

Santeri Karhu

Radioverkkosimulointityökalun hyödyntäminen protokollakerrosten kehityksessä

Bittium

Insinööri (AMK)

Tieto- ja viestintätekniikka

Kevät 2023



KAMK • University
of Applied Sciences

Tiivistelmä

Tekijä(t): Karhu Santeri

Työn nimi: Radioverkkosimulointityökalun hyödyntäminen protokollakerrosten kehityksessä

Tutkintonimike: Insinööri (AMK), tieto- ja viestintätekniikka

Asiasanat: OMNest, Simulointi, Protokollakerros, Ohjelmistotestaus

Tämä opinnäytetyö on tehty työsuhteessa ja toimeksiantosopimuksen alaisena Bittium Wirelesillä, joka on tilannut työn. Bittium on Oululaislähtöinen insinööritalo, joka tuottaa tietoturvallisia viestintäratkaisuja. Työ tehtiin osana suurempaa projektia päämääränä kehittää projektin ohjelmistotestaustyökaluja ja auttaa ohjelmistoarkkitehtia spesifikaatioiden luonnissa. OMNest oli työn alussa tekijälle täysin tuntematon, ja työhön kuului työkaluun liittyvään materiaaliin tutustuminen ja työkalun käytön opiskelu.

Työssä integroitiin projektissa kehitysvaiheessa käytettyyn ilmarajapintasimulaattoriin radiokanavasimulointiominaisuus. Integraatio on toteutettu OMNestilla, kaupallisella sovelluksella. OMNest valikoitui sen kattavien simulointiominaisuuksien vuoksi.

Integraation tuloksena oli ohjelmistokokonaisuus, jota voidaan hyödyntää ohjelmistokehittäjän toimesta yksikkötesteissä sekä ohjelmistoarkkitehdin käsissä spesifikaation luomiseen.

Tehdyn integraation avulla voidaan suorittaa PC-ympäristössä kehittäjän toimesta sellaisia simulaatioskenaarioita, joita ei ilman OMNestia ole ennen kyetty suorittamaan. Näin voidaan varmistaa sellaista toiminnallisuutta, mitä on kyetty testaamaan vain oikeilla laitteilla. Jatkossa OMNest-ohjelmaan voidaan lisätä toiminnallisuutta OMNestin ja sen kumppanikirjasto INET:in avulla.

Abstract

Author(s): Karhu Santeri

Title of the Publication: Utilization of a Radio Network Simulator in the Development of Protocol Layers

Degree Title Bachelor's Degree in Information and Communication Technology, Bachelor of Engineering (AMK)

Keywords: OMNest, Simulation, Protocol layer, Software testing

This thesis was commissioned by Bittium Wireless via a commissioning agreement contracted with the thesis worker. Bittium is an engineering house based in Oulu, Finland that is focused on secure communications devices and software. The thesis was done as a part of a larger project, with its goal being the improvement of the software testing process within the project and help the software architect in creating specifications in the project. The software integrated was unknown to the thesis worker prior to the thesis, and so the study and familiarizing oneself with it was part of the process.

The work done consists of an integration of OMNest, a commercial simulation program, unto an in-house created over-the-air radio network simulator. The integration of OMNest adds radio channel simulation capabilities to the simulator and was chosen for the task for its vast array of available simulation functionalities.

The result of the integration is a combination of software that can be used to the benefit of software developers by creating new unit tests and aid software architects in creation of specifications.

With the integration making possible to run simulation scenarios that weren't feasible in a PC-environment without OMNest, new tests ensuring old functionality of source code can be created. In the future additional simulation functionality can be implemented with the aid of OMNest and its companion library INET.

Alkusanat

Haluaisin kiittää Bittium Wirelessiä työn tilaamisesta ja mahdollistamisesta. Erityiset kiitokset Janne Romppaiselle, Jari Nurmelle ja työn ohjaajalle Aki Rönkälle työn ohjaamisesta ja hyvistä neuvoista.

Kiitokset vanhemmilleni Virve ja Timo Karhulle, jotka ovat olleet suurena apuna tyttäreni hoitamisessa työn aikana.

Sisällys

1	Johdanto	1
2	OSI-malli.....	2
2.1	Alemmat kerrokset.....	3
2.2	Ylemmät kerrokset	4
3	Radioverkot	5
3.1	Ohjelmistotestaus ja simulointi.....	6
3.2	Bittiumin ilmarajapintasimulaattori	7
4	OMNest.....	8
4.1	Radioverkkojen luominen	9
4.2	Radiokanavan simulointi	13
4.3	Datan keräys.....	14
5	Radiokanavasimulaatiotoiminnallisuuden toteuttaminen.....	17
5.1	OMNest-ohjelma	17
5.2	OMNestin integrointi ilmarajapintasimulaattoriin.....	18
5.3	Muutokset ilmarajapintasimulaattoriin	19
5.3.1	Piiloasemaongelma	20
5.4	Simulaatioympäristön käyttäminen	22
6	Arviointi ja pohdinta	23
	Lähteet	24

Symboliluettelo

MAC	Medium Access Control. datakerroksen alikerros
MANET	Mobile Ad hoc Network. Langaton mobiiliverkko
OSI-Malli	Open System Interconnection. Avoimen järjestelmän yhteys
ISO	International Organization for Standardization
HTTP	Hyper Text Transfer Protocol. Selainten tiedonsiirtoprotokolla
IP	Internet Protocol
Debuggaus	Lähdekoodin tutkiminen ja korjaaminen vioilta
gdb	yleinen työkalu lähdekoodin debuggaukseen
Valgrind	Staattinen koodianalysaattori
Ilmarajapinta	rajapinta radioiden välillä
Olio-ohjelmointi	Ohjelmointiparadigma

1 Johdanto

Opinnäytetyön tarkoituksena oli integroida radiokanavasimulointitoiminnallisuus Bittiumin sisäisesti kehittämään ilmarajapintasimulaattoriin. Toiminnallisuus on lisätty toteuttamalla OMNestilla radiokanavan simulointiympäristö, joka on konfiguroitavissa kehittäjän tai ohjelmistoarkkitehdin haluamalla tavalla.

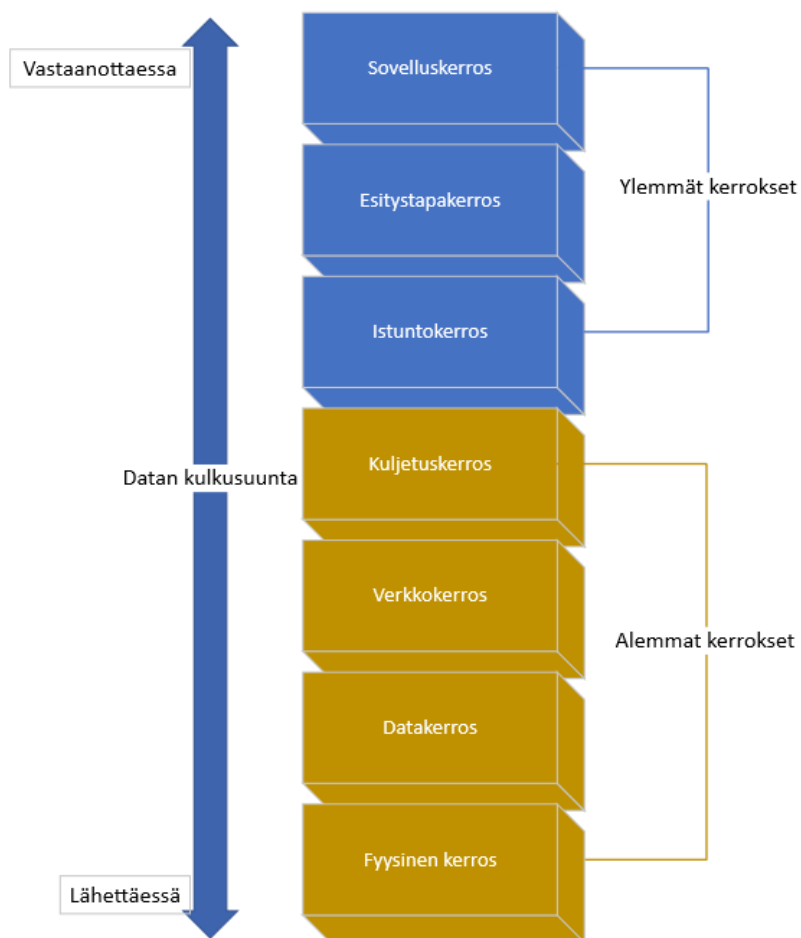
Radioverkkojen ja kanavien simuloinnin kehittämisen tarkoituksena on testausprosessin parantaminen. Testaus on olennainen osa modernia ohjelmistokehitystä, ja projekti, jossa työ on tehty, on päättänyt investoida sisäisten ohjelmistotestausvälineiden ja käytänteiden kehittämiseen. Kehittynyt testausprosessi nopeuttaa tuotantoa ja tekee kehittäjän työstä miellyttävämpää. Helppokäyttöisyys ja ihmisen luettavissa oleva testien kerätty data ovat keskipisteessä työn kehityksen motivaatioissa.

Simulaatioiden hyödyntäminen radiolaitteiden ohjelmakoodin testausprosessissa on kustannustehokasta verrattuna oikeilla radioilla testaamiseen moninaisista syistä: Radioiden valmistushinta, sekä maailmanlaajuinen komponenttipula ovat suurimpia tekijöitä. Mitä enemmän komponentteja pystytään kohdentamaan asiakkaille toimitettaviin tuotteisiin, sitä parempi. Tuotettu ohjelma edesauttaa ohjelmistokehittäjien ja arkkitehtien riippumattomuutta oikeista laitteista kehitys- ja suunnitteluvaiheissa tarjoamalla testausympäristön PC:lle, jolla voidaan varmentaa radiossa ajettavan protokollakerroksen toiminnallisuus.

Tuotespesifikaatiota luodessaan ohjelmistoarkkitehdin on tärkeää pystyä selkeästi varmistamaan data, mitä simulaatiosta kerätään, ja OMNestin avulla saadaan ajetuista simulaatioista raakojen lokitiedostojen lisäksi kerättyä graafisesti esitettäviä taulukoita ja histogrammeja, joiden avulla arkkitehdin on helppo varmentaa vanha toiminnallisuus ja kommunikoida kehittäjille ajatuksia ja suunniteltuja ominaisuuksia.

2 OSI-malli

OSI-malli on ISO:n luoma standardi, joka mahdollistaa yleisesti standardisoitujen protokollien käytön yhtenäisyyden kommunikaatiojärjestelmille. Mallissa jaetaan kommunikointiin vaadittavat toiminnallisuudet seitsemään protokollakerrokseen (ks. Kuva 1). Jokainen kerros suorittaa omia, toisistaan erillisiä toimintoja. [1]



Kuva 1. OSI-mallin kerrokset

Jokainen OSI-mallin kerros kommunikoi ylä- ja alapuolella olevien kerrostensa kanssa sekä lähettäessä että vastaanottaessa. Lähettäessä dataputken aloittaa sovelluskerros; vastaanottaessa fyysinen kerros toimii ensimmäisenä kerroksena vastaanottamalla dataa bittijonona esimerkiksi

verkkokaapelin kautta. Ylempien kerrosten työnä on pääasiassa sovellusten sisäinen datan käsittely ja alempien kerrosten toiminnallisuus on hoitaa tiedonsiirtoa. Tälle työlle olennaisimpana kerroksena on datakerros, jonka kehityksen parissa työ on pääasiassa toteutettu. [1.]

2.1 Alemmat kerrokset

Fyysinen kerros käsittää nimensä mukaisesti fyysisen datan liikkumisen esimerkiksi verkkokaapeleilla kytkimien välillä. Fyysisessä kerroksessa data muunnetaan bittijonoksi kommunikoivien laitteiden välillä yhteisesti sovitun tiedonsiirtoon käytetyn protokollan määräämällä tavalla. [1.]

Datakerroksen tehtävänä on huolehtia tiedonsiirrosta kahden laitteen välillä samassa verkossa. kerros purkaa verkossa lähetettävät paketit pienemmiksi kehyksiksi (ks. kuva 2) verkkokerrokselle lähetettäväksi. Kehys sisältää otsikon, datan ja vaihtoehtoisia osia riippuen verkon tarpeista. [1.]



Kuva 2. Geneerinen aikajaoteltu -kehys

Internetin välillä käytävässä kommunikaatiossa, jossa lähes kaikki data liikkuu IP:n määrittämällä paketeilla, käytetään verkkokerrosta. Sen tehtävänä on hoitaa reititys, IP-pakettien osoitteistaminen, sekä valmistaminen niitä lähettäessä ja paketteja vastaanottaessa niiden purkaminen. [1.]

Fragmentointi tarkoittaa lähetyksen sisältämän datan jakamista pienempiin osiin. Mikäli lähetyksen keskeytyy, sen sijaan että lähetyksen jouduttaisiin aloittamaan alusta, voidaan lähetyksen jatkua siitä mihin lähetyksen jäi. [1.]

IP-protokollan mukaan luodut paketit sisältävät otsikon (headerin), jonka avulla vastaanottaja osaa käsitellä paketin asianmukaisesti, sekä rungon, joka sisältää lähetettävän datan. Verkkojen välinen viestintä reititetään suoraan laitteelta toiselle tai erillisen reitittimen kautta. [1.]

2.2 Ylemmät kerrokset

Kahden laitteen välisessä yhteydessä kuljetuskerroksessa pidetään huolta istutokerroksen välittämän datan fragmentoinnista ja vuonohjauksesta sekä virheentunnistuksesta. Vuonohjaus varmistaa, että nopea lähettäjä ei ylikuormita hidasta vastaanottajaa päällekkäisillä lähetyksillä. Kerroksen suorittama virheentunnistus on vastaanotetun tiedonsiirron täydellisyyden varmistamista sekä pyyntöjen tekemistä tilanteissa missä näin ei ole. [1.]

Resurssien säästämisen nimissä laitteiden välinen yhteys avataan ja suljetaan sopivina aikoina istutokerroksen avulla. Istutokerros luo tiedonsiirron virhetilanteita varten tallennuspisteitä, jotta keskeytynyttä tiedonsiirtoa ei tarvitse aloittaa alusta. Esimerkiksi yhden gigatavun kokoisen siirtoon kerros voisi luoda pisteitä viiden megatavun välein, ja yhteyden katketessa toimitus voidaan aloittaa viimeksi luodusta pisteestä. [1.]

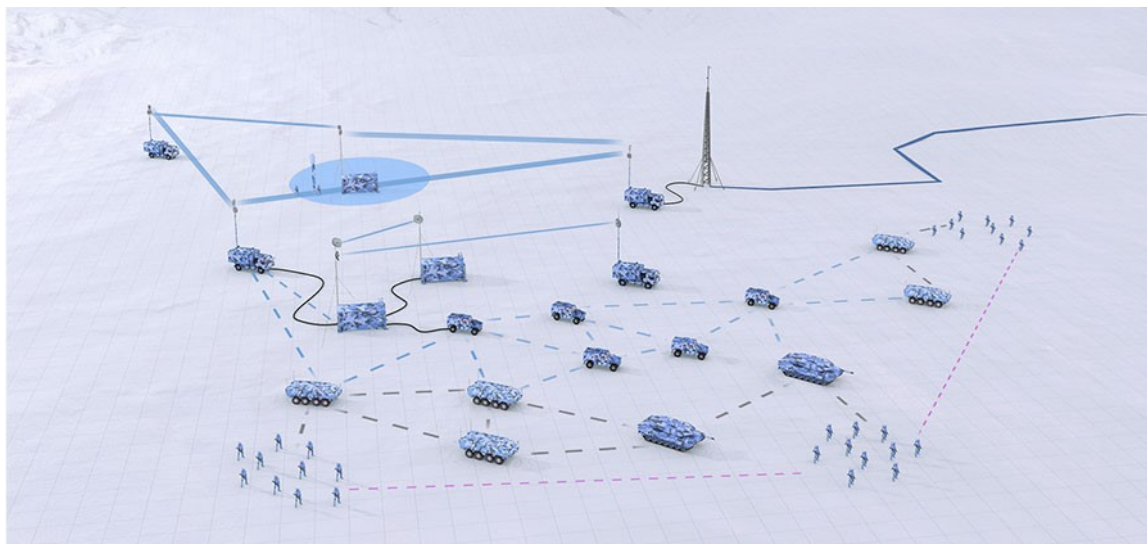
Esitystapakerroksen tehtävänä on varmistaa, että sovelluskerroksen valmistama lähetys on sopeva lähetettäväksi. Tämä voi sisältää datan muuttamista syntaksiin, joita erilaiset laitteet ymmärtävät, datan kryptausta sitä vaadittaessa sekä datan pakkaamista sen lähettämistä varten. [1.]

Sovelluskerros toimii käyttäjän kosketuspintana esitettävään dataan. Tuttuna esimerkkinä kerroksesta on HTTP, joka toimii World Wide Webin perustana. HTTP-pyyntöjen avulla avataan hyperlinkkien kautta nettisivuja. Kerrosta ei pidä sekoittaa itse sovellukseen, jonka käyttäjä näkee. Käyttäjän näkymä ei ole sovelluskerros, vaan sovellus tekee käyttäjän syötteen perusteella pyyntöjä sovelluskerrokseen. [2.]

3 Radioverkot

Verkoksi määritellään sellainen kokonaisuus, jossa on useita laitteita, jotka ovat yhdistetty toisiinsa, ja laitteiden välillä on tiedonsiirtoa. Laitteet voivat olla yhteydessä toisiinsa esimerkiksi verkkopiuhilla tai langattomasti.

Tämän työn käsittelemät radioverkot ovat MANET-verkkoja (ks. kuva 3). MANET-verkon olennaisia ominaisuuksia ovat sen nopeasti käynnistettävissä oleva langaton verkko ja sen itsehallinnoiva infrastruktuuriton toiminta. Itsehallinnoivassa rakenteettomassa verkossa jokainen solmu toimii reitittimenä, ja näin yksikään verkon solmu ei ole korvaamaton, kun verkko mukautuu itsenäisesti solmujen poistuessa ja saapuessa verkkoon. Tämän seurauksena MANET-verkot ovat edukseen sotilaskäytössä ja ulkotiloissa. [3.]



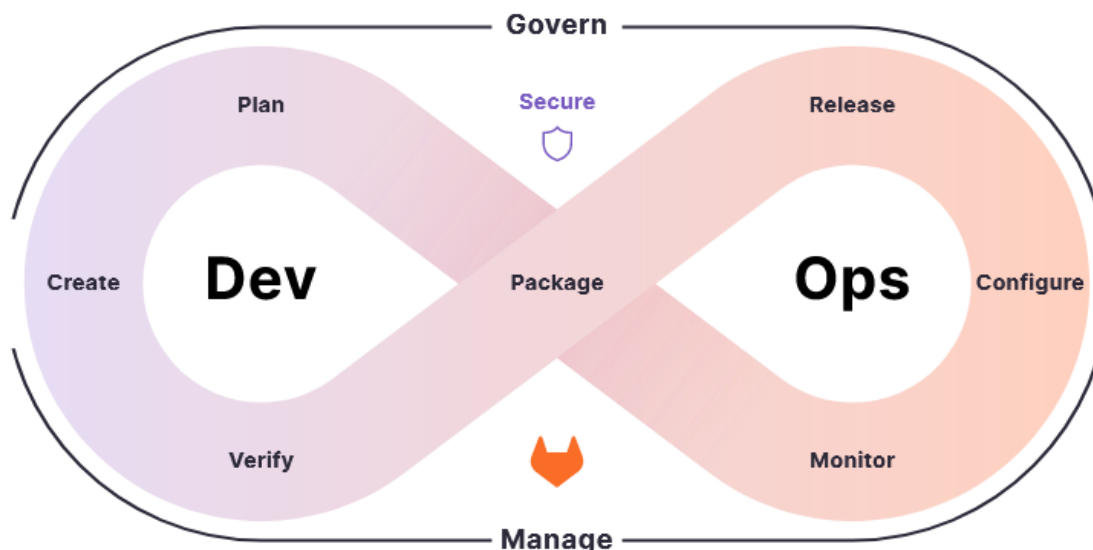
Kuva 3. Bittiumin taktinen IP-verkko

Näin verkko muodostuu vapaasti ja jokaisella solmulla on oma näkymä verkosta. Näkymä luodaan lähettämällä merkkiviestejä pulsseina, joihin saatujen vastausten perusteella kuva luodaan. [4.]

3.1 Ohjelmistotestaus ja simulointi

Ohjelmistokehityksen yksi olennaisimmista osista on testaus. Testaus koostuu monesta eri osasta, jotka muodostavat oikein toteutettuna laajan kokonaisuuden. Yksikkötestaus ja integraatiotestaus ovat osa laajempaa ketterän ohjelmistokehityksen mallia, jolla modernit ohjelmistotalot tuottavat ohjelmistoja.

DevOps on moderni metodologia ohjelmistokehitykseen, joka määrittää eri ohjelmistokehityksen vaiheet, jotka visualisoidaan äärettömyyden symbolilla (ks. Kuva 4). Mallin päätavoitteena on nopeuttaa ohjelmistokehitysprosessista, mikä tekee siitä mielekkäämpää kehittäjille ja edullisempaa yritykselle. Olennaisimpana työlle on mallin kehitys- ja varmistusvaihe, joiden aikana ohjelmistotestausta tehdään. [5.]



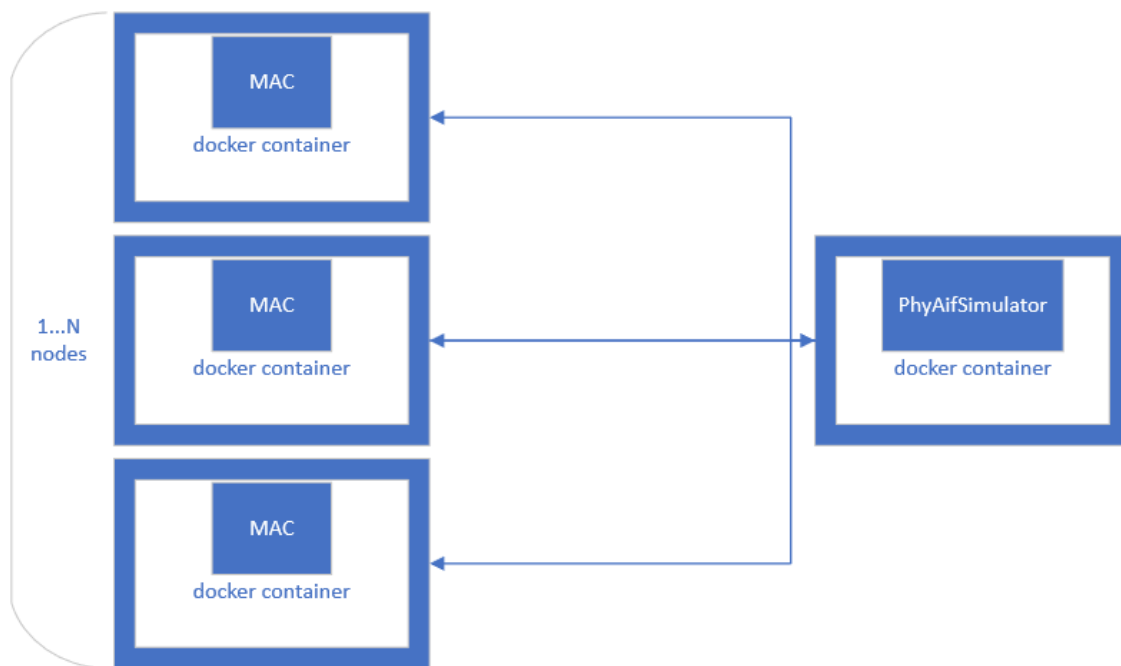
Kuva 4. DevOps-metodologiassa yleisesti käytetty kehityspotki. [5]

Tämän työ keskittyy kehittämään Plan- ja Verify-vaiheita. Plan eli suunnitteluvaiheessa ohjelmistoarkkitehti suunnittelee ja spesifioi valmistettavan ohjelman ominaisuudet ja vaatimukset. Verify, eli varmistusvaiheessa tapahtuu kaikki testaukseen liittyvä. Tässä työssä keskitytään manuaalitestaustapahtumaan, jota kehittäjä tekee ja se rajataan yksikkötestaukseen manuaalitestauksen saralta. Yksikkötestauksella varmistetaan ohjelmistokoodin eri osien toiminnallisuutta erillisillä testeillä. [5.]

3.2 Bittiumin ilmarajapintasimulaattori

Simulaattori on Bittiumin ennestään kehittämä työkalu datakerroksen testaukseen, jolla voidaan simuloida ilmarajapinta usealle samassa testiverkossa olevalle solmulle. Työkalu on hyödyllinen protokollakerroksen kehityksessä, sillä sen avulla voidaan ajaa ja debugata PC-simulointiympäristössä tismalleen samaa ohjelmistokoodia kuin mitä ajetaan Bittiumin radioissa (ks. kuva 5).

Docker on virtualisointityökalu, jonka avulla luodaan virtualisointiympäristöjä, joiden avulla luodaan löysästi eristettyjä ympäristöjä, eli kontteja. Simulaattori ja simulaattoria vasten ajettavat solmut ajetaan Docker-konteissa. Docker-kontteja käytetään simulointiympäristön luontiin ja ajamiseen vähäisen yleisrasitteen, nopean ja helpon käytettävyyden sekä niiden tarjoamien jatkokehitysmahdollisuuksien vuoksi. [6.]



Kuva 5. Ilmarajapintasimulaattori ja solmuja Docker-konteissa

4 OMNest

OMNest on Cogitative Software FZE:n kehittämä kaupallinen verkkojen ja radiokanavien simulointiin käytettävä työkalu. Verkot, joita OMNestin avulla kyetään luomaan, ovat täysin käyttäjän konfiguroitavissa, ja ne voivat koostua lähestulkoon mistä vain, langattomista Ad hoc -verkoista lehmien ammunissimulaatioihin. [7.]

Välttääkseen sekaantumista on tärkeää tietää OMNestin ja OMNet++:n ero. Toisin kuin OMNest, OMNet++ on avoimen lähteen yleisön kehittämä avoin työkalu (ks. kuva 6). Tämä ohjelmiston julkaisumalli mahdollistaa opiskelijoille ja yksityishenkilöille tutustumisen OMNest-ympäristöön ilman kallista lisenssiä, ja ilmaisversio tarjoaakin käyttäjälleen laajalti olennaisimmat opeteltavat OMNestin ominaisuudet. [8.]

	OMNet++	OMNEST
License	Academic Public License ¹	Commercial License
Commercial use	not allowed ¹	allowed
Simulation kernel, tools, examples, documentation	yes	yes
Eclipse-based Simulation IDE	yes	yes
Support for all major operating systems ²	yes	yes
Windows installer	no (distributed as zip)	yes
Pre-compiled (and tested) simulation libraries for Windows	no	yes ³
Support for Microsoft Visual C++	no	yes
Support for the GCC Compiler ⁴	yes	yes
Documentation Generation (example: INET ⁵)	yes under Creative Commons ⁵	yes
SVG Image Export ⁶	no	yes
SystemC Integration ⁷	no	yes
HLA Support ⁸	no	yes
Support	informal, via the mailing list	guaranteed 48-hour email support available
Service Releases	informal	guaranteed after significant fixes, but at least every 6 months

Kuva 6. OMNestin ja OMNet++:n eroja [8]

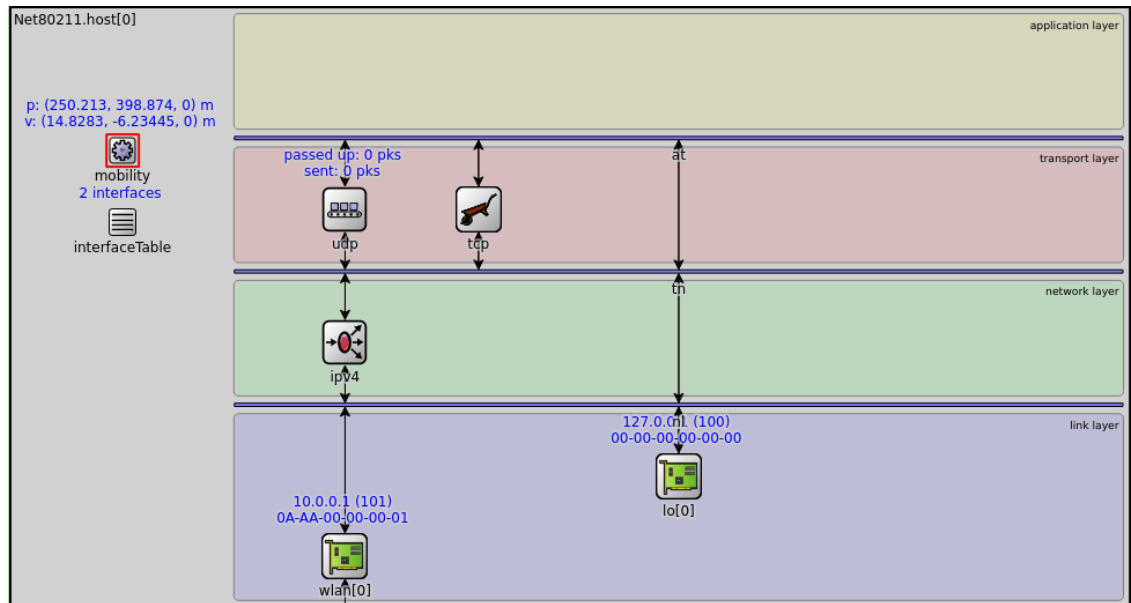
OMNest on kehitetty skaalautuvuus mielessä, ja näin luodut verkot ovat todella tehokkaita. Simulaatioiden ajaminen on erittäin nopeaa, nopeus käytännössä rajoittuu vain niitä ajavan tietokoneen suorituskykyyn. Sen ohjelmakehys on pääasiallisesti kehitetty C++:lla, ja siihen tutustuneelle käyttäjälle tämä näkyy tuttuna syntaksina. Ohjelmiston yhtenä myyntivalttina onkin, että verkkojen kehittäminen on olio-ohjelmointia harrastaneelle käyttäjälle nopeasti opeteltavissa. [7.]

OMNestia on käytetty työssä sen radiokanavan simulointiominaisuuden sekä datankeräysominaisuuden vuoksi. Se on liitetty osaksi ilmarajapintasimulaattoria sen sijaan, että sitä käytettäisiin luomaan koko simulaatioympäristö. Näin on päätetty, koska olemassa olevat protokollakerrosten simulointityökalut palvelevat tarkoitustaan oikein hyvin, eikä ole kustannustehokasta lähteä rakentamaan uudestaan valmiiksi kehitettyjä ja hyvin toimivia työkaluja.

Valinta käyttää OMNestia radiokanavasimulointiin perustuu sen monipuolisuuteen, käyttäjäystävällisyyteen ja datankeräysominaisuuksiin. OMNest tuo ohjelmistokehitysprosessin suunnittelu- ja kehitysvaiheisiin ketteryyttä. Testatessa ohjelmistokoodia PC-simulointiympäristössä voidaan testata erilaisia valmiiksi luotuja skenaarioita, jotka testaavat ohjelmiston kriittistä toiminnallisuutta ja luovat ihmisen luettavaa tietoa verkon toiminnallisuudesta ilman oikeita radiolaitteita erilaisten graafien ja lokien avulla. Tämänkaltainen testaus luo kestävän ja toistettavan tiedotusputken kehittäjältä suunnittelijalle, josta ohjelmistokehitysprosessi vain hyötyy.

4.1 Radioverkkojen luominen

OMNestin verkkojen ohjelmoinnin tärkeimpiä käsitteitä ovat moduulit. Moduulit ovat korkean tason abstraktio, joilla helpotetaan verkkojen sisältämien ominaisuuksien ohjelmointia ja tarkastelua. Moduuleja asetellaan usein sisäkkäin, ja verkossa olevan solmun simuloiva moduuli voi esimerkiksi sisältää protokollakerrosten ominaisuuksia erillisinä alimoduuleina sisällään (ks. kuva 7). [9.]



Kuva 7. Ad hoc -moduulin rakenne ja alimoduulit.

Moduuleja voidaan käyttäjän toimesta ohjelmoida C++:lla OMNestin omalla kirjastolla. Moduulit käsitellään C++-koodissa luokkina, joihin peritään ominaisuuksia OMNest-kirjastosta C++-ohjelmointikieleltä käyttäneelle tutulla tavalla. Koodissa luodut luokat määritellään NED-tiedostossa C++-koodia vastaaviksi, yleensä Simple-tyyppisiksi olioiksi (ks. kuva 8).

```
simple node
{
    parameters:
        @signal[arrival](type="long");
        @statistic[hopCount](title="hop count"; source="arrival"; record=vector,stats; interpolationmode=none);
    gates:
        inout gate[];
}

using namespace omnetpp;

class node : public cSimpleModule
{
protected:
    virtual void initialize() override;
    virtual void handleMessage(cMessage *msg) override;
    virtual void forwardMessage(frame *msg);
    virtual frame *generateMessage(recipient rec, int recId = 0);
private:
    cLongHistogram hopCountStats;
    cOutVector hopCountVector;
    long msgsSent;
};
```

Kuva 8. moduulin NED ja C++ toteutus, joihin on peritty cSimpleModule.

Moduulien ohjelmointi voidaan myös välttää OMNestin kumppaniohjelma INET:n avulla. INET on komponenttikirjasto OMNestille, jolla voidaan tuoda valmiiksi ohjelmoituja moduuleja luomaan toiminnallisuutta verkkoon (ks. kuva 9).


```

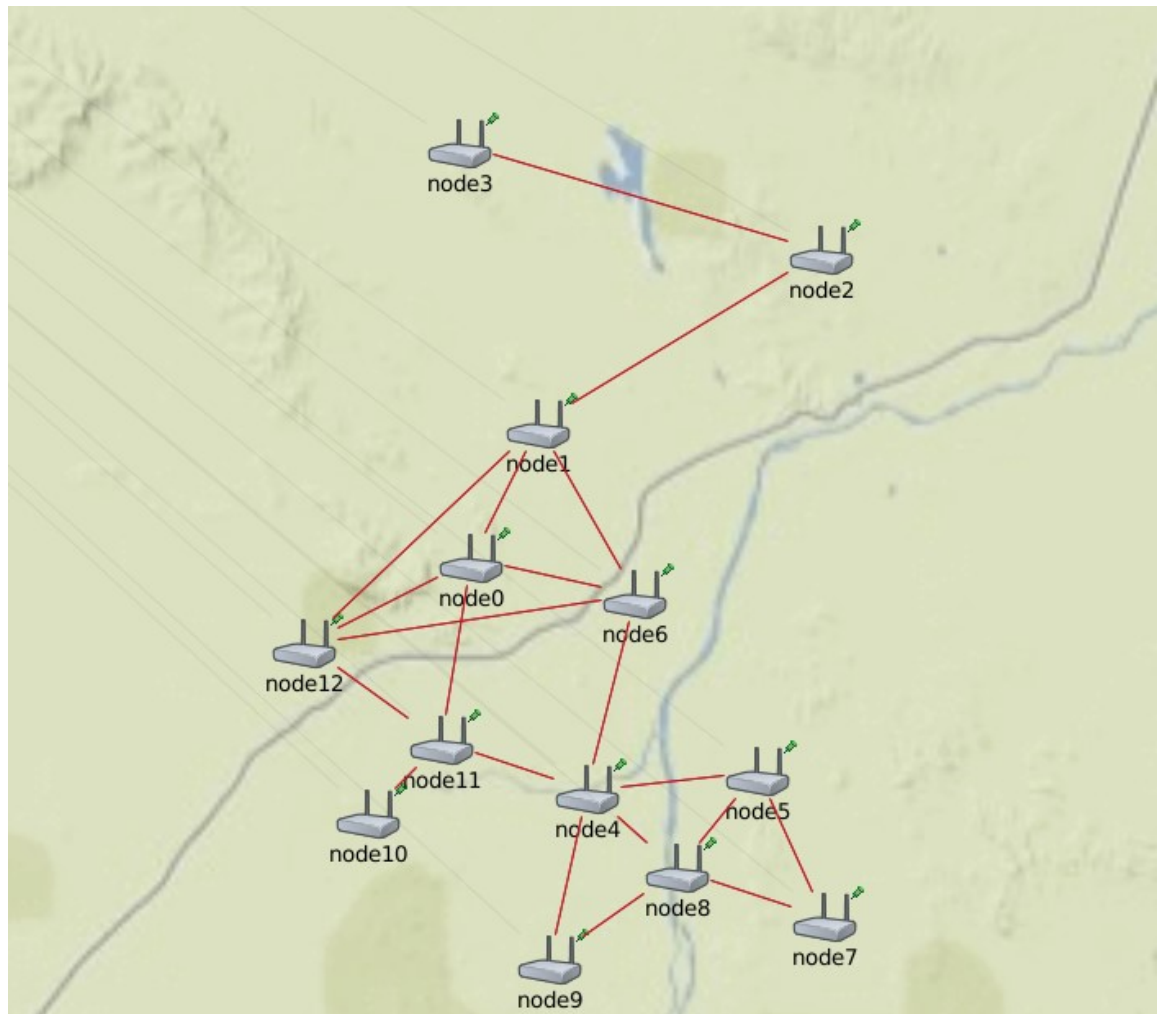
import inet.node.inet.INetworkNode;

network WirelessB extends WirelessA
{
    submodules:
        hostR1: <default("WirelessHost")> like INetworkNode {
            @display("p=250,300");
        }
        hostR2: <default("WirelessHost")> like INetworkNode {
            @display("p=150,450");
        }
        hostR3: <default("WirelessHost")> like INetworkNode {
            @display("p=350,450");
        }
}

```

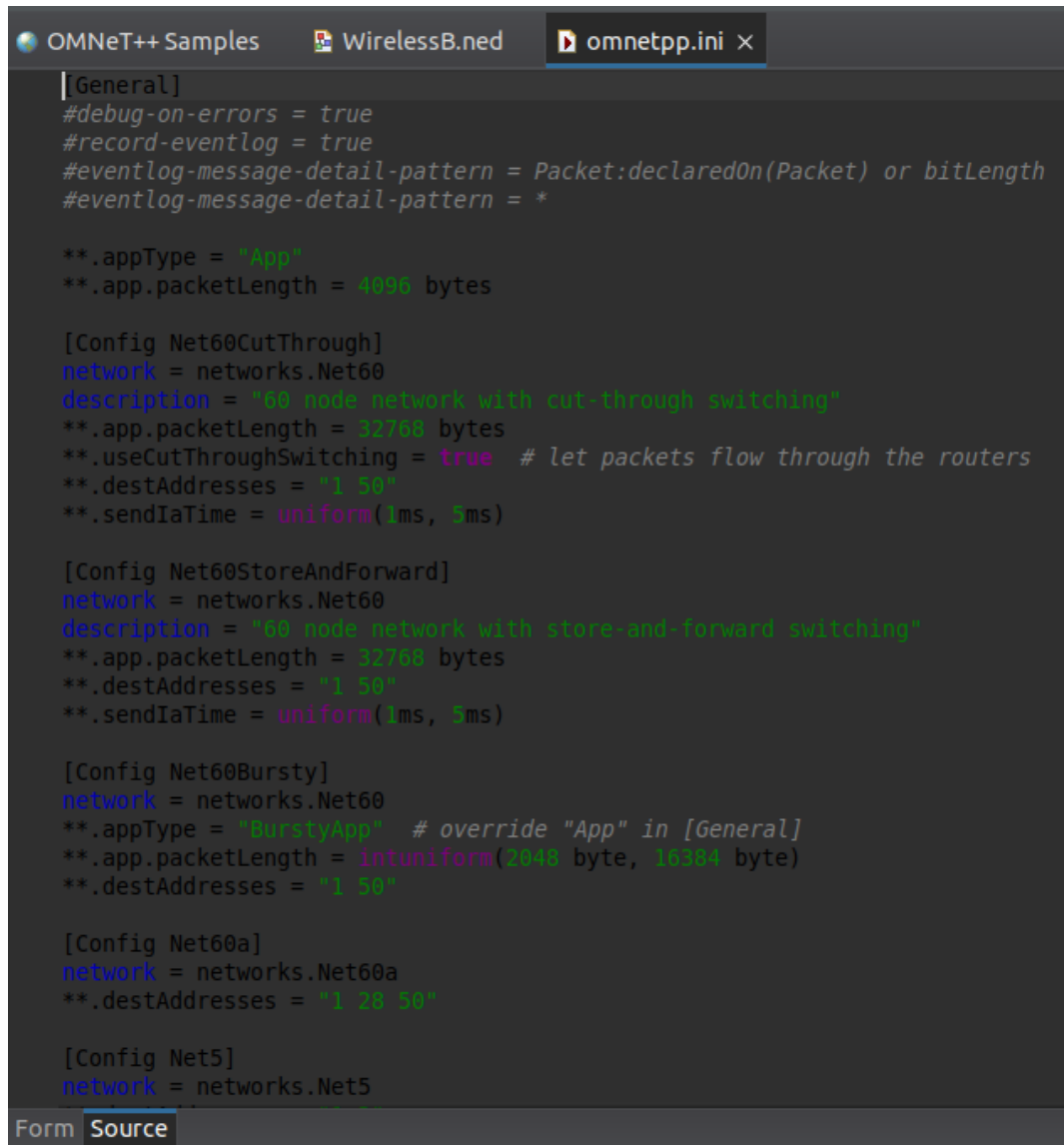
Kuva 9. moduulien luonti WirelessB-nimisen verkon sisällä Käyttäen INET:iä

Radioverkkojen luomiseen käytetään OMNestin omaa NED-tiedostomuotoa. Tiedostoja voidaan luoda Java-ohjelmointikielen tyyllisellä syntaksilla lähdekoodia muokkaamalla (ks. kuva 9) tai graafisella liittymällä (ks. kuva 10), jolla voi luoda topologioita ja määritellä radiokanavan asetuksia. Yleensä verkon luominen tapahtuu siten, että NED-tiedostoon luodaan verkko, jossa määritellään radiokanava, luodaan moduuleita ja signaaleita, joilla nauhoitetaan dataa simulaatiosta. [10]



Kuva 10. graafinen näkymä verkosta NED-editorissa

Simulaatio parametrisoidaan .ini-tiedostomuotoa olevalla tiedostolla (yleensä omnetpp.ini) (ks. kuva 11). Näin voidaan asettaa verkolle vakiosta poikkeavaa toiminnallisuutta ja antaa parametreja mallin topologialle. [9.]



```

[General]
#debug-on-errors = true
#record-eventlog = true
#eventlog-message-detail-pattern = Packet:declaredOn(Packet) or bitLength
#eventlog-message-detail-pattern = *

**.appType = "App"
**.app.packetLength = 4096 bytes

[Config Net60CutThrough]
network = networks.Net60
description = "60 node network with cut-through switching"
**.app.packetLength = 32768 bytes
**.useCutThroughSwitching = true # let packets flow through the routers
**.destAddresses = "1 50"
**.sendIaTime = uniform(1ms, 5ms)

[Config Net60StoreAndForward]
network = networks.Net60
description = "60 node network with store-and-forward switching"
**.app.packetLength = 32768 bytes
**.destAddresses = "1 50"
**.sendIaTime = uniform(1ms, 5ms)

[Config Net60Bursty]
network = networks.Net60
**.appType = "BurstyApp" # override "App" in [General]
**.app.packetLength = intuniform(2048 byte, 16384 byte)
**.destAddresses = "1 50"

[Config Net60a]
network = networks.Net60a
**.destAddresses = "1 28 50"

[Config Net5]
network = networks.Net5

```

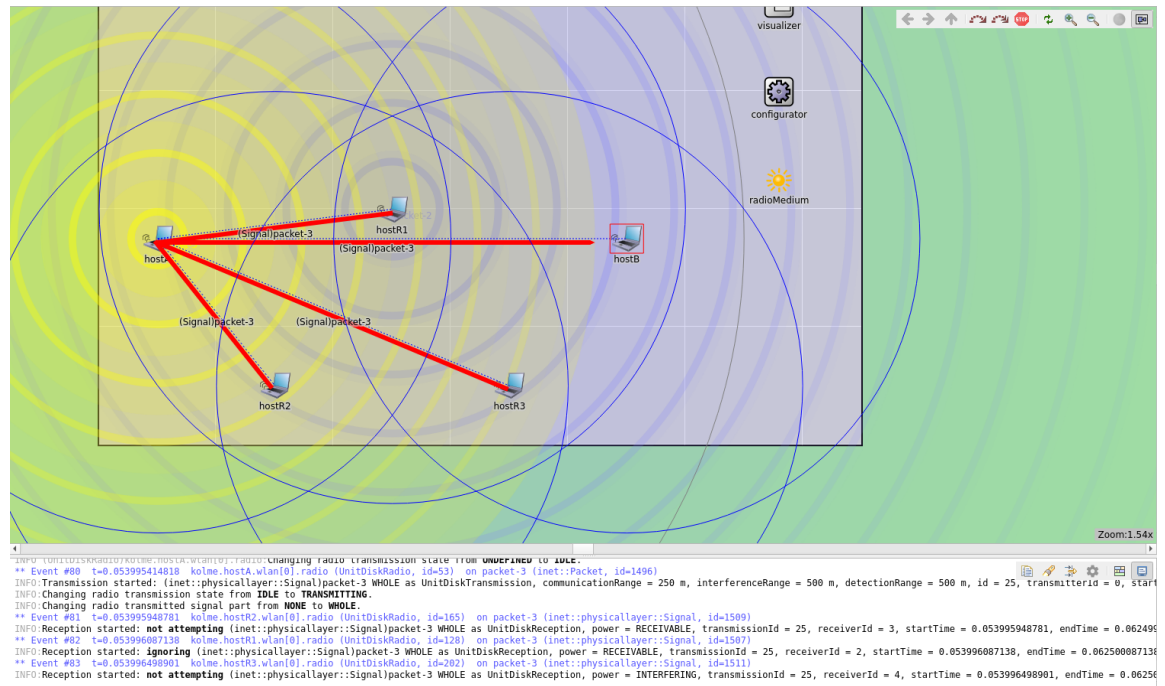
Kuva 11. OMNeT++:n INET-esimerkin initialisointitiedosto

4.2 Radiokanavan simulointi

NED-tiedostossa määritellään radiokanava, jonka välityksellä voidaan simulaatiossa lähettää viestejä. Näihin yleensä peritään jokin OMNeT++ kirjaston oma kanava, jota sitten laajennetaan omilla asetuksilla. Näistä kanavista olennaisimmat ovat delayChannel ja dataChannel. delayChannelin avulla määritellään yksinkertaisesti, kuinka kauan viesteillä kestää saapua solmulta toiselle. dataChannelin avulla voidaan määritellä tarkemmin kanavan toimintaa, tiedonsiirtonopeutta ja viivettä.

OMNestin kumppanikirjasto INET tarjoaa OMNestille valmiita radiokanavia ja verkonrakennustyökaluja, joille voidaan määritellä tarkempia asetuksia kanavan toiminnallisuuteen.

Simulaatiot voidaan kääntää käyttöliittymän kautta, jonka jälkeen simulaatiot voidaan ajaa komentolinjalta tai kyseisessä käyttöliittymässä (ks. kuva 12). Komentolinjalla ajamisessa on etuna se, että siinä voidaan käyttää gdb:tä, joka on yleisesti käytössä oleva debuggaukseen käytetty työkalu, joten lähdekoodin debuggaukseen ei tarvitse opetella uutta työkalua.



Kuva 12. INETin Ad Hoc -verkon ajaminen graafisessa käyttöliittymässä

4.3 Datat keräys

OMNest tarjoaa kattavat työkalut tilastien keräykseen erilaisiin muotoihin. Tietoa simulaatioista voidaan tallentaa monenlaisen eri tiedostomuotoon monella tapaa.

Signaalit ovat yksi olennaisimmista tavoista kerätä simulaatiosta tietoa (ks. kuva 13). Signaaleja voidaan asettaa lähetettäväksi, mistä vain oman C++-koodin sisältä. Näin on helppo suunnitella kustomoitu simulaatiosta luotu datakokonaisuus. Signaalien keräämää dataa kerätään erilaisiin OMNestin tarjoamiin C++-luokkiin käärittyihin säiliöihin. [9.]

```

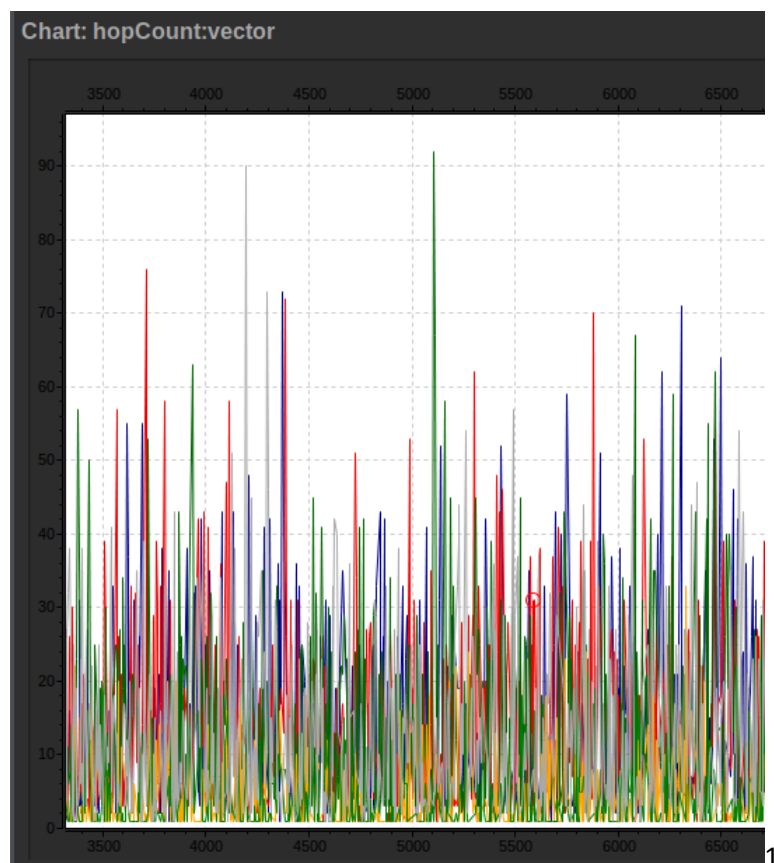
class Txc17 : public cSimpleModule
{
private:
    simsignal_t arrivalSignal;
}

simple Txc17
{
    parameters:
        @signal[arrival](type="long");
        @statistic[hopCount](title="hop count"; source="arrival"; record=vector,stats; interpolationmode=none);
    emit(arrivalSignal, hopcount);
}

```

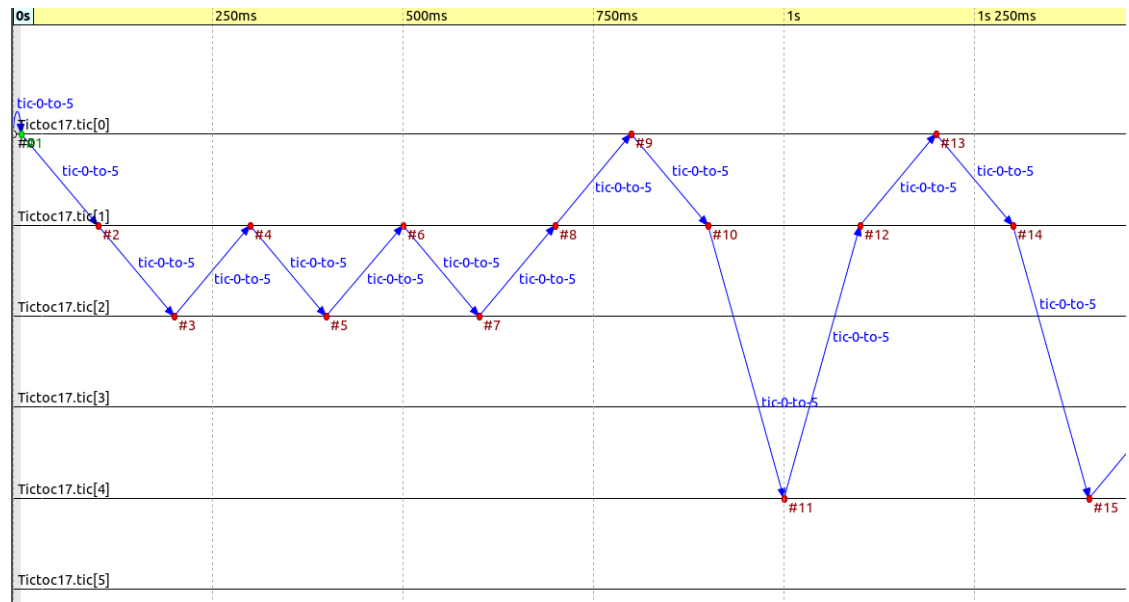
Kuva 13. Signaalin määrittely, datan kerääminen signaalista säiliöön, ja emit-funktio, jolla päästetään signaalin avulla "hopcount" niminen muuttuja

cOutVector-olioon tallennetaan koko simulaation ajalta useita datanäytteitä, joita tallennetaan record-funktiolla suoraan koodista. Näitä voi olla esimerkiksi eri lähetysten ajat hetkittäiset verkon solmujen tilat. Kerättyä dataa voidaan lukea ja visualisoida haluamallaan tavalla OMNestin käyttöliittymällä (ks. kuva 14). [9.]



Kuva 14. vektoridataa visualisoituna OMNest:n plot-työkalulla.

Kertaluontoisia tallenteita, kuten verkon elinikä tai viestien kokonaismäärä, tallennetaan cScalar-luokan alulla. Asettamalla simulaation .ini-tiedostossa asetus "record-eventlog=true", saadaan simulaation tapahtumista nauhoitettua elog-muotoinen tiedosto (ks. kuva 15). Sequence Chart -työkalulla voidaan tarkastella luotua tiedostoa OMNestin IDE:n sisällä. [9.]



Kuva 15. OMNet++:n tictoc-esimerkkisimulaation elog-dataa. [10]

cHistogram-luokalla tehdään histogrammeja, vaikkapa signaalien välittämästä datasta (ks. kuva 16). Histogrammien esittämistyyliä voidaan muuttaa, sekä sillä voidaan visualisoida useita datapisteitä yhtä aikaa.



Kuva 16. Histogrammiin kerättyä hop count -dataa

5 Radiokanavasimulaatiotoiminnallisuuden toteuttaminen

Työn tekoon kuului suunnittelu- sekä toteutusvaihe. Työn suorittamiseen annettiin projektin toimesta vaadittavat ominaisuudet ohjelmalle, ohjelmistorakenteen ja siihen käytettävät tekniikat olivat suorittavan tahon päätettävissä.

Ohjelman kehityksen pohjana oleva käyttöjärjestelmä oli alun perin Ubuntu 18.04, mutta tämä muuttui kehityksen aikana Windows 10:een ja sen jälkeen päädyttiin lopulta Debian 10:een. Käyttöjärjestelmän vaihtuvuuteen vaikutti turvaluokiteltu ympäristö, missä sovelluskehitys tapahtui, ja näin ohjelmistojen ja työkalujen saatavuudessa oli viivästyksiä.

Ilmarajapintasimulaattoriin tehtiin myös muutoksia, jotta se saatiin mukautumaan OMNestiin. Olennaista toiminnallisuutta simulaattorista ei muutettu, siihen vain lisättiin uutta toiminnallisuutta.

OMNest-ohjelma oli määrä integroida ilmarajapintasimulaattoriin siten, että käyttäjälle se olisi intuitiivista ja helppoa. Radiokanavan asetukset täytyi olla kehittäjän muutettavissa, jotta skenaarioiden muokkaaminen olisi nopeaa ja helppoa. Simulaatioista datankeräys oli määrä rakentaa siten, että kerättyä dataa pystyisi tarkastelemaan visuaalisesti, kuten histogrammien tai OMNestin oman elog-tiedoston avulla.

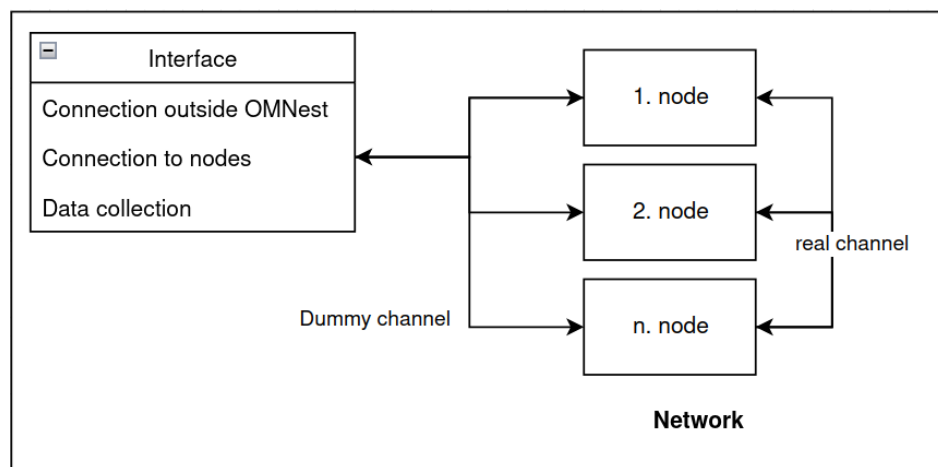
Työ saatiin aikataulussa valmiiksi ja tuloksena oli ohjelma, jolla saatiin lisättyä halutut ominaisuudet ilmarajapintasimulaattoriin. Simulaatioympäristön käyttämiseen luotiin käyttöohjeet, jotta kehittäjä, jolle simulaatioympäristö on ennestään tuntematon, kykenisi itsenäisesti käynnistämään ympäristön. Käyttöohjeessa käydään läpi testausympäristön rakentaminen, simulaation ajaminen ja ajetun simulaation datan kerääminen.

5.1 OMNest-ohjelma

OMNest oli tuntematon prosessin alussa tuntematon työkalu, ja niin suunnittelu- ja tutkimusvaiheessa siihen tutustuttiin erilaisten verkkomateriaalien ja ohjeistuksien avulla. Ohjelmistolle syntyi rakenne tutustumisen jälkeen, ja siitä tuli lopulta ohjelmistorakenteeltaan tavanomainen. Sellaiselle, jolle olio-ohjelmointiparadigma on tuttu, on helppoa silmäyksellä nähdä ohjelman rakenne.

Ohjelma rakentuu Node- ja Interface-luokista. Interface luokan tarkoituksena on toimia rajapintana ilmarajapintasimulaattoriin ja toimia keskipisteenä sekä suorittaa simulaatioiden datankeräys. Node-luokan avulla luodaan nimensä mukaisesti verkossa elävät solmut, jotka lähettävät keskenään viestejä radiokanavalla sekä kommunikoivat Interfacen kanssa lähetettyjen viestien tilasta.

Ohjelman verkossa on kaksi erilaista kanavaa: oikeaa kanavaa mallintava kanava sekä dummy-kanava (ks. kuva 17). Dummy-kanavaa käytetään Interfacen ja solmujen välillä, jotta saadaan karstittua datasta simulaatioon kuulumatonta dataa pois. Oikealla kanavalla, jolla simulaation viestit lähetetään, on käytössä tiedonsiirtonopeuden laskenta sekä lähetysten propagaatiolaskenta.

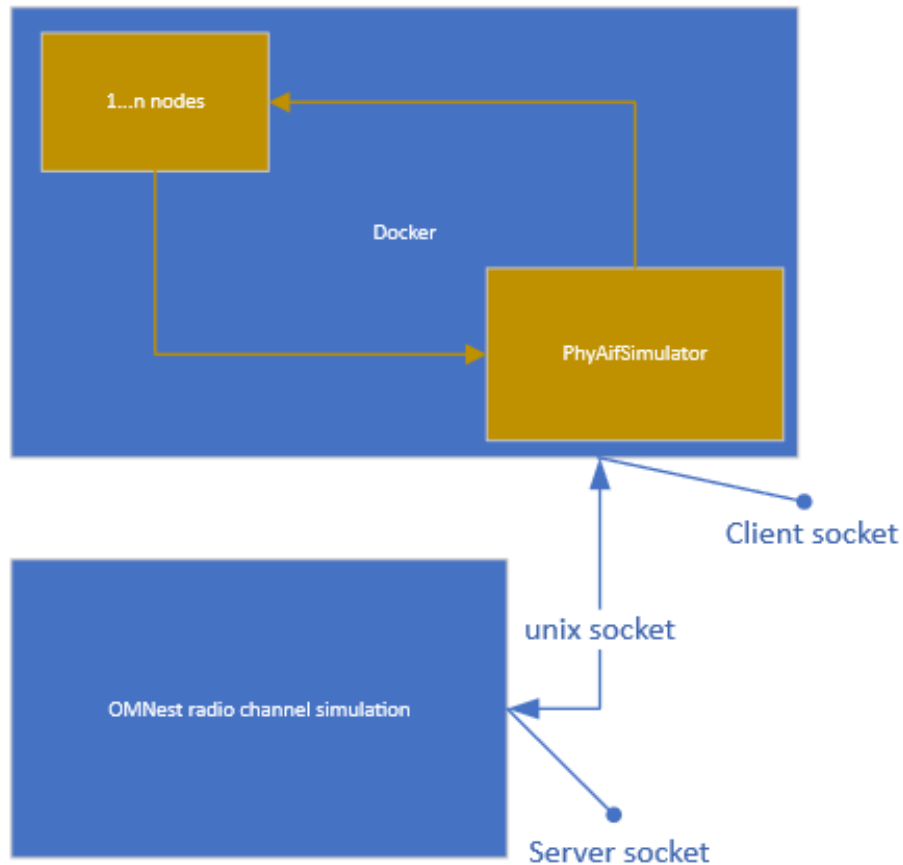


Kuva 17. OMNest-ohjelman kaavio

5.2 OMNestin integrointi ilmarajapintasimulaattoriin

Pistoke on ohjelmoinnissa yleisesti käytetty työkalu prosessien väliseen kommunikaatioputken luomiseen. Työhön valittiin Unix-pistoke prosessien väliseksi kommunikaatiotavaksi, sillä se toimii nopeasti vähäisellä koodimäärällä. Unix-pistokkeen yleinen tuntemus myös varmistaa sen, että tulevaisuudessa koodia lukeva ohjelmistokehittäjä tietää nopeasti, miten rajapinta on toteutettu ja osaa tehdä siihen mahdollisia muutoksia ilman, että hänen täytyy tutustua johonkin vähän tunnettuun IPC-ratkaisuun.

Unix-pistoke on paikallisten prosessien väliseen kommunikaatioon tarkoitettu työkalu. Se on suunniteltu olemaan mahdollisimman tehokas resurssien säästämiseksi. Unix-pistokkeen luominen tapahtuu samalla tavalla kuten muidenkin ohjelmoinnissa käytettyjen pistokkeiden luominen [11]. Pistokkeen voi luoda monella ohjelmointikielellä, tässä työssä se on toteutettu C++:lla. Serveripistoke on luotu OMNestiin, ja asiakaspistoke ilmarajapintasimulaattoriin. Simulaattori voi avata monta pistoketta, jos useampi solmu haluaa lähettää viestejä yhtä aikaa (ks. kuva 18).



Kuva 18. OMNest-simulaation ja radioverkkosimulaation välinen yhteys

5.3 Muutokset ilmarajapintasimulaattoriin

Simulaattoriin lisättiin asiakaspään unix-pistoke viestien lähetykseen, jolla lähetetään OMNestille tarvittavat tiedot lähetyksestä, jotta OMNest voi simuloida radiokanavallaan lähetyksen sekä suorittaa nauhoituksen simulaatiosta. Ilmarajapintasimulaattorin välittää pistokkeella tiedon sol-

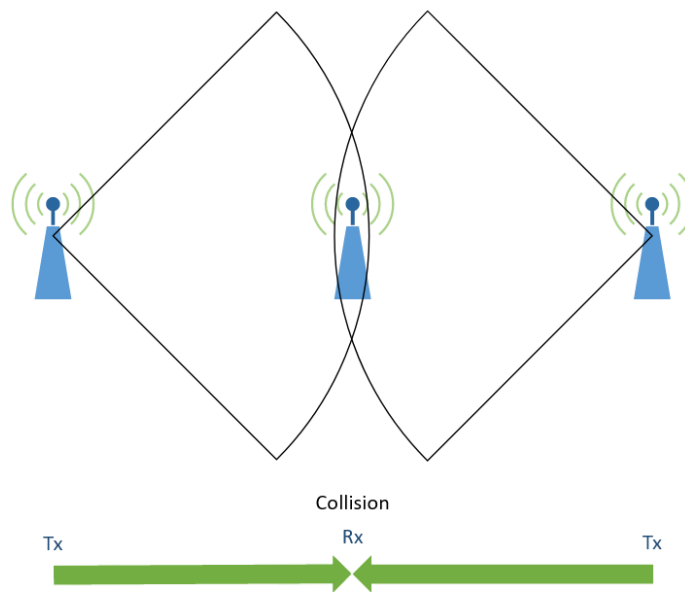
muista, joiden välillä tulisi tapahtua viestinlähetys. Suoritettuaan annetun viestinlähetysten OMNest palauttaa pistokkeella bool-arvolla, oliko viesti mahdollista lähettää. Simulaattori sitten jatkaa toimintaansa tämän arvon perusteella.

Ilmarajasimulaattorin käynnistysvaiheeseen lisättiin C++:lla toteutettu NED-tiedostojen automaattinen generointi. Simulaattori näkee verkossa olevat solmut XML-tiedostossa olevasta konfiguraatietiedostosta, joka sisältää solmujen sijaintidatan kaksikulotteisessa verkossa sekä muita verkon ominaisuuksia. Tämän konfiguraatietiedoston perusteella luodaan OMNestille NED-tiedosto simulaattorin käynnistystyksen yhteydessä.

Simulaattorissa oli myös työn alussa ratkaisematon piiloasemaongelmaan (ks. kappale 5.1.1) liittyvä bugi. Simulaattori ei hyödyntänyt kanavaa, mikäli se havaitsi piiloasemaongelman, vaikka lähetykset olisivatkin olleet tarkoitettu eri solmuille ja törmäystä ei olisi tapahtunut. Tämä korjattiin tarkastelemalla törmäävien lähetysten suuntia ja tekemällä lähetyspäättös näiden vertailuiden perusteella.

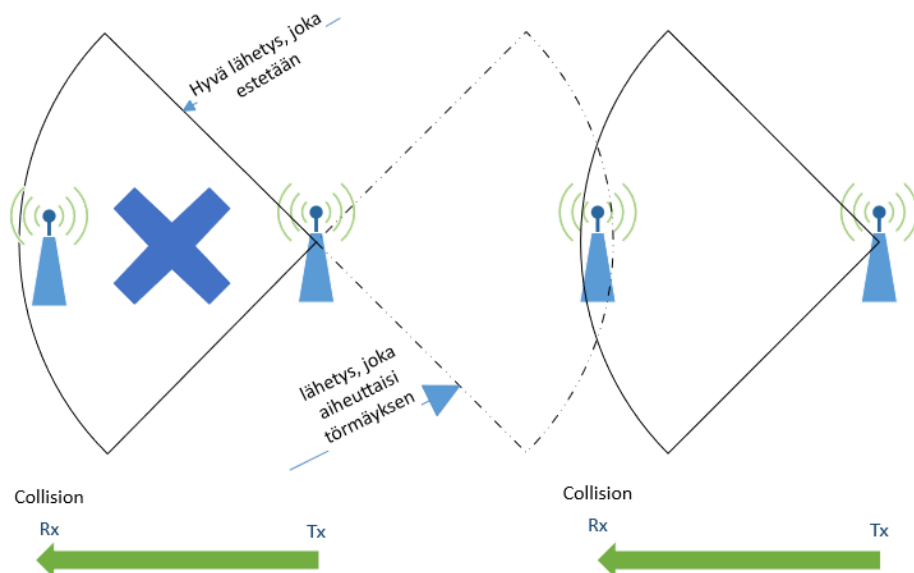
5.3.1 Piiloasemaongelma

Eräs olennaisimmista ongelmista radioverkkojen kehityksessä on piiloasemaongelma. Ongelma muodostuu, kun kaksi verkossa olevaa laitetta ei näe toisiaan, eivätkä osaa täten tarkistaa, onko muilla sama lähetysvuoro. Kun kaksi solmua, jotka luulevat yhdenaikaisesti heidän lähetysvuoronsa tulleen lähettävät viestiä samalle solmulle, tapahtuu lähetysten törmäys (ks. kuva 19), ja lähetys tuhoutuu.



Kuva 19. Piiloasemaongelman aiheuttama lähetyksen törmäys

Ongelmaan on kehitetty monenlaisia ratkaisuja, ja se voidaan ratkaista esimerkiksi lisäämällä lähettävien solmujen lähetystehoja tai liikuttamalla solmuja. Nämä eivät kuitenkaan ilmeisistä syistä ole kestäviä ratkaisuja. [1.]



Kuva 20. Epäihanteellinen ratkaisu ongelmaan

Kuvassa 20 ehdotettu ratkaisu voi johtaa tilanteeseen, jossa verkoissa, joissa solmuilla on suunta-antenni, solmut haluavat lähettää viestin eri suuntaan, mutta tällä ei ole lähetysvuoroa. Tällaisten

suunta-antenneille sopivien lähetysten estäminen lisää verkon hallintaviestintää, koska solmut joutuvat neuvottelemaan lähetysvuoroista. Lisätty signaointi lisää verkon yleisrasitetta. Näille tapauksille onkin sopivaa keksiä ratkaisu. Tämän voi ratkaista tekemällä verkon lähetysvuoroja ylläpitävään algoritmiin muutoksia, jotta algoritmi osaa ottaa suunta-antennit huomioon sallimalla eri suuntaiset lähetykset, jotka eivät törmää, vaikka lähetysvuoro solmuilla onkin päällekkäinen. [1]

5.4 Simulaatioympäristön käyttäminen

Simulaation ajaminen on nopeaa ja helppoa ympäristön pystyttämisen jälkeen. Ensin käynnistetään ilmarajapintasimulaattori ja OMNest, minkä jälkeen käynnistetään simulaatio ja verkossa olevat solmut.

Dataa kerätään simulaation ajan ilman käyttäjän syötettä, ja sitä voidaan tarkastella OMNestin työkaluilla simulaation päätyttyä. Simulaation jokainen osa, eli OMNest-simulaatio, ilmarajapintasimulaattori ja jokainen erillinen verkon solmu voidaan ajaa erikseen gdb:llä tai valgrindilla. Tämä on tehokas ominaisuus, sillä nämä työkalut ovat yleisessä tiedossa ja on hyvin todennäköistä, että ympäristön käyttäjä osaa näitä käyttää ennestään.

6 Arviointi ja pohdinta

Työn tehtiin osana projektia Bittiumilla, ja työn aikana haluttiin kehittää ohjelmistonkehityksen testaus- ja suunnitteluprosessia lisäämällä radioverkkosimulaattoriin radiokanavan simulointiominaisuus. Työlle luotiin yhdessä opinnäytetyöohjaajan sekä tilaajan kanssa vaatimukset ja suunnitelma sekä rajattiin työlle aihe. Rajaus tehtiin hyvin, laajuus oli sopiva opinnäytetyölle ja aihe oli sopivan haastava. Aikataulua noudatettiin, eikä työaika venynyt alkuperäisen määrätyn aikataulun ulkopuolelle.

Työvaihe alkoi hitaasti, työkalujen saatavuus haluttuun kehitysympäristöön on ympäristön turvallituksen takia aikaa vievä prosessi, joten työn alussa ei päästy tutustumaan kunnolla ympäristöön ja aika kului odotteluun. Kun työvaihe saatiin käyntiin kunnolla alkuvaikeuksien jälkeen, työnteko onnistui ilman kummempia vaikeuksia, ja hyvä tekninen tuki opinnäytetyön ohjaajalta auttoi monessa vaikeassa kohdassa. Varsinaista asiantuntijuutta ei OMNestin käyttöön ollut opinnäytetyön tekemiseen. Tämä toisaalta tarkoittaa sitä, että tekijän kannalta työ on ollut todella hedelmällistä, koska erilaisten uusien ja kompleksien työkalujen hallinta nopeasti auttaa kehittämään insinöörinä.

Suunniteltu ratkaisu toimii, kuten suunniteltiin käyttäen ennestään valmiita työkaluja, mutta tämä jättää pöydälle OMNestin tarjoaman koko verkon simulointimahdollisuuden. Halutessaan OMNestillä voitaisiin suorittaa kaikki simulaation osat ja Ilmarajapintasimulaattori voitaisiin jättää käyttämättä. Tämä lähestymistapa toisi erilaisia haasteita kehitystyölle kuin se mitä käytettiin, mutta kumpikaan tyyli ei ole toista parempi tai huonompi, sillä ne sisältävät molemmat erilaisia hyviä ja huonoja puolia. Tehty työ palvelee kuitenkin asiakkaan tilaamaa toimintoa, joten voidaan sanoa työn onnistuneen.

Tehty integraatio muodostaa vankan pohjan jatkokehitystyölle. Kenties parhaana lisänä nykyiseen ympäristöön voidaan pitää INETistä tuotuja realistisempia radiokanavamalleja, joilla mahdollistettaisiin vielä enemmän testiskenaarioita ja useampien toiminnallisuuksien varmentamista. Toisena lisänä voitaisiin kehittää tuki liikkeelle, myös INETillä. Tällä hetkellä simulaatiot koostuvat staattisten solmujen verkoista ja solmujen liikkuvuus lisäisi kyvyn simuloida skenaarioita, mitä ei tällä hetkellä pystytä PC-ympäristössä testaamaan.

Lähteet

1. Cloudflare, Inc: What is the OSI Model? [Internet] Tuntematon vuosi [Viitattu 09.01.2023] Saatavilla: <https://www.cloudflare.com/learning/ddos/glossary/open-systems-interconnection-model-osi/>
2. Cloudflare, Inc: What is HTTP? [Internet] Tuntematon vuosi [viitattu 28.01.2023] Saatavilla: <https://www.cloudflare.com/learning/ddos/glossary/hypertext-transfer-protocol-http/>
3. Bittium Wireless Oy: Tactical Manet product page. [Internet] Tuntematon vuosi [viitattu 05.02.2023] Saatavilla: [MANET - Mobile Ad-Hoc Networks | Bittium](#)
4. Jere Alanen: Hajautetun MANET-verkon tilatallenteiden yhdistäminen yhdeksi tietokannaksi. [Opinnäytetyö] 2021 [viitattu 17.02.2023] Saatavilla: <https://urn.fi/URN:NBN:fi:amk-2021052110273>
5. Gitlab: What is DevOps? [Internet] Tuntematon vuosi [Viitattu 05.02.2023] Saatavilla: <https://about.gitlab.com/topics/devops/>
6. Docker documentation: Docker overview. [Internet] Tuntematon vuosi [Viitattu 05.02.2023] Saatavilla: <https://docs.docker.com/get-started/overview/>
7. Cogitative Software FZE: What is OMNet++? [Internet] Tuntematon vuosi [Viitattu 05.02.2023] Saatavilla: (<https://omnetpp.org/intro/>)
8. Cogitative Software FZE: OMNEST - OMNeT++ Comparison [Internet] Tuntematon vuosi [Viitattu 31.01.2023] Saatavilla: <https://omnest.com/comparison>
9. Cogitative Software FZE: OMNEST overview [Internet] Tuntematon vuosi [Viitattu 02.02.2023] Saatavilla: <https://omnest.com/overview>
10. Cogitative Software FZE: OMNet++ Simulation Manual [Internet] Tuntematon vuosi [viitattu 17.02.2023] Saatavilla: <https://doc.omnetpp.org/omnetpp/manual/>
11. Cogitative Software FZE: OMNet++ introduction: Tictoc statistic collection [Internet] Tuntematon vuosi [viitattu 17.02.2023] Saatavilla: <https://docs.omnetpp.org/tutorials/tictoc/part5/>

12. Michale Kerrix: Linux man-pages, Unix socket. [Internet] 2021 [viitattu 29.01.2023] Saatavilla kaikkien Unix-tietokoneiden sisäisessä manuaalissa. Saatavilla myös verkossa 29.01.2023: <https://man7.org/linux/man-pages/man7/unix.7.html>
13. Cogitative Software FZE: OMNET++ User Guide Version 6.0 [Internet] 2021 [Viitattu 16.02.2023] Saatavilla: <https://doc.omnetpp.org/omnetpp/UserGuide.pdf>
14. Al-Shareera Mahmood A., Manickam Selvakumar: Man-in-the-Middle Attacks in Mobile Ad Hoc Networks (MANETs): Analysis and Evaluation. [Internet] 2022 [viitattu 28.01.2023] Saatavilla: <https://kamezproxy01.kamit.fi:2405/login.aspx?direct=true&db=a9h&AN=158943722&site=ehost-live>