Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan koulutusohjelma

Mobiilien tietoliikennejärjestelmien emulointi

Kandidaatintyö

3. joulukuuta 2012

Kimmo Ahokas

Aalto-yliopisto
Perustieteiden korkeakoulu
Tietotekniikan koulutusohjelma

KANDIDAATINTYÖN
TIIVISTELMÄ

Tekijä:	Kimmo Ahokas
Työn nimi:	Mobiilien tietoliikennejärjestelmien emulointi
Päiväys:	3. joulukuuta 2012
Sivumäärä:	20
Pääaine:	Tietoliikenneohjelmistot
Koodi:	T3005
Vastuuopettaja:	Ma professori Tomi Janhunen
Työn ohjaaja(t):	Tutkija Matti Siekkinen (Tietotekniikan laitos)

Tässä kandidaatintyössä tutkitaan mobiilien tietoliikennejärjestelmien emulointiratkaisuja. Tarkoituksena oli löytää erilaisia ratkaisuja, jotka soveltuvat suosituimipien käytössäolevien mobiilialustojen tutkimukseen. Erityisesti keskityttiin täysin ohjelmallisiin ratkaisuihin ja ulkopuolelle rajattiin fyysiseen laitteistoon perustuvat ratkaisut. Työ suoritettiin kirjallisuustutkimuksena ja minkäänlaista kokeellista osuutta ei liitetty mukaan.

Työssä löydettiin kaksikin erillistä käynnissä olevaa tutkimusta Android-järjestelmän tietoliikenne-emuloinnista, mutta ei vielä julkaistuja tuloksia. Muille suosituille mobiilialustoille ei löydetty lainkaan viitteitä samansuuntaisesta tutkimuksesta. Samalla todettiin, että tutkittavalla mobiilialustalla on suuri vaikutus käytettävään emulointirajapintaan, joka taas vaikuttaa käytettävän tietoliikennesimulaattorin valintaan. Lisäksi emulaattorit keskittyvät erilaisten asioiden tarkkaan mallintamiseen, jolla on myös vaikutusta emulaattorin valintaan. Mobiilijärjestelmien yhteydessä järkevää voikin olla valita emulaattori joka on erityisesti keskittynyt langattomien verkkojen mallintamiseen.

Vähäisestä tutkimustiedon määrästä voidaan päätellä, että mobiilien tietoliikennejärjestelmien emulointi on haastava sovellusalue. Aiheesta kuitenkin selkeästi tarvitaan jatkotutkimusta, sillä mobiilitiedonsiirron lisääntyessä tällaisten järjestelmien tutkimukseen tarvitaan luotettavia työkaluja.

Avainsanat:	tietoliikenne, emulointi, mobiili, simulaattori, simulointi, iOS, Android, Windos Phone
Kieli:	Suomi

Sisältö

1 Johdanto

Tämä kandidaatintyö käsittelee tietoliikennejärjestelmien emuloinnissa käytettyjä ratkaisuja, niiden eroja ja ongelmia keskittyen erityisesti mobiilien tietoliikennejärjestelmien emulointiin.

Tietoliikennejärjestelmien emuloinnissa kantava ajatus on erilaisten tietoliikenneprotokollien ja -ohjelmistojen toiminnan tarkastelun helpottaminen. Teknologian avulla on mahdollista muodostaa ohjelmallinen verkko kahden tietokoneen, virtuaalikoneen, tietokoneohjelman tai muun verkkolaitteen välille. Tämän verkon ominaisuuksia on mahdollista muuttaa helposti lisäämällä esimerkiksi viivettä, rajoittamalla nopeutta, aiheuttamalla virheitä tiedonsiirtoon tai pudottamalla osa tietoliikennepaketeista. Näin ohjelman toimintaa voidaan tarkastella erilaisissa verkko-olosuhteissa ilman oikean verkon rakentamista.

Tässä työssä on tarkoitus tutustua erilaisiin tietoliikennejärjestelmien emulointiin kehitettyihin ohjelmistopohjaisiin ratkaisuihin ja niiden ominaisuuksiin. Pääasiallinen huomio keskittyy mobiileihin ratkaisuihin, eli ohjelmistoihin, joilla voidaan emuloida esimerkiksi langaton verkko älypuhelimille suunnatun ohjelmiston tarkasteluun.

Älypuhelimille suunnattujen ohjelmien määrä on kasvanut räjähdysmäisesti muutaman viime vuoden aikana. Samalla kun puhelimien tietoliikenneyhteydet paranevat, myös ohjelmien siirtämä datamäärä kasvaa. Saataville on tullut esimerkiksi BitTorrent-ohjelmistoja älypuhelimille. Tällaista valtavia tietomääriä siirtävää ohjelmaa kehittäessä on tärkeää saada tietoa siitä, kuinka ohjelma käyttäytyy erilaisissa verkko-olosuhteissa. Toistaiseksi tälläiseen analysointiin ei ole ollut saatavilla erityisen monipuolisia työkaluja.

Tässä kandidaatintyössä etsitään erityisesti vastausta kysymykseen siitä, millaisia mobiililaitteiden kanssa käytettäväksi soveltuvia tietoliikennejärjestelmien emulointiratkaisuja on olemassa. Toimivatko järjestelmät suosituimpien älypuhelinalustojen, kuten Applen iOS, Googlen Android tai Microsoftin Windows Phone -käyttöjärjestelmien kanssa? Millaisia rajoitteita näillä järjestelmillä on? Millaisia esteitä mobiilien tietoliikenneemulaattoreiden kehitykselle on?

Työn tavoitteena on löytää mahdollisimman realistisia tuloksia antavia tietoverkkojen emulointiratkaisuja. Näistä ratkaisuista halutaan löytää parhaat ominaisuudet ja vakavimmat puutteet. Samalla tutkitaan alan tulevia kehityssuuntia sekä tärkeimpiä parannuskohteita.

Kirjallisuustutkimuksen ulkopuolelle jätetään erilaiset laitteistopohjaiset verkkoemulointiratkaisut, sillä erillisen laitteiston hankkiminen ei varsinaisesti tue tekniikan käyttöönoton helppoutta tai haluttuja kustannussäästöjä. Näistä ratkaisuista ei myöskään ole saatavilla akateemista tutkimustietoa yhtä hyvin kuin ohjelmistopohjaisista ratkaisuista.

Työssä huomattiin, että mobiilien tietoliikennejärjestelmien emulointiin ei ole juurikaan kehitetty ratkaisuja. Useat emulaattorit tukevat langattomia lähiverkkoja ja jopa 4G-verkkoja, kuten LTE, mutta kirjallisuudessa ei mainittu käyttöä mobiililaitteiden tai -alustojen kanssa. Tilanteen parantamiseksi esitetään muutamia ratkaisuja, joilla mobiililaitteet olisi mahdollista yhdistää nykyisiin tietoliikennejärjestelmien emulointiratkaisuihin.

Aluksi perehdytään siihen mitä on tietoliikennesimulointi, sillä emulointi perustuu hyvin vahvasti tähän teknologiaan. Tämän jälkeen määritellään mitä tietoliikennejärjestelmien emulointi oikeastaan tarkoittaa ja mihin teknologiaa käytetään. Neljäs luku keskittyy erilaisiin emulointiratkaisuihin ja niiden ominaisuuksiin. Viidennessä luvussa kerrotaan ratkaisujen puutteista sekä tulevista kehityssuunnista. Lopuksi esitetään yhteenveto tutkimuksesta ja tärkeimmistä tuloksista.

2 Tietoliikennejärjestelmien simulointi

Koska tietoliikennejärjestelmien emulointi perustuu voimakkaasti tietoliikennesimulointiin, on aluksi tärkeää ymmärtää, mitä simulointi on. Tässä luvussa esitellään simuloinnin käsite yleisesti sekä perehdytään hiukan tietoliikennejärjestelmien simulointiin siltä osin kuin se on tarpeellista tämän kandidaatintyön varsinaisen aiheen kannalta.

Banks et al. [2] määrittelee simuloinnin olevan oikean prosessin tai järjestelmän toiminnan jäljittelyä ajan kuluessa. Samalla luodaan järjestelmän tilasta keinotekoinen historia, jonka perusteella tehdään johtopäätöksiä oikean järjestelmän toiminnasta. Simulaatioita voidaan käyttää esimerkiksi järjestelmää suunniteltaessa erilaisten ratkaisujen toiminnan arvioimiseen erilaisissa olosuhteissa.

Simulaatioiden ymmärtämiseksi on tärkeää ymmärtää muutaman peruskäsitteen sisältö. Ensinnäkin *entiteetti* on yksittäinen kiinnostuksen kohde. Tietoliikennejärjestelmien tapauksessa entiteettejä voivat olla esimerkiksi ohjelmat, tietokoneet tai tietoliikennepaketit. *Järjestelmä* on entiteettien ja niiden välisten suhteiden muodostama kokonaisuus. Esimerkiksi verkko voi sisältää entiteetit tietokone, kytkin, reititin ja linkki. Kolmas oleellinen käsite on *malli*, joka tarkoittaa matemaattista esitystä jostakin järjestelmästä. Sillä kuvataan tutkittavan asian kannalta oleelliset järjestelmän osat helpommin käsiteltävässä muodossa. Simulaatioiden yhteydessä malli on yleisesti esitettävä tietokoneen ja simulaattorin ymmärtämässä muodossa, toisin sanoen ohjelmakoodina. [17]

Wehrle et al. [17] mukaan tietoliikennejärjestelmiä simuloitaessa yleisin simulaatiotyyppi on diskreettien tapahtumien simulointi (discrete-event simulation). Tässä simulaatiotyypissä järjestelmän tila voi muuttua vain yksittäisinä ajanhetkinä, ei jatkuvasti. Simulaatio koostuu *tapahtumista*, jotka tapahtuvat ennalta määrättyinä ajanhetkinä. Tapahtuma voi

muuttaa järjestelmän tilaa ja se voi ajastaa uusia tapahtumia jollekin tulevalle ajanhetkelle. Esimerkiksi paketin lähetys verkkoon muuttaa järjestelmässä verkon tilaksi varattu ja ajastaa uuden tapahtuman, kun paketti vastaanotetaan verkosta. Simulaatio käsittelee tapahtuman kerrallaan ja etenee sitten seuraavaan tapahtumaan. Tätä jatketaan niin kauan kuin järjestelmässä on tapahtumia jäljellä tai ennalta määritetty lopetushetki saavutetaan. Jokaisen tapahtuman hetkellä systeemin tilaa kuvaavista muuttujista tallennetaan tarpeellinen määrä tietoa analysoitavan ilmiön ymmärtämiseksi. Mahdollista on esimerkiksi tallentaa muuttujien tilastollisia arvoja tai jopa kaikki muuttujien saamat arvot.

Wehrle et al. [17] mainitsee muista tunnetuista simulaatiotyypeistä esimerkkeinä Monte Carlo -simulaation, jatkuvan simuloinnin (continuous simulation), taulukkolaskentasimulaation (spreadsheet simulation) sekä jälkivetoisen simulaation (trace-driven simulation). Tässä työssä näistä simulaatiotyypeistä ei kuitenkaan kerrota tämän enempää, sillä tietoliikennejärjestelmiä simuloitaessa niitä ei juurikaan käytetä [17].

Tietoliikennejärjestelmiä simuloitaessa halutuista verkon laitteista ja ohjelmistoista muodostetaan malli, joka kuvaa tutkittavan ilmiön kannalta oleelliset asiat tarkasti, mutta voi jättää huomiotta ilmiöön liittymättömiä yksityiskohtia. Esimerkiksi reititintä tarkasteltaessa malliin kuvattavia ominaisuuksia voivat olla puskurin koko, pakettien jonotusviive, liitäntöjen nopeus ja viive. Yksittäistä linkkiä kuvattaessa taas kiinnostavia voivat esimerkiksi olla kaistanleveys ja viive. Abstraktiotaso on kuitenkin tärkeää valita tutkittavan sovelluksen mukaan. Matalan tason tietoliikenneprotokollaa simuloitaessa linkin fyysisillä ominaisuuksilla voi olla merkitystä, kun taas jotakin sovellusta simuloitaessa fyysisten ominaisuuksien simulointi tekisi simulaatiosta vain hitaamman tarjoamatta tutkimuksen kannalta oleellista tietoa.

Eri simulaattoreista löytyy valmiita malleja useille eri verkon laitteille ja protokollille. Tämän takia onkin usein käytännöllisintä valita käytettävä simulaattori sen mukaan, mikä simulaattori tarjoaa valmiina mahdollisimman suuren osan suoritettavaan tutkimukseen tarvittavista malleista. Tietenkin samalla on myös tarkistettava, että valmiit mallit ovat riittävän tarkkoja ja niistä saadaan tarvittu tutkimusdata ulos käyttökelpoisessa formaatissa.

Simulaattoria valitessa verkkotopologioiden mallintamisen helppoudella voi oilla huomattava merkitys. Joissakin tapauksissa topologioilla ei ole väliä, vaan riittää esimerkiksi yksi linkki kahden solmun (node) välillä, kunhan vain linkille saadaan simuloitua kaistanleveys, viive ja mahdollisesti häiriöitä. Toisissa tapauksissa taas tarvitaan suuri määrä solmuja ja niiden välille erilaisista linkeistä muodostuta monimutkainen verkko. Lisäksi mobiileja järjestelmiä simuloitaessa langattomien verkkojen liikkuvuusmalleilla voi olla hyvinkin paljon käyttöä.

Käytännössä tietoliikennejärjestelmiä simuloitaessa halutuista verkon laitteista, linkeistä ja ohjelmistoista muodostetaan kokonaisuus, joka vastaa jotakin realistista tai teoreettista tilannetta. Simuloidut ohjelmistot tuottavat liikennettä jonka simuloidut laitteet välittävät haluttuihin simuloituihin kohteisiin. Esimerkiksi tietoliikenneprotokollaa suunniteltaessa simulaatioon lisätään ohjelmisto, joka tuottaa protokollan mukaista liikennettä. Lisäksi voidaan lisätä muunlaista liikennettä tuottavia ohjelmistoja, jotta nähdään kuinka tutkittava protokolla toimii, kun verkossa on muutakin liikennettä. Koska simulaatiossa liikenteen määrää ja reittejä sekä verkon rakennetta ja linkkien kapasiteetteja on helppo muuttaa, on simulaatio hyvin käyttökelpoinen tapa tietoliikennejärjestelmien tutkimuksessa.

Simulaatioiden muokattavuus ja monipuolisuus on tavallaan myös tämän lähestymistavan suurin ongelma. Valmiit tietoliikenneprotokollien ja -ohjelmistojen toteutukset eivät useimmiten sovellu suoraan simulaattorin kanssa käytettäväksi, vaan ne täytyy toteuttaa uudelleen käytetyn simulaattorin ymmärtämässä muodossa. Uudelleentoteutus voi olla hyvinkin aikaavievä, vaikea ja virhealtis operaatio. Virheellisillä malleilla saadut testitulokset voivat pahimmillaan antaa täysin väärää tietoa ja ohjata suunnittelua väärään suuntaan.

3 Tietoliikennejärjestelmien emulointi

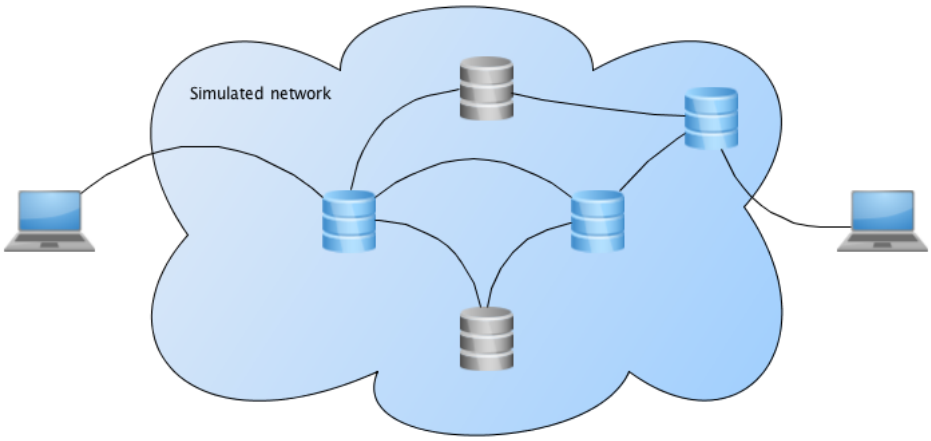
Tässä luvussa kerrotaan mitä tietoliikennejärjestelmien emulointi on, mihin sitä voidaan käyttää ja millaisia mahdollisia tulevaisuuden käyttökohteita tekniikalla on.

Fall [5] määrittelee tietoliikennejärjestelmien emuloinnin (network emulation) tarkoittavan järjestelyä jossa aidot protokollatoteutukset voivat vaikuttaa ajettavaan simulaatioon. Teknologian avulla aidon ohjelmiston tuottama liikenne voidaan syöttää simulaattoriin ja vastaavasti simulaattorin tuottama liikenne ulkopuoliselle ohjelmalle. Lisäksi emulaatiossa simuloidut tietoliikennejärjestelmien osat voivat toimia yhteistyössä fyysisten järjestelmän osien, kuten verkkolaitteiden kanssa. Liu [9] mukaan ero simulaation ja emulaation välillä onkin juuri siinä, että simulaatio on täysin virtuaalista, mutta emulaatio keskittyy yhteisvaikutukseen oikeiden ohjelmistotototeutusten ja simuloitujen toteutusten välillä.

Liu [9, kapale 5] määrittelee käsitteen reaaliaikainen tietoliikennejärjestelmien simulointi (real-time network simulation) tarkoittavan ratkaisua jossa tietoliikenneverkkoa simuloidaan reaaliajassa ja simulaatio vuorovaikuttaa oikeiden sovellusten ja oikean verkkoliikenteen kanssa. Näinollen sekä tietoliikennejärjestelmien emulointi että reaaliaikainen simulointi voidaan nähdä hyvin samanlaisina tekniikoina.

Emulointi yhdistää fyysisten testijärjestelmien ja simulaation parhaita puolia. Useim-

mat tietoliikennejärjestelmien emulointiratkaisut perustuvat tietoliikennesimulaattoreihin, joissa on jonkinlainen rajapinta oikean verkkoliikenteen kanssa vuorovaikutukselle. Näin muokkaamattomien ohjelmisto- tai protokollatoteutusten tuottama liikenne voidaan altistaa simulaattorin tuottamaan helposti muokattavaan ja toistettavissa olevaan verkkoympäristöön. Tutkittavaa protokollaliikennettä voidaan häiritä tiputtamalla paketteja, muuttamalla pakettien saapumisjärjestystä, aiheuttamalla virheitä tiedonsiirtoon ja useilla muilla tavoilla. Verkkoliikenteeseen voidaan lisätä viivettä tai käytettävissä olevaa kaistaa rajoittaa, jolloin voidaan havainnoida hitaan tietoliikennelinkin vaikutusta ohjelmiston toimintaan. Simuloituun verkkoon voidaan myös lisätä muuta verkkoliikennettä joka häiritsee tutkittavaa ohjelmistoa. Näin voidaan varmistua siitä, että sovellus toimii myös oikeissa käyttötilanteissa joissa verkon toimintaan ei voi ikinä luottaa täydellisesti. Kuvassa 1 on esitetty esimerkki siitä, kuinka kaksi tietokonetta voidaan yhdistää toisiinsa simuloidulla verkolla.



Kuva 1: Esimerkki tilanteesta, jossa kaksi oikeaa tietokonetta kommunikoivat kytkimistä ja reitittimistä koostuvan simuloidun verkon välityksellä.

Emuloinnista puhuttaessa on hyvä mainita myös suuret testipedit, jotka yhdistävät emulointia ja fyysisiä verkkoja. Eräs hyvin tunnettu esimerkki tällaisesta järjestelmästä on Emulab [19], joka kuvaa emuloidun verkkotopologian fyysisiin tietokoneisiin ja verkkolaitteisiin. Emulab koostuu sadoista toisiinsa liitetyistä tietokoneista, jotka on yhdistetty toisiinsa nopealla lähiverkolla, jonka rakennetta voidaan muuttaa ohjelmallisesti käyttäen virtuaalisia lähiverkkoja (Virtual LAN, VLAN). Tämän lisäksi linkkien kapasiteettia, virheherkkyyttä ja muita ominaisuuksia voidaan kontrolloida ohjelmallisesti. Koska jokainen emulaation solmu kuvataan yhteen fyysiseen laitteeseen, voidaan tutkittavien ohjelmien resurssien käyttöä havainnoida tarkasti. Samalla kaikki käytettävät käyttöjärjestelmät, protokollat ja ohjelmistotototeutukset toimivat täysin muokkaamattomina, jolloin testitulokset ovat realistisia. Samankaltaisia toteutuksia ovat myös esimerkiksi langattomien

verkkojen emulointiin suunnattu Orbit [13] sekä planeetan kattavien suuren mittakaavan verkkojen tutkimukseen suunniteltu Planetlab [12].

Tietoliikennejärjestelmien emuloinnin suurin etu on kuitenkin siinä, että teknologia mahdollistaa hyvinkin monimutkaisten verkkotopologioiden luonnin ilman fyysisten laitteiden hankintaa. Verkot voivat sisältää esimerkiksi langallisia lähiverkkoja, valokuitulinkkejä, langattomia lähiverkkoja tai LTE-verkkoja. Varsinkin uusiin langattomiin verkkoihin tarvittavat laitteet ovat vaikeasti saatavia ja hinnakkaita, joten näiden verkkojen korvaaminen simulaatiolla on erittäin houkutteleva vaihtoehto. Lisäksi langattomiin verkkoihin vaikuttavat ulkopuoliset häiriöt, joten testitulokset eivät välttämättä ole toistettavissa. Emuloidulla verkolla tällaisia ongelmia ei ole. Lisäksi emulaatiossa verkkoihin on todella helppo syöttää ylimääräistä liikennettä häiritsemään tutkittavaa sovellusta. Fyysisiä verkkoja käytettäessä taas muun verkkoliikenteen käyttäminen ei välttämättä onnistu helposti, tai liikenne ei ole toistettavissa samanlaisena testikertojen välillä.

4 Emulointitekniikat

Tässä luvussa perehdytään erilaisiin tietoliikennejärjestelmien emulointiratkaisuihin. Erityisesti huomiota kiinnitetään siihen, kuinka nämä ratkaisut soveltuvat mobiilien järjestelmien emulointiin. Erilaiset emulointiratkaisut pyritään ryhmittelemään loogisesti, jolloin erilaisten emulaattoreiden tunnusomaisia piirteitä on helpompi vertailla keskenään. Työssä esitellään pintapuolisesti aiemmassa kirjallisuudessa esitettyjä emulaattoreiden ryhmittelyvaihtoehtoja jonka jälkeen perehdytään tarkemmin emulointirajapintojen luokitteluun.

4.1 Emulaattoreiden luokittelu

Aiemmin kirjallisuudessa esitettyjä ryhmittelyperusteita ovat esimerkiksi emulaattorin tärkein sovellusalue, tietoisuus topologiasta tai hajautettavuus. Tässä kandidaatintyössä esitelty lista erilaisista ryhmittelyperusteista ei ole missään nimessä kattava, vaan se on esitetty lähinnä esimerkkinä ja pohjatietona.

Liu [9] jakaa emulaattorit kolmeen luokkaan, linkkikeskeisiin, solmukeskeisiin ja verkkokeskeisiin emulaattoreihin. Nimensä mukaisesti ensimmäisen ryhmän emulaattorit keskittyvät yhden tai muutaman linkin emulointiin. Esimerkiksi DummyNet [14] kuuluu tähän ryhmään. Solmukeskeiset emulaattorit taas erikoistuvat yhden solmun sisäisen toiminnan emulointiin. Tämän ryhmän emulaattoreissa pyritään jäljittelemään mahdollisimman tarkasti oikeaa verkkopinoa. Esimerkkejä solmukeskeisistä emulaattoreista ovat ENTRAPID [7], sekä NCTUns [16]. Verkkokeskeisten emulaattorien päämääränä on mallintaa koko-

naisen suuren verkon vaikutusta verkkoliikenteeseen. Tämän ryhmän emulaattoreilla voidaan mallintaa esimerkiksi internetin tyyppinen verkko, jossa on hyvinkin monimutkainen topologia. Esimerkiksi ModelNet [15] voidaan luokitella verkkokeskeiseksi emulaattoriksi.

Linkkikeskeisiä emulaattoreita käytetään yleisesti silloin, kun emuloitavan verkon rakenteella ei ole väliä, vaan tutkimuksen kannalta on riittävää, että emulaattori muokkaa tutkittavan sovelluksen verkkoliikennettä halutulla tavalla. Toisin sanoen siis merkityksellistä on verkon kapasiteetti, viive, kadonneiden pakettien määrä ja siirtovirheet. Solmukeskeisiä emulaattoreita taas on yleisesti käytetty tutkimukseen silloin kun merkityksellistä on protokollan tarkka toiminta oikean käyttöjärjestelmän sisällä, eikä niinkään verkon toiminta. Tämän ryhmän emulaattorit usein toteuttavat suoraan esimerkiksi käyttöjärjestelmän verkkorajapintaa vastaavan rajapinnan. Tällaista emulaattoria varten kehitetty ohjelma voidaan yleisesti siirtää oikeaan käyttöön hyvin helposti, sillä rajapinnat ovat lähes identtisiä. Verkkokeskeisiä emulaattoreita taas käytetään silloin, kun merkityksellistä on koko verkon toiminta kokonaisuutena. Näiden emulaattoreiden suunnittelussa on siis erityisesti keskitytty siihen, kuinka useat samaa linkkiä kuormittavat tietovirrat vaikuttavat toisiinsa.

Gu ja Fujimoto [6] jakaa emulaattorit kahteen kategoriaan sen mukaan, mallintavatko ne emuloitavan verkon topologiaa. Topologiatiedottomat (topology-unaware) emulaattorit abstraktoivat emuloidun verkon yhdeksi linkiksi jolla on tietyt parametrit. Esimerkkejä tämän ryhmän emulaattoreista ovat NIST Net [4], DummyNet [14] ja ModelNet [15]. Topologiatietoiset (topology-aware) emulaattorit, kuten jo aiemmin mainittu Emulab [19] ja EMPOWER [20] taas kuvaavat emuloitavan verkon useisiin tietokoneisiin. Jälkimmäisestä ryhmästä käytetään myös nimitystä hajautettu verkkoemulointi [6].

Jako topologiatietoisten ja -tiedottomien emulaattoreiden välillä voidaan tavallaan nähdä myös jakona täysin ohjelmallisten ja osittain laitteistopohjaisten emulaattoreiden välillä. Topologiatiedottomat emulaattorit toteuttavat linkkien parametrit täysin ohjelmallisesti, samoin kuin ainakin osan linkeistä. Topologiatietoisissa taas jokaista emuloitua linkkiä ja solmua vastaa myös fyysinen linkki ja solmu.

4.2 Emulointirajapintojen luokittelu

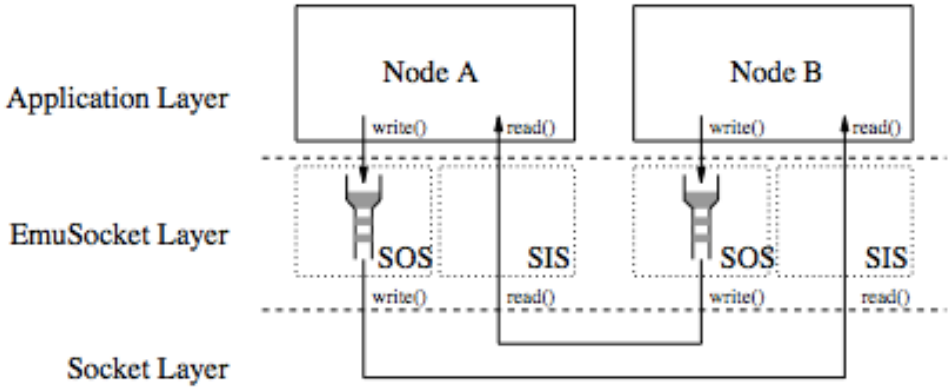
Mobiilijärjestelmien emulointia ajatellen mikään yllämainituista luokitteluperusteista ei erityisen hyvin tue soveltuvan emulaattorin valintaa. Erilaiset mobiilijärjestelmät asettavat hyvinkin paljon toisistaan poikkeavia vaatimuksia emulointirajapinnalle, eli tavalle jolla emulaattori kaappaa oikeiden sovellusten liikenteen. Bradford et al. [3] esittelee erilaisia emulointirajapintoja, joihin perehdytään seuraavaksi hiukan tarkemmin. Jokaisesta rajapinnasta esitellään niiden ominaisuudet sekä yritetään arvioida miten rajapinta mahdollisesti soveltuu käytettäväksi mobiilijärjestelmien kanssa. Jokaisesta ryhmästä

annetaan myös esimerkkejä ryhmään kuuluvista emulaattoreista.

Ensimmäinen ryhmä on käyttöjärjestelmän ohjelmistotasolla (application level) tiedonsiirron kaappaavat emulaattorit. Tässä tekniikassa emulaattori tarjoaa suoritettaville ohjelmille samantyyppisen kommunikoinnin kuin käyttöjärjestelmän socket-rajapinta. Emulaattori siis käytännössä asettuu sovelluksen ja käyttöjärjestelmän väliin kaappaamaan liikenteen. Esimerkkinä tämän kategorian emulaattoreista voidaan mainita EmuSocket [1] joka on Java-ohjelmille suunnattu tietoliikenne-emulaattori. EmuSocket korvaa Javan virtuaalikoneessa OutputStream- ja InputStream-luokat omilla toteutuksillaan, jotka eivät välitä liikennettä käyttöjärjestelmän verkkopinolle, vaan emulaattorille. Tämä tekniikka antaa sovelluskehittäjille kaksi eri vaihtoehtoista tapaa sovelluksen liittämiseksi emulaattoriin. Ensinnäkin kehittäjä voi itse kirjoittaa sovelluksen käyttämään Emusocket-työkalupaketin tarjoamaa socket-rajapintaa haluamissaan kohdissa, jolloin kaiken ohjelmiston liikenteen ei tarvitse välittyä emulaattorille. Vaihtoehtoisesti kaikki järjestelmän socket-rajapinnalle tarkoitetut kutsut voidaan ohjata EmuSocket-rajapinnalle, jolloin ohjelman lähdekoodia ei tarvitse muokata. Kuvassa 2 on esitetty kuinka EmuSocket kaappaa ohjelmiston liikenteen. Kun ohjelma tahtoo lähettää liikennettä verkkoon, se kutsuu emulaattorin rajapinnan tarjoamaa lähetysfunktiota, joka muokkaa liikennettä asetetun verkkoprofiilin mukaan. Tämän jälkeen emulaattori välittää liikenteen käyttöjärjestelmän verkkopinolle.

Ohjelmistotasolla liikenteen kaappaavat emulaattorit eivät todennäköisesti sovellu mobiilijärjestelmien kanssa käytettäväksi kovinkaan hyvin, sillä ratkaisu vaatii joko ohjelmiston tai käyttöjärjestelmän muokkaamista. Useimmista kiinnostavista mobiiliohjelmistoista ei ole lähdekoodia saatavilla, joten ensimmäinen ratkaisu ei ole useimmiten käytettävissä. Yleisimmistä mobiilialustoistakin vain Android on avointa koodia, joten useissa tapauksissa käyttöjärjestelmääkään ei ole mahdollista muokata. Toisaalta taas vaikka käyttöjärjestelmän muokkaaminen olisi mahdollista, ei tämä välttämättä ole järkevää, sillä tällöin liikenteen kaappaus joudutaan tekemään joko tehoiltaan rajatussa mobiililaitteessa tai mobiililaitetta emuloivassa ympäristössä, joka ei välttämättä ole tehokkuudeltaan riittävä tai ei vastaa tarpeeksi tarkasti oikeaa suoritusympäristöä.

Toiseen ryhmään luokitellaan unixin tun/tap-verkkorajapintaa (network tunnel, network tap) käyttävät emulaattorit. Tun ja tap ovat unix-kernelin tarjoamia virtuaalisia verkkorajapintoja, jotka antavat käyttäjätason ohjelmistoille mahdollisuuden käsitellä käyttöjärjestelmän verkkopinon läpi kulkevia paketteja. Kun jokin tietoverkkoa käyttävä ohjelmisto lähettää dataa, sen sijaan että käyttöjärjestelmän verkkopino lähettäisi datan käyttäen fyysistä verkkolaitetta välitetäänkin data tun/tap-rajapintaa hyödyntävälle ohjelmistolle, tässä tapauksessa emulaattorille. Vastaavasti emulaattori voi kirjoittaa dataa tun/tap-rajapinnan avulla, jolloin verkkopino välittää datan verkko-ohjelmistolle aivan kuin se olisi tullut fyysisestä verkosta. Tekniikka on siinä mielessä erittäin käytännölli-

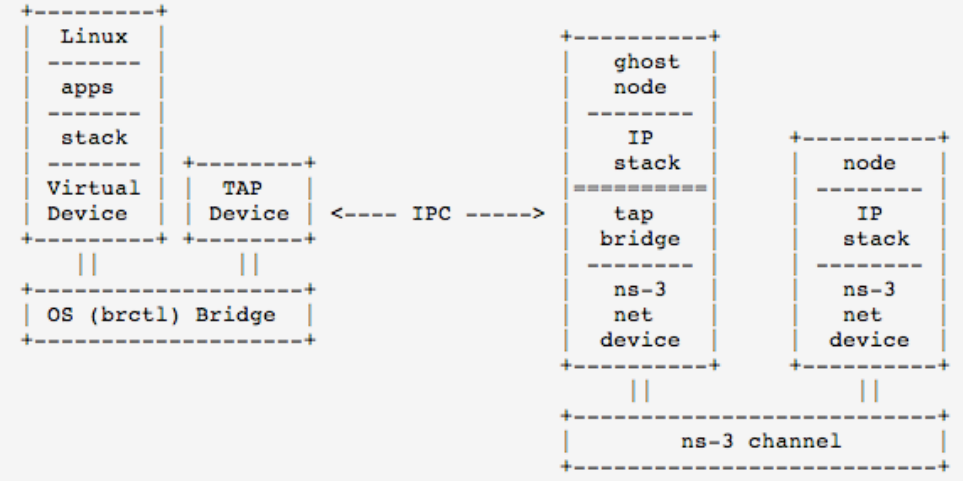


Kuva 2: EmuSocket-emulaattori Korvaa Java-virtuaalikoneen verkkorajapinnan omalla yhteensopivalla rajapinnallaan ja siten kaappaa ohjelmiston tuottaman verkkoliikenteen. [1]

nen, että verkkoa käyttävää sovellusta ei tarvitse muokata millään tavalla emulaattoriin yhdistettäessä. Sovelluksen ei myöskään tarvitse olla jollakin tietyllä ohjelmointikielellä toteutettu, vaan tekniikka toimii minkä tahansa sovelluksen kanssa. Esimerkiksi ns-3 simulaattori sisältää TapBridge-komponentin, jonka avulla simulaattori voidaan yhdistää tap-rajapintaan ja käyttää sitä emulaattorina [11]. Kuvassa 3 on esitetty kuinka virtuaalikone on mahdollista yhdistää ns-3 simulaattorin TapBridge-komponentin avulla. Ohjelmiston tuottama verkkoliikenne välitetään normaalisti käyttöjärjestelmän verkkopinolle, jossa se etenee virtuaaliselle tap-verkkolaitteelle. Tältä verkkolaitteelta liikenne etenee erityisen sillan (bridge) kautta toiselle tap-verkkolaitteelle joka vuorostaan välittää liikenteen simulaattorille.

Mobiilijärjestelmiä emuloitaessa tun/tap-pohjainen emulointirajapinta voi olla hyödynnettävissä alustasta riippuen. Esimerkiksi Android-emulaattori tukee tun/tap-rajapintaa, jolloin emulaattorin sisällä ajattavien tietoliikenneohjelmistojen liikenteen kaappaaminen onnistuu isäntäjärjestelmästä käsin. Tästä ratkaisusta kerrotaan enemmän luvussa 5.2. Oikeisiin mobiililaitteisiin teknologia taas ei ole sovellettavissa helposti, sillä niiden käyttöjärjestelmät ovat hyvin rajoittuneita eivätkä siksi yleisesti tarjoa tämäntyyppisiä rajapintoja. Lisäksi liikenne pitäisi tällaiselta rajapinnalta välittää vielä ulkopuoliselle emulaattorille, sillä mobiililaitteiden rajatut tehot eivät riitä kunnolliseen emulointiin.

Kolmas ryhmä on ulkopuoliseen yhdyskäytävään luottavat emulaattorit. Tämä teknologia on yleisimmin käytössä jo aikaisemmin mainituissa hajautetuissa emulaattoreissa. Sovelluksen verkkoliikenne välitetään fyysistä verkkoa pitkin yhdyskäytävään, joka vastaanottaa liikenteen ja välittää sen emulaattorille. Yhdyskäytävänä voi toimia esimerkiksi verkon oletusyhdyskäytävä (default gateway) tai VPN-palvelin (Virtual Private Network). Ratkaisun huonona puolena on välissä olevan fyysisen verkon liikenteeseen

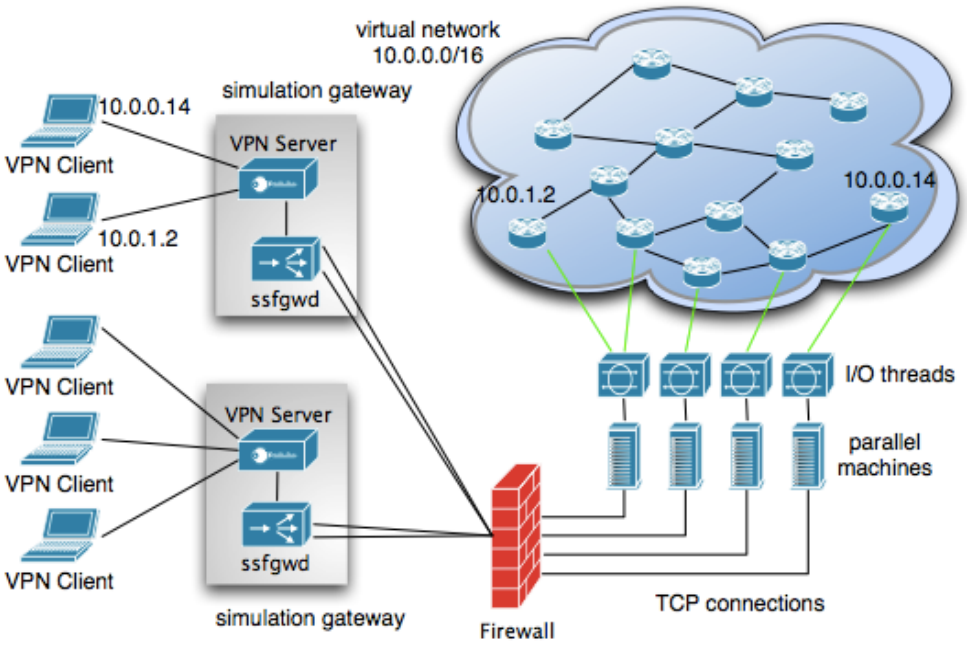


Kuva 3: Ns-3 simulaattorin TapBridge-komponentti kaappaa ohjelmiston liikenteen käyttöjärjestelmän verkkopinosta hyödyntän tap-rajapintaa. [11]

mahdollisesti aiheuttamat häiriöt, mutta toisaalta se taas mahdollistaa liikenteen syöttämisen emulaattorille jopa ilman käyttöjärjestelmän apua. Kuvassa 4 on havainnollistettu yhdyskäytäväpohjaista emulointirajapintaa hyödyntävän PRIME-emulaattorin toimintaa. Siinä tutkittavan ohjelman liikenne ohjataan VPN-tunneliin, jonka toisessa päässä muokattu VPN-palvelin vastaanottaa liikenteen ja välittää sen sitten emulaattorille [10].

Yhdyskäytäväpohjainen emulointiratkaisu todennäköisesti soveltuu käytettäväksi minkä tahansa mobiilialustan kanssa. Esimerkiksi langattomia lähiverkkoja tukevat mobiililaitteet voidaan liittää langattomaan lähiverkkoon jonka kaikki liikenne syötetään verkon yhdyskäytävän kautta simulaattoriin. Kuten jo mainittu, fyysinen verkko simulaattorin ja sovelluksen välillä voi vaikuttaa tuloksiin. Varsinkin langaton verkko on hyvin altis ulkopuolisille häiriöille. Siksipä ulkoiseen yhdyskäytävään perustuvaa emulointirajapintaa käytettäessä täytyy punnita hyvin tarkoin häiriöiden vaikutuksia mittaustuloksiin.

Mobiilien tietoliikennejärjestelmien emuloimiseksi tulisi pystyä valitsemaan emulaattori, johon valittu mobiililaite tai -sovellus onnistutaan liittämään. Liitettävyyteen voi vaikuttaa valittu sovellusalusta ja sen tarjoamat palvelut ja kehitystyökalut. Yhdyskäytäväpohjainen emulointiratkaisu voisi soveltua käytettäväksi minkä tahansa mobiilialustan kanssa, sillä liikenne on tällöin mahdollista kaapata esimerkiksi mobiililaitteen käyttämän langattoman tukiaseman avulla. Tällöin tosin testijärjestelyssä on mukana fyysinen verkko, joka voi vaikuttaa testituloksiin. Useille mobiilialustoille on tarjolla virtuaalikoneita joiden avulla mobiiliohjelmien testaus onnistuu tietokoneella. Virtuaalikoneen liittäminen emulaattoriin voi onnistua esimerkiksi tun/tap-rajapinnan avulla, jos vain virtuaalikoneohjelmisto tätä tukee.



Kuva 4: PRIME-emulaattorin VPN-tunneliin perustuva yhdyskäytäväpohjainen emulointirajapinta [10].

5 Emulointiratkaisujen puutteet ja tulevat kehityssuunnat

Tässä luvussa perehdytään nykyisten tietoliikennejärjestelmien emulointiratkaisujen puutteisiin sekä todennäköisiin tuleviin kehityssuuntiin.

5.1 Emulointiratkaisujen puutteet

Toistaiseksi mikään kirjallisuudessa esitelty tietoliikennejärjestelmien emulointiratkaisu ei ole erityisesti suunnattu mobiilien tietoliikennejärjestelmien emulointiin. Samalla kun älypuhelimet, tabletit ja muut internetiin jatkuvasti yhteydessä olevat mobiililaitteet yleistyvät on tärkeää löytää mahdollisimman helppoja ja totuudenmukaisia tuloksia antavia menetelmiä näiden laitteiden tietoliikenneratkaisujen tarkasteluun. Tähän tarkoitukseen emulaattorit ovat luonnollinen valinta, sillä niiden avulla on mahdollista tarkastella ohjelmistojen toimintaa millaisessa tahansa verkossa, kuten langattomassa lähiverkossa, LTE-verkossa tai 3g-verkossa.

Koska emulointi pääsääntöisesti asettaa järjestelmälle vaatimuksen reaaliaikaisuudesta on emulaattoreiden tehokkuus tärkeää. Tutkimuksessa usein kiinnostuksen kohteena olevat verkot ovat internetin tyyppisiä, hyvin suuria solmumääriä sisältäviä verkkoja. Emulointiratkaisuissa suurin osa verkon topologiasta toteutetaan tietoliikennesimulaattorin avul-

la, jolloin simulaattorin tulee pystyä välittämään paketteja niin tehokkaasti, että luodun verkon vaikutukset näkyvät tietoliikenteessä oikeanlaisina. Tämän takia useat suurten verkkojen emulointiin kehitetyt ratkaisut ovatkin hajautettu useammalle koneelle. Osassa ratkaisuista fyysinen verkko on osa emuloitua verkkotopologiaa, joka asettaa rajoitteita simuloidulle verkkotopologialle.

Mobiilijärjestelmiä emuloitaessa käytetyt mobiiliohjelmistot ja -laitteet rajaavat emulointirajapinnan valintaa ja siten käytettävän verkkosimulaattorin valintaa. Käytettävä emulointirajapinta taas voi vaikuttaa mittausten tuloksiin haitallisesti. Esimerkiksi yhdyskäytäväpohjaista emulointirajapintaa käytettäessä mobiililaitteen ja simulaattorin välillä voi pahimmillaan olla häiriöaltis langaton verkko. Ohjelmistopohjaista emulointirajapintaa käytettäessä taas pahimmillaan koko mobiilikäyttöjärjestelmää joudutaan suorittamaan natiivista poikkeavassa laitteistossa, joka voi myös vaikuttaa tuloksiin.

Yllättäen kirjallisuudessa ei juurikaan ole esitetty viitteitä tutkimuksiin, jossa osana emulointijärjestelmää olisi käytetty mobiilialustoja. Yksikään löytämäni tutkimus ei käsitellyt esimerkiksi Android, iOS tai Windows Phone -järjestelmien tietoliikenne-emulointia. Tilannetta saattaa selittää se, että mobiililaitteet ovat suhteellisen rajattuja ja niiden verkkoliikenteen kaappaaminen on tästä johtuen mahdollisesti haastavaa. Sandian kansallinen laboratorio on julkaissut uutisen kehittäneensä järjestelmän Android-laitteiden vuorovaikutusten tutkimukseen [8], mutta tästä järjestelmästä ei vielä ole julkaistu tarkempaa tietoa tai tutkimusraporttia.

Android-emulaattorista löytyy sisäänrakennettuna toiminto, jolla järjestelmän tietoliikenneyhteyden nopeutta voidaan rajoittaa ja viivettä kasvattaa. Tämä jo sinällään auttaa ohjelmistojen testauksessa, mutta ei käytännössä ole tarpeeksi tehokas työkalu suuremman luokan tutkimukseen. iOS- ja Windows Phone -kehitystyökalut eivät dokumentaation mukaan sisällä edes tälläistä toimintoa. Koska suurin osa tietoliikennejärjestelmien emulointiratkaisuista on kehitetty Unix-alustoille, mutta iOS-kehitystyökalut ovat saatavilla vain Mac-alustalle ja Windows Phone -kehitystyökalut Windowsille voi tämäkin osaltaan vaikuttaa emulointiratkaisujen puutteeseen.

5.2 Emulointiratkaisujen tulevia kehityssuuntia

Eräs mielenkiintoisimmista uutuuksista tietoliikennejärjestelmien emuloinnissa on synkronoitu emulointi. Weingärtner et al. [18] esittelee SliceTime-järjestelmän, jossa oikeiden sovellusten on mahdollista vuorovaikuuttaa simulaation kanssa ilman että simulaation tarvitsee olla reaaliaikainen. Aikaisemmin tarkkojen tutkimustulosten saavuttamiseksi emulaattorin on pitänyt toimia tiukasti reaaliaikaisesti kaikissa tilanteissa. Tämän vuoksi monimutkaisten ja suurien järjestelmien emulointi on vaatinut huomattavan määrän laskentaresursseja. SliceTime poistaa vaatimuksen reaaliaikaisuudelle, jolloin tarkkoja

tutkimustuloksia on mahdollista saada myös aiempaa pienemmillä resursseilla.

SliceTime-järjestelmässä virtuaalikoneessa ajettavat sovellukset ovat yhteydessä ns-3-simulaattoriin. Vaatimus reaaliaikaisuuteen poistetaan synkronoimalla virtuaalikoneen ja simulaattorin suoritus niin sanotulla *puomisynkronointialgoritmilla* (barrier synchronization algorithm). Tässä lähestymistavassa virtuaalikoneelle ja verkkosimulaattorille annetaan lupa edetä jokin lyhyt ennalta määrätty aikajakso. Kun komponentti on suorittanut aikajakson se jää odottamaan muiden komponenttien suoritusta. Kun kaikki komponentit ovat suorittaneet aikajakson annetaan niille taas lupa suorittaa seuraava jakso. Algoritmi varmistaa sen, että simulaattorin ja virtuaalikoneiden kellojen ero on korkeintaan yhden aikajakson suuruinen.

Olen itse mukana Aalto-yliopiston perustieteiden korkeakoulun tietoliikenneohjelmistojen tutkimusryhmässä kehittämässä SliceTime-järjestelmästä versiota Android-käyttöjärjestelmälle. Tämä versio perustuu ohjelmistokehittäjille suunnattuun Android-emulaattoriin, jonka avulla ARM-pohjaista Android-järjestelmää voidaan ajaa virtuaalikoneen tavoin normaalissa tietokoneessa. Android-emulaattori tukee tap-rajapintaa, jonka avulla emulaattorissa ajettavien Android-ohjelmistojen liikenne voidaan ohjata ns-3-verkkosimulaattoriin. Koska Android-emulaattori nimensä mukaisesti emuloi ARM-käskykantaa tukevaa mobiilijärjestelmää yleisesti tietokoneissa käytetyn x86-käskykannan päällä, ei emuloidun käyttöjärjestelmän tehokkuus aivan vastaa natiivisti suoritettua käyttöjärjestelmää. Tämä näkyy emulaattorissa muun muassa ajoittaisena hitautena ja ajan epälineaarisena etenemisenä. Koska SliceTime poistaa vaatimuksen reaaliaikaisuudesta, teknologia saattaa tulevaisuudessa myös parantaa ohjelmistoille näkyvän laitteiston tehokkuutta. Kehitysversiolla on saatu jo lupaavia testituloksia, mutta koska kehitys on vielä kesken, ei järjestelmästä ole julkaistu tutkimustuloksia.

Tulevaisuudessa toivottavasti myös nähdään sekä Applen iOS- , että Microsoftin Windows Phone -älypuhelinalustoille suunnattuja tietoliikenne-emulointiratkaisuja. Android avoimuutensa ansiosta on houkutellut tutkijoita, mutta hyvin suosittu iOS ja kovaa vauhtia nouseva Windows Phone ovat myös erittäin mielenkiintoisia tutkimuksen kohteita. Jollei näille alustoille nähdä täysin ohjelmistopohjaista emulointiratkaisua, niin ainakin oikeisiin mobiililaitteisiin ja yhdyskäytäväpohjaiseen emulointirajapintaan perustuvan emulointiratkaisun kehittäminen näyttää mahdolliselle.

6 Yhteenveto

Tässä kandidaatintyössä löydettiin useita emulaattoreita, jotka lähestyivät samaa ongelmaa hyvinkin erilaisista lähtökohdista. Jokaisella erilaisella ratkaisulla on hyvät ja huonot puolensa ja ne soveltuvat hyvinkin erilaisiin emulointitarpeisiin. Tästä johtuen käyttäjä

joutuukin valitsemaan emulaattorin punnitsemalla eri ominaisuuksia ja valitsemalla kulloiseenkin käyttötarkoitukseen parhaiten sopivan ratkaisun.

Työn tavoitteena oli löytää erityisesti mobiilien tietoliikennejärjestelmien emulointiin hyvin soveltuva ohjelmistopohjainen ratkaisu. Valitettavasti kuitenkaan yksikään löydetyistä ratkaisuista ei tue suoraan mobiiliohjelmien tai -laitteiden käyttöä liikenteen tuottamisessa. Tällä hetkellä kaksikin eri tutkimusryhmää suorittaa tutkimusta Android-järjestelmän käytöstä tietoliikenne-emulaattoreiden kanssa, mutta kummastakaan tutkimuksesta ei vielä ole saatavilla julkisia tutkimustuloksia. Yllättäen yksikään löydetty tutkimus ei käsitellyt iOS tai Windows Phone -järjestelmien käyttöä tietoliikenne-emulaattoreiden yhteydessä.

Mobiilien tietoliikennejärjestelmien emuloinnin tulevan kehityksen suhteen huomattiin, että tutkittava mobiilijärjestelmä rajaa huomattavasti käytettävää emulointirajapintaa, joka taas vuorostaan rajaa käytettävän tietoliikennesimulaattorin valintaa. Oikeiden mobiililaitteiden kanssa käytettäväksi soveltuu ainoastaan yhdyskäytäväpohjaista emulointirajapintaa hyödyntävä emulaattori. Jos taas mobiilijärjestelmää voidaan ajaa virtuaalikoneessa esimerkiksi linux-alustalla, käytettävissä on myös ohjelmallisia emulointirajapintoja tarjoavia simulaattoreita, jolloin simulaattorin valinnassa on enemmän vaihtoehtoja. Android-alustan kehitystyökaluista löytyi ohjelmallinen rajapinta liikenteen kaappaamiseen, mutta iOS ja Windows Phone -alustojen yhteydessä tällaista rajapintaa ei löydetty. Käytännössä tämä siis tarkoittaa, että näille alustoille on todennäköisesti vaikeampi kehittää luotettavasti toimivia tietoliikenne-emulaatioratkaisuja.

Koska mobiilien tietoliikennejärjestelmien emuloinnista löytyi hyvin vähän tutkimustietoa, tullaan tällä sovellusalueella varmasti näkemään tulevaisuudessa paljon mielenkiintoista kehitystä. Mobiili tiedonsiirto kasvaa jatkuvasti, joten tarkkoja, toistettavissa olevia mittauksia varten tarvitaan tutkimusjärjestelmiä. Tällaiselle järjestelmälle varmasti on käyttöä mobiililaitteiden lisääntyessä ja tiedonsiirtomäärien kasvaessa jatkuvasti.

Lähteet

[1] M. Avvenuti ja A. Vecchio. Application-level network emulation: the emusocket toolkit. *Journal of Network and Computer Applications*, 29(4):343 – 360, 2006. ISSN 1084-8045. doi: 10.1016/j.jnca.2005.01.002. URL `http://www.sciencedirect.com/science/article/pii/S1084804505000044`.

[2] Jerry Banks, John S. Carson, Barry L. Nelson ja David M. Nicol. *Discrete-event system simulation*. Pearson education international, neljäs painos, 2005. ISBN 0-13-129342-7.

[3] R. Bradford, R. Simmonds ja B. Unger. Packet reading for network emulation. *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2001. Proceedings. Ninth International Symposium on*, sivut 150 –157, 2001. doi: 10.1109/MASCOT.2001.948864.

[4] Mark Carson ja Darrin Santay. Nist net: a linux-based network emulation tool. *SIGCOMM Comput. Commun. Rev.*, 33(3):111–126, heinäkuu 2003. ISSN 0146-4833. doi: 10.1145/956993.957007. URL `http://doi.acm.org/10.1145/956993.957007`.

[5] K Fall. Network emulation in the vint/ns simulator. *International Symposium on Computers and Communications*, sivut 244–250. IEEE, 1999. ISBN 0-7695-0250-4. doi: 10.1109/ISCC.1999.780820.

[6] Yan Gu ja Richard Fujimoto. Adaptive model update algorithms for remote network emulation. *Proceedings of the 22nd Workshop on Principles of Advanced and Distributed Simulation*, PADS '08, sivut 15–22, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3159-5. doi: 10.1109/PADS.2008.26. URL `http://dx.doi.org/10.1109/PADS.2008.26`.

[7] X.W. Huang, R. Sharma ja S. Keshav. The entrapid protocol development environment. *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, osa 3, sivut 1107 –1115 vol.3, mar 1999. doi: 10.1109/INFCOM.1999.751666.

[8] Sandia National Laboratories. Sandia builds self-contained, android-based network to study cyber disruptions and help secure hand-held devices, October 2012. URL `https://share.sandia.gov/news/resources/news_releases/sandia-builds-self-contained-android-based-network-to-study-cyber-disruptions-and-help-secure-hand-held-devices/`.

[9] J. Liu. A primer for real-time simulation of large-scale networks. *41st Annual Simulation Symposium*, sivut 85–94. IEEE, 2008. doi: 10.1109/ANSS-41.2008.18.

[10] Jason Liu, Yue Li, Nathanael Van Vorst, Scott Mann ja Keith Hellman. A real-time network simulation infrastructure based on openvpn. *Journal of Systems and Software*, 82(3):473 – 485, 2009. ISSN 0164-1212. doi: 10.1016/j.jss.2008.08.015. URL `http://www.sciencedirect.com/science/article/pii/S016412120800188X`.

[11] ns 3 Project. Ns-3 tap netdevice, 2012. URL `http://www.nsnam.org/docs/release/3.15/models/html/tap.html`.

[12] Larry Peterson, Tom Anderson, David Culler ja Timothy Roscoe. A blueprint for introducing disruptive technology into the internet. *SIGCOMM Comput. Commun. Rev.*, 33(1):59–64, tammikuu 2003. ISSN 0146-4833. doi: 10.1145/774763.774772. URL `http://doi.acm.org/10.1145/774763.774772`.

[13] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu ja M. Singh. Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols. *Wireless Communications and Networking Conference, 2005 IEEE*, osa 3, sivut 1664 – 1669 Vol. 3. IEEE, march 2005. doi: 10.1109/WCNC.2005.1424763.

[14] Luigi Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *SIGCOMM Comput. Commun. Rev.*, 27(1):31–41, tammikuu 1997. ISSN 0146-4833. doi: 10.1145/251007.251012. URL `http://doi.acm.org/10.1145/251007.251012`.

[15] Amin Vahdat, Ken Yocum, Kevin Walsh, Priya Mahadevan, Dejan Kostić, Jeff Chase ja David Becker. Scalability and accuracy in a large-scale network emulator. *SIGOPS Oper. Syst. Rev.*, 36(SI):271–284, joulukuu 2002. ISSN 0163-5980. doi: 10.1145/844128.844154. URL `http://doi.acm.org/10.1145/844128.844154`.

[16] S.Y. Wang, C.L. Chou ja C.C. Lin. The design and implementation of the nctuns network simulation engine. *Simulation Modelling Practice and Theory*, 15(1):57 – 81, 2007. ISSN 1569-190X. doi: 10.1016/j.simpat.2006.09.013. URL `http://www.sciencedirect.com/science/article/pii/S1569190X06000803`.

[17] Klaus Wehrle, Mesut Güneş ja James Gross. *Modeling and Tools for Network Simulation*. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-12330-6. doi: 10.1007/978-3-642-12331-3.

[18] Elias Weingärtner, Florian Schmidt, Hendrik vom Lehn, Tobias Heer ja Klaus Wehrle. Slicetime: A platform for scalable and accurate network emulation. *8th USENIX Symposium on Networked Systems Design and Implementation*. USENIX, Berkeley, CA, USA, 2011. URL `http://static.usenix.org/events/nsdi11/tech/full_papers/Weingartner.pdf`.

[19] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb ja Abhijeet Joglekar. An integrated experimental environment for distributed systems and networks. *Proc. of the Fifth Symposium on Operating Systems Design and Implementation*, sivut 255–270, Boston, MA, joulukuu 2002. USENIX Association.

[20] Pei Zheng ja L.M. Ni. Empower: a cluster architecture supporting network emulation. *Parallel and Distributed Systems, IEEE Transactions on*, 15(7):617 – 629, july 2004. ISSN 1045-9219. doi: 10.1109/TPDS.2004.21.