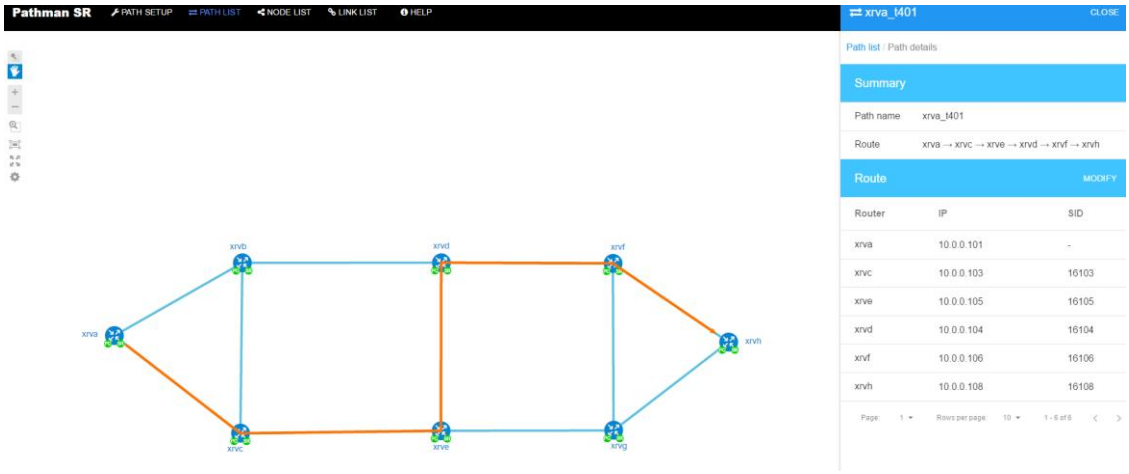
Verkkoviiveellä tai verkkolatenssilla tarkoitetaan aikaa, joka paketilta kuluu kulkea pisteestä toiseen. Latenssia tapahtuu molempiin suuntiin, ja edestakainen viive (Round Trip Time) saadaan laskemalla yhteen molempiin suuntiin vaikuttava latenssi. Verkkoliikenne voidaan ohjata pienimmän latenssin polkua pitkin SR-alueella ja täyttää vaatimukset latenssikriittisille sovelluksille tai parantaa palveluiden käyttökokemusta.

Latenssin mittauksessa topologiaan (liite 1) lisättiin kaksi WAN-Bridge-virtuaalikonetta [36], joilla luotiin latenssia verkkoon keinotekoisesti. WAN-Bridge-virtuaalikoneet sijoitettiin XRv-B- ja XRv-D- sekä XRv-E- ja XRv-G-linkkien väliin, ja ne asetettiin tuottamaan 250 ms:n RTT-viive molempiin lyhimmän polun reitteihin Client 1 -pääteasemalta

Server 1- ja Server 2 -palvelimiin. Palvelimet 1 ja 2 toimivat web-palvelimina ja Client 1 suoritti niille HTTP GET -kutsuja Curl-työkalulla.

Testi suoritettiin 90 sekunnin aikajaksona, jossa vertailtiin vasteaikaa palvelimien 1 ja 2 välillä, joissa on identtiset konfiguraatiot. Lähtötilanteessa liikenne ohjattiin lyhintä polkua pitkin, ja koska topologiassa oli sama metriikka jokaisella linkillä, reitittimeltä XRv-A oli kaksi lyhintä polkua reitittimelle XRv-H, jolloin liikenne ohjattiin käyttäen ECMP-kuormantasausta. Palvelimelle 1 suunnattu liikenne muutettiin 30 sekunnin kohdalla käyttämään sellaista polkua, joka vältti korkean latenssin linkkejä. Viimeisessä vaiheessa 60 sekunnin kohdalla palvelimelta 1 palautuva liikenne ohjattiin samaa polkua vastakkaiseen suuntaan, jolloin liikenne kulki kokonaan välttäen korkean latenssin linkkejä.

Kuvassa 39 nähdään, kuinka palvelimelle 1 kohdistettu liikenne asetettiin välttämään korkean latenssin linkkejä 30 sekunnin kohdalla. Liikenne ohjattiin reitittimien XRv-A → XRv-C → XRv-E → XRv-D → XRv-F → XRv-H kautta.



Kuva 39. Liikenteen ohjaus pienimmän latenssin polkuun.

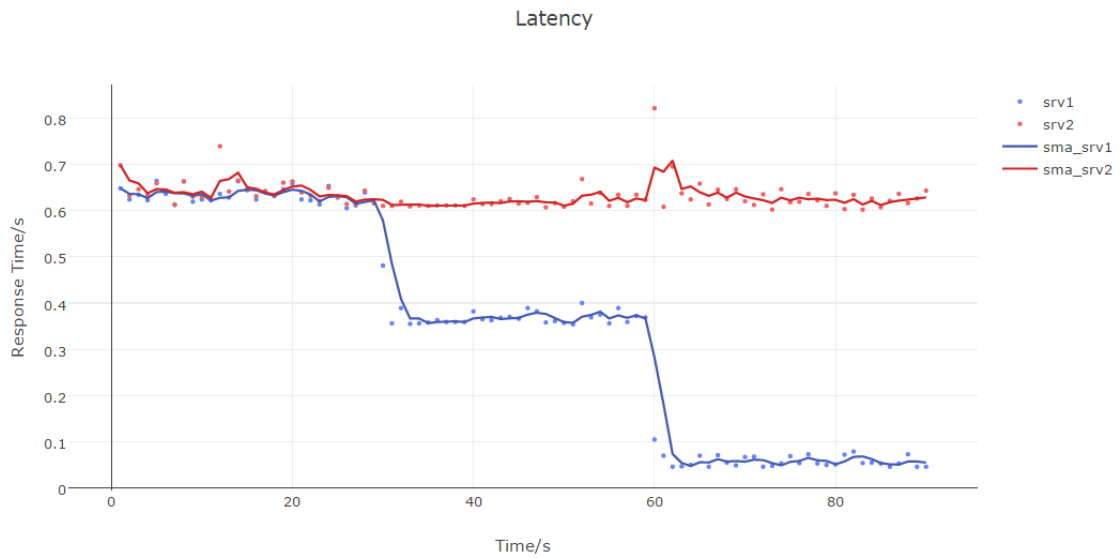
Reitittimeltä XRv-A suoritettiin `traceroute` kohti palvelinta 1, jolla voitiin varmistaa, että liikenne kulkee oikeaa polkua pitkin eikä siinä esiinny suurta latenssia:

```
RP/0/0/CPU0:xrva#traceroute 10.30.1.31 source 10.0.0.101 probe 1

Mon Feb 1 13:50:37.389 UTC
Tracing the route to 10.30.1.31

 1  10.10.20.103 [MPLS: Labels 16105/16104/16106/16108 Exp 0] 9 msec
 2  10.10.50.105 [MPLS: Labels 16104/16106/16108 Exp 0] 9 msec
 3  10.10.70.104 [MPLS: Labels 16106/16108 Exp 0] 0 msec
 4  10.10.60.106 [MPLS: Label 16108 Exp 0] 0 msec
 5  10.10.90.108 9 msec
 6  10.30.1.31 9 msec
```

Testin tulokset nähdään kuvasta 40, jonka graafi näyttää miten vasteaika muuttui palvelimelta 1, kun siihen suunnattu liikenne ohjattiin lyhimmän polun reitiltä latenssioptimoituun polkuun. Testitulos näyttää, että vasteaika laski lähes puoleen verrattuna palvelimeen 2, johon liikenne ohjattiin lyhintä polkua pitkin. Palvelimelta 1 palautuva liikenne kulki kuitenkin vielä lyhintä polkua pitkin, joten paluusuuntaan vaikuttava latenssi näkyi vasteajassa. Kun myös palvelimelta 1 palautuva liikenne ohjattiin pienen latenssin polkua pitkin, laski vasteaika noin 50 millisekuntiin, joka oli optimaalisin tulos verkossa.



Kuva 40. Latenssioptimoidun polun vasteaika suhteessa lyhimpään polkuun

Testitulos osoitti, että SR-reititys yhdessä keskitetyn kontrollerin kanssa mahdollistaa nopean ja vaivattoman tavan ohjelmoida latenssikriittinen verkkoliikenne käyttämään sellaista polkua, joka välttää korkean latenssin linkkejä. Latenssia optimoidessa tulee kuitenkin ottaa huomioon, että verkkoliikenne täytyy optimoida molempiin suuntiin, kos-

ka LSP on aina yksisuuntainen polku ja verkkoliikenne ohjataan oletuksena lyhintä reittiä takaisin.

5.3 Inter-AS-topologia

Inter-AS-topologian testauksessa katsottiin, miten SR-reititys toimii usean autonomisen alueen välillä. Verkon topologia jaettiin kahteen autonomiseen alueeseen, joista topologia muodostettiin keskitetylle kontrollerille BGP-LS-protokollalla ja testattiin BGP-Prefix-SID-segmenttitunnisteen toimintaa. Verkon topologia nähdään liitteessä 2. Lisäksi testattiin, miten staattisesti reitittimeen luotu LSP-polku voidaan delegoida PCE:lle.

Verkko jaettiin kahteen IGP-alueeseen, jotka toimivat eri AS-alueiden sisällä. Niiden välille muodostettiin eBGP-sessio reitittimillä XRv-D ja XRv-F. Reititin XRv-D ja ODL-kontrolleri sijaitsivat saman AS-alueen sisällä, joten ne muodostivat iBGP-session samalla tavalla kuin liitteen 1 topologiassa. Lisäksi XRv-D-reititin ohjasi XRv-F-reitittimeltä tulleet BGP-LS-mainostukset ODL-kontrollerille.

Kuvassa 41 näkyvät konfiguraatiolisäykset reitittimelle XRv-D verrattuna edellisen topologian konfiguraatioon (liite 5). Myös XRv-F-reitittimelle lisättiin lähes identtinen konfiguraatio.

```
1       route-policy LO
2         if destination in (0.0.0.0/0 eq 32) then
3           pass
4         else
5           drop
6         endif
7       end-policy
8       !
9       route-policy SID($SID)
10        set label-index $SID
11      end-policy
12      !
13      route-policy PASS
14        pass
15      end-policy
16      !
17      router static
18       address-family ipv4 unicast
19        10.10.60.106/32 GigabitEthernet0/0/0/1
20       !
21      !
22      router isis srlab
23       address-family ipv4 unicast
24        redistribute bgp 65001 route-policy LO
25       !
26      !
27      router bgp 65001
28       bgp router-id 10.0.0.104
29       address-family ipv4 unicast
30        network 10.0.0.104/32 route-policy SID(104)
31        redistribute isis srlab route-policy LO
32        allocate-label all
33       !
34       address-family link-state link-state
35       !
36       neighbor 10.10.60.106
37        remote-as 65002
38        description eBGP to xrvf
39        address-family ipv4 labeled-unicast
40         route-policy PASS in
41         route-policy PASS out
42        !
43        address-family link-state link-state
44         route-policy PASS in
```

Kuva 41. XRv-D:n konfiguraatiomuutos.

Kuvan 41 riveillä 1-15 lisättiin kolme reitityspolitiikkaa: LO, SID ja PASS. LO-reitityspolitiikkaa käytettiin reititysprotokollien välisessä redistribuutiossa, joka sallii vain verkkomaskilla /32 olevien prefiksien mainostuksen reititysprotokollasta toiseen. Tätä politiikkaa käyttäen rajoitettiin prefiksien mainostus vain loopback-osoitteisiin. SID-reitityspolitiikalla määriteltiin BGP-Prefix-SID-segmenttitunnisteen indeksiarvo. Reitityspolitiikassa määriteltiin muuttuja SID-indeksiarvolle, joka liitettiin haluttuun prefiksiin BGP-konfiguraatiossa. PASS-reitityspolitiikka on yksiselitteinen, jota käyttäen sallittiin kaikkien prefiksien mainostus. Sitä käytettiin sallimaan kaikkien prefiksien mainostus eBGP-sessiossa, joka ei mainosta mitään eikä vastaanota mainostuksia naapureilta ilman reitityspolitiikkaa.

Riveillä 17-19 määriteltiin staattinen reitti osoittamaan BGP-naapurin prefiksiin siinä portissa, joka oli liitetty kyseisen naapurin reitittimeen. Ilman tätä CEF (Cisco Express Forwarding) ei osaa selvittää BGP-LU-reiteille MPLS-leimoja.

Riveillä 22-24 IS-IS-instanssi määriteltiin mainostamaan BGP-instanssin mainostamat loopback-osoitteet LO-reitityspolitiikalla.

Riveillä 30-32 reitittimen loopback-osoite sidottiin BGP-Prefix-SID-tunnisteeseen, IS-IS-instanssin kautta opitut loopback-osoitteet mainostettiin BGP-instanssissa ja reititin määriteltiin allokoimaan lokaali leima kaikille prefikseille, jotka ovat lähtöisin reitittimeltä.

Riveillä 36-44 määriteltiin eBGP-naapuri, jolle sallittiin kaikkien prefiksien mainostus IPv4 Labeled-Unicast-protokollaperheellä, mutta siltä sallittiin Link State -protokollaperheen mainostukset vain vastaanotettaviksi, koska linkkitilatiedot toimitettiin ODL-kontrollerille, eikä niitä ollut tarpeellista mainostaa toisen AS-alueen reitittimille.

Kun BGP-sessio oli muodostettu reitittimien XRv-D ja XRv-F välille, voitiin tarkastaa, että naapurireititin mainostaa BGP-Prefix-SID tunnistetta, joka nähdään kuvassa 42. Reititin XRv-F mainosti BGP-Prefix-SID-tunnistetta indeksiarvolla 106, joka lisättiin SRGB-alueen ensimmäiseen lukuun, jolloin XRv-F-reitittimen leimalle saatiin arvo 16106.

```
RP/0/0/CPU0:xrvd#sh bgp ipv4 labeled-unicast 10.0.0.106/32
Wed Feb 2 20:19:37.671 UTC
BGP routing table entry for 10.0.0.106/32
Versions:
  Process           bRIB/RIB  SendTblVer
  Speaker                55          55
    Local Label: 16106
Last Modified: Feb 2 20:11:03.907 for 00:08:33
Paths: (1 available, best #1)
  Not advertised to any peer
  Path #1: Received by speaker 0
  Not advertised to any peer
  65002
    10.10.60.106 from 10.10.60.106 (10.0.0.106)
      Received Label 3
      Origin IGP, metric 0, localpref 100, valid, external, best, group-best
      Received Path ID 0, Local Path ID 0, version 55
      Origin-AS validity: not-found
      Prefix SID Attribute Size: 10
      Label Index: 106
```

Kuva 42. BGP-Prefix-SID-segmenttitunnisteen mainostus.

Seuraavaksi reitittimeen XRv-A konfiguroitiin staattinen LSP-polku, jonka ylläpito delegoitiin PCE:lle. Kuvassa 43 nähdään LSP-polun konfiguraatio. Riveillä 1-4 määriteltiin LSP-polun reitti lisäämällä ne segmenttitunnisteet, joiden kautta polku kulkee. Tämän jälkeen riveillä 6-14 luotiin uusi tunneli, jonka kohteeksi määriteltiin XRv-H-reitittimen loopback-osoite ja valittiin tunneli käyttämään sr1-polkua. Lisäksi tunnelin hallinta delegoitiin PCE:lle.

```
1   explicit-path name sr1
2    index 10 next-label 16105
3    index 20 next-label 16107
4    index 30 next-label 16108
5
6   interface tunnel-te201
7    ipv4 unnumbered Loopback0
8    destination 10.0.0.108
9    path-selection
10    segment-routing adjacency unprotected
11   !
12   path-option 1 explicit name sr1 segment-routing
13   pce
14    delegation
```

Kuva 43. Staattinen SR LSP -polku.

Liikenne ohjattiin tunneliin määrittelemällä staattinen reitti, jonka seuraavaksi hypyksi valittiin tunneli tunnel-te201 seuraavalla komennolla:

```
router static address-family ipv4 unicast 10.0.0.108/32 tunnel-te201
```

Tämän jälkeen testattiin, että liikenne kohti XRv-H-reitittimen loopback-osoitetta kulki sr1-polkua pitkin, joka nähdään kuvassa 44.

```
RP/0/0/CPU0:xrva#traceroute 10.0.0.108 source 10.0.0.101 probe 1

Tracing the route to 10.0.0.108

 1  10.10.20.103 [MPLS: Labels 16105/16107/16108 Exp 0] 9 msec
 2  10.10.50.105 [MPLS: Labels 16107/16108 Exp 0] 0 msec
 3  10.10.70.104 [MPLS: Labels 16107/16108 Exp 0] 0 msec
 4  10.10.60.106 [MPLS: Labels 16107/16108 Exp 0] 9 msec
 5  10.10.100.107 [MPLS: Label 16108 Exp 0] 9 msec
 6  10.10.110.108 9 msec
```

Kuva 44. Traceroute reitittimeltä XRv-A kohti XRv-H-reititintä.

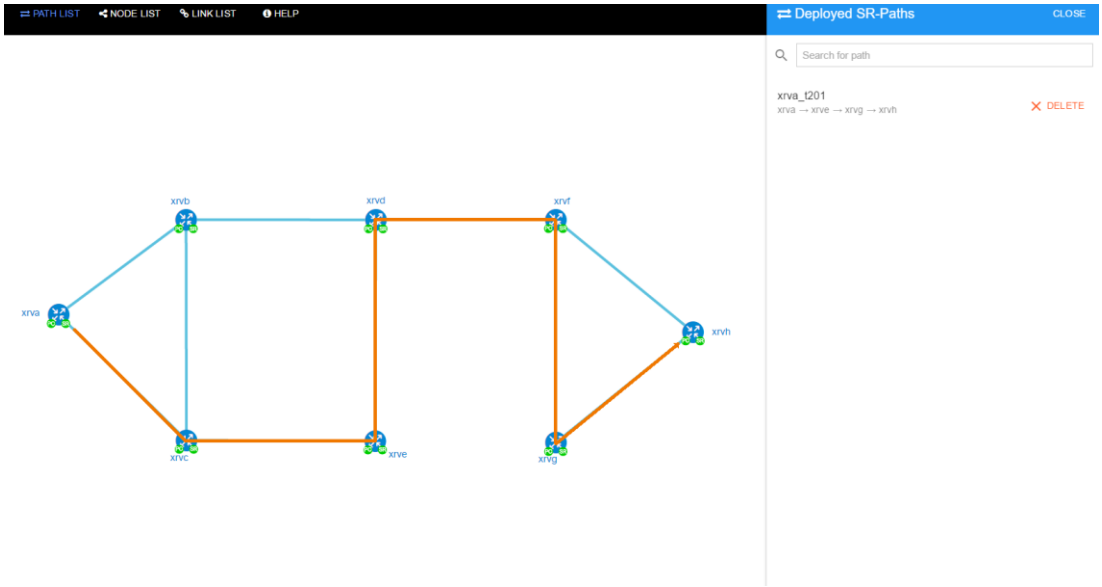
Reitittimeltä XRv-D tarkastettiin, että sen BGP-LS-kannasta löytyi molempien AS-alueiden linkkitilatiedot, jotka se mainosti ODL-kontrollerille. Kuvassa 45 nähdään XRv-D-reitittimen BGP-LS-kanta reitittimien osalta. Siitä nähtiin, että AS-alueen 65002 tiedot

tulivat reitittimen XRv-F kautta. Lisäksi BGP-LS toimitti reitittimien välisten linkkien sekä IP-prefiksien tiedot.

```
RP/0/0/CPU0:xrvd#sh bgp link-state link-state
Wed Feb 1 12:59:22.448 UTC
BGP router identifier 10.0.0.104, local AS number 65001
Status codes: s suppressed, d damped, h history, * valid, > best
              i - internal, r RIB-failure, S stale, N Nexthop-discard
Origin codes: i - IGP, e - EGP, ? - incomplete
Prefix codes: E link, V node, T IP reacheable route, u/U unknown
              I Identifier, N local node, R remote node, L link, P prefix
              L1/L2 ISIS level-1/level-2, O OSPF, D direct, S static/peer-node
              a area-ID, l link-ID, t topology-ID, s ISO-ID,
              c confed-ID/ASN, b bgp-identifier, r router-ID,
              i if-address, n nbr-address, o OSPF Route-type, p IP-prefix
              d designated router address
   Network            Next Hop            Metric LocPrf Weight Path
*> [V][L2][I0x0][N[c65001][b10.0.0.104][s0100.0000.0101.00]]/328    0.0.0.0 0 i
*> [V][L2][I0x0][N[c65001][b10.0.0.104][s0100.0000.0102.00]]/328    0.0.0.0 0 i
*> [V][L2][I0x0][N[c65001][b10.0.0.104][s0100.0000.0103.00]]/328    0.0.0.0 0 i
*> [V][L2][I0x0][N[c65001][b10.0.0.104][s0100.0000.0104.00]]/328    0.0.0.0 0 i
*> [V][L2][I0x0][N[c65001][b10.0.0.104][s0100.0000.0105.00]]/328    0.0.0.0 0 i
*> [V][L2][I0x0][N[c65002][b10.0.0.106][s0100.0000.0106.00]]/328    10.10.60.106    0 65002 i
*> [V][L2][I0x0][N[c65002][b10.0.0.106][s0100.0000.0107.00]]/328    10.10.60.106    0 65002 i
*> [V][L2][I0x0][N[c65002][b10.0.0.106][s0100.0000.0108.00]]/328    10.10.60.106    0 65002 i
```

Kuva 45. BGP-LS-kanta XRv-D-reitittimellä.

Lopuksi tarkastettiin, että ODL saa tiedot molemmista AS-alueista, joista se pystyi muodostamaan verkkotopologian. Kuvassa 46 nähdään Pathman-SR-sovelluksen muodostama topologia kahdesta AS-alueesta ja se, kuinka XRv-A:n delegoima LSP näkyy käyttöönotettujen polkujen listalla.



Kuva 46. Inter-AS-topologia Pathman-SR-sovelluksessa.

Inter-AS-topologian testaus osoitti, että BGP-LS-protokollalla voidaan muodostaa kokonaiskuva usean AS-alueen yli ulottuvasta verkosta. Verkkoliikenne voidaan ohjata

AS-alueiden yli haluttua polkua pitkin ilman, että kohde-AS-alueen reitittimiin tarvitsee koskea vaan reitityspäätös voidaan tehdä ulkopuolisesta kontrollerista käsin.

BGP-Prefix-SID-tunnisteella voidaan mainostaa segmenttitunnisteita AS-alueelta toiselle. Tästä voidaan hyötyä erityisesti konesaliympäristöissä, joiden verkossa käytetään ainoastaan BGP-protokollaa. Verkkoliikenne voidaan myös ohjata määrättyä polkua pitkin päästä päähän esimerkiksi kahden konesalin välillä, joiden välinen liikenne reititetään MPLS-runkoverkon kautta.

6 Johtopäätökset

Insinöörityössä tutkittiin Segment Routing -teknologiaa ja sitä, miten SR-reititystä voidaan hallita reitittimien ulkopuolisesta komponentista käsin sekä miten operoitavasta verkosta voidaan muodostaa topologianäkymä keskitetylle kontrollerille. Lisäksi virtuaaliympäristössä testattiin, miten SR-teknologiaa hyödyntäen voidaan optimoida ja parantaa verkon suorituskykyä ja miten verkkoliikenne voidaan ohjata päästä päähän useiden autonomisten alueiden yli lähdereitittimestä käsin.

Segment Routing yksinkertaistaa ohjaustasoa MPLS-verkoissa, koska LDP- tai RSVP-TE-protokollia ei tarvita vaan SR-informaatio voidaan jakaa käyttäen laajennuksia IGP-protokolliin, kuten OSPF tai IS-IS, ja BGP-protokollaan. SR tarvitsee myös vähemmän MPLS-leimoja käyttöön eikä tilaa tarvitse pitää yllä LSP-polun varrella olevissa reitittimissä vaan tila on paketissa itsessään.

Segment Routing yhdessä SDN-kontrollerin kanssa mahdollistaa keskitetyn verkkoliikenteen hallinnan ja tarjoaa visuaalisen näkymän operoitavaan verkkoon. Tämä yhdistelmä yksinkertaistaa ja nopeuttaa reagointia erilaisiin verkon tilanteisiin, joissa verkkoliikenne halutaan ohjata eri reittejä pitkin kuin lyhintä polkua. Tietynlaiselle verkkoliikenteelle voidaan myös ajastaa enemmän resursseja määrättynä ajankohtana. Esimerkiksi yöaikaan, jolloin tuotantoliikennettä on vähemmän, voidaan allokoida enemmän resursseja backup-liikenteelle.

Insinöörityön tutkimus ja tulokset osoittivat, että Segment Routing yhdessä SDN-kontrollerin kanssa ovat potentiaalinen yhdistelmä verkkoliikenteen optimoimiselle. Testauksissa ei kuitenkaan käyty läpi erilaisia ongelmatilanteita, joita voi syntyä, mikäli topologiassa tapahtuu odottamattomia muutoksia. BGP-LS-protokollan kautta tuleva tieto SDN-kontrollerille ei välttämättä ole reaaliaikaista, koska IGP-protokollan täytyy ensin injektoida tieto BGP-protokollaan, joka välittää tiedon kontrollerille, jossa voi syntyä viivettä.

Työssä tutkittiin ja testattiin, miten keskitettyä Segment Routing -reititystä käyttäen voidaan optimoida verkkoliikennettä. Haasteita aiheutti saatavilla olevan tiedon rajallinen määrä, koska Segment Routing -teknologia on vielä suhteellisen uusi asia ja vielä standardoinnin alla. Työn tavoitteisiin kuitenkin päästiin ja testaukset saatiin suoritettua onnistuneesti.

Lähteet

1 Filsfils C., Michielsen K. & Talaulikar K. 2016. Segment Routing, Part 1. Cisco Systems.

2 Segment Routing: Prepare Your Network for New Business Models. 2015. Verkkodokumentti. Cisco. http://www.cisco.com/c/en/us/solutions/collateral/serviceprovider/application-engineered-routing/white-paper-c11-734250.html. Luettu 22.11.2016.

3 Segment Routing and Path Computation Element. 2015. Verkkodokumentti. Alcatel-Lucent. http://www.tmcnet.com/tmc/whitepapers/documents/whitepapers/2015/11525-segment-routing-path-computation-element-using-traffic-engineering.pdf. Luettu 22.11.2016.

4 Segment Routing Architecture. 2016. Verkkodokumentti. IETF. https://tools.ietf.org/html/draft-ietf-spring-segment-routing-09. Luettu 10.11.2016.

5 IPv6 Segment Routing Header (SRH). 2015. Verkkodokumentti. IETF. https://tools.ietf.org/html/draft-previdi-6man-segment-routing-header-08. Luettu 10.11.2016.

6 Segment Routing with MPLS data plane. 2016. Verkkodokumentti. IETF. https://tools.ietf.org/html/draft-ietf-spring-segment-routing-mpls-05. Luettu 12.11.2016.

7 Multiprotocol Label Switching Architecture. 2001. Verkkodokumentti. IETF. https://tools.ietf.org/html/rfc3031. Luettu 12.12.2016.

8 MPLS Label Stack Encoding. 2001. Verkkodokumentti. IETF. https://tools.ietf.org/html/rfc3032. Luettu 12.12.2016.

9 The Segment Routing Architecture. 2015. Verkkodokumentti. IMDEA Networks Institute. http://eprints.networks.imdea.org/1306/1/the_segment-routing_architecture_2015.pdf. Luettu 3.12.2016.

10 IS-IS Extensions for Segment Routing. 2016. Verkkodokumentti. IETF. https://tools.ietf.org/html/draft-ietf-isis-segment-routing-extensions-09. Luettu 10.12.2016.

11 Segment Routing Prefix SID extensions for BGP. 2016. Verkkodokumentti. IETF. https://tools.ietf.org/html/draft-ietf-idr-bgp-prefix-sid-04. Luettu 19.12.2016.

12 Software-Defined Networking: A Comprehensive Survey. 2014. Verkkodokumenti. IEEE. https://arxiv.org/pdf/1406.0440.pdf. Luettu 28.12.2016.

13 Shah P., Hassan S. & Chayapathi R. 2016. Network Functions Virtualization (NFV) with a Touch of SDN. Addison-Wesley Professional.

14 Stallings, William. 2015. Foundations of Modern Networking: SDN, NFV, QoE, IoT, and Cloud. Addison-Wesley Professional.

15 Accelerate SDN Adoption and Deployment. 2017. Verkkodokumentti. Ixia. https://www.ixiacom.com/sites/default/files/2017-01/915-3697-01-T-SB-IxNetwork-SDN.pdf. Luettu 12.1.2017.

16 Voruganti S. & Subramanian S. 2016. Software-Defined Networking (SDN) with OpenStack. Packt Publishing.

17 Anderson J. & Morreale P. 2015. Software Defined Networking. CRC Press.

18 OpenDaylight Boron. 2016. Verkkodokumentti. Opendaylight. https://www.opendaylight.org/odlboron. Luettu 20.12.2016.

19 A Path Computation Element (PCE)-Based Architecture. 2006. Verkkodokumenti. IETF. https://tools.ietf.org/html/rfc4655. Luettu 4.12.2016.

20 An Analysis of Scaling Issues in MPLS-TE Core Networks. 2009. Verkkodokumentti. IETF. https://tools.ietf.org/html/rfc5439. Luettu 6.12.2016.

21 Path Computation Element Protocol (PCEP) Numbers. 2016. Verkkodokumentti. IANA. http://www.iana.org/assignments/pcep/pcep.xhtml. Luettu 13.12.2016.

22 Advanced PCE. 2013. Verkkodokumentti. Old Dog Consulting. http://www.olddog.co.uk/AdvancedPCE.pdf. Luettu 15.12.2016.

23 Path Computation and Instantiation in Software Defined Networks Using Stateful PCE. 2013. Verkkodokumentti. Cisco. http://www.slideshare.net/getyourbuildon/mplssdn-2013-international-conference-mpls2013-iso-core. Luettu 13.12.2016.

24 Applicability of a Stateful Path Computation Element (PCE). 2016. Verkkodokumentti. IETF. https://tools.ietf.org/html/draft-ietf-pce-stateful-pce-app-08. Luettu 14.12.2016.

25 Path Computation Element (PCE) Communication Protocol (PCEP). 2009. Verkkodokumentti. IETF. https://tools.ietf.org/html/rfc5440. Luettu 16.12.2016.

26 PCEP Extensions for Stateful PCE. 2016. Verkkodokumentti. IETF. https://tools.ietf.org/html/draft-ietf-pce-stateful-pce-18. Luettu 18.12.2016.

27 PCEP Extensions for PCE-initiated LSP Setup in a Stateful PCE Model. 2016. Verkkodokumentti. IETF. https://tools.ietf.org/html/draft-ietf-pce-pce-initiated-lsp-07. Luettu 17.12.2016.

28 North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP. 2016. Verkkodokumentti. IETF. https://tools.ietf.org/html/rfc7752. Luettu 13.12.2016.

29 Cisco IOS XRv 9000 Router Installation and Configuration Guide. 2015. Verkkodokumentti. Cisco. http://www.cisco.com/c/en/us/td/docs/routers/virtual-routers/configuration/guide/b-xrv9k-cg.pdf. Luettu 5.1.2017.

30 Cisco IOS XRv Router Overview. 2014. Verkkodokumentti. Cisco. http://www.cisco.com/en/US/docs/ios_xr_sw/ios_xrv/install_config/b_xrvr_432_chapter_01.pdf. Luettu 10.1.2017.

31 Cisco IOS XR Software Release 5.2.0 for Cisco ASR 9000 Series Routers. 2014. Verkkodokumentti. Cisco. http://www.cisco.com/c/en/us/products/collateral/routers/asr-9000-series-aggregation-services-routers/bulletin-c25-732696.html. Luettu 12.1.2017.

32 OpenDaylight Pathman SR App. 2016. Verkkodokumentti. CiscoDevNet. https://github.com/CiscoDevNet/pathman-sr. Luettu 7.1.2017.

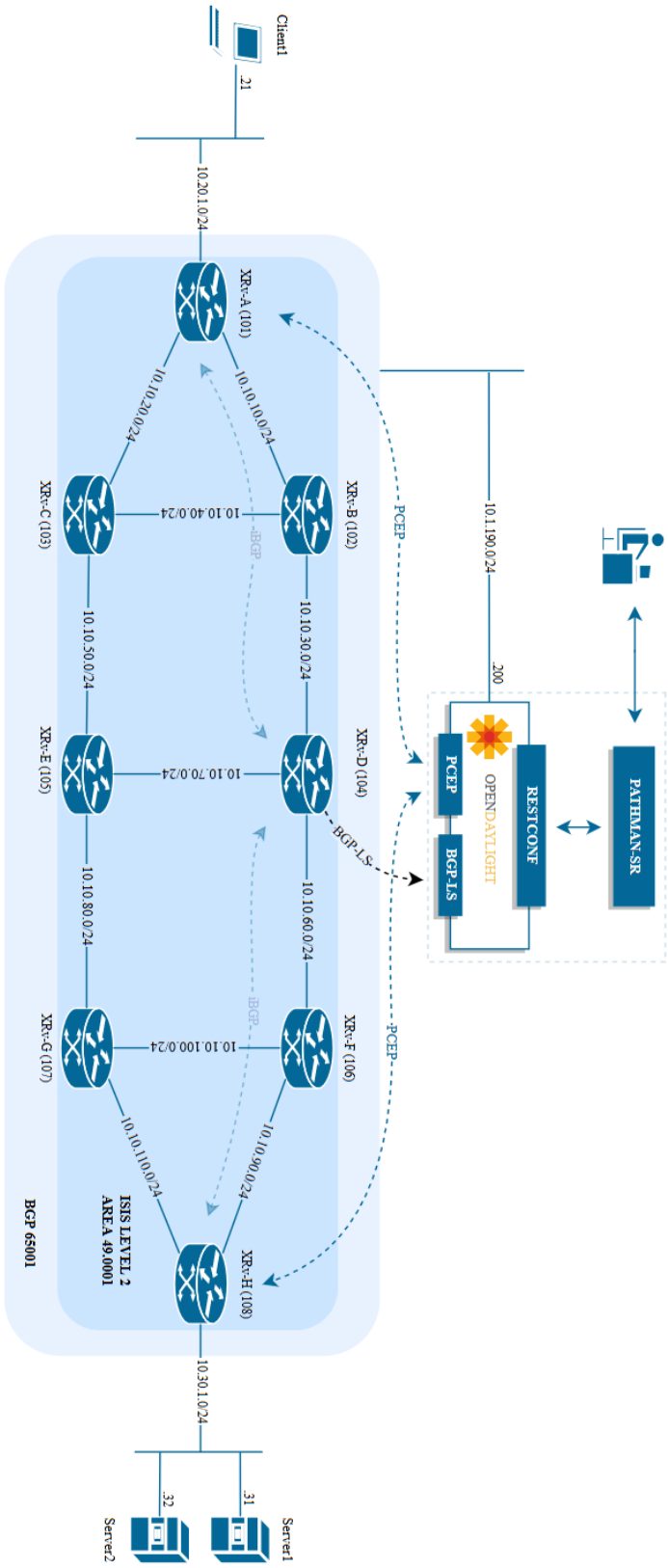
33 A Border Gateway Protocol 4 (BGP-4). 2006. Verkkodokumentti. IETF. https://tools.ietf.org/html/rfc4271. Luettu 14.12.2016.

34 Boron-SR2. 2016. Verkkodokumentti. OpenDaylight. https://www.opendaylight.org/software/downloads/boron-sr2. Luettu 16.1.2017.

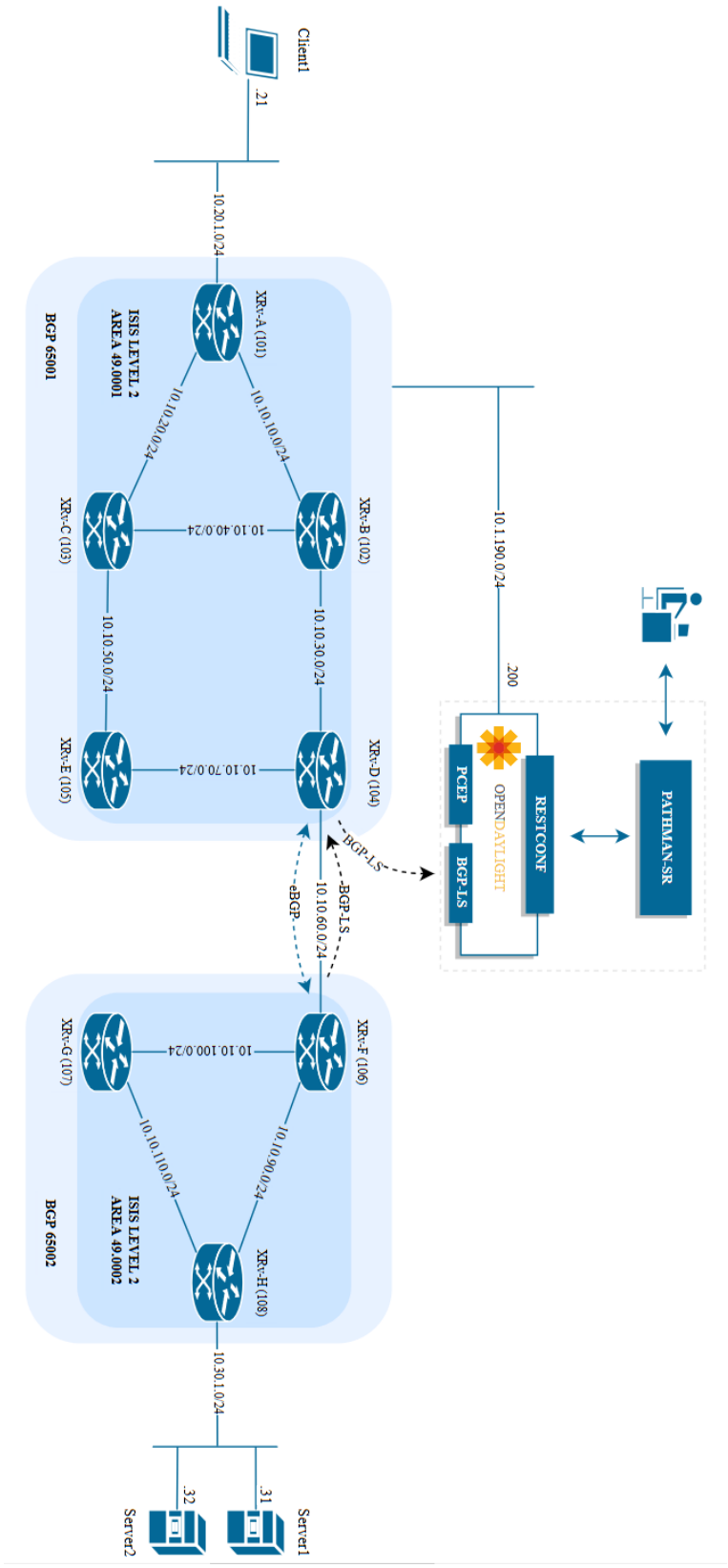
35 File Exchange. 2017. Verkkodokumentti. Cisco. https://upload.cisco.com/cgi-bin/swc/fileexg/main.cgi?CONTYPES=Cisco-IOS-XRv. Luettu 16.1.2017.

36 Wanbridge. 2010. Verkkodokumentti. Google Code. https://code.google.com/archive/p/wanbridge/downloads. Luettu 16.1.2017.

Verkkotopologia 1



Verkkotopologia 2



Reitittimien IP osoitteet

Reititin	Interface	IP osoite
XRv-A	Loopback0	10.0.0.101/32
	MgmtEth0/0/CPU0/0	10.1.190.101/24
	GigabitEthernet0/0/0/0	10.10.10.101/24
	GigabitEthernet0/0/0/1	10.10.20.101/24
	GigabitEthernet0/0/0/2	10.20.1.101/24
XRv-B	Loopback0	10.0.0.102/32
	MgmtEth0/0/CPU0/0	10.1.190.102/24
	GigabitEthernet0/0/0/0	10.10.10.102/24
	GigabitEthernet0/0/0/1	10.10.30.102/24
	GigabitEthernet0/0/0/2	10.10.40.102/24
XRv-C	Loopback0	10.0.0.103/32
	MgmtEth0/0/CPU0/0	10.1.190.103/24
	GigabitEthernet0/0/0/0	10.10.20.103/24
	GigabitEthernet0/0/0/1	10.10.40.103/24
	GigabitEthernet0/0/0/2	10.10.50.103/24
XRv-D	Loopback0	10.0.0.104/32
	MgmtEth0/0/CPU0/0	10.1.190.104/24
	GigabitEthernet0/0/0/0	10.10.30.104/24
	GigabitEthernet0/0/0/1	10.10.60.104/24
	GigabitEthernet0/0/0/2	10.10.70.104/24
XRv-E	Loopback0	10.0.0.105/32
	MgmtEth0/0/CPU0/0	10.1.190.105/24
	GigabitEthernet0/0/0/0	10.10.50.105/24
	GigabitEthernet0/0/0/1	10.10.70.105/24
	GigabitEthernet0/0/0/2	10.10.80.105/24
XRv-F	Loopback0	10.0.0.106/32
	MgmtEth0/0/CPU0/0	10.1.190.106/24
	GigabitEthernet0/0/0/0	10.10.60.106/24
	GigabitEthernet0/0/0/1	10.10.90.106/24
	GigabitEthernet0/0/0/2	10.10.100.106/24
XRv-G	Loopback0	10.0.0.107/32
	MgmtEth0/0/CPU0/0	10.1.190.107/24
	GigabitEthernet0/0/0/0	10.10.80.107/24
	GigabitEthernet0/0/0/1	10.10.100.107/24
	GigabitEthernet0/0/0/2	10.10.110.107/24
XRv-H	Loopback0	10.0.0.108/32
	MgmtEth0/0/CPU0/0	10.1.190.108/24
	GigabitEthernet0/0/0/0	10.10.90.108/24
	GigabitEthernet0/0/0/1	10.10.110.108/24
	GigabitEthernet0/0/0/2	10.30.1.108/24

XRv-A:n konfiguraatio

```
hostname xrva
!
cdp
!
ipv4 unnumbered mpls traffic-eng Loopback0
interface Loopback0
 description Router ID
 ipv4 address 10.0.0.101 255.255.255.255
!
interface MgmtEth0/0/CPU0/0
 description Management
 ipv4 address 10.1.190.101 255.255.255.0
!
interface GigabitEthernet0/0/0/0
 description link to xrv-b
 cdp
 ipv4 address 10.10.10.101 255.255.255.0
!
interface GigabitEthernet0/0/0/1
 description link to xrv-c
 cdp
 ipv4 address 10.10.20.101 255.255.255.0
!
interface GigabitEthernet0/0/0/2
 description link to Branch1
 cdp
 ipv4 address 10.20.1.101 255.255.255.0
!
router isis srlab
 is-type level-2-only
 net 49.0001.0100.0000.0101.00
 address-family ipv4 unicast
  metric-style wide
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  segment-routing mpls
 !
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 101
  !
 !
 interface GigabitEthernet0/0/0/0
  point-to-point
  address-family ipv4 unicast
```

```
   metric 10
  !
 !
 interface GigabitEthernet0/0/0/1
  point-to-point
  address-family ipv4 unicast
   metric 10
  !
 !
!
router bgp 65001
 bgp router-id 10.0.0.101
 address-family ipv4 unicast
 !
 neighbor 10.0.0.104
  remote-as 65001
  update-source Loopback0
  description iBGP to xrv-d
  address-family ipv4 unicast
   route-reflector-client
  !
 !
!
mpls traffic-eng
 pce
  peer ipv4 10.1.190.200
  !
  segment-routing
  stateful-client
   instantiation
  !
 !
 auto-tunnel pcc
  tunnel-id min 1 max 199
 !
 reoptimize timers delay installation 0
!
netconf-yang agent
 ssh
!
segment-routing
 global-block 16000 16199
!
ssh server v2
ssh server netconf vrf default
```

XRv-D:n konfiguraatio

```
hostname xrvd
!
cdp
!
ipv4 unnumbered mpls traffic-eng Loopback0
interface Loopback0
 description Router ID
 ipv4 address 10.0.0.104 255.255.255.255
!
interface MgmtEth0/0/CPU0/0
 description Management
 ipv4 address 10.1.190.104 255.255.255.0
!
interface GigabitEthernet0/0/0/0
 description link to xrvr-b
 cdp
 ipv4 address 10.10.30.104 255.255.255.0
!
interface GigabitEthernet0/0/0/1
 description link to xrv-f
 cdp
 ipv4 address 10.10.60.104 255.255.255.0
!
interface GigabitEthernet0/0/0/2
 description link to xrv-e
 cdp
 ipv4 address 10.10.70.104 255.255.255.0
!
router isis srlab
 is-type level-2-only
 net 49.0001.0100.0000.0104.00
 distribute bgp-ls
 address-family ipv4 unicast
  metric-style wide
  mpls traffic-eng level-2-only
  mpls traffic-eng router-id Loopback0
  segment-routing mpls
 !
 interface Loopback0
  address-family ipv4 unicast
   prefix-sid index 104
  !
 !
 interface GigabitEthernet0/0/0/0
  point-to-point
  address-family ipv4 unicast
```

```
   metric 10
  !
 !
 interface GigabitEthernet0/0/0/1
  point-to-point
  address-family ipv4 unicast
   metric 10
  !
 !
 interface GigabitEthernet0/0/0/2
  point-to-point
  address-family ipv4 unicast
   metric 10
  !
 !
!
router bgp 65001
 bgp router-id 10.0.0.104
 address-family ipv4 unicast
 !
 address-family link-state link-state
 !
 neighbor 10.0.0.101
  remote-as 65001
  update-source Loopback0
  description iBGP to xrv-a
  address-family ipv4 unicast
   route-reflector-client
  !
 !
 neighbor 10.0.0.108
  remote-as 65001
  update-source Loopback0
  description iBGP to xrv-h
  address-family ipv4 unicast
   route-reflector-client
  !
 !
 neighbor 10.1.190.200
  remote-as 65001
  update-source MgmtEth0/0/CPU0/0
  description BGP-LS to ODL
  address-family ipv4 unicast
   route-reflector-client
  !
  address-family link-state link-state
   route-reflector-client
  !
 !
```

```
!
mpls traffic-eng
 pce
  peer ipv4 10.1.190.200
  !
  segment-routing
  stateful-client
   instantiation
  !
 !
 auto-tunnel pcc
  tunnel-id min 1 max 199
 !
 reoptimize timers delay installation 0
!
netconf-yang agent
 ssh
!
segment-routing
 global-block 16000 16199
!
ssh server v2
ssh server netconf vrf default
```