# Reunalaskenta arkkitehtuurit

Lauri Vene

Pro gradu -tutkielma
HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen laitos

Helsinki, 13. huhtikuuta 2018

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

| Tiedekunta — Fakultet — Faculty | Laitos — Institution — Department |
|---|---|
| Matemaattis-luonnontieteellinen | Tietojenkäsittelytieteen laitos |

Tekijä — Författare — Author
Lauri Vene

Työn nimi — Arbetets titel — Title

Reunalaskenta arkkitehtuurit

Oppiaine — Läroämne — Subject
Tietojenkäsittelytiede

| Työn laji — Arbetets art — Level | Aika — Datum — Month and year | Sivumäärä — Sidoantal — Number of pages |
|---|---|---|
| Pro gradu -tutkielma | 13. huhtikuuta 2018 | 31 |

Tiivistelmä — Referat — Abstract

Tiivistelmä.

Avainsanat — Nyckelord — Keywords
reuna, pilvi, tietojenkäsittelytiede

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

# Sisältö

# Huomioita

- reuna
- reunasolmu

Tutkielmassa puhutaan mobiililaitteesta, mutta monet toiminnallisuudet ovat sovellettavissa kaikenlaisiin mobiileihin laitteisiin, esimerkiksi kannettaviin tai älykkäisiin ajoneuvoihin.

Listan Mobile Computingin neljästä pääasiallisesta rajoitteesta.
*Unpredictable variation in network quality, lowered trust and robustness of mobile elements, limitations on local resources imposed by weight and size constraints, and concern for battery power consumption* [13]

[13]M. Satyanarayanan, "Fundamental Challenges in Mobile Computing,"Proc. 15th ACM Symp. Principles of Dist. Comp., Philadelphia, PA, May, 1996

# 1 Johdanto

# 2 Reunalaskennan perusteet

## 2.1 Motivaatio

Reunalaskennan ideana on täydentää ja avustaa resurssiköyhiä asiakaslaitteita. Mobiililaitteisiin voitaisiin teoriassa lisätä enemmän resursseja, mutta ne tulisivat kannettavuuden ja käyttöajan kustannuksella. Avustamiseksi voidaan ajatella esimerkiksi tilanne, jossa akkuvirtaa säästääkseen, mobiililaite välittää resurssi intensiivisen laskutoimituksen reunasolmulle laskettavaksi. Reunasolmun voi ajatella kevennetyksi versioksi palvelimesta joten sen rajoitteet ovat hyvin erilaiset kuin asiakaslaitteen. Täydentävä toiminta reunalaskennan avulla tarkoittaa asiakaslaitteen resurssin puutetta suoriutua jostakin tehtävästä. Esimerkiksi muistin riittämättömyys kuvankäsittelyyn. Tällöin voitaisiin esimerkiksi ottaa etäkäyttöyhteys reunasolmuun jolla käsittely tehdään. Asiakaslaitteelle jäisi tässä tilanteessa tehtäväksi ainoastaan esittää reunasolmun tilaa käyttäjälle.

Satyanarayana [Satyanarayanan, 2001] esitti artikkelissaan pervasive computing esimerkkejä jokapaikan tietotekniikasta. Ympäristöön sijoitettujen laitteiden yhteistoiminnan avulla, asiakkaalle voidaan tarjota parempaa ja täsmällisempää palvelua. Yhtenä palvelun laadun ehtona on kyky ennakoida asiakkaan toimintaa. Nykyään mobiililaitteilla on mahdollista hyödyntää langattomia yhteyksiä. Suurin osa palvelinresursseista ja palveluiden tuottamiseen käytettävästä tiedosta on keskittyneenä pilveen. Käytännössä minkä tahansa palvelun käyttäminen mobiililaitteella edellyttää yhteyttä näihin pilvipalveluihin. Pilvipalveluiden ylläpitäminen

Satyanara etal. argumentoivat julkaisussaan että verkon lantenssia ei voida enää juurikaan pienentää. Ratkaisuksi jää palveluiden tuominen lähemmäksi.

Toinen painopiste on siinä että tieto seuraa käyttäjää. Esimerkiksi pöytäkoneelta mobiililaitteeseen.(Satyanarayanan, 2001). Cyber foraging, on termi jota käytetään kuvaamaan paradigmaa jossa laite etsii ympäristöstä hyödynnettävää tietoa ja avustajia/korvikkeita. Avustajan (surrogate) rooli on täydentää lähtökohtaisesti resurssirajallista laitetta, esimerkiksi suorittamalla laskentaa, jotta asiakaslaite voisi esimerkiksi säästää akkua. Tähän toimintaan liittyy kuitenkin useita haasteita. Esimerkiksi kuinka asiakaslaite löytää avustajan? Mitäs jos avustaja on ruuhkautunut? Kuinka avustaja alustetaan ja kauanko siinä kestää? Nämä ovat keskeisiä kysymyksiä myös reunalaskennassa. Vastuun jakaminen asiakaslaitteen sekä reunanklusterin välillä on riippuvainen siitä, kuinka paljon avustusta asiakaslaite tarvitsee. Toiseen ääripäähän vietynä asiakaslaite on niin sanottu kevyt asiakaspääte (thin client), jolla ei olisi resursseja juurikaan mihinkään. Tällainen asiakaslaite joutuisi jakamaan kaiken laskennan eteenpäin avustajalle. Tämän kaltainen asiakaslaite olisi riippuvainen reunan mahdollisuuksista suorittaa palveluiden vaatimia toiminnallisuuksia. Seuraavana askeleena kohti itsenäisempää suoriutumista olisi asiakaslaite, joka pystyy osittain tarjoamaan käyttäjälle palveuita. Tämä laite tarvisi reunaklusterilta avustusta ainoastaan joissain tapauksissa. Viimeisenä toimijana olisi kokonaan itsenäinen asiakaslaite, joka tarvitsisi reunalta ainoastaan palveluita täydentäviä ominaisuuksia. Tämä laite saattaisia turvautua reunalaskentaan esimerkiksi jos akku on vähissä, tai laitteella itsellään ei ole kaikea tarvittavaa tietoa laskennan suorittamiseksi.

**a**

- Mistä reunalaskenta koostuu? (Suurimmat toimijat, keskeisimmät toiminnot)
- Mikä on MEC?
- Reunalaskenta vai reunapalvelu?
- MCC vai MEC?
- Pilvi vai palvelinkeskus, palvelinresursseja.
- Mitä reunalaskenta on?
- Miksei siirretä laskentaa pilveen?
- Mitkä ovat mobiilin ongelmat nykyisellään?
- Mitkä ovat reunalaskennan haasteet?

Pilvipalvelulla tarkoitetaan palvelua, joka sijaitsee internetissä. Palvelut tarjotaan käyttäjälle verkkoyhteyden välityksellä. Pilvipalvelut sisältävät usein suuria määriä laskenta- ja tallennusresursseja. Palvelut usein sijoitettu siten että ne ovat runkoverkossa kiinni, jolloin niiden voidaan ajatella sijaitsevan internetin "keskustassa". Yleensä palveluita ylläpidetään keskitettyinä korkeintaan muutamaan eri konesaliin. Pilvipalvelun palvelun kohde on usein asiakaslaite (UE, user equipment), joka sijaitsee internet-topologian näkökulmasta lehtisolmussa.

Reunalaskenta on yksi hajautetun laskennan muoto jossa

Mobiililaitteiden yleisiä ominaisuuksia ovat resurssien vähyys ja akkuvirran rajallisuus. Mobiililaitteiden käyttö on myös usein riippuvaista langattomista verkkoyhteyksistä. Palveluiden toiminta mobiililaitteilla on siis riippuvainen näiden kolmen ominaisuuden asettamista rajoista. Laskentaresurssien lisääminen johtaisin lyhyempään käyttöaikaan akkuvirralla. Suurempi akku mahdollistaa pidemmän käyttöajan, mutta se tekisi laitteesta suuremman. Akun kokoa ja laskentaresurssien määrää pyritäänkin tasapainottamaan. Voidaan pyrkiä minimoimaan laitteen virrankulutus esimerkiksi laittamalla laitteeseen heikkotehoisempi suoritin. Tämä näkyy siinä, millaisia palveluita mobiililaitteella voidaan tarjota. Esimerkiksi kuvankäsittelyä tai muuta raskaampaa laskentaa vaativaa toiminnallisuutta ei voida tällaisella laitteella tehdä. Seuraava vaihtoehto olisi lähettää laskentaa tehokkaammille laitteille pilveen. Laskennan siirtäminen vie aikaa ja tällöin ei voida tarjota kovin reaaliaikaisia palveluita. Siirrettyyn laskentaan kuluva aika koostuu pääasiassa verkon viiveestä, siirrosta ja itse laskennasta. Kokonaisuudessa siirtämiseen ja laskentaan kuluvan ajan määrään vaikuttaa pilven sijainti, verkon ruuhkaisuus, verkon kapasiteetti, sekä käytössä olevan laskentakapasiteetin määrä. Reunalaskenta on konsepti, jonka avulla laskenta voidaan tuoda lähemmäksi käyttäjää.

Reunalaskennassa (MEC,Mobile Edge Computing vai MCC Mobile Cloud Computing?) on tarkoitus tuoda palvelinresursseja lähemmäksi käyttäjää. Reunalaskenta on osassa kirjallisuutta jaettu kahteen kategoriaan sen mukaan tarjotaanko palvelua RAN (Radio access network) vai yleisesti WAN yhteydessä. RAN verkossa toimivaa reunalaskentaa kutsutaan Multi-access edge computing ja muita reunalaskennan ratkaisumalleja nimillä kuten sumu(fog computing) tai cloudlet[Taleb et al., 2017] . Tässä kontekstissa reunalla tarkoitetaan käyttäjän ja pilven väliin jäävää tilaa. TCP/IP-mallissa sovellustasolla olevia toimintoja ei siis esiinny tällä välillä. Reunalaskenta siis mahdollistaa palveluiden tuottamisen lähempänä käyttäjää. Lähempänä on hieman harhaan johtava termi, koska mikä tahansa pilveä lähempänä oleva palvelu on lähempänä, eikä siis välttämättä konkreettisesti lähellä.

Reunalaskennalle ei vielä ole olemassa kokonaisvaltaista arkkitehtuuria. Ongelmakentän voi jakaa karkeasti kahteen osaan. Fyysiseen arkkitehtuuriin ja sovellustason arkkitehtuuriin. Nämä ovat toisistaan riippuvaisia. Arkkitehtuuriratkaisut ovat riippuvaisia tarjottavista palveluista. Toiset arkkitehtuu-

riratkaisut tukevat toisia palveluita paremmin kuin toiset, kompromisseilta on siis vaikea välttyä.

## 2.2 Keskeiset käsitteet

Tässä kappalessa esitellään tämän tutkielman kannalta tärkeät toimijat. Reunaan kuuluvien toimijoiden lisäksi on hyvä tietää reunan sijoittuminen muiden reunan ulkopuolisten toimijoiden suhteen. Tässä yhteydessä muut toimijat on rajattu reunajärjestelmän osalta oleellisiin toimijoihin.

Perinteinen asiakas-palvelin -malli koostuu vain kahdesta toimijasta. Reunajärjestelmän lisääminen tähän malliin luo omanalaisen hierarkian joka, muistuttaa enemmänkin asiakas-reuna-palvelin muotoista järjestelmää. Seuraavaksi käydään läpi hieman tarkemmin mitä nämä osat sisältävät ja mikä niiden keskeisin rooli reunajärjestelmän kontekstissa on.

### 2.2.1 Asiakaslaite

Asiakaslaite (user equipment) on yleisnimitys laitteelle, joka on reunan tai pilven tarjoamien palveluiden asiakas. Asiakaslaitteen tyyppiä ei ole rajattu ja asiakaslaite voi olla esimerkiksi älypuhelin, älylasit tai vaikka verkkoyhteydellä varustettu auto. Yhdistävänä tekijänä on siis jonkinlainen yhteys reuna- tai pilvipalveluihin. Yleisesti yhteys on tyypiltään langaton. Tässä tutkielmassa asiakaslaitteella tarkoitetaan älypuhelinta, jollei toisin mainita.

Verkkohierarkian näkökulmasta asiakaslaitteet ovat lehtisolmuja. Tämä tarkoittaa että asiakaslaitteet toimivat ainoastaan palveluiden kuluttajina eivätkä siis tarjoa itse palveluita. Myöskään kulutettavien palveluiden tyypillä ei ole juurikaan merkitystä, kun asiaa käsitellään infrastruktuuri/arkkitehtuuritasolla.

Esimerkkinä asiakaslaitteen toiminnasta reunapalvelua hyödyntäen voisi olla ajoneuvo, joka käyttää reunapalveluita tiedon välittämiseen muille lähistöllä oleville ajoneuvoille.

### 2.2.2 Reuna

Reuna koostuu useista toiminnallisista entiteeteistä, jotka voidaan jakaa sekä fyysisiin, että loogiisin entiteetteihin. On kuitenkin hyvä ymmärtää koko reunan käsittävä reuna-alue, joka esitellään ensimmäisenä. Tämän jälkeen esitellään reunasolmu, joka on reunajärjestelmän keskeisin fyysinen rakennuspalanen. Lopuksi esitellään pääasiassa ohjelmallisen tason toimivat toimijat reuna-alusta ja reunasovellus.

**Reuna-alue** Reuna-alue tai reuna ei ole mikään tarkasti rajattu alue, vaan reunalla usein tarkoitetaan jonkin tietyn kontekstin mukaista reunaa. Yleisesti reunalla voidaan viitataan alueeseen, joka ulottuu asiakaslaitteelta

runkoverkkoon asti. Reunalle ei siis ole tarkkaa määritelmää. Reuna-alue rajautuu siinä esiintyvien toimijoiden mukaan jollekkin välille. Esimerkiksi mobiiliverkon tukiasemien yhteyteen rakennettua reunajärjestelmää käsiteltäessä, reunalla tarkoitetaan ainoastaan reunajärjestelmän asiakaslaitteita palvelevien osien muodostamaa vyöhykettä. Yleisenä nyrkkisääntönä voidaan pitää verkkoyhteyksien viivettä, suhteessa muuhun internettiin, koska reuna-alueella viiveiden tulisi siis olla muuta internettiä nopeampia. Toisin sanoen palveluiden tulisi sijaita lähempänä.

**Reunasolmu** Tämän tutkielman kontekstissa reunasolmulla (mobile edge host) tarkoitetaan yksittäistä reunalaskentaa suorittavaa entiteettiä. Reunasolmu voi koostua esimerkiksi mobiilitukiaseman ja palvelinlaitteiston muodostamasta kokonaisuudesta. Reunasolmu sisältää vähintään reunasovelluksien suorittamiseen tarvittavan laitteiston, sekä toiminnallisesti reunasovelluksien suorittamiseen tarvittavat hallinnolliset toimet. Kuten aiemmin mainittu, reunajärjestelmä koostuu joukosta reunasolmuja, jotka on hajautettu maantieteellisesti. Reunasolmut siis eroavat toisistaan vähintään sijainnin perusteella, mutta voivat erota myös käytettävissä olevien laskenta ja tallennus resurssien osalta. Reunasolmun sijaintiin vaikuttaa käytössä oleva reuna-arkkitehtuuri. Teoriassa reunasolmu voi sijainta asiakaslaitteesta *yhden hypyn päässä*, jolloin asiakaslaitteella olisi suora yhteys reunasolmuun, mutta on myös mahdollista, että reunasolmu sijaitsee kauempana esimerkiksi runkoverkon reunalla. Laskentaresurssien osalta reunasolmu voi olla mitä tahansa vähäisillä laskenta ja tallennus resursseilla varustetun WiFi-tukiaseman ja kokonaisen palvelinklusterin väliltä.

**Reuna-alusta** (Edge platform) on ohjelmistotason toimija, joka tarjoaa rajapinnan reunasovelluksien suorittamista varten. Toisin sanoen siis tarjoaa reunasovelluksille toimintaympäristön. Reuna-alustan tehtävät eivät rajaudu ainoastaan yksittäiseen reunasolmuun, vaan sen lisäksi se hoitaa hallinnollisia tehtäviä kuten tietoliikenteen ohjausta. Lisäksi reuna-alustan tehtäviin voidaan lukea reunasovelluksia suorittaviin virtuaalikoneisiin liittyvät hallinnolliset toimet. Esimerkiksi myöhemmin kappaleessa 4.3 esiteltävä virtuaalikoneiden migraatio reunasolmulta toiselle on reuna-alustan vastuulla. Reunasovelluksien lisäksi reuna-alusta voi itsessään tarjota jonkinlaista toiminnallisuutta, joka ei suoranaisesti ole reunasovellus vaan esimerkiksi kommunikaatioväylä laitteelta-laitteelle (machine-to-machine) esimerkiksi ajoneuvojen väliselle viestinnälle.

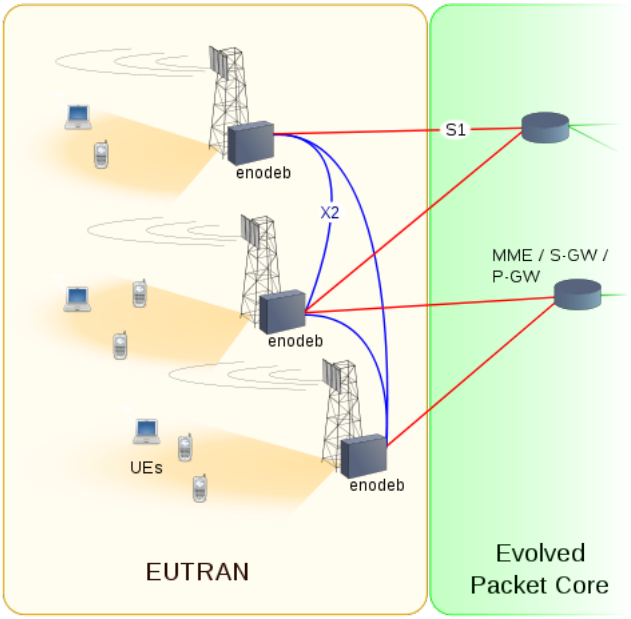
**Reunasovellus** (Edge application) on yksittäinen reunasolmulla suoritettava ohjelmisto, jonka kuluttajana voi toimia asiakaslaite tai toinen reunasovellus. Reunasovellus ei ota kantaa millaista palvelua sillä tuotetaan ja sen rajoitteet asettaa pääasiassa käytössä oleva reuna-alusta. Reunasovelluksen

tuottama palvelu voi olla käyttäjäkohtainen, esimerkiksi virtuaalikone johon käyttäjä voi ottaa etäyhteyden tai tukevaa laskentaa, esimerkiksi kasvojen tunnistusta suorittava palvelinohjelma. Esimerkki ei-käyttäjäkohtaisesta reunasovelluksesta olisi pelipalvelin, jota voivat käyttää reunasolmun lähistöllä olevat pelaajat.

### 2.2.3 Pilvi

Pilvellä tarkoitetaan internetin ytimen läheisyydessä sijaitsevaa aluetta. Pilven voidaan ajatella laajenevan kohti reunaa. Koska pilven ja reunan raja ei ole selkeä ne voivat olla limittäin. Tavallisessa asiakas-palvelin -mallissa palvelun tuottavan palvelimen voidaan ajatella sijaitsevan pilvessä. Kun mukaan otetaan reunajärjestelmä, palveluiden tuottaminen hajautuu reunan ja pilven kesken. Reunanjärjestelmän ei ole tarkoitus korvata pilveä vaan täydentää sitä.

### 2.2.4 Mobiiliverkko




Kuva 1: Mobiiliverkon rakenne

MEC ehdotuksia tarkasteltaessa, olemassa olevien mobiiliverkkojen arkkitehtuurit ovat keskeisessä asemassa. Useat reunalaskentaa käsittelevät ehdotukset on tarkoitettu integroitaviksi osaksi olemassa olevaa mobiiliverkkoarkkitehtuuria. Erityisesti asiakaslaitteiden ollessa mobiililaitteita ja mahdollisimman pieniä viiveitä tavoiteltaessa, reunalaskenta ratkaisujen in-

tegroiminen osaksi mobiiliverkkoa on väistämätöntä. Seuraavaksi käydään pääpiirteittäin läpi 4G mobiiliverkon arkkitehtuurin funktionaaliset osat.

Yksinkertaistettuna mobiiliverkko koostuu kahdesta osasta: radiorajapinnasta ja mobiilinverkon runkoverkosta (puhelinkeskuksesta?). 3GPP kehittämässä LTE standardissa radiorajapinnan sisältävä osuus on nimeltään E-UTRAN (Evolved UMTS Terrestrial Radio Access Network) ja runkoverkon osuus EPC (Evolved Packet Core).

E-UTRAN tehtävänä on toimia rajapintana asiakaslaitteen ja EPC:n välillä. Asiakaslaitteiden suuntaan yhteys on radiosignalointia ja yhtenä E-UTRAN tehtävänä onkin radioresurssien hallinointi. E-UTRAN sisältää verkon puolella pääasiallisena toimijana eNodeB tyyppisiä tukiasemia [ETSI, 2017a], myös muutamia poikkeustapauksia on, mutta ne jätetään käsittelemättä. Tukiasema on asiakaslaitetta lähimpänä sijaitseva funktionaalinen verkon osa ja sen seurauksena se on houkutteleva kohde MEC ratkaisuille. Tukiasemaa voikin ajatella *reunan* viimeisenä etappina ennen asiakaslaitetta. Tämä tutkielma käsittelee pääasiassa ratkaisuja, jotka keskittyvät LTE verkkoihin kohdistuvia ratkaisuja, joten tukiasemista puhuttaessa tarkoitetaan nimenomaan E-UTRAN mukaista eNodeB:tä.

ENodeB:n tehtävänä on kommunikoida radioyhteyttä käyttäen asiakaslaitteen kanssa ja välittää molemminsuuntaista liikennettä EPC:n (Evolved packet core) suuntaan S1 rajapinnan kautta. eNodeB:t ovat myös yhteydessä toisiinsa X2 rajapintaa käyttäen. X2 rajapintaa käytetään tukiasemien väliseen kommunikointiin, joka sisältää handoverin yhteydessä tehtävää asiakaskontekstin siirtoa ja erinäisiä muita hallinnollisia toimintoja. Handoverin vaikutukset näkyvät myös MEC puolelle ja handoveria toimintaa käsitellään tarkemmin kappaleessa XYZ.

Mobiiliverkkojen tietoliikenne koostuu kahdesta eri tasosta: kontrollikerroksesta ja datakerroksesta. Kontrollikerroksella on hallinnointiin liittyviä protokollia joiden tehtävänä on välittää esimerkiksi asiakaslaitteeseen liittyviä kontekstitietoja entiteetiltä toiselle. Datakerros puolestaan välittää IP liikennettä asiakkaan ja palveluiden välillä. EPC koostuu kolmesta alikomponentista jotka ovat MME (Mobility Management Entity), SGW (Serving Gateway) ja PGW (PDN gateway) [ETSI, 2016a]. SGW:n eli palveluyhdyskäytävän tehtävänä on välittää datakerroksen liikennettä asiakaslaitteen ja ulkoisten palveluiden välillä. SGW on yhdistettynä PGW:hen, jonka tehtävänä on toimia IP tason reitittimenä EPC:n ja ulkoisen verkon välillä. PGW toimii asiakkaan ulkoverkon yhteyksien kiintopisteenä [Firmin, 2017]. MME on EPC:n hallinnollinen komponentti ja se toimii ainoastaan kontrollikerroksessa.

## 2.3 Reuna-arkkitehtuuri

Tässä tutkielmassa käsiteltävät arkkitehtuurit ovat pääasiassa tyypiltään oletusarkkitehtuureja (framework). Tämänkaltaisen arkkitehturin tarkoituk-

sena on jakaa järjestelmä toiminnallisiin osiin fyysisellä ja ohjelmallisella tasolla [Tuovinen, 2017]. Se siis kuvaa järjestelmän rakenneosien tehtävät. Oletusarkkitehtuuria voidaan hyödyntää varsinaista toteutettavaa järjestelmää suunniteltaessa.

Kappaleessa 2.2.2 kuvattujen reunan toimijoiden osalta reuna-arkkitehtuurit keskittyvät kuvaamaan reunasolmun ja reuna-alustan tehtäviä. Lisäksi reuna-arkkitehtuurit kuvaavat koko järjestelmän osana toista järjestelmää, jolloin reunajärjestelmä itsessään on komponentti suuremmassa järjestelmässä. Tällä tasolla tarkasteltu arkkitehtuuri jättää avoimeksi reunasovelluksien toiminnan ja keskittyy kuvaamaan reunasovelluksien sijaintia järjestelmässä.

Tämän tutkielman puitteissa arkkitehtuurilla tarkoitetaan reuna-alustan ja reunasolmujen muodostamaa järjestelmää. Reuna-alustan toiminnallisuuksia voisi eritellä vielä tarkemmin, etenkin reunasovelluksien hallinnan osalta. Cloudlet on konkreettinen toteutus reuna-alustan reunasovelluksia hallinnoivasta osasta, mutta se ei kata kuin yhden reunasolmun kerrallaan. Tämän lisäksi olemassa on reunasolmujen välisestä hallinnasta vastaava kerros, tosin tämä kerros voi olla ulkoistettu erillisille hallinnolliselle toimijalle, jolloin hallinnollinen osa reuna-alustasta on varsinaisten reunasolmujen ulkopuolella.

Reunasovelluksien toimintaa ei käsitellä tarkemmin. Taxonomy paperissa on hyvä jaottelu erilaisista sovelluksista. Tämä aihepiiri laajenee sovelluksien jakamiseen osittain reunalla suoritettaviin (offloading) tai mobiilisovelluksen suoritusta tukeviin sovelluksiin.

# 3 Mahdollistavat teknologiat

Seuraavaksi käsitellään reunalaskennan mahdollistavat teknologiat SDN, NFV ja virtualisointi. Tähän esiteltäväksi valittujen teknologioiden joukko perustuu tämän tutkielman kappaleessa **??** esitettyjen arkkitehtuuriratkaisujen yhteydessä esiintyneisiin teknologioihin. Edellä mainittujen teknologioiden lisäksi reunalaskentaa käsittelevien julkaisujen yhteydessä on esitetty myös muita teknologioita kuten verkon viipalointi (network slicing) [Taleb et al., 2017], jonka käsittely sivuutetaan tämän tutkielman yhteydessä.

## 3.1 Virtuaalikoneet

Virtualisoinnilla tarkoitetaan tietokoneohjelmiston suorittamista ohjelmallisesti toteutetun rajapinnan päällä. Toisin sanoen virtualisointi erottaa suoritettan ohjelmiston suorittavasta laitteistosta. Usein virtualisoitava ohjelmisto on kokonainen käyttöjärjestelmä, jolloin suoritettavaa instanssia kutsutaan virtuaalikoneeksi. Virtualisointi mahdollistaa laitteiston resurssien jakamisen useamman toisistaan erillään olevan virtuaalikoneen kesken. Juurikin resurssien jakamisen mahdollisuus yhdessä palvelinlaitteiston suori-

tuskyvyn kasvun kanssa on johtanut virtuaalikoneiden käytön yleistymiseen palvelinsaliympäristössä.

Virtuaalikoneen ja laitteiston välillä toimivaa ohjelmistoa kutsutaan hypervisoriksi. Hypervisorin pääasiallinen tehtävänä on tarjota rautatason resursseja virtuaalikoneiden käytettäväksi. Yksi hypervisorin tehtävistä on virtuaalikoneiden luominen. Sen avulla voidaan määrittä virtuaalikoneen käytössä olevat resurssit, esimerkiksi virtuaalikoneen käytettävissä olevan muistin tai prosessorien määrä. Hypervisorilla on myös monia muita toiminnallisuuksia, joita ei tässä tutkielmassa käydä läpi tämän enempää.

Virtuaalikoneet ovat vahvasti esillä reunalaskenta-arkkitehtuurien yhteydessä. ETSI:n reunalaskennan referenssiarkkitehtuurissa [ETSI, 2016b] virtuaalikoneet esitetään reunapalvelun tuottamisen välineenä ja Taleb et al. [Taleb et al., 2017] esittävät kirjallisuuskatsauksessaan virtuaalikoneet yhdeksi reunalaskennan mahdollistavista teknologioista. Virtuaalikoneiden dynaamisuus verrattuna tavallisesti suoritettavaan ohjelmistoon on reunalaskennan näkökulmsta haluttu ominaisuus. Virtuaalikoneet tarjoavat myös helpon tavan jakaa reunasolmun resursseja usean toisistaan erillisen palvelun välillä. Toisena etuna on virtuaalikoneiden vahva eristys muusta järjestelmästä ja muista virtuaalikoneista.

Virtuaalikoneet eivät aseta rajoitteita tarjottavan palvelun tyypille ja niiden avulla onkin mahdollista tarjota monia erilaisia palvelumalleja. Yksi ehdotetuista palvelumalleista on käyttäjäkohtaisen virtuaalikoneet [Satyanarayanan et al., 2009, Wang et al., 2015].

Tulevissa kappaleissa perehdytään tarkemmin virtuaalikoneiden hyödyntämistä osana reunapalveluita. Kappaleessa 4.3 käsittellään virtuaalikoneiden siirtelyä suorituslaitteistolta toiselle, ilman että itse järjestelmän suorittamista tarvitsee keskeyttää siirron ajaksi. Kappaleessa 6.1 käsitellään cloudletiksi nimettyä virtuaalikoneisiin pohjautuvaa reunapalvelun tuottamisjärjestelmää.

### Säiliöt

Koska virtuaalikoneet eivät jaa ohjelmisto- tai käyttöjärjestelmäresursseja, saattavat samat kirjastot olla ladattuna muistiin useamman kerran – jokaiselle virtuaalikoneelle omansa. Onkin siis huomattava, että kokonaisen käyttöjärjestelmän virtualisointi on hyvin resurssi-intensiivistä. Tällöin yksittäisen virtuaalikoneen käyttäminen esimerkiksi yksittäisen kevyen palvelinsovelluksen suorittamiseksi voitaisiin ajatella resurssien tuhlaamiseksi.

Käyttöjärjestelmä-tason virtualisointi mahdollistaa sovelluksien suorittamisen omalla eristetyllä muistialueellaan. Tälle toiminnallisuudelle on useita nimityksiä, mutta tässä tutkielmassa käytetään termiä säiliö (container). Säiliö on virtuaalikonetta kevyempi ratkaisu. Tämä johtuu siitä että säiliön suoritus tapahtuu isäntäjärjestelmän päällä ja täten se kykenee hyödyntämään isäntäjärjestelmän toimintoja kuten käyttöjärjestelmäkutsuja. Rajusti

yksinkertaistaen säiliö on vain tiukat rajat sisältävä suoritusympäristö sovellukselle.

Säiliöiden käyttöä on ehdotettu myös reunalaskentaympäristöön [Pahl and Lee, 2015]. Säiliöiden eduiksi laskettakoon jo edellä mainittu keveys. Se näkyy sekä säiliön käyttöönottossa, että pienempänä yleisrasitteena sovelluksia suoritettaessa. Säiliöiden nykyinen hallinnointimenetelmä ei kuitenkaan ole täysin yhteensopiva reunalaskentaympäristön asettamille vaatimuksille ja täten se vaatisi oman hallinnointijärjestelmän kehittämisen [Farris et al., 2017a].

Kaiken kaikkiaan säiliöt esittävät hyvän kandidaatin virtuaalikoneiden korvaajaksi. Tässä tutkielmassa reunalaskennan käsittelyä jatketaan virtuaalikoneita hyödyntävien ratkaisujen parissa. Joskaan tämä ei tarkoita, etteikö suuri osa esiteltävistä ratkaisuista olisi sovellettavissa myös säiliöihin.

## 3.2 Software-defined networking

Perinteinen tietoliikenneverkko koostuu joukosta erikoistuneista verkkolaitteita. Tällaisia laitteita ovat esimerkiksi kytkin, palomuuri ja reititin. Näiden laitteiden joukko muodostaa hajautetun rakenteen, jossa jokainen laite täytyy konfiguroida erikseen. Laitevalmistajat tarjoavat hallinnointityökaluja, joiden avulla valmistajan omia laitteita on mahdollista konfiguroida suljettua rajapintaa käyttäen. Verkkoinfrastruktuurin hallintaa kuitenkin vaikeuttaa eri laitteiden konfiguraatioiden erilaisuus, sekä puute kokonaisvaltaiselle ohjelmalliselle konfiguroitavuudelle. Tästä johtuen verkkoon tehtävät muutokset ovat työläitä ja riskialttiita [Kreutz et al., 2015].

Perinteisessä tietoliikennelaitteistossa tiedonvälityskerros ja kontrollikerros ovat tiukasti toisistaan riippuvaisia. Tiedonvälityskerroksella tarkoitetaan itse tietoliikennepakettien välittämiseen käytettävää kerrosta, eli tietoliikennettä ohjaavia laitteistoja. Kontrollikerroksella tarkoitetaan tietoliikenteen reitittämiseksi käytettävää logiikkaa, kuten esimerkiksi reititystauluja. Kontrollikerros on siis tällä hetkellä hajautettuna laitteiston mukana ympäri verkkoa. Tästä johtuen verkko on usein hyvin staattinen ja muutokset kankeita.

SDN (Software-defined networking) eli ohjelmallisesti määritetty verkko on yleistyvä paradigma verkkoympäristöissä. SDN on ratkaisu, jossa tiedonvälitys- ja kontrollikerros on erotettu toisistaan. SDN:ssä ei ole erikoistunutta verkkolaitteistoa vaan nykyinenen tiedonvälitykseen käytetty laitteisto korvattaisiin yleisillä reitittävillä laitteilla[1]. Kontrollikerroksen toiminnasta vastaa SDN Controller. Se on loogisesti keskitetty entiteetti, joka vastaa näiden reitittävien laitteiden ohjaamisesta.

SDN Controllerin ja reitittävän laitteiston välille oletetaan hyvin määritelty rajapinta, jonka kautta reitittäviä laitteita voidaan hallita [Kreutz et al., 2015]. SDN siis toteuttaa *Separation of Concerns* -periaatetta jakamalla verkon

[1]Reitittämisellä tarkoitetaan tässä yhteydessä pakettien ohjausta ja välitystä yleisessä mielessä

reititysmäärittelyjen konfiguroinnin ja itse laitteistopohjaisen toteutuksen omiin osiinsa. SDN Controller tarjoaa rajapinnan ylöspäin ohjelmalliselle verkkokonfiguroinnille ja hoitaa sääntöjen tulkkaamisen alaspäin. OpenFlow on yksi tunnetuimmista SDN Controllerin ja verkkolaitteiden välisestä protokollasta[2].

Tarve SDN pohjaisille ratkaisuille kumpuaa jo aiemmin mainitusta konfiguraation työläydestä ja dynaamisuuden tarpeesta. Esimerkiksi virtuaalikoneiden käytön yleistyessä tarve verkon ohjelmalliselle hallittavuudelle on kasvanut [Jammal et al., 2014]. Virtuaalikoneiden sijainti verkossa ei välttämättä ole kiinteä, vaan virtuaalikone saattaa siirtyä esimerkiksi migratoinnin seurauksena. Virtuaalikoneita saattaa tulla ja poistua verkosta. Perinteisessä verkkoympäristössä esimerkiksi MAC osoitetaulujen päivittäminen saattaa aiheuttaa yhteyskatkoksia palvelimeen [Jammal et al., 2014]. SDN pohjaisia ratkaisuja on jo olemassa, mutta niiden käyttö ei vielä ole korvannut perinteisiä verkkolaitteita.

## 3.3 Network Function Virtualization

NFV (Network function virtualization) lähtee liikkeelle ongelmasta, jossa nykyisen verkkolaitteiston käyttöikä on lyhyttä ja toiminnot ovat hajautettuina useisiin proprietary (omisteisiin?) laitteisiin [ETSI, 2012].

NFV:n tarkoituksena on eriyttää verkkolaitteisto ja verkkotoiminnot. Tämä toteutuisi siten, että erillistä verkkolaitteistoa ei enää tarvittaisi ja nykyisten verkkolaitteiden toiminnallisuudet toteutettaisiin tavallisella palvelinlaitteistolla ohjelmatasolla. Periaate on hyvin samankaltainen kuin perinteisten palvelimien siirtyminen virtuaalikoneita hyödyntävään ympäristöön. Virtualisoidulla verkkotoiminnallisuudella (jossain yhteyksissä VNF, Virtualized network function) tarkoitetaan ohjelmallisesti toteutettua verkkotoimintoa. Tämä mahdollistaa verkkotoiminnallisuuksien suorittamisen tavallisella palvelinlaitteistoilla hyprvisorin päällä.

Verkkotoimintojen toteuttaminen virtuaalisina, mahdollistaisi useamman toiminnon sijoittamisen samaan laitteistoon. Tämä ainakin teoriassa mahdollistaisi myös paremman skaalautuvuuden. Myös verkkotoiminnallisuuksien käyttöönotto helpottuu mikäli erillisen laitteiston asennusta ei tarvita. Virtuaalisten verkkotoimintojen avulla on myös mahdollista säästää kaappitilaa, sekä pienentää sähkönkulutusta [Nguyen and Cheriet, 2016].

NFV on soveltuva mille tahansa data-tason prosessoinnille ja kontrollitason toiminnoille [ETSI, 2012]. NFV siis soveltuu toteutustavaksi monille erilaisille verkkoitoiminnoille mobiiliverkoissa ja perinteisissä tietoliikenneverkoissa. Esimerkkeinä käyttökohteista ETSI:n NFV whitepaperissä on esitetty muunmuassa reitittimet, palomuurit, kuormantasaajat, eNodeB:t

[2] `https://www.opennetworking.org/software-defined-standards/specifications/`

ja MME:t. ETSI on pohtinut NFV:n laajamittaisen käyttöönoton mahdollisuuksia 5G-verkkoingrastruktuurissa ja mainitsee reunalaskennan yhtenä sen käyttötapauksena [ETSI, 2017b].

Mobiiliverkossa toimivan reunalaskennan näkökulmasta NFV:n potentiaali on ilmeinen. Jos suurin osa mobiiliverkon toiminnoista voitaisiin virtualisoida ja siirtää tavalliselle palvelinlaitteistolle, voisi reunalaskentaa suorittaa samalla laitteistolla, eikä se vaatisi erillistä laitteistoa omille toiminnoilleen.

NFV:n haasteet? Periaatteessa vielä hyvin epäkypsä tekniikka, mutta suunniteltu 5G integraatio.

# 4 Ominaisuudet

Reunalaskennan keskeisin tarkoitus on laskennan siirtäminen reunalla toimivalle reunalaskentaklusterille. Reunaa voidaan lähestyä pilven ja käyttäjälaitteiden puolelta. Käyttäjälaitteiden, kuten älypuhelimien laskentateho on suhteellisen heikkoa, lisäksi ne ovat akkuvirrasta riippuvaisia. Mobiililaitteen käyttöajan pidentämiseksi voidaan pyrkiä tekemään mahdollisimman vähän akkuakuluttavaa laskentaa paikallisesti, siirtämällä sitä reunalaskentaklusterille (Etsi se lähde jossa verrataan tietokoneita ja mobiililaitteita - eroa oli yhden kertaluokan verran). Esimerkiksi verkkoliikenne mobiililaitteen ja kohdepalvelimen välillä on merkittävä viive toisi reunaklusteri palvelun huomattavasti lähemmäksi ja pienentäisi viivettä palvelussa. Viiveen pienenemisen seurauksena monet reaaliaikaisuuttaa tai nopeaa reagointia vaativat palvelut ovat mahdollisia. Lisäksi verkon viiveestä tai ruuhkasta johtuen, mobiililaitteella on usein huomattavasti nopeampi yhteys fyysisesti lähelle itseään verra

Lisäksi pilven tai konesalien suunnasta asiaa lähestyttäessä runkoverkko tukkeutuu. Siirtämällä osan palveluvastuusta reunalle, runkoverkon rasitteen tulisi ainakin periaatteessa pienentyä.

## 4.1 Kommunikaatio

Reunan keskeisin haaste kommunikaatiossa on asiakaslaitteen ja reunasolmun välisen kommunikaation ylläpitäminen.

## 4.2 Etälaskenta

Etälaskennan toteuttamiseksi tarvitaan vastaukset seuraaviin kysymyksiin. Mitä siirretään ja minne siirretään?

MOCAssa oli selitetty kuinka muodostetaan yhteys in-network cloudiin.

Etälaskenta voidaan karkeasti jakaa kahteen tyyppiin: Binääriseen ja osittaiseen [Mao et al., 2017]. *Onkohan ok lainata surveytä?*. Binäärisessä ohjelmasta suoritetaan selkeitä kokonaisuuksia joko reunasolmulla tai asia-

kaslaitteella. Osittaisessa etälaskennassa suoritusta siirretään dynaamisesti reunasolmulle.

Offloading on varmaan samankaltainen termi. Reunalla suoritettava etälaskenta saattaa siirtää ohjelman suorituksen kokonaan tai osittain asiakaslaitteelta reunasolmulle. Laskennan tulos lähetetään takaisin asiakaslaitteelle ja sitä käytetään osana muuta laskentaa. Laskennan hallinta suoritetaan asiakaslaitteella. Etälaskentaa motivoi raskaiden operaatioiden siirtäminen asiakaslaitteelta reunalle. Erityisesti mobiililaitteilla akkuvirran säästäminen on keskinen tekijä. Etälaskennan kannattavuus puhtaasti akkuvirran näkökulmasta muuttuu kannattavaksi, kun suoritettavan ohjelman lähettäminen ja tuloksen vastaanottaminen kuluttavat vähemmän akkua kuin ohjelman suorittaminen paikallisesti asiakaslaitteella. Todellisuudessa pelkästään akkuvirran säästäminen ei riitä, sillä muuten laskentaa voitaisiin siirtää pilveen. QoS kuitenkin heikkenee, mikäli laskennan suorittamiseen kuluva aika pitenee huomattavasti. Reunalle on teoriassa nopeampi yhteys ja nopeampi vastaus. Voitaisiin siis laskea että suoritusajassa mitaten ohjelman siirtäminen reunalle on kannattavaa kun suoritettavan ohjelman lähettäminen palvelimelle, sen suorittaminen ja tuloksen vastaanottaminen kestävät vähemmän aikaa kuin ohjelman suorittaminen paikallisesti. Ongelmana on että suorituksien aikavaatimus ei ole eksakti vaan ainoastaan arivoitavissa. Lisäksi suoritusaikainen aika-arvion tekeminen vie myös aikaa. Ajan ja akkuvirran säästämiseksi tehtävät toimet ovat siis keskeisimmät haasteet etälaskentaa toteutettaessa. Näiden käsittelyä ei tämän enempää tässä tutkielmassa käsitellä niiden monimutkaisuuden vuoksi.

## 4.3 Live migraatio

> Tarkista käsitelläänkö myös tapaus jossa virtuaalikonetta ei liikuteta vaan ainoastaan varmistetaan yhteyden säilyminen?

Live migraatio, eli suorituksen aikanen siirto, tarkoittaa ohjelman tai virtuaalikoneen siirtämistä laitteistolta toiselle, siten että ohjelman tai virtuaalikoneen käyttö ei keskeydy. Useimmiten tällaista toiminnallisuutta käytetään palvelinkeskuksissa virtuaalikoneiden siirtelyyn. Palvelinkeskuksissa siirtämiseen käytetään nopeita sisäisiä yhteyksiä, jolloin tiedon siirtoon käytettävät väylät ovat nopeita. Syitä live migraation suorittamiseksi on muutamia. Perinteisessä palvelinympäristössä live migraation syinä ovat virrankulutuksen optimointi, kuorman tasaaminen tai fyysisen laitteiston huoltoon ottaminen [Soni and Kalra, 2013].

Reunalaskentaan sovellettuna live migraation tehtävänä on laskentaa tai palvelua suorittavan virtuaalikoneen siirtäminen reunasolmujen välillä. Syyt joiden vuoksi reunalaskentaympäristössä tehdään live migraatiota koskevat pääasiassa asiakaslaitteen liikkumista verkossa, jolloin virtuaalikone siirretään asiakaslaitetta lähimpänä olevalle reunasolmulle. Reunalaskentaa suori-

tettaessa voi myös tulla tilanne, jossa laskentaa suorittavalla reunasolmulla ei ole resursseja laskennan suorittamiseksi. Tällöin asiakkaan virtuaalikone joudutaan siirtämään toiselle reunasolmulle, jolla on enemmän resursseja.

### Handover/handoff

Virtuaalikoneille tehtävän live migraation ja mobiiliverkossa suoritettavan handoverin ideat ovat samankaltaisia. Handoverilla tarkoitetaan asiakaslaitteen yhteyden siirtämistä tukiasemalta toiselle tukiasemalle. Handover tehdään yleensä tilanteessa, jossa asiakaslaite on liikkunut mobiiliverkossa siten että se on lähempänä toista tukiasemaa. Handover voidaan myös tehdä mikäli tukiasema on ruuhkautunut. [**?**] 4G LTE:ssä yhteys on tyypiltään tunnelimainen ja handoverissa tunnelin toinen pää siirretään toiselle tukiasemalle. Yhteyden siirron alkaessa tukiasema välittää kohteena olevalle tukiasemalle asiakaslaitetta koskevat tilatiedot. Kun tilatiedot on välitetty, asiakaslaitteen ja tukiaseman välinen yhteys katkaistaan ja asiakaslaite muodostaa yhteyden uudelle tukiasemalle. Asiakaslaitteen tietoliikennettä puskuroidaan S-GW:n toimesta sillä välin kun yhteys on katkaistuna. Yleensä handover on niin nopea toimenpide, että laitteen käyttäjä ei sitä huomaa. Handover on nopea toimenpide koska sen aikana siirrettävän tiedon määrä on vähäinen ja se koostuu pääasiassa erilaisista asiakaslaitteen ja tunneleiden tunnisteista.

### Live migraation toiminta

Live migraatiossa virtuaalikone siirretään palvelimelta toiselle palvelimelle, ilman että virtuaalikoneen käyttö keskeytyy. Perinteinen live migraatio on toteutettu siten, että virtuaalikoneen suoritustilan eli prosessorin tilan ja muistin siirretään toiselle palvelimelle.

Siirtäminen suoritetaan iteraatioittan siten, että ensimmäisellä iteraatiolla kaikki muistisivut siirretään kohdelaitteelle. Seuraavilla kierroksilla lähtölaitteelta siirretään vain ne muistisivut joille on tapahtunut muutoksia edellisen siirron alkamisen jälkeen. Tätä jatketaan kunnes muutoksia sisältävien muistisivujen määrä ei vähene iteraatioittan. Tässä vaiheessa lähtöpisteenä oleva virtuaalikone pysäytetään ja loput virtuaalikoneen muistisivut ja tilatiedot siirretään kohdelaitteelle. Tämän jälkeen virtuaalikone käynnistetään uudessa sijainnissa.

Live migraatioon liittyy myös erilaisia optimointeja ja lähestymistapoja joita ei tässä tutkielmassa sen tarkemmin avata. Konesali ympäristössä on myös yleistä että varsinaista tallennustilaa ei ole tarpeen siirtää, koska se on toteutettu levypalvelimen avulla, jolloin ainoastaan yhteys täytyy siirtää. Mikäli näin ei ole, myös tallennustila tulee siirtää laitteelta toiselle.

Live migraation suorituskykyä mitataan seuraavilla metriikoilla [Soni and Kalra, 2013]:

- Migraation kesto: Aika joka kuluu migraation suorittamiseen

- Katkon kesto: Aika jona palvelut eivät käytettävissä
- Siirretyn datan määrä: Migraatiosta aiheutuva tietoliikenteen määrä
- Migraation yleisrasite: Paljonko migraatio vie järjestelmän resursseja
- Suorituskyvyn alentuma: Siirrettävän virtuaalikoneen suorituskyvyn heikentyminen siirron aikana

Näistä kolme ensimmäistä ovat keskeisimpiä [Farris et al., 2017b]. Tavoitteena perinteisessä live migraatiossa on mahdollisimman lyhyt käyttökatko. [Ha et al., 2015]

#### Reunalaskennassa

Reunalaskentaympäristössä suoritettavan live migraation toiminnallisuus on pääpiirteittäin sama kuin edellä kuvattu. Keskeisenä erona palvelinsaliympäristöön on juurikin suoritusympäristö.

Päällimmäisenä erona on palvelinsaleissa tehtävään migraatioon on käytettävien yhteyksien nopeus. Reunasolmujen väliset yhteydet ovat oletettavasti hitaampia ja nopeus saattaa vaihdella[Ha et al., 2017]. Hitaat yhteyden johtavat pitkään siirtoaikaan, joka puolestaa johtaa pidempään käyttökatkokseen ja täten palvelun laadun heikkenemiseen. Siirtoajan minimoimiseksi on siis pyrittävä pitämään siirrettävän datan määrä mahdollisimman pienenä.

Palvelinsaleja ja reunalaskentaympäristöä erottaa myös se, että palvelinsaliympäristössä virtuaalikoneiden live migraatiot voidaan pääsääntöisesti tehdä koordinoidusti ilman tiukkkoja aikarajoitteita. Reunalaskentaympäristössä migraatiopäätös perustuu usein johonkin ulkoiseen tapahtumaan. Todennäköisintä onkin että migraatiopäätös tehdään samankalataisin perustein kuin edellä esitelty handover. Keskeisin syy migraatiolle oletettavasti on asiakaslaitteen liikkuminen verkossa. Asiakaslaitteiden liikkumiseen pohjautuva migraatio yhdistettynä heikkoihin tietoliikenneyhteyksiin on kuitenkin ongelmallinen. Voidaan esimerkiksi kuvitella tilanne jossa aamulla kaupungin keskustaan saapuvat työmatkailijat aiheuttavat "migraatiotulvan", joka ruuhkauttaa reunasolmujen käytössä olevat tietoliikenneyhteydet.

Lähes kaikki läpikäydyt reunalaskenta-arkkitehtuurit tiedostivat tarpeen laskennan migraatiolle, mutta kovin monessa sen tarkempaa toiminnallisuutta ei määritelty. Cloudletit ovat yksi tunnetuimpia reunasolmujen välisten migraatioiden ratkaisumalleja. Cloudlettien live migraatio ratkaisuja on käsitelty tarkemmin tulevassa kappaleessa 6.1.1. MobiScud:n yhteydessä käytettiin Xen hyperviisorin migraatiotoiminnallisuutta suoraan sovellettuna reunalaskentaympäristöön[Wang et al., 2015].

## 4.4 Integraatio mobiiliverkkoihin

> Tarkista onko käsitelty myös niin että laitteistovaatimukset tulevat esiin. Esimerkiksi vaatiiko reunalaskennan toteuttaminen erikoisvalmisteisia laitteita vai selvitäänkö tavallisilla palvelimilla?

Useat reunalaskentaan liittyvät arkkitehtuurit on suunniteltu integroitaviksi osaksi mobiiliverkkoa. Motivaatio on ilmeinen, koska lähes kaikki mobiiliverkossa toimivat laitteet kuuluvat reunalaskennan kohderyhmään. Reunalaskentatoiminnallisuutta lisättäessä olemassa olevaan järjestelmään, yhdeksi suurimmista tekijöistä tulee hinta. Kuten aiemmin mainittu, mobiiliverkon laitteisto on usein suljettu, eikä siihen välttämättä ole mahdollista tehdä muutoksia tai laajennoksia. Tämä tarkoittaa että ainakin osa olemassa olevasta laitteistosta jouduttaisiin korvaamaan uudella, mikäli reunalaskentatoiminnallisuus halutaan lisätä. Tämän vuoksi on esitetty tekniikoita joiden avulla laitteiston uusimiselta voidaan välttyä, tai ainakin minimoida korvattavan laitteiston määrä. Yksi erottavata tekijä on reunalaskentaratkaisun sidonnaisuus mobiiliverkkoon. Tämä tarkoittaa että löyhästi sidonnaisten ratkaisujen tuottajana voi jokin kolmas osapuoli, mutta yleisesti tämä myös tarkoittaa että reunapalvelu tuotetaan kauempana reunasta. Jokaisessa ratkaisussa on siis hyvät ja huonot puolensa.

Tavat joilla reunalaskenta voidaan lisätä osaksi mobiiliverkkoa voidaan jakaa kolmeen pääryhmään

- **Suorat integraatiot** sisältävät uusien toimintojen lisäämistä osaksi olemassa olevaa arkkitehtuuria. Tämänkaltainen ratkaisu edellyttää muutoksia myös olemassa olevien komponenttien toimintaan.
- **Epäsuorat itegraatiot** eivät edellytä toiminnallisia muutoksia mobiiliverkon toimintoihin.
- **Läpinäkyvät integraatiot** vaativat muutoksia mobiiliverkkoon, mutta eivät vaadi muutoksia olemassa olevien komponenttien toimintaan.

Suorat integraatiot edustavat niiden ratkaisujen joukkoa jotka muokkaavat olemassa olevan mobiiliverkkoarkkitehtuuria. Näiden ratkaisujen voidaan olettaa olevan kalleimpia, koska yhteistoiminnallisten toimijoiden lisääminen olemassa olevaan infrastruktuuriin vaatii muidenkin toimijoiden päivittämistä. Pääasiassa nämä ratkaisut tähtäävät lisäämään reunalaskentaa viidennen generaation mobiiliverkkoihin, mutta myös LTE:hen pohjautuvia ratkaisuja on esitetty. Koska viidennen generaation määrittelytyö on vielä kesken, ehdotetut ratkaisut rakentuvat osittain oletuksien päälle. Esimerkkinä suorasta integraatiosta on CONCERT (esitellään kappaleessa **??**). Suora integraatio mahdollistaa reunalaskennan tuomisen niin lähelle reunaa kuin mahdollista.

Epäsuoralla integraatiolla tarkoitetaan ratkaisuja, joiden pääasiallinen toiminnallisuus on sijoitetu mobiiliverkko-arkkitehtuurin ulkopuolelle. Tällaiset ratkaisut mahdollistavat reunalaskentainfrastruktuurin tuottamisen kolmansilla osapuolilla. Esimerkkinä tällaisesta ratkaisusta on FMC (käsitellään tarkemmin kappaleessa 6.2). Haasteena tämänkaltaisissa ratkaisuissa on optimaalisen reunasolmun valinta, koska se joudutaan tekemään asiakaslaitteen antamien tietojen pohjalta. Onkin siis huomattava, että asiakaslaite joutuu osallistumaan reunalaskentaan liittyviin hallinnollisiin toimiin.

Läpinäkyvissä ratkaisuissa reunalaskennan mahdollistavat toiminnot on toteutettu siten että suoria yhteyksia mobiiliverkon toimintoihin ei ole. Periaate on hieman samankaltainen kuin läpinäkyvässä välipalvelimessa. Käytännössä tämä tarkoittaa jonkinlaiseen tietoliikennemonitoriin pohjautuvaa ratkaisua. Monitorin tehtävänä on tarkkailla mobiiliverkon tapahtumia, sekä mahdollistaa halutun tietoliikenteen ohjaaminen reunalaskennalle. Näiden toimintojen toteuttamisen helpottamiseksi käytössä on tai käyttöön oletetaan SDN, NFV tai molemmat. Joskaan läpinäkyvät ratkaisut eivät ole osa mobiiliverkkoa, ne sijaitsevat sen välittömässä yhteydessä. Tämä sitoo ne osaksi mobiiliverkon infrastruktuuria ja käytännössä tämä tarkoittaa että reunapalvelualustan tuottajana on mobiiliverkon-operaattori. Esimerkkinä tällaisesta ratkaisusta on MobiScud, joka käydään tarkemmin läpi kappaleessa 6.4.1. Tämänkaltaiset ratkaisut ovat asiakaslaitteen näkökulmasta "näkymättömiä".

## 4.5 Hallinta

Reunan hallinta koostuu joukosta hallinnollisia toimia toteuttavia entiteettejä. Tämän tutkielman yhteydessä hallinnolliset toimet rajautuvat pääasiassa niihin toimintoihin, jotka jotka mahdollistavat, ylläpitävät tai säätelevät asiakaslaitteen ja reunasolmun välistä kommunikaatiota. Arkkitehtuuritasolla keskeisimmät hallinnolliset toimijat liittyvätkin juuri reunasolmujen saavutettavuuden varmistamiseen. Tämän tukielman esittelyjen ulkopuolelle jäävät siis ne hallinnolliset toimijat, jotka esiintyvät arkkitehtuurikuvauksissa pääasiassa mainintana. Esimerkiksi virtuaalikoneisiin pohjautuvien järjestelmien resurssien oikeanlainen utilisointi [Yousaf and Taleb, 2016, Taleb et al., 2017] on keskeinen haaste johon arkkitehtuuritasolla ei oteta juurikaan kantaa ja sen suunnittelu jätetään järjestelmän toteuttajan vastuulle.

Pääasiassa hallinnollisten toimijoiden rooli esiintyy reunalaskenta-arkkitehtuurissa passiivisena tarkkailijana, jonka tehtävä on reagoida mobiiliverkon tapahtumiin. Esimerkki tällaisesta tapahtumasta on mobiiliverkossa tapahtuva handover, johon hallinnollinen entiteetti saattaa haluta reagoida esimerkiksi aloittamalla reunalaskennan migraation lähemmäksi käyttäjää.

Arkkitehtuurien esittelyiden yhteydessä hallinnosta vastaava entieetti esitellään kaikissa arkkitehtuureissa. Hallinnosta vastaavat entiteetit arkkitehtuureittain ovat SCM (Small Cell Cloud Manager) Small Cell Clou-

dissa [Lobillo et al., 2014, Gambetti et al., 2015], conductor CONCERT:ssa [Liu et al., 2014], FMC Controller FMC:ssä [Taleb and Ksentini, 2013], MC (MobiScud Controller) MobiScud:ssa [Wang et al., 2015], ja edge orchestrator ETSI MEC refenssiarkkitehtuurissa [ETSI, 2016b]. *Tarkista onko kaikki*

Hallinnollisten toimien osalta arkkitehtuuri voi olla hajautettu tai keskitetty. Hajautetuissa malleissa lähestytään vertaisverkko -tyylistä ratkaisua. Siinä hallinnolliset entiteetit organisoivat hallinnolliset toimet keskenään. Hajautetussa mallissa hallinnolliset entiteetit usein vastaavat jostain osajoukosta reunasolmuja ja esimerkiksi virtuaalikoneiden migraation yhteydessä hoitavat hallinnollisen vastuun siirtämisen vertaiselleen. Tämä muistuttaa eNodeB:n kaltaista ratkaisua, jossa handoverin yhteydessä vastuu asiakaslaitteen yhteydestä siirtyy handoverin kohteena olevalle eNodeB:lle. Hallinnollisten toimien ollessa keskitettynä yhdelle entiteetille. Esimerkiksi SCM:n [Lobillo et al., 2014] tapauksessa oletetaan, että SCM:llä on saatavillaan kokonaiskuva verkon tilasta ja sen pohjalta SCM voi tehdä päätöksiä esimerkiksi resurssien käyttöönotosta sekä reunalaskennan migraatiosta.

# 5 Reunan rakenne

Rakenteeseen vaikuttaa ehkä eniten seuraavat kaksi valintaa

- Kerroksittain
- Lähelle vähän vai kauemmaksi enemmän

Molempiin ratkaisuihin liittyy omat haasteensa. Etenkin voidaan olettaa että lähelle hajalleen aiheuttaisi huomattavasti enemmän ylläpitotyötä. [Mao et al., 2017]

## 5.1 Rakenne

Reunan rakenne on osittain arkkitehtuuristen komponenttien määräämä hierarkia. On kuitenkin huomattava että osa reuna-alustoista on niin vapaamuotoisia että varsinaisia rajoitteita ei ole, mutta periaatteen vuoksi tässä tutkielmassa noudatetaan arkkitehtuurin esittäjän ehdottomaa rakennetta, mikäli sellainen löytyy.

fog, 2-tier 3-tier

# 6 Esitetyt ratkaisut

Reunalaskennan toteuttamiseksi on esitetty useita erilaisia ratkaisuja. Sen sijaan että esitettäisiin ratkaisu yksittäiselle toiminnolle, reunalaskenta-arkkitehtuuri pyrkii toteuttamaan tarpeeksi suuren osajoukon toteuttamisen kannalta kriittisistä ominaisuuksita. Seuraavaksi käydään läpi ehdotettuja arkkitehtuureja ja lopuksi tarkastellaan ETSI:n dokumentaatioiden esittämä

reunalaskennan referenssiarkkitehtuuri ja reunalaskennalle asetetut vaatimukset.

## 6.1 Cloudlet

Minkä ongelman cloudlet ratkaisee? Miten cloudlet toimii? Mitkä ovat cloudletin ongelmat?

Cloudlet on reunasolmulla suoritettava virtuaalikone-teknologiaan perustuva ratkaisu. [Satyanarayanan et al., 2009] Virtuaalikoneisiin pohjautuva cloudlet on yksi tunnetuimmista reunapalveluiden tuottamisen konsepteista. Cloudletin ideana on hyödyntää muokattuja virtuaalikone-instansseja, joita voidaan suorittaa reunasolmussa. Cloudletillä tarkoitetaan järjestelmää, joka suorittaa reunasolmulla virtuaalikoneiden hallintaan liittyviä toimia. Yhdellä reunasolmulla voidaan siis suorittaa usean eri käyttäjän virtuaalikoneita.

Reunan luonteesta johtuen cloudletin oletetaan ylläpitävän ainoastaan sellaisia palveluita jotka käyttävät reunaa hetkellisesti tai väliaikaisesti [Satyanarayanan et al., 2009]. Cloudlettiä ei siis ole tarkoitettu pidempiaikaiseen tallennukseen, vaan enemminkin suoritusaikaisten tarpeiden täyttämiseen.

Cloudletin keskeisimmäksi rakennuspalikaksi on valittu virtuaalikone. Verrattuna prosessi tai sovellus tason virtualisointiin, virtuaalikoneiden käyttö ei aseta rajoitteita palvelun toteuttamiseen käytettävään ohjelmointikieleen [Satyanarayanan et al., 2009]. Lisäksi virtuaalikoneiden käyttö cloudlettien perustana mahdollistaa helpon tavan jakaa reunasolmun resursseja ja pitää asikaskohtaiset instanssit toisistaan erillään.

Tavallisen virtuaalikone-kuvan suuruus tallennettuna on noin 8 gigatavua. Koska reunasolmujen pääasiallinen idea on tarjota suoritusympäristö palveluille, ei näin suuren tiedoston tallentaminen reunasolmulle jokaista käyttäjää kohden ole tarkoituksenmukaista. Ja kuten jo aiemmin mainittu, reunasolmujen resurssit oletetaan rajallisiksi verrattuna palvelinsaliympäristöön. Täten pidempiaikainen tiedostojen tallennus on parempi tehdä keskitettyihin ratkaisuihin. Sen lisäksi että tallentaminen reunasolmuille ei ole mahdollista, kokonaisen virtuaalikoneen siirtäminen verkon yli olisi huomattavan hidasta ja se tukkisi verkon. Siirrettävyys korostuu etenkin tilanteessa jossa siirrettäviä virtuaalikoneita olisi useita.

Cloudletin keskeisimpiä oivalluksia onkin käyttää virtuaalikoneissa yhteistä pohjaa. Yhteisen pohjan käyttäminen mahdollistaa sen että jokaisesta virtuaalikoneesta voidaan tallentaa vain pohjaan tehdyt muutokset. Muutokset sisältävää tiedostoa kutsutaan pinnoitteeksi (VM overlay). Pinnoitteen ja pohjan yhdistämiseen käytetään dynaamista virtuaalikone synteesiä. Dynaamisuudella tarkoitetaan sitä, että käyttäjän alkaessa käyttämään cloudlet infrastruktuuri rakentaa käyttäjän muutokset sisältävästä pinnoitteesta ja virtuaalikone-pohjasta, suoritettavan virtuaalikoneen. Dynaaminen virtuaalikoneiden syntetisointi edellyttää että cloudletiltä löytyy

oikeanlainen virtuaalikonepohja. Kun virtuaalikone halutaan sulkea cloudlet tallentaa virtuaalikoneeseen tehdyt muutokset pinnoitteeksi ja pakkaa tiedoston. Satayanaran ensimmäisessä cloudletteja käsittelevässä julkaisussa [Satyanarayanan et al., 2009] muutoksien tallentaminen pinnoitteeksi tuottaa noin 100-200 megatavun kokoisen tiedoston. Tämän kokoinen tiedosto on nykyisillä tiedosiirtonopeuksilla siirrettävissä inhimillisessä ajassa, jopa langattomien yhteyksien yli.

Cloudlettejä voi olla useita erilaisia, siispä cloudlet toiminta edellyttää että jokaisella reunasolmulla tulee olla oikean cloudlet-pohja saatavilla ja käyttäjien cloudlettien on pohjauduttava johonkin rajalliseen määrään erilaisia cloudlet pohjia.

**Hallinta**

### 6.1.1 VM handoff kuten arkittelissa

Cloudlettien välinen migraatio ja handoff. Cloudlettien väliselle migraatiolle voi olla useita eri syitä. Migraatio tilanteessa jossa virtuaalikone siirretään paikasta toiseen odottamaan käyttöönottoa on keskeisin. Pelkän pinnoitteen siirtäminen riittää, mikäli tarvittava virtuaalikonepohja löytyy kohteena olevasta cloudletistä. On kuitenkin huomattava että pinnoiteen(VM overlay) tallettamiseen voi olla kannattavaa käyttää myös asikkaan laitetta, mikäli se vain on mahdollista. Perusteluina asiakaslaitteen käyttölle tallennusmediana voisi olla tapaus, jossa asiakas matkustaa ulkomaille ja pinnoite tulisi siirtää hyvin kaukana sijaitsevalta palvelimelta kohteena olevalle cloudletille. Tällöin saattaisi olla nopeampaa siirtää pinnoite suoraa asiakaslaitteelta.

**RAW – Handoff**

## 6.2 FMC - Follow me cloud

Asiakaslaitteiden määrän kasvu, sekä tietoliikenne-intensiivisien sovelluksien käytön yleistyminen aiheuttaa suurta kuormaa verkkoinfrastruktuurille. Ongelmat johtuvat pääasiassa keskitetystä rakenteesta, jossa tietoliikenneyhteydet kerätään kulkemaan keskitetyn pisteen kautta. Tämän lisäksi yhteyksissä on huomattavasti viivettä, koska asiakkaan ja palvelimen väliset yhteydet ovat pitkiä. FMC ehdottaa ratkaisua jossa operaattorin keskitetty verkkorakenne muutettaisiin hajautetuksi. [Taleb and Ksentini, 2013] Käytännössä tämä tarkoittaisi, että EPC:n ja ulkoverkon välisten yhdyskäytävien, eli P-GW, määrää lisättäisiin. Toinen osa ideaa on, että P-GW:t voitaisiin hajauttaa palvelinsaleihin lähemmäksi asiakkaan käyttämiä palveluita. Tämä siksi, että myös palveluiden tarjoajat ovat huomanneet hajautustarpeen ja alkaneet hajauttaa palveluitaan. FMC:n arkkitehtuuri koostuu hajautetusta EPC:stä ja hajautetusta palvelinkeskuksista. Asiakaslaitteen yhdistäessä verkkoon verkkoyhteyksien kiintopisteenä toimii todennäköisimmin lähimpä-

nä sijaitseva P-GW. Perinteisessä mobiiliverkossa asiakaslaite pysyy samana P-GW:n asiakkaana, vaikka liikkuisi mobiiliverkossa. FMC lähteekin ratkaisemaan ongelmaa jossa asiakaslaitteelle tarjottaisiin aina optimaalisin[3] yhteys tavoiteltuun palveluun.

FMC:n ratkaisemat ongelmat voidaan jakaa kahteen osaa, optimaalisen reitin valitseminen ja optimaalisen reitin ylläpitämiseen käyttäjän liikkuessa.

Optimaalisella reitinvalinnalla tarkoitetaan, että käyttäjän yhteys kohdepalveluun olisi mahdollisimman lyhyt tai nopea. Tämä jakautuu matkaan, jonka asiakkaan yhteys kulkee mobiiliverkossa, eli tarkemmin ottaen EPC:n sisällä, ja matkaan P-GW:ltä kohdepalvelimeen. Tämän saavuttamiseksi mobiiliverkon arkkitehtuurin tarvitaan palvelinkeskuksien ja P-GW välisiä mappauksia tekevä entiteetti.

FMC tarjoaa optimaalisen reitin löytämiseen ratkaisua, jossa mobiiliverkkoon lisättäisiin hallitseva entiteetti – FMC Controller. Optimaalisen asiakaslaite-palvelinsali -yhteyden ylläpitäminen vaatii jonkin verran muutoksia tapaan jolla käyttäjän siirtymiseen mobiiliverkossa reagoidaan. P-GW:n vaihtaminen ja mahdollisesti myös palvelun tarjoavan konesalin vaihtaminen tarkoittavat että käyttäjän ja palvelun välinen IP-yhteys katkeaa osoitteiden muuttuessa. Kuten kappaleessa 6.3.1 mainittiin, perinteisessä mobiiliverkko arkkitehtuurissa käyttäjän liikenne ulkoverkkoon kulkee GTP-tunnelia pitkin ja käyttäjän liikkuminen tukiasemien välillä ei näy asiakkaslaitteen yhteyksissä. FMC Controllerin tehtävänä on hoitaa avattujen yhteyksien yhteys-tunnisteita sekä tehdä päätöksiä palveluiden migratoinnista palvelinkeskuksien välillä asiakkaan liikkuessa.

FMC eroaa muista mobiiliverkon ratkaisuista sillä, että se ei tuo palvelinresursseja osaksi mobiiliverkkoa ja pyrkii tekemään mobiili infrastruktuuriin mahdollisimman vähän muutoksia. Sen sijaan FMC pyrkii takaamaan nopeimman mahdollisen yhteyden haluttuun palveluun. FCM jättääkin auki mahdollisuuden, että reunapalvelut tarjoaa joku muu kuin operaattori itse.

## 6.3 Small Cell Cloud

Small Cell Cloud (SCC, pienisoluinen pilvi?) perusidea on reunapalveluiden integroiminen osaksi LTE-verkkoinfrastruktuuria. Reunaopalveluita tuotettaisiin tukiasemiin (Base Station) sijoitetuilla palvelinresursseilla. LTE mahdollistaa heterogeenisten verkkojen luomisen. Matkapuhelinverkkojen tapauksessa tällä tarkoitetaan sitä, että yksittäisen solun sisällä saattaa olla pienempiä soluja. Verkko saattaa siis koostua useammasta kerroksesta erikokoisia soluja. Kun solut ovat muuttumassa pienemmiksi tukiasemat tulevat fyysisesti lähemmäksi käyttäjää.

SCC tarkoituksena on hyödyntää solujen pienentymistä, tuomalla niihin reunapalveluiden tuottamisen mahdollistavaa teknologiaa. SCC reunalasken-

[3] optimaalisella tarkoitetaan tässä yhteydessä parasta mahdollista, käyttäen mittareina fyysistä sijaintia ja kuormaa

tapalvelut tuotetaan SCeNBce klustereissa. SCeNBce (Small-cell eNodeB computing-enhanced) on palvelinresursseilla varustettu eNodeB tukiasema [Lobillo et al., 2014]. Verrattuna cloudletteihin, SCeNBce:ssä palvelinresurssit ovat tiiviisti osana tukiasemaa.

SCM (Small Cell Manager) on SCC arkkitehtuurissa hallinnasta vastaava komponentti [Gambetti et al., 2015]. Normaalisti LTE verkossa keskeinen hallinnosta vastaava komponentti on Mobility Management Entity (MME). Sen tehtäviin kuuluu on hoitaa muunmuassa MME:ltä toiselle tapahtuvat handoverit ja asiakkaiden autentikointi. Täten on luonnollista että SCM liitetään MME:n yhteyteen, jossei suorasti niin ainakin jonkin yhteyden välityksellä. Eräs ehdotettu tapa toteuttaa SCM onkin integroida se suoraan MME:n yhteyteen jolloin tämä komponentti vastaa sekä reunapalveluista ja LTE-verkon hallinnasta[Lobillo et al., 2014]. SCM tehtävänä on organisoida resurssien utilisointia SCeNBce klustetereiden välillä. Tähän kuuluu esimerkiksi palvelinresurssien sammuttaminen mikäli ne eivät ole käytössä ja mahdollisesti yliutilisoidulta SCeNBce:ltä kuorman siirtäminen toiselle SCeNBce:lle. Tämän lisäksi esimerkiksi virtuaalikoneiden migraatiopäätöksien tekeminen on SCM tehtävä.

Koska SCC:n tapauksessa palveluiden tuottamiseen käytettäisiin matkapuhelinverkkoa, rajaisi se myös asiakaslaitteet älypuhelimiin ja muihin tätä verkkoa käyttäviin laitteisiin. Reunapalveluiden sitominen tiukasti matkapuhelinverkkoon vaikuttaa erikoiselta. Ratkaisulla on hyvät ja huonot puolensa. Reunapalveluiden tarjoamisen muihin verkkoihin, esimerkiksi WiFi yhteydellä ei onnistu suoraan ehdotetulla

SCC:n arkkitehtuurin haasteena saattaa olla niiden ylläpidettävyys, jossa matkapuhelinverkko-operaattori on vastuussa reunasolmun ylläpidosta, koska se on tiukasti sidoksissa tukiasemaan[Lobillo et al., 2014]. Tukiasemaan tiukasti sidotulla ratkaisulla on kuitenkin mahdollista hyödyntää laajemmin tukiaseman ominaisuuksia, kuten radioyhteyksiä toisiin tukiasemiin. SCeNBce tukiasemien hankkiminen vaatii operaattoreilta investointeja sekä lisää ylläpitotyön määrää. (Näiden vuoksi ehkä kantsis miettii, että miksi se on niin tiukasti kiinni siinä televerkossa. Muutenkin useampi operaattori samalla alueella tarkoittaa että jokaisella on myös omat reunapalvelut, joka taas meinaa kokonaiskuvassa että sama palvelu pitää duplikoida monelle operaattorille.)

### 6.3.1 Kommunikointi reunapalveluun

SCC pohjaisessa järjestelmässä tavallinen verkkoon menevä tietoliikenne ja reunapalvelulle menevä tietoliikenne on ehdotettu eroteltavaksi toisistaan [Puente et al., 2015]. Eristyksen seurauksena LTE-A arkkitehtuuriin ei tarvitse koskea muutoin kuin lisäyksien osalta. Liikenne SCeNBce:llä tarjottuihin reunapalveluihin on tehtävä tarkoitusta varten olevan rajapinnan kautta. Asiakaslaitteelta tuleva liikenne reititetään SCeNBce:llä. Pakettien reitittämi-

seksi SCeNBce joutuu päättelemään, onko paketin määränpää reunapalvelu vai tavallinen tietoliikenne.

Koska LTE-tukiasemat eivät toimi IP tasolla, ei reunapalvelulta tuleva pakettiliikenteen sisältämä asiakaslaitteen IP-osoite riitä identifioimaan kohdelaitetta. Yleisesti LTE verkossa, internetistä asiakaslaitteelle suuntautuvan liikenteen osalta pakettien ohjaamiseksi, käytetään GTP-tunnelia. GTP-tunnelilla tietoliikenne saadaan ohjattua oikealle tukiasemalle. Tukiaseman ja asiakaslaitteen välisen kommunkaatioväylän selvittämiseksi joudutaan tekemään TEID (Tunnel endpoint IDentifier) ja DRB-ID (Data Radio Bearer ID) muunnos.

Vastaava ongelma on myös reunapalvelulta asiakaslaitteelle suuntautuvassa tietoliikenteessä. SCC:n yhteydessä ehdotetussa ratkaisussa reunapalvelulta tulevan liikenteen yhdistäminen asiakaslaitteen IP-osoitteeseen voidaan tehdä käyttäen TEID perustuvaa ohjaustaulua. TEID on asiakaskohtaisen GTP-tunnelin indetifioiva tunniste. TEID:stä ja asiakaslaitekohtaisesta IP-osoitteesta voidaan muodostaa SCeNBce:lle reititystaulu. Reititystaulun avulla voidaan ohjata IP-osoitte muuntaa DRB liikkenneväyläksi ja ohjata tietoliikenne asiakaslaitteeseen.

## 6.4 SMORE ja MobiScud

SMORE (Software defined network Mobile Offloading aRchitecturE) on toinen televerkkoihin suunniteltu MEC ratkaisu. [Cho et al., 2014] SMORE:n keskeisin sisältö on ottaa SDN (Software Defined Networking) käyttöön reunapalveluiden saavuttamiseksi. SDN käyttöönotolla tavoitellaan sitä, että televerkon toimintaan ei tarvitsisi tehdä muutoksia. Minkään olemassa olevan komponentin toiminta ei siis muutu. SMORE:n toiminta on jaettu kahteen osaa, joita varten SDN on käytössä. Ensimmäisenä on mobiiliverkon kontrollitasolla (control-plane) tapahtuva viestien monitorointi. Toisena toimintona on monitoroitujen tietojen pohjalta tehtävät SDN hallintatoimenpiteet. Näiden avulla voidaan ohjata haluttu osa tietoliikenteestä haluttuihin reunalaskentayksikköihin. Kontrollitasolla viestejä monitorointia suorittaa SMORE monitori (SMORE monitor) ja tietoliikenteen ohjauspäätöksistä vastaava komponentti on SMORE kontrolleri (SMORE controller).

Tietojen välitys SMORE kontrollerin ja SMORE monitorin välillä on toteutettu yhteisen tietokannan kautta. SMORE monitor tallentaa poimitut tiedot tietokantaan ja antaa tiettyjen tapahtumien yhteydessä SMORE kontrollerille herätteen tehdä asianmukaiset muutokset SMORE:n SDN reitityksiin. Herätteitä laukaisevat tapahtumat ovat asikkaan liittyminen verkkoon ja asikkaan liikkuminen verkossa.

SMORE monitori tarkkailee asiakaslaitteen ja LTE/EPC:n välistä liikennettä. Asiakaslaitteen liittyessä mobiiliverkkoon, asiakaslaite ja LTE/EPC:n sisäiset komponenti muodostavat tunneleita ja asiakaslaitteeseen liitetään erilaisia tunneleita koskevia osoite ja metatietoja. SMORE monitorin tehtä-

vänä on poimia asiakaslaitteen liittyessä ja yhteydenmuodostuksen aikana asiakaslaitteisiin liittyviä tietoja. Asiakaslaitteen lähettäessä liittymispyynnön (attach request) SMORE monitori poimii pyynnöstä asiakaslaitteen IMSI (international Mobile Subscriber Identity) ja TAI:n (Tracking Area Identifier). Tämän jälkeen SMORE monitori poimii MME:n asiakaslaitteelle lähettämästä liittymispyynnön hyväksymis -viestistä (attach accept) monitori poimii asiakaslaitteelle annetun IP-osoitteen, SGW:n IP-osoitteen, SGW:n TEID:n ja asiakaslaitteen GUTI:n (Globally Unique Temporary Id). Tämän jälkeen eNodeB neuvottelee asiakaslaitteen kanssa radioyhteydestä ja tämän lopputulos välitetään MME:lle. SMORE monitor poimii eNodeB:n ja MME:n välisestä kommunikaatiosta eNodeB:n IP-osoitteen ja eNodeB:n TEID:n. Kun edellä mainitut tiedot on tallennettu SMORE:n tietokantaan, SMORE monitor lähettää SMORE kontrollerille herätteen päivittää SDN reitityksiä.

Toinen tapahtuma josta SMORE kontrolleri on kiinnostunut on handover, eli mikäli asiakaslaite siirtyy mobiiliverkossa eNodeB:den välillä. SMORE kontrollerin on tässä tapauksessa kiinnostunut mobiiliverkossa tapahtuvista muutoksista. Handover alkaa kun tällä hetkellä käytössä oleva eNodeB päättää, että on aika siirtää asiakaslaitteen yhteys toiselle eNodeB:lle (kohde). Alkuperäinen eNodeB välittää pyynnön kohteena olevalle eNodeB:lle joka oletettavasti hyväksyy sen. Tämän jälkeen, kohteena oleva eNodeB pyytää MME:ltä reitityksen muutosta. Tässä välissä oleva SMORE monitor poimii reitityksen muutosta koskevan tiedon ja välittää ne SMORE kontrollerille. Tämän tiedon pohjalta SMORE kontroller voi tehdä SDN muutokset siten että vanhat reititykset voidaan poistaa ja korvata uusilla.

### 6.4.1 MobiScud

MobiScud [Wang et al., 2015] on SMORE:n ideoita ammentava versio reunalaskentapalveluiden tuottamisesta mobiiliverkoissa. MobiScud:in perustoiminnallisuus on hyvin samankaltainen kuin SMORE:ssa. MobiScud painottaa enemmän käyttäjän tarvetta liikkua verkossa. Lisäksi MobiScud käyttää hyödykseen oletusta mobiiliverkkojen hajautettua rakennetta.

Kuten SMORE, MobiScud käyttää hyödykseen SDN:n tarjoamia ominaisuuksia ja siten olettaa sen käyttöönottoa palveluntarjoajan infrastruktuurissa. Lisäksi MobiScudin toiminnallisuudet on ajateltu toteutettavan Network Function Virtualisantionin (NFV) avulla. MobiScud Controller (MC) koostuu kahdesta loogisesta kokonaisuudesta: monitorista ja kontrollerista. Näiden toimintaperiaate on käytännössä identtinen SMORE:n vastaavan nimisiin toimijoihin. MC:n sijaintia ei kuitenkaan ole keskitetty kuten SMORE:ssa vaan sen oletetaan sijaitsevan hajautettna RAN ja EPC välissä. Käyttäjän ja reuna-infrastruktuurin välinen yhteys toteutetaan hyvin samalla tavalla kuin SMORE:ssa.

MobiScudin reunapalvelut on ajateltu toteutettavan hyvin cloudletmäisesti. Mahdollisimman lähellä reunaa sijaitsevassa "pilvessä"on palvelinresursse-

ja, joita käyttäjät hyödyntävät yksityisien virtuaalikoneinstanssien muodossa (Private Virtual Machinem, PVM).

MobiScudin tavoitteena on tarjota asiakkaalle mahdollisimman nopea yhteys asiakkaan ja PVM:n välille. MobiScud hyödyntää televerkon omia kontrollitason viestejä asiakkaan liikkumisen seuraamiseen. Kun handoverista tulee viesti MC:n monitoroivalle entiteetille, alkaa MC organisoimaan reunalaskentaan liittyviä muutoksia.

Asiakkaan siirtyessä tukiasemalta toiselle PVM:n siirto aloitetaan livemigraationa kohteena olevalle "pilvelle". Livemigraation ollessa käynnissä kohteena olevan pilven MC huolehtii että asiakaslaitteella on edelleen yhteys alkuperäiseen sijaintiinsa. Tämä hoidentaan SDN reititysmuutoksilla. Kun PVM:n livemigraatio on saatu suoritettua alkuperäisen sijainnin MC ilmoittaa tästä kohdesijainnin MC:lle, joka puolestaa päivittää SDN reitityksset ohjaamaan siirretylle PVM:lle.

MobiScudin testeissä livemigraation ja SDN reitityksien avulla RTT (round trip time) saatiin pidettyä pienenä. Kuitenkin yhteyksissä aiheutui noin kahden sekunnin mittaisia katkoksia sillä hetkellä kun livemigraation viimeiset muutokset lähetetään. Yhteyskatkoksen pituuteen vaikuttavat monet asiat ja artikkelissa huomautetaankin että nykyinen toteutus oli optimoimaton ja vaatii jatkotutkimuksia.

## 6.5 CONCERT

Nykyinen LTE televerkko koostuu joukosta laitteita, joilla on hyvin spesifit tehtävät. CONCERT [Liu et al., 2014] pyrkii seuraavan sukupolven televerkkoihin keskittyvässä ratkaisussaan vähentämään tehtäväspesifien laitteiden tarvetta. CONCERT arkkitehtuuri koskee siis pääasiassa seuraavan sukupolven televerkkoja, vaikkakin sen ratkaisut mahdollistavat tuen myös vanhemmille teknologioille. CONCERT esittää uudenlaista toiminnallisuuksien toteuttamismallia, joka korvaisi valmistajakohtaisia telelaitteita (kuten tukiasemat) virtualisoiduilla ja ohjelmistopohjaisilla ratkaisuilla. NFV ja SDN käyttöönotto toimii siis tämänkin ehdotuksen selkärankana. Verkon toimintojen virtualisointi mahdollistaa laitteiston siirtämisen kauemmaksi reunasta, joka osaltaan vähentää hajautettavan laitteiston määrää. CONCERTin ideana on, että hajautetuksi jäisi telelaitteistosta ainoastaan välttämättömin osa, eli radiorajapinnan mahdollistava RIE (Radio Interfacing Equipment) ja hierarkisesti jaetut reunalaskentayksiköt. Televerkon muita toiminnallisuuksia voitaisiin keskittää ja tarjota virtuaalisina ohjelmistoteutuksina.

Conductor on CONCERT arkkitehtuurissa hallinnollisessa keskiössä ja se vastaa päällysverkon hallinnasta. CONCERT arkkitehtuurissa control ja data planet on erotettu toisistaan ja Conductorin tehtävänä esittää data planella esiintyvät fyysiset resurssit virtuaalisina resursseina. Reunalaskentayksiköiden resurssien jakaminen sekä niiden välisten SDN kytkimien kautta muodostettujen yhteyksien hoitamisesta vastaa conductor. Conductorin toi-

minta on toistaiseksi kuvattu vain korkealla tasolla, joten sen toteutustekniset ratkaisut ovat avoimia.

Vaikka laitteiston virtualisointi ja palveluiden keskittäminen kustannuksien säästämiseksi kuulostaakin houkuttelevalta, on kesittämisessä myös omat ongelmansa. Etenkin liiallinen keskittäminen siirtää verkon toiminnallisuuksia kauemmaksi reunasta, joka osaltaa johtaa korkeampiin latensseihin ja heikentää palvelun laatua. Vaikka conductor onkin esitetty yksittäisenä loogisena entiteettinä, sen toiminnallisuuksia on mahdollista porrastaa ja hajauttaa paremman skaalautuvuuden ja pienempien latenssien tavoittamiseksi.

CONCERTin ratkaisu reunapalveluiden tuottamiseksi on hierarkinen kolmeen tasoon jaettu arkkitehtuuri. *Paikalliset* reunalaskentaresurssit sijaitsevat kaikkein lähimpänä verkon reunaa, esimerkiksi RIE:n kanssa samassa sijainnissa sijaitseva palvelin. Paikallisen reunalaskentayksikön laskentaresurssit ovat rajalliset ja sen tehtävänä onkin suorittaa ainoastaan kaikkein tiukimman aikavaatimuksen laskentaa. *Alueelliset* reunalaskentaresurssit kattavat jonkin pienen alueen reunalaskenta tarpeita ja korkeimman tason *keskus* reunalaskentaresurssit kattavat jonkin suuremman alueen resurssi intensiivistä reunalaskentaa. CONCERT:ssa reunalaskentaresurssien jakamisesta vastaa conductorin sisäinen komponentti LCM (Location-aware computing management).

Reunalaskennan osalta CONCERTissa resurssit on jaettu hierarkisesti kolmeen tasoon, siten että lähimpänä käyttäjää on *paikalliset* palvelimet, jotka on tarkoitettu kaikkein tiukimman aikavaatimuksen sovelluksille. Seuraavat kaksi tasoa, *alueellinen* ja *keskus (central)*, kasvattavat palvelinresurssien määrää, mutta ovat kauempana reunasta. Reunalaskentayksiköt voivat olla yhteydessä toisiinsa, jolloin ne mahdollistavat nopeat M2M (Machine-to-Mahcine) yhteydet. Tämä mahdollistaisi muunmuassa autojen välisen kommunikaation reunaverkon välityksellä.

Toistaiseksi CONCERT on korkean tason suunnitelma ja se käyttää hyväkseen vielä olemassa olemattomia teknologisia ratkaisuja kuten tukiasemien virtualisointia. Arkkitehtuuri mahdollistaa erilaisien ratkaisujen toteuttamisen, eikä aseta tiukkoja rajoitteita resurssien tai hallinnon sijoittelun suhteen.

## 6.6 ETSI MEC

ETSI (European Telecommunications Standards Institute) on eurooppalainen telealan standardointijärjestö. ETSI on aloittanut reunalaskennan arkkitehtuurin, sekä sen touteuttamiseksi vaadittavien toimintojen standardoimisen.

ETSIn MEC spesifikaatio määrittelee reunapalvelun tuottamiseksi vaadittavat ominaisuudet, jotka reunainfrastruktuurin tulee toteuttaa. Spesifikaatio listaa myös mahdollisia, mutta ei vaadittuja toimintoja. Listaamalla vaaditut toiminnot stanrardi pyrkii yhtenäistämään reunalaskennan konsepteja.

### 6.6.1 Vaatimukset

Vaatimukset on jaettu kategorioihin sen mukaan mihin toiminnallisuuteen vaatimus liittyy. Vaatimuksien kategoriat ovat yleiset vaatimukset (generic requirements), palvelu vaatimukset (service requirements), hallinta vaatimukset (operation and management requirements) ja viimeisenä kategoriana on kokoelma vaatimuksista, joiden teemoina ovat turvallisuus, sääntely ja veloitus (Security, regulation, charging requirements)[ETSI, 2016c].

Yleiset vaatimukset ovat luonteelta korkean tason kuvauksia reuna-infrastruktuurin toiminnallisuuksista. Yleiset vaatimukset kategorisoitu seuraaviin luokkiin: viitemallista, reunapalvelun sovelluksien elinkaaresta (application lifecycle), reunapalvelun sovellusympäristöstä (application enviroment) sekä liikkuvuuden tuesta (support of mobility). Viitemallin tulee vaatimuksien mukaan hyödyntää mukaan NFV ratkaisuja hallinnollisten toimien toteuttamiseen, mikäli mahdollista. ETSI:n vaatimuksen mukaan reunapalvelun tulee voida sijaita käytännössä missä tahansa kohdassa radiomaston runkoverkon reunan välillä. Sijainnin vapaus myös tarkoittaa että reuna-alusta(?) ei voi olla riippuvainen alla olevasta infrastruktuurista. Reunapalvelun sovelluksien elinkaareen liittyvät vaatimukset koskevat pääasiassa reunapalvelun toimijoiden oikeuksia päättää reunan sovelluksien käynnistämisestä ja sulkemista. Reunapalveluiden sovellusympäristöä koskevissa vaatimuksissa esitetään, että sovelluksien autenttisuus ja eheys pitää pystyä varmentamaan. Sovellusympäristön täytyy myös mahdollistaa reunasovelluksen käyttöönotto toisella reuna-isännällä, ilman erikoisempaa mukautusta (without specific adaptation) [ETSI, 2016c]. Mobiiliverkkojen ollessa kyseessä, asiakaslaitteiden liikkuminen tukiasemalta toiselle, on keskeinen käyttötapaus. Tämä heijastuu myös vaatimukseen liikkuvuuden tukemisesta reunapalveluissa. Vaatimuksena on että asiakaslaitteen ja reunapalvelun välisen yhteyden tulee säilyä, vaikka asiakaslaite siirtyisi solusta toiseen tai asiakaslaite siirtyisi sellaiseen soluun joka on toisen reuna-isännän vastuualuetta.

Palveluvaatimukset on joukko vaatimuksia, jotka keskittyvät takaamaan reuna-infrastruktuurin perimmäiset palveluperiaatteet. Lista palveluvaatimuksista on pitkä, joten tähän tutkielmaan on poimittu ainoastaan osa vaatimuksista. Täydellinen lista löytyy [ETSI, 2016c] julkaisusta. Palveluvaatimukset kuvaavat toiminnallisuuksia, joiden avulla reunapalveluita voidaan tuottaa. Tällaisesta esimerkkinä tietoliikenteen reitittämiseen liittyvät vaatimukset, joiden keskeinen tehtävä on kuvata mahdolliset tietoliikennereitit reunapalveluun ja ulos reunapalvelusta. Yksi ehkä keskisimmistä palveluvaatimuksista on reuna-alusta mahdollisuus suodattaa ja muokata verkkoliikennettä. Lisäksi kuvataan että reunapalveluiden toimintaa ei haluta rajoittaa pelkästään asiakas-palvelu tyyppisen toimintamalliin. Reunapalveluiden tuottamiselle onkin annettu mikropalvelu -tyyppinen (microservice) kuvaus, jossa palveluntuottajat voivat toimia myös toisten reunapalveluiden kuluttajana (consumer).

Taulukko 1: Reunalaskenta-arkkitehtuurien ominaisuudet

### 6.6.2 Viitekehys ja referenssiarkkitehtuuri

ETSIn esittämän viitekehys esittää reunalaskentaan liittyvät korkean tason entiteetit. Nämä entiteetit on jaettu kolmeen tasoon: järjestelmä, isäntä ja verkko(system, host ja network). Verkkokerros sisältää verkkoyhteyksistä vastaavat entiteetit. MEC:n tapauksessa verkkokerros koostuu ainakin kolmesta osasta: sisäverkko, ulkoverkko ja televerkko. Isäntäkerros koostuu reunapalveluiden virtualisointiin ja reunapalvelun hallinointiin keskittyvistä entiteeteistä. Järjestelmäkerros koostuu korkeamman tason hallinnosta vastaavista elementeistä.

Referenssiarkkitehtuurissa on esitetty funktionaaliset entiteetit. Funktionaalisten entiteettien toiminta on kuvattu Referenssiarkkitehtuurissa reuna-isännällä (edge-host) tarkoitetaan entiteettiä, joka tarjoaa virtualisointi-infrastruktuurin, sekä reunalaskennan toteuttamiseen vaadittavat resurssit (laskenta, tallennus ja verkko). Reuna-alustalla (Mobile edge platform) tarkoiteen sitä entiteettiä joka mahdollistaa reunapalveluiden käyttämisen. Reuna-alusta siis mahdollistaa reunapalveluiden saavuttamisen, eli käytännössä tarjoaa rajapinnan asiakaslaitteen suuntaan. Tähän kuuluu siis palvelurekisterin ylläpitäminen, reitityssääntöjen ylläpitäminen, sekä liikenteen välittäminen reunapalveluille. Reuna-alusta voi myös itse tarjota palveluita. Esimerkkinä tällaisesta voisi olla tunnistautumispalvelu. Reuna-applikaatioilla tarkoitetaan reuna-alustalla suoritettavia virtuaalikoneita, jotka suorittavat reunapalveluiden tuottamiseksi tarkoitettuja ohjelmistoja.

## 6.7 Vertailu

# Lähteet

[Cho et al., 2014] Cho, J., Nguyen, B., Banerjee, A., Ricci, R., Van der Merwe, J., and Webb, K. (2014). Smore: software-defined networking mobile offloading architecture. In *Proceedings of the 4th workshop on All things cellular: operations, applications, & challenges*, pages 21–26. ACM.

[ETSI, 2012] ETSI (2012). Network functions virtualisation. `https://portal.etsi.org/NFV/NFV_White_Paper.pdf`. [Verkkoaineisto, Luettu 2018-03-19].

[ETSI, 2016a] ETSI (2016a). Lte; evolved universal terrestrial radio access (e-utra) and evolved universal terrestrial radio access network (e-utran); overall description; stage 2. `http://www.etsi.org/deliver/etsi_ts/136300_136399/136300/14.05.00_60/ts_136300v140500p.pdf`. [Verkkoaineisto, Luettu 2018-03-13].

[ETSI, 2016b] ETSI (2016b). Mobile edge computing (mec); framework and reference architecture. http://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/01.01.01_60/gs_MEC003v010101p.pdf. [Verkkoaineisto, Luettu 2018-02-23].

[ETSI, 2016c] ETSI (2016c). Mobile edge computing (mec); technical requirements. http://www.etsi.org/deliver/etsi_gs/MEC/001_099/002/01.01.01_60/gs_MEC002v010101p.pdf. [Verkkoaineisto, Luettu 2018-02-23].

[ETSI, 2017a] ETSI (2017a). Lte; evolved universal terrestrial radio access network (e-utran); architecture description (3gpp ts 36.401 version 14.0.0 release 14). http://www.etsi.org/deliver/etsi_ts/136400_136499/136401/14.00.00_60/ts_136401v140000p.pdf. [Verkkoaineisto, Luettu 2018-03-14].

[ETSI, 2017b] ETSI (2017b). Network functions virtualisation. http://portal.etsi.org/NFV/NFV_White_Paper_5G.pdf. [Verkkoaineisto, Luettu 2018-03-19].

[Farris et al., 2017a] Farris, I., Taleb, T., Flinck, H., and Iera, A. (2017a). Providing ultra-short latency to user-centric 5g applications at the mobile network edge. *Transactions on Emerging Telecommunications Technologies*.

[Farris et al., 2017b] Farris, I., Taleb, T., Iera, A., and Flinck, H. (2017b). Lightweight service replication for ultra-short latency applications in mobile edge networks. In *Communications (ICC), 2017 IEEE International Conference on*, pages 1–6. IEEE.

[Firmin, 2017] Firmin, F. (2017). The evolved packet core. http://www.3gpp.org/technologies/keywords-acronyms/100-the-evolved-packet-core. [Verkkoaineisto, Luettu 2018-03-14].

[Gambetti et al., 2015] Gambetti, F., Fogliuzzi, F., de Marinis, E., Pascual, A., Muñoz, O., Lagén, S., Agustín, A., Puente, M. A., Barbarossa, S., and Sardellitti, S. (2015). Distributed computing, storage and radio resource allocation over cooperative femtocells. https://cordis.europa.eu/docs/projects/cnect/4/318784/080/deliverables/001-D61Ares20152101644.pdf. [Verkkoaineisto, Luettu 2018-04-03].

[Ha et al., 2015] Ha, K., Abe, Y., Chen, Z., Hu, W., Amos, B., Pillai, P., and Satyanarayanan, M. (2015). Adaptive vm handoff across cloudlets. *Technical report, Technical Report CMU-C S-15–113, CMU School of Computer Science*.

[Ha et al., 2017] Ha, K., Abe, Y., Eiszler, T., Chen, Z., Hu, W., Amos, B., Upadhyaya, R., Pillai, P., and Satyanarayanan, M. (2017). You can teach elephants to dance: agile vm handoff for edge computing. In *Proceedings of the Second ACM/IEEE Symposium on Edge Computing*, page 12. ACM.

[Jammal et al., 2014] Jammal, M., Singh, T., Shami, A., Asal, R., and Li, Y. (2014). Software defined networking: State of the art and research challenges. *Computer Networks*, 72:74–98.

[Kreutz et al., 2015] Kreutz, D., Ramos, F. M., Verissimo, P. E., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2015). Software-defined networking: A comprehensive survey. *Proceedings of the IEEE*, 103(1):14–76.

[Liu et al., 2014] Liu, J., Zhao, T., Zhou, S., Cheng, Y., and Niu, Z. (2014). Concert: a cloud-based architecture for next-generation cellular systems. *IEEE Wireless Communications*, 21(6):14–22.

[Lobillo et al., 2014] Lobillo, F., Becvar, Z., Puente, M. A., Mach, P., Presti, F. L., Gambetti, F., Goldhamer, M., Vidal, J., Widiawan, A. K., and Calvanesse, E. (2014). An architecture for mobile computation offloading on cloud-enabled lte small cells. In *2014 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–6.

[Mao et al., 2017] Mao, Y., You, C., Zhang, J., Huang, K., and Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys Tutorials*, PP(99):1–1.

[Nguyen and Cheriet, 2016] Nguyen, K. K. and Cheriet, M. (2016). Virtual edge-based smart community network management. *IEEE Internet Computing*, 20(6):32–41.

[Pahl and Lee, 2015] Pahl, C. and Lee, B. (2015). Containers and clusters for edge cloud architectures–a technology review. In *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pages 379–386. IEEE.

[Puente et al., 2015] Puente, M. A., Becvar, Z., Rohlik, M., Lobillo, F., and Strinati, E. C. (2015). A seamless integration of computationally-enhanced base stations into mobile networks towards 5g. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*, pages 1–5.

[Satyanarayanan, 2001] Satyanarayanan, M. (2001). Pervasive computing: vision and challenges. *IEEE Personal Communications*, 8(4):10–17.

[Satyanarayanan et al., 2009] Satyanarayanan, M., Bahl, P., Caceres, R., and Davies, N. (2009). The case for vm-based cloudlets in mobile computing. *IEEE Pervasive Computing*, 8(4):14–23.

[Soni and Kalra, 2013] Soni, G. and Kalra, M. (2013). Comparative study of live virtual machine migration techniques in cloud. *International Journal of Computer Applications*, 84(14).

[Taleb and Ksentini, 2013] Taleb, T. and Ksentini, A. (2013). Follow me cloud: interworking federated clouds and distributed mobile networks. *IEEE Network*, 27(5):12–19.

[Taleb et al., 2017] Taleb, T., Samdanis, K., Mada, B., Flinck, H., Dutta, S., and Sabella, D. (2017). On multi-access edge computing: A survey of the emerging 5g network edge cloud architecture and orchestration. *IEEE Communications Surveys & Tutorials*, 19(3):1657–1681.

[Tuovinen, 2017] Tuovinen, A.-P. (2017). Ohjelmistoarkkitehtuurin suunnitteluperiaatteita.

[Wang et al., 2015] Wang, K., Shen, M., Cho, J., Banerjee, A., Van der Merwe, J., and Webb, K. (2015). Mobiscud: A fast moving personal cloud in the mobile network. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*, pages 19–24. ACM.

[Yousaf and Taleb, 2016] Yousaf, F. Z. and Taleb, T. (2016). Fine-grained resource-aware virtual network function management for 5g carrier cloud. *IEEE Network*, 30(2):110–115.