

POLITECNICO DI TORINO

Master's Degree in Communications and Computer
Networks Engineering



Master's Degree Thesis

Dynamic management of cellular networks with Base Stations on wheels

Supervisors:

Professor Gianluca RIZZO

Professor Marco AJMONE MARSAN

Candidate:

Masoud SOLTANIAN

September 2020

Summary

This thesis's outcome is a moving base station simulator based on developing SimuLTE, which is an LTE cellular network simulator project written in C++ on the OMNeT++ framework. After adding more than 1000 lines of C++ code throughout this thesis, now SimuLTE has the moving base stations feature that facilitates future researches on this topic. You can find the complete code of the last version of SimuLTE featured by moving base stations through my GitHub:

<https://github.com/Masoudsultanian/MovingBaseStations.git>

C++ is one of the most complex programming languages in the world, and developers able working with this language consider it as their badge of honor. Difficulties of this thesis in terms of development and investigation were:

- Finding an open-source extendable cellular network simulator that implements the maximum number of LTE cellular network functionalities.
- No documentation is available for SimuLTE even at a very low level, like how to run the project and to deal with simple possible configuration errors.
- Understanding not only how this massive project works but how to change and develop it according to requirements for moving base stations.

A brief description of each chapter:

- In the first chapter, you will see an introduction to moving networks where the necessity of moving base stations for the future generation of cellular networks (5G) is discussed.
- Second chapter is related to our investigation on finding the best cellular network simulator among several possible options discussing their pros and cons.
- Third chapter is the place where the result of chapter 2 is explained in detail. All network elements which are already available in SimuLTE are described. The most crucial network element for development is eNodeB because in

SimuLTE, the nature of eNodeB is designed to be stationary, and we want to make it moving.

- Moving base station simulator is demonstrated in chapter 4, where you can find many details about implementation tricks and difficulties.
- In order to show that the simulator came out from chapter 4, is working perfectly fine, we decided to devise a set of experiments with results and put them in chapter 5.

To make explanations more sensible, we put some algorithms and codes inside chapter 4. Different colors are used in code presentations: light blue background codes are the ones that are completely written within this thesis, and gray background codes are the ones that are in the original version of SimuLTE, but they are manipulated for this thesis.

In the appendix, most of the codes written in this thesis are available.

Acknowledgements

First and foremost, praises and thanks to God, the Almighty, for His blessings throughout my thesis work to complete it successfully. I would like to express my deep and sincere gratitude to my research supervisors:

Professor Gianluca Rizzo at University of Applied Sciences Western Switzerland, Professor Marco Ajmone Marsan at Politecnico di Torino, Italy, for the continuous support of my master thesis study and research, for their patience, motivation, enthusiasm, and immense knowledge without which this work was impossible for me.

Special thanks to my wife Mina and my friends Marco Malinverno, Cyril Dinesh, Gaetano Manzo, Marie-Bernadette Rey, who supported me technically and emotionally, and Antonio Viridis, one of the creators of SimuLTE, who answered all my questions.

Great appreciation goes to my father, Mohammad-Ali, and my mother, Mitra, who guided me to be a good son.

Table of Contents

List of Tables	VIII
List of Figures	IX
1 Introduction To Moving Base Stations	1
1.1 Introduction	2
1.2 Why Moving Base Stations?	2
1.3 Moving Base Stations	3
1.4 Issues And Challenges	5
1.5 The goal of this work	6
2 Extendable Simulator?	7
2.1 What do we need?	8
2.2 A test-bed for network simulation	8
2.2.1 JSim	8
2.2.2 ns-3	9
2.2.3 OMNeT++	9
2.2.4 Comparison table	10
2.3 A simulator for road traffic	10
2.3.1 MATSim	11
2.3.2 Treiber’s Microsimulation	12
2.3.3 SUMO	12
2.3.4 Comparison table	13
2.4 An extensible cellular network simulator project on OMNeT++ . .	13
2.4.1 VeinsLTE	13
2.4.2 SimuLTE-Veins	14

2.4.3	Comparison table	14
3	SimuLTE-Veins	16
3.1	SimuLTE	17
3.1.1	Veins and SUMO	18
3.1.2	INET	19
3.1.3	SimuLTE integration on OMNeT++	19
3.2	Network Elements	22
3.2.1	Zone settings	22
3.2.2	Server	24
3.2.3	Router	24
3.2.4	Packet Gateway (PGW)	25
3.2.5	EnodeB	25
3.2.6	User Equipment (UE)	26
3.2.7	Cars	28
4	Developing ENodeB for Moving Base Stations (Moving Small Cells)	31
4.1	Approaches and challenge	32
4.2	Moving eNodeB	34
4.2.1	Mobility models	35
4.2.2	BonnMotionMobility utilization	36
4.3	Increase eNodeB population	39
4.4	Appear and disappear eNodeB	39
4.4.1	Appear eNodeB	40
4.4.2	Disppear eNodeB	41
4.5	Macro and micro eNodeBs	41
4.5.1	eNodeB height	42
4.6	Statistic collection	43
4.6.1	Physical layer throughput and delay	44
4.6.2	Uplink SINR	45
4.6.3	Per User number of Handovers	46
5	Experiments and results	47
5.1	Funtional experiments	48

5.1.1	EnodeB movement experiment	48
5.1.2	eNodeB On and off experiment	53
5.2	Performance test experiment	54
5.3	Conclusion	62
A		63
Bibliography		93

List of Tables

2.1	Comparison of Simulators	10
2.2	Comparison of road traffic simulators	13
2.3	Comparison between 2 cellular network simulators	15
4.1	statistics measured by SimuLTE have ✓. For handovers uplink/- downlink doesn't make sense but for coherency in table format we mentioned both	43
5.1	Experiment specifications	48
5.2	Experiment specifications	51
5.3	Experiment specifications	53
5.4	Experiment specifications	57
5.5	Experiment Results, DL: downlink, avg: average, CI: Confidence Interval, Throughput (Bytes/s) is measured at physical layer , black dot : macro eNodeB position, red dots: micro eNodeB positions . .	58
5.6	Experiment specifications	60
5.7	Experiment Results, DL: downlink, avg: average, CI: Confidence Interval, Throughput is measured Bytes/s	60
5.8	Experiment Results, DL: downlink, avg: average, CI: Confidence Interval, Throughput is measured Bytes/s	61

List of Figures

1.1	Introduction to moving base stations	1
1.2	the twice mobile networking concept[1]	5
2.1	Some network simulators	7
3.1	SimuLTE-Veins	16
3.2	Veins architecture taken from Veins official website	19
3.3	Installing Files	20
3.4	Existing Project into Workspace	21
3.5	Search for nested projects	21
3.6	lte dependencies	21
3.7	Veins-inet dependencies	21
3.8	SimuLTE Cars	22
3.9	SimuLTE-Veins Ready to work	22
3.10	$1432 \times 1432m^2$ yellow zone with 7×7 blocks grid	23
3.11	Server on play ground	24
3.12	Server, Router, Packet Gateway	26
3.13	UE and eNodeB architectures	27
3.14	NIC architecture	27
3.15	Scenario instance	27
3.16	7×7 grid street in SUMO	28
3.17	Zoomed junction	28
3.18	Cars in SUMO	30
3.19	Cars in SimuLTE	30
4.1	Moving Base Station	31

4.2	Communication schematic	37
4.3	Car position trajectory with time suitable for BonnMotionMobility	38
5.1	Experiments and results	47
5.2	initial position of UE	48
5.3	A position close to eNodeB	48
5.4	last position far from eNodeB	48
5.5	SINR vs time for Moving UE	49
5.6	Moving UE distance from stationary eNodeB vs time	49
5.7	SINR vs Distance	50
5.8	SINR vs Distance (zoomed)	50
5.9	initial position of Moving eNodeB	51
5.10	A position close to fixed UE	51
5.11	last position far from fixed UE	51
5.12	SINR vs time for moving eNodeB	51
5.13	UE distance from moving eNodeB vs time	52
5.14	SINR vs Distance	52
5.15	SINR vs Distance (zoomed)	52
5.16	Moving eNodeB is entering to playground	53
5.17	moving eNodeB position closest to fixed UE	53
5.18	Moving eNodeB is about to leave the playground	53
5.19	SINR vs time for moving eNodeB	54
5.20	UE distance from moving eNodeB vs time	55
5.21	SINR vs Distance	55
5.22	SINR vs Distance (zoomed)	56
5.23	one of five figures of this setup	57
5.24	one of five figures of this setup	59

Chapter 1

Introduction To Moving Base Stations

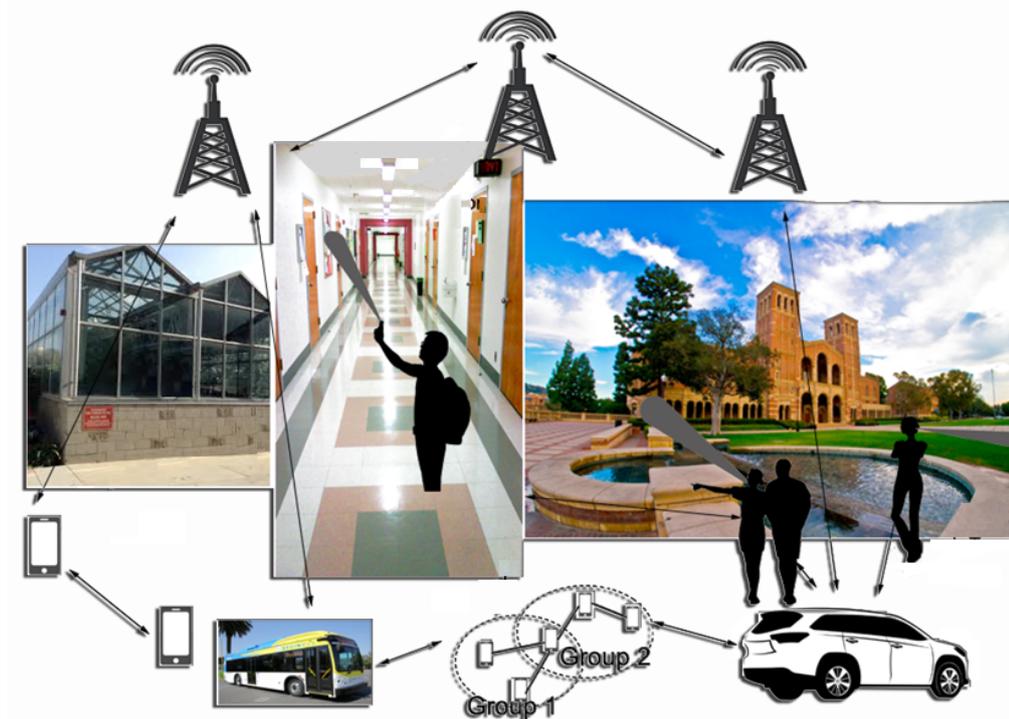


Figure 1.1: Introduction to moving base stations

1.1 Introduction

The next-generation of mobile networks called 5G is essential not only because of its enormous capacity but also because of a high potential revolution that it brings to human life. It provides higher accessibility for the user to the internet and makes the connection of billions of devices to each other possible. With 5G, home and industry automation is entirely possible, and health and security levels increase rapidly. There are tons of benefits for 5G; we do not count since it is not hidden for anyone. As the importance of implementation of the fifth generation of mobile networks (5G) comes to focus, issues and essential technologies to overcome the problems will be highlighted. Here we count a couple of issues to achieve 5G and explain proposed solutions on these issues.

5G should be able to transfer a tremendous amount of information in a very short time slot, which requires very high frequency in either uplink or downlink. As a solution, millimeter waves with the capacity of from 30 GHz to 300 GHz is proposed. As 5G technology grows, subscribers expect to have ubiquitous 5G antennas everywhere and backward compatibility with 4G matters more. To address this issue, small cells are suggested, which are portable low-powered base stations that all together can form a dense and adaptable infrastructure. This technique is also well-known as densification, and this thesis deals with this technique. Another vital concern is efficiency and leveraging speed. To address this issue, MIMO (multiple-input-multiple-output) comes to play, which allows both transmitter and receiver to have many antennas working simultaneously. There is an interference potential in MIMO that makes the use of beam-forming necessary.

In this thesis, among the three mentioned issues and corresponding solutions, we worked on deploying small cells. In the following sections of this chapter, you will see what we mean by moving base stations and why they are essential, and finally, we count some issues and challenges we face throughout this work.

1.2 Why Moving Base Stations?

In today's world, by the dramatic increase of mobile data traffic, deployment of more cells in places where user equipment (UEs) presence is high is necessary. In environments such as stadiums, universities, and shopping centers where user traffic density is high, network densification is vital to provide a good quality of service to each UE. The latter implies increasing the base stations (BSs), which burdens more costs on mobile network operators (MNOs). Moreover, when there is no event taking place in a stadium or the hours that shopping centers and universities are closed, additional base stations will be useless. Even in daily life, users commute

to work in the morning of working days, making business areas full of people during business hours. In this duration, MNOs require more capacity provided by dense small cell (SC) radio access network (RAN) deployments to have coverage over working areas. After working hours, users get out of their business place, so that the RAN capacity required in an industrial area drastically falls, and many of the mounted SCs become unnecessary. The opposite happens in residential quarters, where capacity is more demanding at night, typically not in business hours. Furthermore, RAN densification in different areas may be needed when traffic jams appear during travel to and from work [1].

These days we see network traffic with the mobility of users jointly has a sharp increase; for instance, in concert halls, Instagram or YouTube live video traffic requires additional network capacity. These kinds of circumstances force MNOs to cost on the dense deployment of SCs while at the end of events like concerts or sports matches, network traffic will have a drastic downfall. By observing these situations, we realize that dense coverage relates to UEs' presence in a time-varying manner from day by day.

To make the situation better, it would be advantageous to create many SCs in working areas in the daylight duration only, and in residential quarters during the evening only. Deploying dense SC coverage in both business and residential, then turning them on and off based on demand is one possibility to achieve this goal. This way, we can bring savings in OPEX (operational expenditures), specifically in energy consumption, but this approach does not reduce the CAPEX (capital expenditures) caused by cell deployment. Another option is to carry SCs from living areas to commercial districts and back for bringing cells' capacity at the needing place. Today, MNOs already utilize truck-mounted BSs for rapid service provisioning in areas where service is more demanding [2], and several papers have proposed drones to support communications in disaster areas. These solutions are satisfactory in specific use cases, but they are not scalable, e.g., large metropolitan areas, where MNOs have to pull out and synchronize hundreds of trucks mounted with BSs or drones[1] .

1.3 Moving Base Stations

Based on the new approach proposed by [1], We choose BSs as the moving part of the infrastructure, private cars are suitable for moving in accordance with the mobile network clients. Thus, in a working center during working hours, both UEs and vehicles are present. A temporary dense SC is created if a considerable part of those cars mounted with BS. This cellular temporary dense will be recreated in residential quarters when clients get back home with their cars. Besides, situations like traffic jam areas during rush traffic hours, football matches in stadium

or concert halls will be provisioned similarly in the lack of network capacity. A vehicle-mounted BS deployment has some prominent advantages:

- Network densification will be move-able to places in which data traffic is more demanding. This implies drastically increase in efficiency. [3].
- With a tiny fraction of cars carrying SCs, the enormous capacity increase will happen. capacity gains over 100% are possible with just 1% of vehicles providing SC support as reported in[1].
- radio access network (RAN) densification is obtained very rapidly and with a very limited cost for the MNO, since SCs are carried by vehicles, and are used by the MNO with a small cell as a service (SCaaS) paradigm, with no installation cost[1].
- Interconnection of a group of BSs for providing local service can happen in areas where the cellular network infrastructure due to disasters is not operational [1].
- Delay-tolerance in the Internet of Things (IoT) can be very effectively supported in applications like smart meter reading, by down-loading data while they pass by smart objects[1].

Now we describe the concept of moving base stations, taken from Twice Mobile Network (TMN) proposed by [1]. According to Figure 1.2, either a fixed or vehicle-mounted one is connected to the backbone network via a backhaul link, which can be either wired or wireless for fixed BSs but must be wireless for Mobile Base Station (MoBSs). Millimetre-wave link is suggested for the connection between a MoBS and the stationary network to sufficiently support the capacity of the traffic between end-users and MoBSs, and also interference avoidance with the lower frequency channels[1].

Millimetre-wave link between MoBSs is also essential to allow the creation of a back-haul network when a direct link from a MoBS to the fixed network is not present because of obstacles, so neighbor MoBSs provides back-haul opportunities as well[1].

Finally, real-time management of the extremely complex Moving Network is done by orchestrator and several controllers. MoBSs must be switched on and off for better service provisioning, and to maximize the system performance and energy consumption[1].

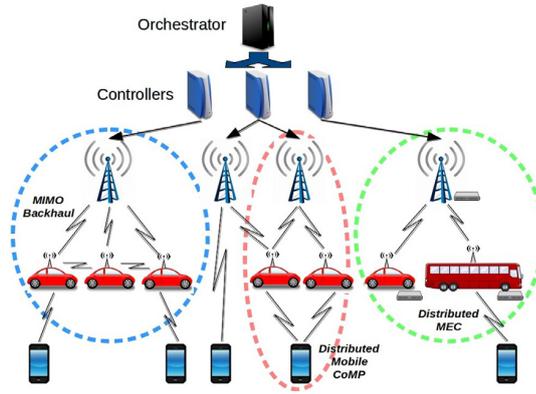


Figure 1.2: the twice mobile networking concept[1]

1.4 Issues And Challenges

There are two categories of issues and challenges we face at the beginning of this thesis:

- General issues:

By considering the difference between vehicle-based SCs and the ones carefully planned by network operators, we realize that pre-planning of the position of the base station to obtain minimum interference and maximum throughput in the case of vehicle-based SC deployment does not work anymore. Because fixed position BSs are planned to be in such a way to avoid interference and maximize the efficiency, whereas vehicle-mounted base stations are positioned according to drivers' will. This requires strongly dynamic network solution management.

Moreover, dynamic change in the position of the BS engaged with UE creates more frequent handovers, and this again makes the rapid and on-demand network management necessary. Random positioning of BS may lead to a high signal to interference and noise ratio (SINR), and some actions are needed to be done to cope with this problem. The connection of each moving BS to back-haul is also challenging. The type of physical layer technology and connectivity control due to the movement of BS is also crucial.

- Issues for our work:

- When we investigate network simulators, we do not find any simulator that can simulate moving small cell networks, so the first problem is

developing a module or existing project to meet our requirements, such as creating a set of trace-based moving base stations. For the latter as a second problem, testing different frameworks and projects, debugging and verifying them is cumbersome and time-consuming.

- we have to figure out how much existing simulators or projects can offer detailed attributes of today’s cellular network technology. For this purpose, a comprehensive investigation of available projects is required.
- Issues for development:
 - Programming language and available libraries that give us possibility to develop accurate and flexible codes and at the same time easy to run on different platforms.
 - Finding frameworks with Integrated Development Environment (IDE) that provide debugging mode for faster debug and ease of development. For example, ns-3, despite it is a fast and well-structured network simulator, it does not provide IDE and, as a consequence, no debugging mode so difficult, complicated and time consuming development process.
 - Performance parameters measurements and statistics such as Throughput, Delay, SINR, Handovers (counting entirely for all users or per each user), Channel quality indicator (CQI), etc. should be either embedded in the available framework (project) or developed.

1.5 The goal of this work

Our aim in this thesis is to develop a project on one of the existing simulation frameworks to achieve maximum possible approximated implementation of the Moving Base Stations concept. We attempt to obtain numerical results of a framework that can simulate moving small cells because we want not only to have a closer look at dynamics and variants of this system but to achieve a cellular network simulation environment with the possibility of moving small cells for future researches. For this purpose, we need to solve some of the critical issues mentioned in the previous section, such as finding a well and detailed made simulator of today’s cellular technology and developing, bringing it up to a maturity level that can properly simulate moving small cell base stations.

In the next chapter, one will see the procedure and criteria of choosing simulators fit our work, and at the end, what simulator and which project meet our requirements and attract our attention to start with.

Chapter 2

Extendable Simulator?



Figure 2.1: Some network simulators

2.1 What do we need?

In this section, we will talk about the requirements of our work and explain how we came up with the solution. Moreover, in some parts, we will talk about the pros and cons of each choice. Here we mention briefly three main requirements which should be met for our work:

- Comprehensive simulation test-bed able to simulate networking features of real-world network and it either should support road traffic simulation or can be integrated with a road traffic simulator.
- A road traffic simulator in the case in which the network simulator does not have a road traffic simulator
- An extendable cellular network simulation project on the above-mentioned test-bed

Based on what we said above, we started looking for a test-bed in which there is a possibility of having both road traffic simulation and network simulation. Since Moving Small Cells (MSC) is a quit novel idea, we did not find any simulator being able of simulating moving base stations jointly with road traffic, and because of the importance of modularity and extend-ability, we went through available open-source network test-beds such as ns-3, OMNeT++, and JSim to check the possibility of extension according to our needs. In the following, you will see that we divided our research focus in two parts: first, finding proper test-bed fits for simulation of cellular networks and second, finding a road traffic simulator with the capability of integrating with cellular network simulator.

2.2 A test-bed for network simulation

Our criteria for investigation among network test-beds were open-source, modular, extendable, discrete-event based, high processing speed, and visualization. From a processing speed perspective, we desired a test-bed implemented in C++ or Java. These programming languages are useful for high processing speed and complex calculations. so we decided to look into the three most popular test-beds based on C++ or java elaborated below.

2.2.1 JSim

JSim is an open-source Java-based simulation and animation environment supporting Web-Based Simulation for building quantitative numeric models and analyzing

them concerning experimental reference data. Its computational engine is quite general and applicable to a wide range of scientific domains[4]. The positive point of JSim is that it includes a specific platform dedicated to network simulation, the Inter-networking simulation platform called INET. The INET framework contains models for numerous wired and wireless protocols, a detailed physical layer model, application models, and more. However, the disadvantage of JSim is that it is a real-time process-driven simulator. Event executions in JSim are carried out in real-time as opposed to fixed time points in discrete event simulation. The latter implies that this simulation test-bed doesn't work in our case. Because discrete event simulators are flexible and variation of the level of detail and complexity of the simulation model is possible. The possibility to model uncertainties and the dynamic behavior of the real system exists, but for having GUI in JSim, we need to integrate a lot of codes manually.

2.2.2 ns-3

it is a discrete-event network simulator, created primarily for research and educational purposes. ns-3 is free software, built using C++ and Python with scripting capability, licensed under the GNU GPLv2 license. It is publicly available for research, development, and use[5]. ns-3 is under development by a large community of scientists and developers with heterogeneous research interests, and it already contains a considerable number of network modules (e.g., WiFi, WiMAX, 802.11s mesh networks, etc.). The latter, along with being a discrete event simulator, are prominent advantages. Moreover, having the possibility of implementing millimeter-wave as an emerging technology for wireless communication makes ns-3 unique. Despite all mentioned advantages above, The influential drawbacks of choosing ns-3 for the purpose of our work are two: first, it doesn't have a built-in integrated development environment (IDE), which is important for programmers in terms of fast editing and debugging of code. Second, it doesn't provide a built-in GUI, and this makes the error detection in network experiments difficult, especially when it is supposed to integrate with a vehicular network or road traffic simulator.

2.2.3 OMNeT++

OMNeT++ (stands for Objective Modular Network Testbed in C++) is an open-source, modular, component-based framework written in C++. It has a lot of useful C++ simulation libraries and extendable for building network simulators. Network protocols like IP or HTTP instead, the main computer network simulation models are available in several external frameworks such as the most commonly used one called INET, which offers various models for all kinds of network protocols

like for IPv6, BGP, etc. INET also offers a set of mobility models like linear, circular and random walk, etc. to simulate the node movement in simulations. The basic OMNeT++ building blocks are modules, which can be either simple or compound. Modules communicate through messages, which make discrete events nicely possible and usually sent and received through connection links. The most common approach to have the messaging process under our control is to manage them by event handlers called by the simulation kernel when modules receive a message. Besides handlers, simple modules have an "initialize" and a "finish" function to write results to a file at the end of the simulation. The latter is really useful to collect and analyze the results after experiments. The kernel includes an event queue, whose events correspond to messages (including those a module sends to itself) and regularly controls all the events throughout the simulation time. It is a stable, mature, and feature-rich framework, much more so than ns-3. It is supported by a large and community of users, from both academia and networking industries, which is a guarantee that any setup time invested in learning is a reserve over a long future. Concerning our criteria, from extendability and modularity points of view and being a well-implemented event-based test-bed, this test-bed meets our requirements.

2.2.4 Comparison table

In his table you see a summarized comparison among 3 investigated simulators.

-	Node Mobility	GUI and IDE	Extendable	Modular	Discrete event
J-Sim	✓	✗	✓	✓	✗
ns-3	✓	✗	✓	✓	✓
OMNeT++	✓	✓	✓	✓	✓

Table 2.1: Comparison of Simulators

2.3 A simulator for road traffic

There are some powerful road traffic simulators (we call them RTS), but not all are free to use. So, we just briefly introduce some popular RTSs then we explain how we made our choice among open-source ones.

1. SUMO (Simulation of Urban MObility) is an open-source, highly portable, microscopic road traffic simulation package designed to handle large road networks.

2. Quadstone Paramics (Parallel Microscopic Simulation) Modeller: it is a modular set of microscopic simulation instruments providing a strongly integrated platform for modeling a complete range of real-world traffic and transportation problems.
3. AIMSUN (Advanced Interactive Microscopic Simulator for Urban and Non-Urban Networks): a simulation package that integrates three types of transport models: static traffic assignment tools; a mesoscopic simulator; and a micro-simulator.
4. CORSIM TRAFVU servers as a traffic simulation viewer and is part of the TSIS CORSIM software package. It provides animation and static graphics of traffic networks, using the CORSIM input and output files created by a licensed user of TSIS CORSIM.
5. SimTraffic: a simulation application part of Trafficware’s Synchro Studio package. It servers as a traffic simulator for Trafficware’s Studio, including traffic lights synchronization application.
6. MATSim (Multi Agent Transport simulator): provides a framework for implementation of large-scale agent-based transport simulations. The framework includes several modules that can be combined or used stand-alone. Modules can be replaced by custom implementations to test single aspects of your own work .
7. Treiber’s Microsimulation is a personal software project created by that author and used in his research of traffic dynamics and traffic modeling .

Among all the above-mentioned RTSs, just three of them are open source: SUMO, Treiber’s Microsimulation, and MATSim. So, we just discuss main features of these three RTSs.

2.3.1 MATSim

MATSim developed by the Polytechnic of Zurich, providing a set of instruments for the implementation of a very large simulation based on agents. It is used for traffic simulation in Zurich (Switzerland), Padang (Indonesia), Berlin (Germany), and Toronto (Canada). MATSim pursues an activity-based approach to demand generation. Unlike other transportation simulation packages, MATSim is throughout agent-based and generates individual activity plans as input to the network loading rather than (time-dependent) origin-destination matrices as typically used in dynamic traffic assignment[6]. MATSim is a simulator that uses GIS (Geographic Information System). It can simulate the traffic of a vast region throughout the day. But it is not interested in detail vehicle behavior, and this is a disadvantage

from our perspective because this reduces the simulator’s modularity. The other drawback of this simulator is that it does not have the necessary scalability to simulate the entire city.

2.3.2 Treiber’s Microsimulation

This simulator has the ability to work under multiple operating systems, and it is useful for heavy runs. It illustrates the utilization of different types of vehicles used in the simulation: trucks and cars. The cars are shown with smaller rectangles in color, while the trucks are larger rectangles in another color. The difference between SUMO and this package is that SUMO is an open-source project developed by two different institutions, while Treiber’s Microsimulation is a personal software project whose source code is available[7]. Additionally, Treiber’s Microsimulator includes some statistical distribution of vehicles where the user can basically define the number of vehicles emitted per hour from a certain intersection as well as the ratio between cars and trucks in certain scenarios[7]. Two drawbacks of this simulator caused that we do not choose it for our work: first, it can not convert traffic networks from most of the other simulators while SUMO has this capability. Second and more importantly, integration with OMNeT++ has some challenges, while SUMO integration with OMNeT++ is way more comfortable.

2.3.3 SUMO

SUMO is open-source and was created and developed at the German Aerospace Center. In this simulator, vehicles can move freely, with the simulation of collision between vehicles. Each vehicle has its route, and this routing is dynamic. The vehicle behavior is considered when changing lanes happen. Roads in SUMO are shown as a population of lanes. Moreover, the width of each lane and the vehicle width is fixed. It does not take into account the many different types of vehicles. SUMO allows modeling of intermodal traffic systems, including road vehicles, public transport, and pedestrians. SUMO can be utilized with custom models and provides a variety of APIs for the remote control of the simulation [6]. The latter is the most important feature which made us able to make a real-time information exchange between OMNeT++ and SUMO.

Some other important capabilities of SUMO are the following:

1. we can manually write up of a traffic network in an XML file. OMNeT++ is compatible with XML and this makes SUMO and OMNeT++ more cooperative.
2. We can Import city road map and also networks created in other traffic

simulation applications .

3. Using an automatic network generator creating three different types of networks: Grid network, Spider network Random network [7].

2.3.4 Comparison table

In this table you see a summarized comparison among 3 investigated road traffic simulators.

-	MATSim	Treiber’s Microsimulation	SUMO
intractability with network simulators	✗	✗	✓
detailed vehicle behaviour	✗	✓	✓
comprehensive documentation	✓	✗	✓
GUI	✓	✓	✓

Table 2.2: Comparison of road traffic simulators

2.4 An extensible cellular network simulator project on OMNeT++

In this section we explain how we found our desired module fitting our work and meeting our most of requirements. By investigating throughout the available extendable project modules on OMNeT++ able to implement features of cellular networks and also vehicular networks we found two packages: VeinsLTE and SimuLTE-Veins. We elaborate pros and cons of each, then we describe how we developed the C++ codes to set up our test-bed to meet maximum possible requirements of moving base station simulator.

2.4.1 VeinsLTE

Veins is an open-source framework for vehicular network simulations. It is based on two well-established simulators: OMNeT++, an event-based network simulator, and SUMO, a road traffic simulator [8]. VeinsLTE is a simulator for heterogeneous vehicular networks. It provides fine-grained simulation of vehicular networks based on IEEE 802.11p and LTE. VeinsLTE was created by integrating three popular frameworks to build a complete simulation suite: SUMO for detailed road-traffic simulation, Veins for a fine-grained model of IEEE 802.11p and SimuLTE

for a detailed model of LTE [9]. A first integration attempt has been performed with VeinsLTE, which integrates a customized version of both simulators in a single package. Nevertheless, this solution defines a third standalone framework, rather than two interconnected simulators, taking snapshots of two independent developments. This makes upgrading difficult, if possible at all [10]. Code extension for developing the simulator according to our aims is very difficult on VeinsLTE. The latter motivated us to take a look at facilities and possibilities on SimuLTE.

2.4.2 SimuLTE-Veins

SimuLTE is an OMNeT++-based simulator for LTE and LTE-Advanced cellular networks. SimuLTE demonstrates a fully modular structure, which makes extension, verification, and integration more comfortable. Moreover, it inherits the benefits of OMNeT++ that is a widely-used and versatile simulation framework, and exploits experiment support and seamless integration with the OMNeT++ network modules and projects, such as INET. This allows SimuLTE users to build up combined scenarios where LTE can be only a part of a wider network. SimuLTE simulates the Evolved Packet Core and data plane of the LTE Radio Access Network [11]. SimuLTE allows simulation of LTE/LTEA in Frequency Division Duplexing (FDD) mode, with heterogeneous eNBs (macro, micro, pico, etc.), using omnidirectional and anisotropic antennas, possibly communicating via the X2 interface. Realistic channel models, MAC, and resource scheduling in both directions are supported. In the current release, the Radio Resource Control (RRC) is not modeled [11]. For an exhaustive explanation of the detailed implementation of SimuLTE, we invite you to take a look at [11].

As it is explained in the above paragraph, we observe that SimuLTE is not only well designed for the detailed implementation of LTE/LTE-A, but it is modular and extendable and by inheriting features of IP and TCP/UDP layer from INET gives us the possibility of implementation of any cellular network scenario. With that said and nice integration of it with Veins and SUMO made us want to choose SimuLTE-Veins as our framework for implementation of MSC.

2.4.3 Comparison table

In his table, you see a summarized comparison between two SimuLTE projects combined with Veins in two different approaches.

-	VeinsLTE	SimuLTE-Veins
Up-gradable	✗	✓
Extendable	✗	✓
Veins intractable	✓	✓
SUMO intractable	✓	✓
INET intractable	✓	✓

Table 2.3: Comparison between 2 cellular network simulators

In the next chapter, you will get familiar with the structure and features of SimuLTE and its integration with Veins.

Chapter 3

SimuLTE-Veins

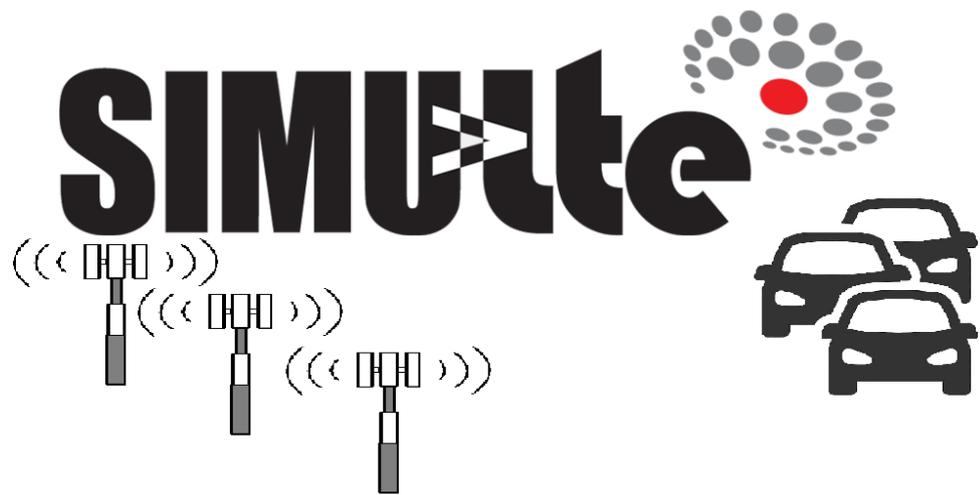


Figure 3.1: SimuLTE-Veins

This chapter is an elaboration of what SimuLTE is and how it is structured. For more information about OMNeT++, which is a testbed on which SimuLTE is run, please visit <http://www.omnetpp.org>.

3.1 SimuLTE

SimuLTE is a well-made project that simulates Long-Term Evolution (LTE) of the UMTS(3GPP-TS 36.300), which is a standard for cellular access networks. SimuLTE simulates the data plane of the LTE/LTE-A Radio Access Network and Evolved Packet Core. SimuLTE allows simulation of LTE/LTE-A in Frequency Division Duplexing (FDD) mode, with heterogeneous eNodeBs (macro, micro, pico, etc.), using omnidirectional and anisotropic antennas, possibly communicating via the X2 interface. Realistic channel models, MAC, and resource scheduling in both directions are supported. SimuLTE implements eNodeBs and UEs as compound modules. These modules can be connected with each other and with other nodes (e.g., routers, applications, etc.) for composed network creation. The Binder module is instead visible by every other node in the system and stores information about them, such as references to nodes. For instance, it is used to locate the interfering eNodeBs to compute the inter-cell interference perceived by a UE in its serving cell. UE and eNodeB are further composed of modules. Every module has an associated description file (.ned) defining its structure and may have a class definition file (.cpp, .h) which implements the module functionalities.

The UDP and TCP modules, taken from the INET package, implement the respective transport layer protocols and connect the LTE stack to TCP/UDP applications. TCP and UDP applications (TCP App and UDP App) are implemented as vectors of N modules, thus enabling multiple applications per UE. Each TCP/UDP App represents one end of a connection, the other end of which may be located within another UE or anywhere else in the topology. SimuLTE comes with models of real-life applications (e.g., VoIP and Video on Demand), but any other TCP/UDP-based OMNeT++ application can also be used. The IP module is taken from the INET package as well. In the UE, it connects the Network Interface Card (NIC) to applications that use TCP or UDP. In the eNodeB, it connects the eNodeB itself to other IP peers (e.g., a web server), via PPP (Point-To-Point Protocol). The NIC module, whose structure is shown in Figure 3.14, implements the LTE stack. It has two connections: one between the UE and the eNodeB and one with the LTE IP module. It is built as an extension of the IWirelessNic interface defined in the INET library, so as to be easily plugged into standard scenarios. This allows one – among other things, to build hybrid connectivity scenarios, e.g., with nodes equipped with both Wi-Fi and LTE interfaces. Each of the NIC sub-modules on Figure 3.14, represents one or more parts of the LTE protocol stack, which is

common to the eNodeB as well. The only module in the UE that has no counterpart in the eNodeB is the Feedback Generator, which creates channel feedbacks that are then managed by the PHY module. The communication between modules takes place only via message exchange; thus, each action starts from a message handler. Cross module calls are used only in the form of getter/setter functions. This allows us to maintain strong control over the interactions between modules, thus limiting possible buggy behaviors.

3.1.1 Veins and SUMO

What is Veins? Veins is an open-source vehicular network simulation framework, works as a set of simulation models for vehicular networks. An event-based network simulator (OMNeT++) executes these models while interacting with SUMO, which is a road traffic simulator. Other components of Veins are in charge of setting up, running, and monitoring the simulation. This constitutes a simulation framework. Veins is built to serve as a framework for writing application-specific simulation codes. While it can be used, with only a few parameters tweaked for a specific use case without modifications, it is also designed to serve as an execution environment for user codes. Typically, these user-written codes will be an application that is to be evaluated by means of simulation itself. The framework cares about the rest: node mobility and modeling lower protocol layers, taking care of the simulation setup and its proper execution, and collecting statistic results during and after the simulation execution.

Veins contains an enormous number of simulation models that can be applied to vehicular network simulation generally. Not all of them are required for every single simulation, in fact, for some of them, it only makes sense to instantiate one in any given simulation execution. The Veins' simulation models serve as a box of tools: much of what is needed to build up a comprehensive, highly detailed simulation of a vehicular network are already accommodated in Veins. Nevertheless, a researcher assembling a simulation is expected to know which of the available models to apply for which task.

Veins is an Open Source simulation framework for vehicular networks. This means that it and all of its simulation models are freely available for download, study, and use. Nothing about its operations are kept secret. Any simulation built and executed with Veins can be shared with interested colleagues not only the results but the complete toolchain required for an interested colleague to reproduce the same results, verify how they were derived, and build upon the research performed.

As mentioned before, with Veins, each simulation is performed by executing two simulators in parallel via a TCP connection: OMNeT++ (for network simulation)

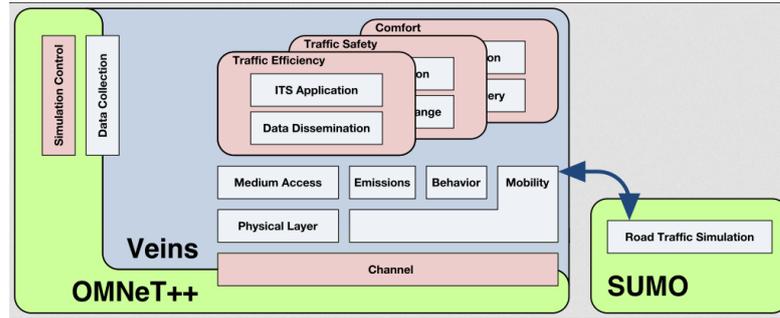


Figure 3.2: Veins architecture taken from Veins official website

and SUMO (for road traffic simulation). A Traffic Control Interface (TraCI) protocol has been standardized for this communication, allowing bidirectionally-coupling simulation of network traffic and road traffic. Vehicles' movement in the road traffic simulator SUMO is reflected as nodes' movement in OMNeT++ simulation. Nodes can interact with the running road traffic simulator.

3.1.2 INET

INET Framework is an open-source model C++ library for the OMNeT++ simulation testbed. It provides protocols and models for researchers and students interested in working with both wired and wireless communication networks. INET is particularly useful when designing and validating new protocols or exploring new or exotic scenarios. INET contains models for the Internet stack such as IPv4, IPv6, TCP, UDP, OSPF, BGP, etc., wired and wireless link layer protocols such as Ethernet, 802.11, PPP, etc., support for mobility, MANET protocols, DiffServ, MPLS with LDP and RSVP-TE signaling, several application models, and many other protocols. Several other simulation frameworks take INET as a base and extend it into specific desired directions, such as overlay/peer-to-peer networks, vehicular networks, or even LTE.

3.1.3 SimuLTE integration on OMNeT++

In order to utilize simuLTE, we need to import it as a project into OMNeT++ jointly with INET and Veins. In this section, we talk about how we did it. Since it is a little challenging, we preferred to mention it in order for newbies not to be in trouble with the installation and importing process. If you are not interested in this part, you can skip to 3.2.

The first release of SimuLTE was working only on Ubuntu 16.04 and OMNeT++ 5.1.1, which mainly we worked on it but supporters of SimuLTE released a new

version of that which is working on Ubuntu 18.04 and OMNeT++ 5.5.1, and we upgrade our work according to the new version, and after some days of challenge finally it is perfectly working. Now all that you see here is based on Ubuntu 18.04, so make sure that your operating system is Ubuntu 18.04 and you have installed OMNeT++ 5.5.1 and make sure that you unchecked the INET installation in the emerging window when we run OMNeT++ because it tries to install the last version on INET which is not desirable for us. First of all create a folder and put these three items with exactly mentioned versions inside:

- INET 3.6.6
- SimuLTE 1.1.0
- Veins 4.6

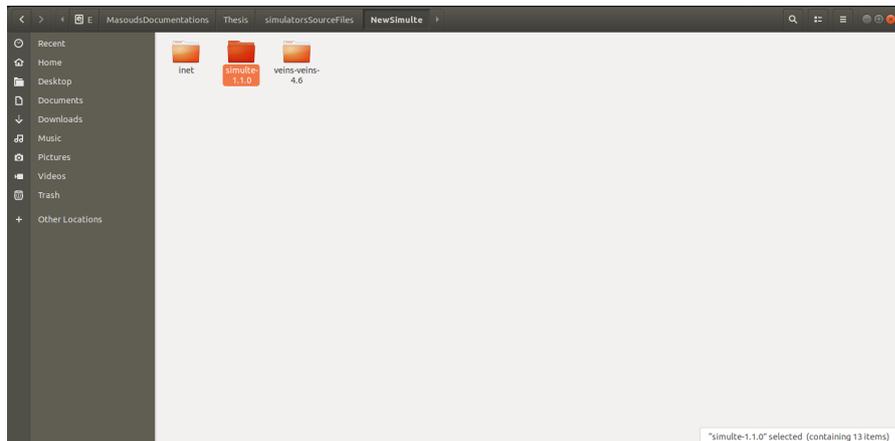


Figure 3.3: Installing Files

Secondly, make an empty folder beside this for workspace, call it whatever you like, and change your new workspace's path. From file/Import, you will see a window shown below, then you select "Existing Project into Workspace," as shown in Figure 3.4 and click next. In the new window, browse for the path in which you have stored the whole 3 installing files, and according to Figure 3.5, make sure you have checked "Search for nested projects." Once you performed all previous steps, ensure that each project's dependencies are given correctly by right click on SimuLTE (LTE) and Veins-INET project folders respectively and choose "properties" according to Figures 3.6, 3.7, and 3.8. Finally, run "Build All" from the "project" menu. After finishing the build process, which takes several minutes, the whole project is ready to use or develop, Figure 3.9.

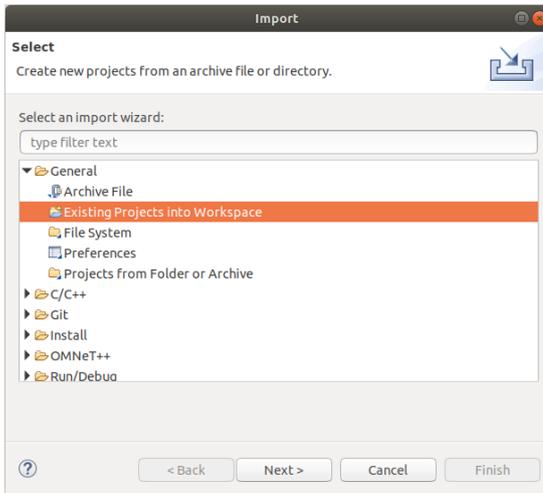


Figure 3.4: Existing Project into Workspace

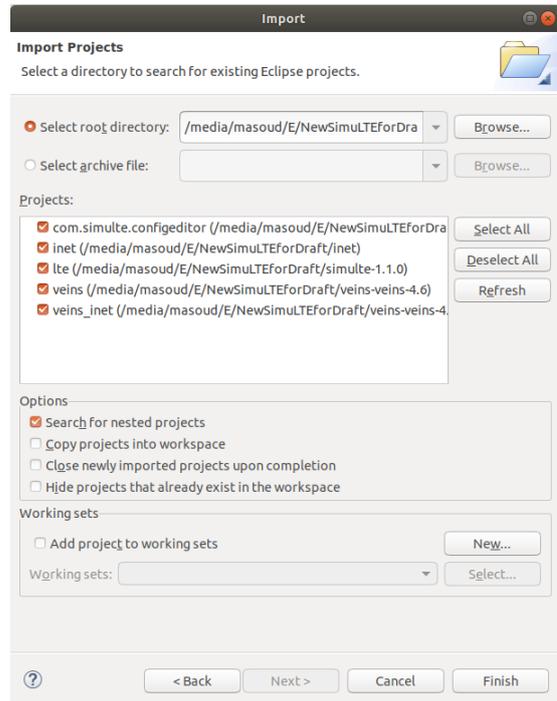


Figure 3.5: Search for nested projects

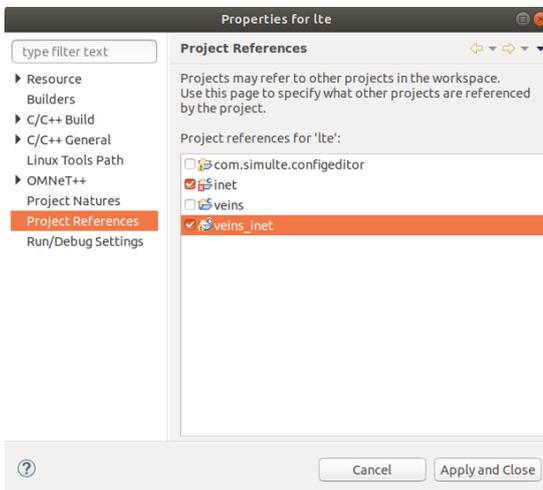


Figure 3.6: lte dependencies

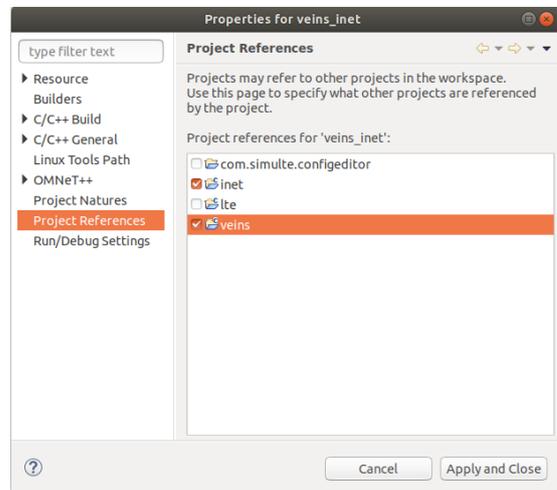


Figure 3.7: Veins-inet dependencies

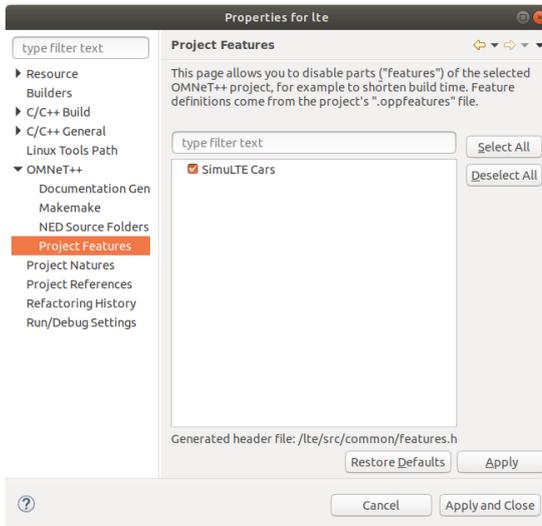


Figure 3.8: SimuLTE Cars

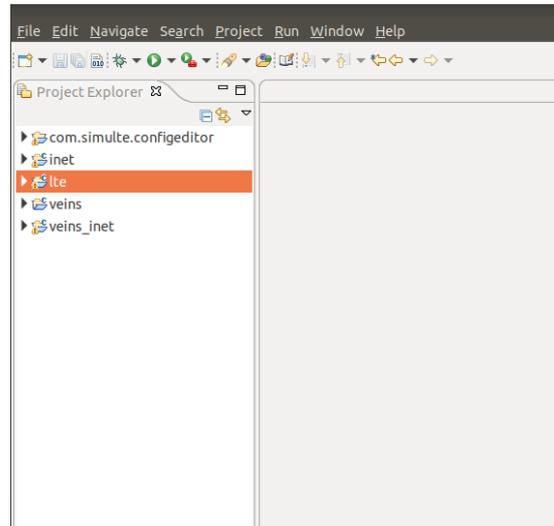


Figure 3.9: SimuLTE-Veins Ready to work

3.2 Network Elements

In this section, we will see what settings and ingredients required to build any LTE network simulation with SimuLTE and how these ingredients can be added to our experiment setup.

3.2.1 Zone settings

At the beginning of every simulation, we have to define zone characteristics on which we want to mount our network elements. In order to define all physical features of our network and its elements, we have to be familiar with the NED language. This language is well made for this purpose by OMNeT++ developers. To learn about syntax and semantic of NED, you have to visit chapter 3 of the OMNeT++ 5.6.1 manual.

For every zone we in OMNeT++, we may want to specify the length(X), width(Y), and height(Z) under the parameter keyword of our network definition file. In the following example, you see NED code for a square zone with $X=1432$ meters, $Y=1432$ meters (because we want to have 7×7 blocks, each block 200 meters + 8 streets 4 meters wide each both vertically and horizontally in SUMO Figure 3.10 shows only in OMNeT++), and $Z=50$ meters:

```

1  double playgroundSizeX @unit(m) = 1432m; // x size of the area
   the nodes are in (in meters)
2  double playgroundSizeY @unit(m) = 1432m; // y size of the area
   the nodes are in (in meters)
3  double playgroundSizeZ @unit(m) = 50m; // z size of the area
   the nodes are in (in meters)

```

Listing 3.1: play ground NED code

There is a command named @display by which you can define some visual attributes of your zone:

```

1  @display("bgb=1432,1432,#FCE94F,#729FCF;bgg=1432,7,black");

```

Listing 3.2: play ground appearance code

Inside the @display command, every inner command is separated by a semicolon, and within a command, each attribute is separated by a comma. The above example with bgb, you specify the zone, and the first two elements describe the scale, and the third and fourth elements describe the color of the zone and its background, respectively. Then we have a second command bgg that puts a grid of 7 blocks on our zone. Finally, with the above attributes, our "Manhattan's quasi-streets" zone is Figure 3.10.

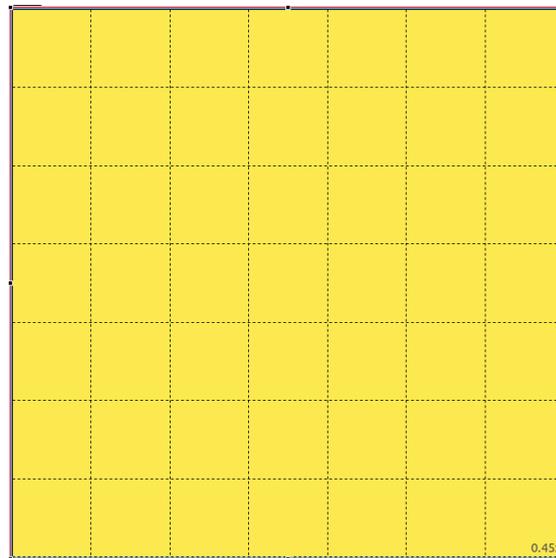


Figure 3.10: $1432 \times 1432 m^2$ yellow zone with 7×7 blocks grid

3.2.2 Server

We need communication at the application level in both uplink and downlink to have near to realistic experiments. One of the best network devices for this purpose is the server. To bring the server to the scenario, we need to write a piece of code according to listing 3.3.

```

1  server: StandardHost
2  {
3      @display("p=1291.8359,120.83499;is=v1;i=device/server");
4  }

```

Listing 3.3: server NED code

The server is a type of StandardHost defined in INET and can support application and transport layer (TCP/UDP protocols). In the appendix, you will find the StandardHost NED code. Once you add server code, you can see the presence of the server in scenario figure 3.10.

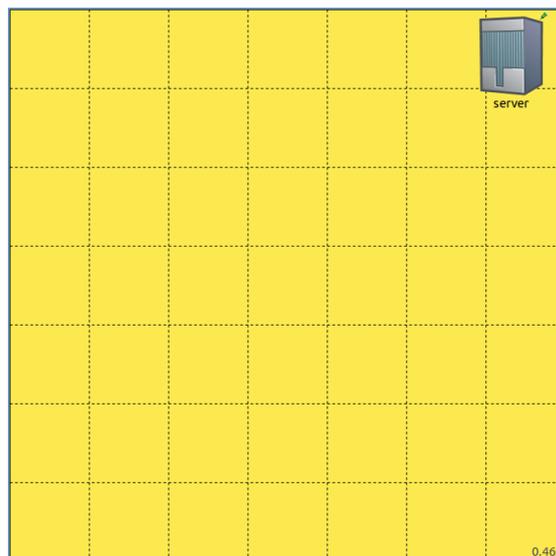


Figure 3.11: Server on play ground

3.2.3 Router

One of the fundamental network devices in realistic scenarios is router. In SimuLTE router inherits its attributes from the router defined in INET, and it has many capabilities in terms of network protocols such as OSPF, RIP, BGP, etc. It is an IPv4 router that supports wireless, Ethernet, PPP, and external interfaces. The

number of Ethernet and PPP ports are dependant on the external connections. The NED code we used to have this router in our scenario is mentioned in Listing 3.4.

3.2.4 Packet Gateway (PGW)

SimuLTE simulates the Evolved Packet Core (EPC) and data plane of the LTE Radio Access Network. Unlike 2G and 3G network architectures that switching voice and data happens through two separate sub-domains: circuit-switched for voice and packet-switched for data, Evolved Packet Core unifies voice and data treating both as Internet Protocol (IP) service. One of the key devices in EPC is Packet Data Node Gateway (PGW), which works as an interface between the LTE network (mostly eNodeB) and other packet switching networks to manage the quality of service (QoS), and performs packet inspection. SimuLTE also supports GPRS Tunneling Protocol (GTP) which is an IP/UDP based protocol used in LTE core networks. GTP encapsulates user data when passing through the core network and carries specific signaling traffic between various core network elements. The corresponding NED code for adding PGW into the scenario is mentioned in Listing 3.4, and Figure 3.12 shows the results of all the NED codes we provided up to now.

```

1 router: Router
2 {
3     @display("p=916.1489,114.243996;is=v1;i=device/smallrouter");
4 }
5 pgw: PgwStandardSimplified
6 {
7     nodeType = "PGW";
8     @display("p=555.84094,114.243996;is=v1");
9 }

```

Listing 3.4: Router and Pacet gateway NED definition

3.2.5 EnodeB

The most sophisticated and well-made LTE/LTE-A element in SimuLTE is eNodeB. It simulates a variety of LTE functionalities, which we mention some of them throughout this subsection. Once each User Equipment (UE), is bounded to an eNodeB, it allocates radio resources to UE, using Single-Carrier Frequency Division Multiplexing (SC-FDMA) in uplink and Orthogonal Frequency Division Multiplexing Access (OFDMA) in the downlink. This allocation is done in both directions as a time/frequency frame of resource blocks (RBs) On each Transmission Time Interval (TTI, 1ms). Depending on the modulation and coding scheme, each RB carries a variable amount of bytes to/from a UE. This means SimuLTE supports

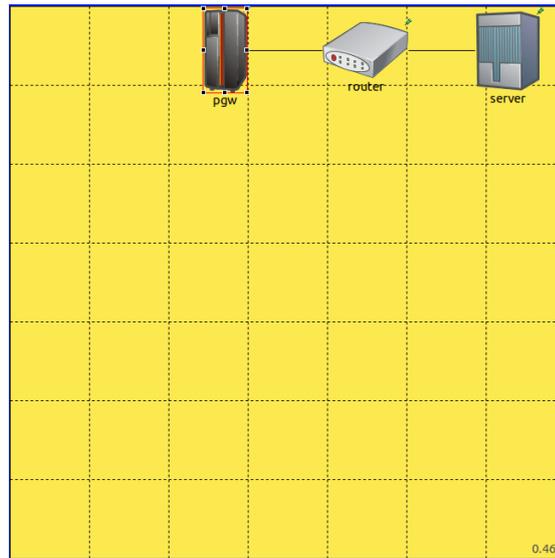


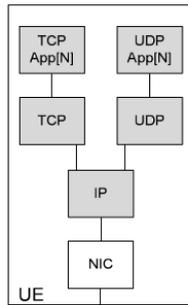
Figure 3.12: Server, Router, Packet Gateway

adaptive modulation coding. The eNodeB chooses the proper coding scheme (QPSK, 16QAM, 64QAM) based on the Channel Quality Indicator (CQI) reported by each UE. CQI measures how each UE perceives the channel. Handovers are supported very well according to CQI from eNodeB and UE point of views. ENodeB in SimuLTE supports two important transmission schemes, such as Coordinated Multi-Point (CoMP) Scheduling and Device-to-Device (D2D) communications.

The architecture of eNodeB in SimuLTE is depicted in Figure 3.13, and the NED code is shown in Listing 3.5. There are some other features which are not supported by eNodeB in SimuLTE such as movement and distinguishing between macro and micro eNodeBs based on transmission power, attenuation computing for different eNodeB antenna heights, etc.. These features and some others are developed in this thesis and will be discussed in chapter 4.

3.2.6 User Equipment (UE)

Figure 3.13 shows how UE as a compound module in OMNeT++ is structured. The Network Interface Card (NIC) is connected to the TCP/UDP layer through an IP module taken from the INET package. SimuLTE has implemented NIC in good detail according to the real LTE data link layer, Figure 3.14. An exemplary scenario is shown in Figure 3.15, which depicts an eNodeB in the middle of Manhattan streets, and 10 UEs are scattered around. The NED code for adding eNodeB and UE is written in Listing 3.5. Please note that only 3 UEs are written for sample generation of UEs. For generating more UEs, you should follow the same code



Binder

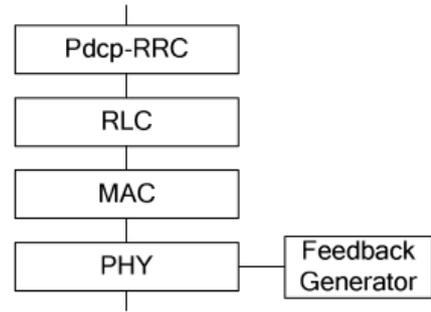
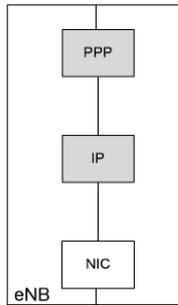


Figure 3.13: UE and eNodeB architectures

Figure 3.14: NIC architecture

pattern or use for loop and use array of UEs.

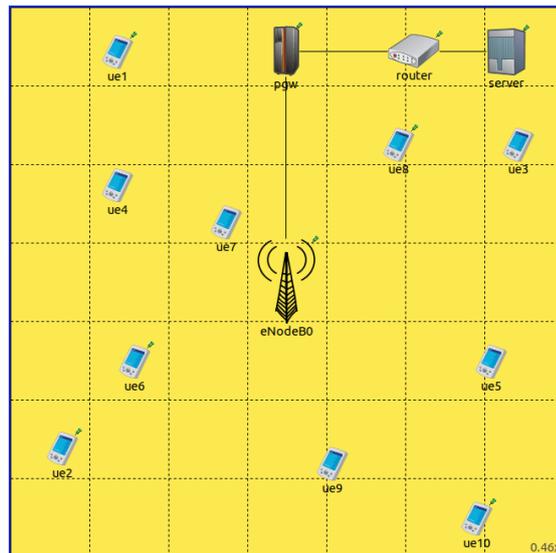


Figure 3.15: Scenario instance

```

1 eNodeB0: eNodeB {
2   @display("p=716,716;is=v1");
3 }
4 ue1: Ue {
5   @display("p=276.822,116.440994");
6 }
7 ue2: Ue {
8   @display("p=134.017,1149.0309");
9 }

```

```

10 ue3: Ue {
11   @display("p=1309.412,344.929");
12 }

```

Listing 3.5: EnodeB and UEs NED definition

3.2.7 Cars

Cars in SimuLTE act as a moving UE, and in terms of internal architecture, they are the same as UEs. For car traffic models and trajectories, we need a car traffic simulator, which is SUMO. Once we set up our car traffic scenario with SUMO, we need to send its information, including car positions, speed, etc. at each time instance to SimuLTE. This connection management is done through Veins, which is a vehicular network simulator.

SUMO (Cars and streets)

Since in the previous chapter, we have talked about SUMO, here we only go a little into detail of how to apply SUMO commands for our purposes. Please consider that the only version of SUMO that SimuLTE works with is 0.30.0. After installation of SUMO 0.30.0 to create a grid similar to Manhattan streets with 7×7 blocks, each block 200 meters long and create the corresponding ".net.xml" file, we need to insert the Listing 3.6 command in terminal (Linux).

```

1 netgenerate --grid --grid.number=7 --grid.length=200 --output-file
  =MySUMOFile.net.xml

```

Listing 3.6: SUMO command for grid of street generation

The result of Listing 3.6 command is shown in Figure 3.16 and Figure 3.17. Now

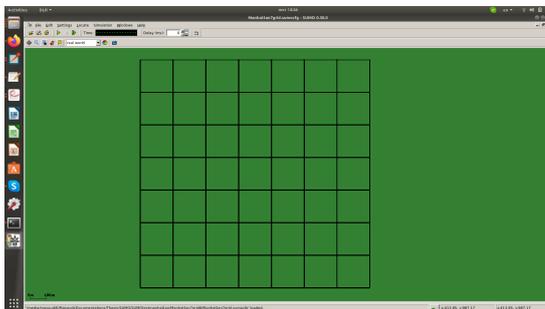


Figure 3.16: 7×7 grid street in SUMO

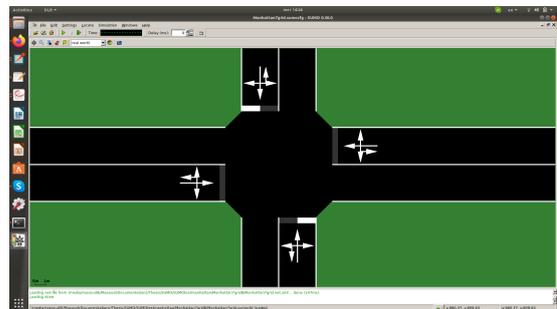


Figure 3.17: Zoomed junction

we need to insert cars into these streets and assign a random trip to each of them. For this purpose, we go to the directory in which SUMO is installed then use the command in Listing 3.7. please pay attention that after -n, you should mention the name network (streets) XML file.

```
1 /media/E/sumo-0.30.0/tools/randomTrips.py -n MySUMOFile.net.xml -
  e 50
```

Listing 3.7: SUMO command for 50 car random trip generation

the above command will generate a name.trip.xml file, but In order to be able to run our scenario in SUMO, we need two other XML files. The first one is the configuration file, and the second in the launch file (name.sumocfg and name.launchd.xml). Please note that in the configuration file, you must insert the name.net.xml file as net parameter and name.trip.xml as rout parameter, and in the launch file, you must insert these two plus the name of the configuration file. In the following, we see the contents of these two files. There are also possibilities to upload real-world street maps in SUMO that we do not mention it here.

```
1 <configuration>
2   <input>
3     <net-file value="MySUMOFile.net.xml"/>
4     <route-files value="MySUMOFile.trips.xml"/>
5   </input>
6   <time>
7     <begin value="0"/>
8     <end value="500"/>
9     <step-length value="0.1"/>
10  </time>
11  <gui_only>
12    <gui-settings-file value="gui-settings.xml"/>
13  </gui_only>
14 </configuration>
```

Listing 3.8: SUMO configuration xml file

```
1 <launch>
2   <copy file="MySUMOFile.net.xml"/>
3   <copy file="MySUMOFile.trips.xml"/>
4   <copy file="MySUMOFile.sumocfg" type="config"/>
5   <copy file="gui-settings.xml"/>
6 </launch>
```

Listing 3.9: SUMO launchd.xml file

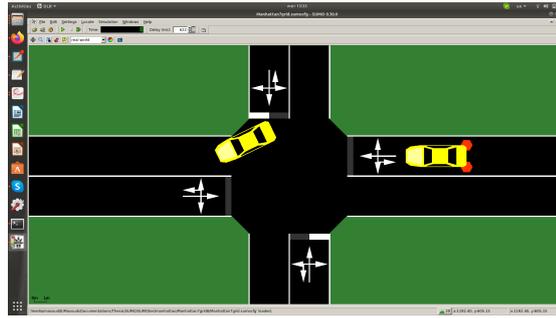


Figure 3.18: Cars in SUMO

Veins

In order to make SUMO collaborative with SimuLTE, Veins comes to play. To do so, we need to insert a NED code into our network.ned file (in addition to what was said in 3.1.3). This code allows us to use the capabilities of Veins and INET in SimuLTE. For more detail, you can find this piece of code in final developed release in my GitHub: <https://github.com/Masoudsultanian/MovingBaseStations.git>.

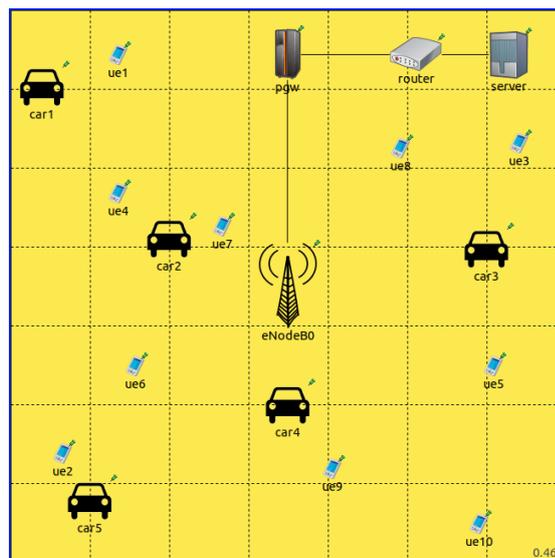


Figure 3.19: Cars in SimuLTE

In the next chapter, you will see the core of the thesis, where we put our hands into code to tackle problems and obstacles on our way of creating Moving Base Stations. During this way, we also added extra statistic measurement capabilities that were not implemented in the SimuLTE original code.

Chapter 4

Developing ENodeB for Moving Base Stations (Moving Small Cells)



Figure 4.1: Moving Base Station

In this chapter, we talk about methods and algorithms we devised to develop eNodeB in SimuLTE for movement support and adding new features to meet Moving Base Stations implementation requirements. You will also see the problems, challenges, and constraints we were facing throughout this journey. Each challenge is introduced in each section. There were also many tests and analyses that did not lead to solutions for challenges so that we avoid mentioning them here because only the analyses led to the solution might be interesting for readers. Please note that codes or algorithms with light blue backgrounds are completely devised in this thesis.

4.1 Approaches and challenge

When we take a look at many features provided in SimuLTE for eNodeB, we figure out that it does not support the movement of eNodeB, and here is the point we had to stop search and start to work on SimuLTE. For this purpose, we considered two methods and evaluated the pros and cons of each.

- Snapshots

In this method, after setting up an exemplary scenario, we had to run the simulation for a short duration. For instance, 100 milliseconds then stop the simulator, gather statistics, update the base stations' position manually, and then resume the simulation run for another 100 milliseconds and collect statistics and continuously repeat this process until reaching desirable total simulation duration. Now we briefly point out some advantages and disadvantages to this method.

- Even though this approach is not complicated in terms of implementation but cumbersome and time-consuming.
- It does not need to get deeply into code or even development of a component, so regardless of time and effort by having only SimuLTE, every cellular communications scenario can be implemented without intervening to code.
- To account for handovers, we need to correlate vehicles' positions from one snapshot to another, which is burdensome.
- computation of SINR is less complex but less realistic in comparison with contiguously moving base stations.

- Simulate moving by animating eNodeB

we imagined a grid of streets being one eNodeB at the center of each square, so many eNodeBs placed in a nice order vertically and horizontally, and by

turning the eNodeB ON and OFF sequentially one after another, we would have a moving eNodeB.

- This approach is less time-consuming than the previous one, but since we increase the number of base stations compared with the previous scenario, we increase the computation complexity.
 - We improve in terms of time spending for our experiments, but this approach is error-prone due to high complexity.
 - We are animating a moving base station by a sequence of base stations, and this is nearer to reality rather than taking snapshots, but still it is not realistic
- Add movement capability to eNodeB features

The only close to the realistic scenario is to physically move eNodeB itself by applying modifications and developments in its features, also adding new features..

- This is the most realistic scenario.
- Computational complexity burden is on functions that we embed in SimuLTE, and taking correlation for car parameters for handovers is not needed since the handover mechanism is well-implemented in SimuLTE.
- The only drawback of this approach is that it needs a huge effort to dive into thousands of lines of code and perform developments to move eNodeB neatly without disrupting other eNodeB functionalities.

After hours of discussions and counting advantages over disadvantages, we decided to implement the last approach. To relieve our concerns about simulation stuck due to high computation complexity, we started to look at how some SINR and handovers are implemented. For SINR, we wanted to see how computed it is, is it pre-computed at the initial phase of each experiment, or is dynamically computed? What do the parameters of the handover computation depend on? Can we include the eNodeB movement into handover computations?

Here you see the answers that we found to the above questions:

- By looking into code and extracting the SINR computation algorithm, we figured out that SimuLTE Computes SINR for each band for each user according to pathloss, shadowing (optional) and multipath fading upon each LTE airframe arrival. This means no pre-computation is applied. In listing 4.1, you can see the algorithm.

```

1   for each arrived Data packet or FeedBack packet
2   {
3       get transmission power;
4       get antenna gain;
5       get the Resource Blocks used to transmit this packet;
6       find object associated to the packet; //if we are in
7       downlink this object is eNodeB otherwise UE
8       get the current position of eNodeB and UE;
9       compute pathloss;
10      compute fading;
11      compute shadowing (optional)
12      compute received Signal strength according to
13          transmission power pathloss, fading, shadowing and
14          antenna gain;
15      compute Interference;
16      compute Noise;
17      SINR = received signal / (Interference + Noise);
18  }

```

Listing 4.1: SINR computation algorithm in SimuLTE

- Handover computation is done dynamically by checking the Received Signal Strength Indicator(RSSI) periodically from neighboring eNodeBs by each UE. At each time interval, each eNodeB broadcasts a handover frame. Whenever UE receives a broadcast from another eNodeB with higher RSSI, it starts the handover process to new eNodeB regardless of moving or stationary eNodeB. By studying the handover process in SimuLTE and performing some experiments, we conclude that handovers will be done correctly in moving eNodeB scenarios as well.

Now we are sure that we can start developing eNodeB to make it movable without worrying about SINR and handover computations.

4.2 Moving eNodeB

Our goal was to have a set of moving eNodeBs that move as if each one mounted on top of each car in all of our scenarios so that the mobility of eNodeB should coincide point by point with the position of the corresponding car at any time instance. In order to develop eNodeB for this purpose, we have to study current features of eNodeB, so we go to the path: /src/corenetwork/nodes and open the NED file of eNodeB, remembering that NED describes the skeleton and body of a

component and C++ is its soul. Among all features, we see that at line 77, there is a "mobility" feature under the sub-module section, which is set to `StationaryMobility` by default. This means the nature of `eNodeB` in `SimuLTE` is stationary. There is a nice feature in `OMNeT++` that allows you by holding the `Ctrl` key and click on `StationaryMobility` it shows the source NED file of `StationaryMobility`, and you can see all attributes of this feature. By observing `StationaryMobility` file, we figure out that these features are part of `INET`.

4.2.1 Mobility models

After studying `INET` mobility models, we understand that there are many mobility patterns that we can have for our `eNodeB` that we explain some of them which are tested by us:

- `LinearMobility`

This module describes a constant speed linear motion. There are parameters to set starting angle, speed, and acceleration. For initial positioning, users can set three parameters (`initialX`, `initialY`, `initialZ`), inside a pre-defined constraint area by themselves. After testing this model on `eNodeB` we realize some drawbacks:

- We do not have speed control over `eNodeB`. Once we set the speed at the initial phase of our scenario, `eNodeB` keeps it, and there is no possibility to manage the speed of motion while the simulation is running.
- No possibility to change the direction of `LinearMobility`. Once we run the simulation, `eNodeB` starts to move on a straight line in the direction according to a given angle, and we can not manage it, and it keeps going forward until it reaches the border of the pre-defined constraint area.

- `TurtleMobility`

It is a programmable mobility model, where we program our moving object with a set of statements in the form of an XML script. There will be commands to set the speed and position, turn to a specified angle, pass from a certain distance, etc. So, various motion patterns can be described by this model. This model has some disadvantages that does not fit our work.

- `TurtleMobility` can not consider coordinates (X, Y, Z) of the area and just follows the path as a result of the degree of direction and speed of movement, which are specified in the XML file program. We can not define every point that we would like the `eNodeB` to traverse.

- **BonnMotionMobility**

It is a trace-based mobility model, in the sense that a pre-recorded trace file specifies nodes' trajectory. The BonnMotionMobility uses a plain native text file format of the BonnMotion simulation tool. Each line of this file describes the motion of one host. A line can consist of either triplet format (t, x, y) or quadruples (t, x, y, z). This means that the given node goes to the position (x,y,[z]) at the time t. There is a boolean parameter called is3D that is responsible for controlling whether lines should be interpreted as including quadruples (3D) or triplets (2D). Here we point out some pros and cons:

- The most distinguishing feature of this mobility model is that we can tell a node where to be at each time instance t, and this is helpful for our purposes, especially when we want to select a car trajectory to mount an eNodeB on that car.
- Disadvantage of this model is that it does not support real-time positioning in the sense that there is no possibility of real-time assignment of car trajectory to eNodeB at the first simulation run. In other words we have to first run simulation with random car path in SUMO, providing TCP connection with OMNeT++ through Veins, extracting each car trajectory point by point at each time instance to have translation of SUMO path as many (t,x,y) triplets in OMNeT++.

We studied some other mobility models like CircularMobility and Gauss-MarkovMobility. As their names suggest, they had some attributes, but those attributes were not in line with our aims. for more information visit [12].

4.2.2 BonnMotionMobility utilization

In order to utilize BonnMotionMobility model, we need to provide a plain text file with "movements" extension i.e., "Car[1].movements" for each eNodeB that we want to make a move. We have to extract each car trajectories from SUMO and translate them into sequences of time and position (t1 x1 y1 t2 x2 y2 ...). For this purpose, we have to study the Application Programming Interface(API) used to communicate between SUMO and SimuLTE and understand how to change the API code to extract our desirable information. To summarize what we studied on this API called TraCI (Traffic Control Interface), a C++ program provides a TCP connection between the traffic simulator as a server and the network simulator as the client. During the connection, SUMO sends information about the traffic to OMNeT++. In Figure 4.2, you see the schematic of the communication.

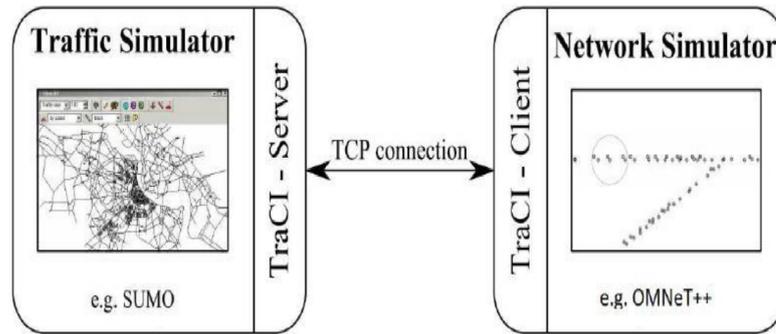


Figure 4.2: Communication schematic

To extract required information from TraCI, there will be two important challenges:

- We have to evaluate the source code of TraCI to see how we can extract each car’s information.
- Since each car has its own ID, we have to extract the car Id and bind it to that car’s trajectory file to understand which car the current trajectory belongs to.

By considering the above challenges, we devised an algorithm to extract each car trajectory as a sequence of time and positions to keep each file’s uniqueness for each car using car IDs. In listing 4.2, this algorithm is shown.

```

1 while simulation is running
2 {
3     for each car[i] in scenario
4     {
5         create a file car[i].movements;
6         write current simulation time;
7         write position.X and position.Y;
8     }
9 }
```

Listing 4.2: Algorithm for time and position extract result of this algorithm in Figure 4.3

To get a desirable result, we found the TraCIScenarioManager.cc file in which we insert our modification after line 950. In listing 4.3, you can see the piece of C++ code we used to extract this information. Please note that this code describes inside the for loop of the above algorithm.

```

1 FILE *fp=fopen("Car[i].movements","a");
2 std::string str = std::to_string(simTime().dbl()) + " "+
   std::to_string(p.x) + " "+ std::to_string(p.y) + " ";
3 fputs(str.c_str(),fp);
4 fclose(fp);

```

Listing 4.3: C++ code inside the for loop

Now we have car trajectories compatible with BonnMotionMobility, each in a separate file suitable for assigning each file to each eNodeB. This way, we can have eNodeBs exactly at car positions at the right time. We have to change the mobility field in eNodeB.ned file as shown in listing 4.5 but do not forget to import the corresponding package on top of the eNodeB.NED file, listing 4.4, then we have to tell eNodeB to move according to the trajectory indicated in the car[i].movements file in omnetpp.ini file, listing 4.6.

```

1 import inet.mobility.single.BonnMotionMobility;

```

Listing 4.4: BonnMotionMobility source package

```

1 mobility: BonnMotionMobility {
2     @display("p=50,175;is=s");
3 }

```

Listing 4.5: Adding BonnMotionMobility to eNodeB features

```

1 *.eNodeB1.mobility.typename = "BonnMotionMobility"
2 *.eNodeB1.mobility.traceFile = "Car[1].movements"
3 *.eNodeB1.mobility.is3D = false
4 *.eNodeB1.mobility.nodeId = -1

```

Listing 4.6: Car trajectory assignment to eNodeB

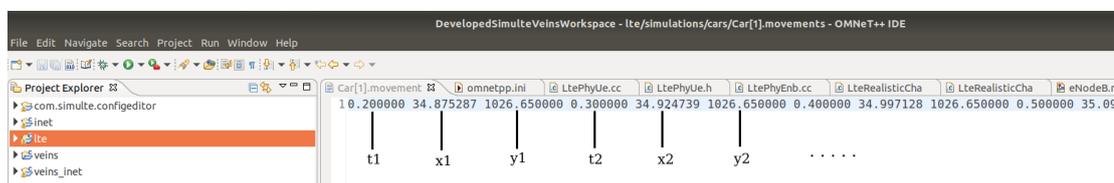


Figure 4.3: Car position trajectory with time suitable for BonnMotionMobility

4.3 Increase eNodeB population

When we want to have more than one eNodeB in our experiments, we have to perform some steps. We are mentioning these steps here as a separate section because it was challenging, and there is no guide or tutorial for it. For every eNodeB which we want to add to our scenario, we have to pass these steps mentioned in the algorithm shown in listing 4.7 in this algorithm; we assume that we have eNodeB0 already in the scenario and we are going to add eNodeB1:

```

1 //step1 : Add the line of code corresponding to new adding eNodeB
   and make sure that you have imported its package
2 import lte.corenetwork.nodes.eNodeB;
3 eNodeB1: eNodeB {
4     @display("p=200,0;is=1");
5 }
6 //step2 : set up the backhaul connection (to packet gateway)
7 pgw.pppg++ <--> Eth10G <--> eNodeB1.ppp;
8 //step3 : open omnetpp.ini file and add these lines
9 *.eNodeB*.numX2Apps = 1
10 *.eNodeB*.x2App[*].server.localPort = 5000 + ancestorIndex(1)
11 *.eNodeB0.x2App[0].client.connectAddress = "eNodeB1%x2ppp0"
12 *.eNodeB1.x2App[0].client.connectAddress = "eNodeB0%x2ppp0"

```

Listing 4.7: Steps for each additional eNodeB

There are some considerations regarding steps 2 and 3:

- In the LTE network, the interconnecting interface between eNodeBs is X2 and supports both Control Plane and User Plane. Whenever we add new eNodeB, we have to put the number of X2 connections equal to the number of eNodeBs that this new one wants to connect to i.e., when adding the fifth eNodeB, we have to put `*.eNodeB*.numX2Apps = 4`. This means now each eNodeB has four X2 connections (one for each of the other eNodeBs).
- In the array of connecting addresses of each eNodeB, we have to specify which other eNodeBs this new eNodeB is connected to (listing 4.7 lines 11 and 12). PPP stands for point to point protocol that is implemented in INET and is used in SimuLTE.

4.4 Appear and disappear eNodeB

According to what is illustrated in Figure 4.2, SUMO as a server tells about conditions of the cars at each time instance to OMNet++ by a TraCI listener inside the Veins. There are moments in which you see that cars enter (appear) to the

scenario and also leave (disappear) after a while. The question that arises is that what should we do with eNodeBs mounted on top of cars which enter the scenario at time t_1 and leave at time t_2 ($t_1 < t_2$)? We divided the problem into two main parts to address the problem and tried to conquer like what you will read in 4.4.1 and 4.4.2.

When we think about appearing, disappearing eNodeB, we envision physically appearing them and then disappearing after some time, just like what is happening for cars in SUMO, which is ideal but it has some disadvantages:

- Appearing and disappearing, a wired node in OMNeT++ needs to manipulate the code of the graphical part of the program, and that would need OpenGL coding knowledge, which was difficult to manage in 6 months with having lots of other works to do.
- Since OMNeT++ depicts all the nodes contributing to the scenario at the beginning (except cars), if we want to frequently appear and disappear nodes, it burdens a huge workload on our computing system, especially in large scenarios with a lot of moving eNodeBs.

4.4.1 Appear eNodeB

By studying the functionalities of eNodeB in SimuLTE, we figure out that there is an initialize function for eNodeB embedded at the physical layer at `LtePhyEnb.cc` file in which "start to work of eNodeB" is written in C++ code. In this function, each eNodeB introduces its presence to other nodes (eNodeBs, UEs, and cars) with a broadcast message. The time of the broadcast message is set to 0, which means at the beginning of the simulation process.

The idea of appearing eNodeB can be implemented by just turning the eNodeB on at the time at which the corresponding car that is supposed to carry this eNodeB appears to the scenario. However, how to turn an eNodeB on? By scheduling the time at which the eNodeB introduces its presence equal to the presence of its carrier car to the scenario. The algorithm is written in listing 4.8.

```

1 //at the end of initialize funtion of LtePhyEnb.cc
2 if ( eNodeB is the one that we want to turn on)
3 {
4     scheduleAt(Time of car presence , broadcast message Starter);
5     Eisable ppp link of eNodeB to server;//Enable back haul
6 //scheduleAt is a function in OMNeT++ used to schedule a self-
7     message, this self-message turns on the eNodeB
8 }

```

Listing 4.8: An algorithm for turning eNodeB on

4.4.2 Disappear eNodeB

After studying and testing the eNodeB behavior, we realized that by setting the transmission power of eNodeB to zero, we turn eNodeB off when the carrier car leaves the scenario. This way, we disable all eNodeB functionalities. How we do it in the code? In the `LtePhyEnb.cc` file, which defines attributes of the physical layer of eNodeB, there is a function named `handleAirFrame(cMessage* msg)`. This function is responsible for all the receiving packets from all nodes in the network and can work only if the transmission power of eNodeB is not zero. By using the algorithm mentioned in the listing 4.9, we turn off the eNodeB.

```

1 //at the beginning of handleAirFrame function of LtePhyEnb.cc
2 if (eNodeB is the one that we want to turn off and simulation time
   = leaving time of the car that carries This eNodeB)
3 {
4     set the transmission power to zero;
5     disable PPP link of eNodeB to the server;//disable backhaul
6 }

```

Listing 4.9: An algorithm for turning eNodeB off

4.5 Macro and micro eNodeBs

Distinguishing between macro and micro eNodeBs is one of challenging parts of developing SimuLTE. By looking at the SimuLTE website and investigating in code and asking creators of SimuLTE, we figure out that having both macro and micro eNodeBs in one experiment simultaneously is not provided in the SimuLTE original code. So we decided to develop this facility for eNodeB as well. Although SimuLTE creators claim that by having different antenna transmission directions i.e., omnidirectional and anisotropic we can have micro and macro base stations in the same scenario, the transmission power and antenna height are two important factors that are not possible to be obtained with just changing omnidirectional to anisotropic. Finally we decided to have both micro and macro eNodeBs at the same time with distinct transmission power and antenna height. there are two important decision-making points:

- In order to distinguish between macro and micro eNodeBs, we have to look for a feature that makes the difference between eNodeBs independently. What comes to mind is eNodeB Id (identifier), but there is also cell Id, which is exclusive to each eNodeB.
- We have to decide what the best place among these forty thousand lines of code is for making the decision about which eNodeBs are about to be macro

and which ones are to be micro

For the first point making a decision depends on the second point, which means in some of the C++ classes defined in SimuLTE, we can work with eNodeB Id in some others with cell Id. After searching and testing some functions, it turns out that in the C++ class header file named `LteCellInfo.h` (in the previous SimuLTE version named `LteDeployer.h`), there is a getter function that only returns the type of eNodeB for either macro or micro type, not both. So we found the best place to make a decision is here. In the listing below, you can see the modified code of the get function in the `LteCellInfo.h` file.

```

1 EnbType getEnbType()
2 {
3     if (cellId_==1 || cellId_==2)
4         eNbType_ = MACRO_ENB;
5     else
6         eNbType_ = MICRO_ENB;
7     return eNbType_;
8 }

```

Listing 4.10: Developed get function that returns eNodeB type based on cell Id (`LteCellInfo.cc`)

When this function is called from the physical layer C++ class of eNodeB (`LtePhyEnb`), the physical layer class initialize function sets the transmission power of each eNodeB base on what this get function return. In `getEnbType()` function, in exemplary way we have described that we have two macro eNodeB for cell Ids equal to 1 or 2, and all the others are micro, but by using an extended if-else command or switch-case command we can have desirable number of macro or micro eNodeBs.

4.5.1 eNodeB height

When we have different types of antennas, we would like to have different antenna altitudes as well. This makes us want to develop SimuLTE more. In SimuLTE, the eNodeB height contributes to the computation of attenuation in the LTE communication channel. For channel simulation, SimuLTE has a component named `LteRealisticChannelModel.ned` in which the attenuation computation is performed there but for macro and micro eNodeBs distinctly. We bring them together in one function with different eNodeB height, to do so, the goal was to choose a feature that based on we could differentiate among eNodeB types in order to assign the different heights.

After investigating this feature, we realize that eNodeB Id is the best choice, but the problem is that this feature is not available in the `getAttenuation` function, which computes attenuation, and we have to find a way to pass eNodeb Id to this

function. We use a search algorithm to find some points in code that we have eNodeB Id available for passing it to the getAttenuation() function to be able to make decision in getAttenuation() function.

```

1 if (Find functions calling getAttenuation function)
2   if(find eNodeB Id)
3     pass Id with getAttenuation arguments
4   else
5     Find eNodeB Id where the caller function of getAttenuation
      is called and pass the Id

```

Listing 4.11: recursive algorithm to find eNodeB Id for getattenuation()

After successfully passing the eNodeB Id to the getAttenuation function, we have to choose which eNodeB will be macro for having higher height. The next step is going to put together attenuation computation and make a decision based on eNodeB height. For more clarity, the developed code for attenuation computation is shown in the appendix. Please note that formulas are taken from the Winner II channel model.

4.6 Statistic collection

SimuLTE has a nice statistic collection mechanism, and some statistic measurements are available in the original code, but SimuLTE creators do not collect some essential ones for our purpose. We decide to create and embed them into simuLTE code. First of all, to highlight our work, we have to see what performance parameter statistics are collected in SimuLTE and what are not. In Table 4.1, we see all the statistic measurements related to our work. Measuring throughput and delay in

Per user performance metrics	Uplink	Downlink
Mac layer throughput	✓	✓
Physical layer throughput	✗	✗
Mac layer delay	✓	✓
Physical layer delay	✗	✗
SINR	✗	✓
Number of Handovers	✗	✗

Table 4.1: statistics measured by SimuLTE have ✓. For handovers uplink/-downlink doesn't make sense but for coherency in table format we mentioned both

the Mac layer is useful when UE does not move from one cell to another because in SimuLTE, when a user moves to a new cell, all statistics measured in the last cell

will be eliminated. This is weird, but that is how simuLTE works; hence, we will explain how delay and throughput measurement are developed in the physical layer of all LTE nodes in SimuLTE. In the following, you will see how we implement statistic measures that they are specified with (✕) in the above table.

4.6.1 Physical layer throughput and delay

The most important question for starting to compute delay and throughput is that what formula is suitable for measuring these parameters? We decide to use formulas for throughput and delay which are used in original SimuLTE code.

```
1 for each arrived packet
2 {
3     Total received bytes += the current packet size
4     Throughput = Total received bytes / current time
5 }
```

Listing 4.12: average throughput

```
1 for each arrived packet
2     Delay = current time - packet time stamp
```

Listing 4.13: Delay

At each time instance we collect each received packet size and its arrival time in a vector for point by point throughput computation.

Downlink

We have to embed corresponding code inside the physical layer of UE for having delay and throughput computed truly in downlink. According to the above formulas, whenever a UE receives a data packet, it computes delay and throughput.

Uplink

Computation of throughput and delay for each UE at uplink is very challenging because, at each time instance, many packets from all UEs that are connected to an eNodeB arrive at the physical layer of eNodeB, so there are two main challenges for developing the throughput and delay at uplink:

- Distinguishing packets in order to understand which UE this packet belongs to.
- Once eNodeB identifies the UE that has sent this packet, it needs to have access to that UE's physical layer to write statistics of this packet to its owner.

For the latter, you may say each eNodeB can save a record of each connected UEs, and for each of them at the end of simulation report throughput and delay. This may seem easier to implement but actually does not work in our case because we have to consider that our eNodeBs are moving and at each time we may have UEs joining and leaving the current eNodeB and keeping track of each single UE and transfer its statistic information to the eNodeB that UE is going to handover on is messier from the coding point of view and also more complicated for the running machine and eventually may not work.

Some coding techniques can be applied to solve the two above mentioned challenges:

- For the first challenge, we propose to use an object pointer of `UserControlInfo` class inside the `handelAirFrame` function inside the physical layer of eNodeB that points to the control information part of the packet and call the function `getSourceId`. This way, we can have access to the packet transmitter's Id, which is one of the UEs connected to this eNodeB.
- The solution we are going to use for the second challenge is more tricky and has some steps:
 - Using forward declaration, we should announce a pointer object that is about to point to UE's physical layer.
 - To specify which UE the pointer object is going to point to, we write a function that uses the module that is in charge of binding UEs to each eNodeB and provides access to the physical layer of UE. This way, we can write all statistics of uplink inside the corresponding UE.

you can see the developed codes in appendix.

4.6.2 Uplink SINR

The SINR computation algorithm in SimuLTE is shown in listing 4.1. Challenges for computation of SINR at uplink are very similar to challenges mentioned for Uplink throughput and delay computation. We used almost the same solutions applied for uplink delay and throughput, for this case. In the `LteRealisticCannelModel.cc`, which is responsible for channel model and conditions, there is a function that is in charge of SINR computation. Whenever this function is called from the physical layer of eNodeB (uplink), it computes per packet SINR, and with the help of a pointer object to the physical layer of UE, we collect SINR and record for that specific UE. The general approach is very similar to what is mentioned in the previous section for uplink throughput and delay. The developed code is in the appendix.

4.6.3 Per User number of Handovers

When we look at the Handover process in SimuLTE, we see that computations and management are done very nicely, but there is a missed statistic collection point, which is per user number of Handovers in the experiment. All the computation and management processes of Handovers are done in the physical layer of each UE. So the approach is to sit in the `doHandover()` function and observe the Handover completion. Once it is done, we increase a private variable belonging to the corresponding UE class's physical layer by 1.

For collecting all the mentioned developed statistics, we used the signal-emit approach. In the next chapter, we will perform a set of functional tests to evaluate what is developed in this thesis, and at the end a performance test on a large scenario with 200 UEs and 6 micro (moving and stationary) and 1 macro stationary eNodeBs.

Chapter 5

Experiments and results



Figure 5.1: Experiments and results

5.1 Funtional experiments

5.1.1 EnodeB movement experiment

As an outcome of our work in this thesis, to ensure that the eNodeB movement is done properly, we will set up two comparative experiments then compare the results for the validity of our work in this thesis.

- First setup: UE moves, eNodeB doesn't move.
 - Goal: Examination of UE movement.
 - Description: In the first experiment, we set up a scenario where one macro eNodeB and one user equipment (UE) are located in a $1432 \times 1432 m^2$ area. The eNodeB position will be fixed for the whole experiment duration, but the position of UE at the beginning of the experiment is near the eNodeB (Figure 5.2) then with a constant speed approaches close to eNodeB (Figure 5.3) then as the time goes on it gradually moves away (Figure 5.4) with the same constant speed (as if the user sits in a car). During the experiment, we measure SINR perceived by UE (downlink) and the distance of UE from eNodeB .

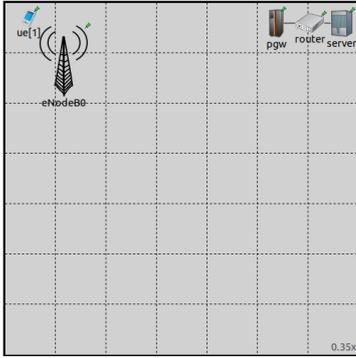


Figure 5.2: initial position of UE

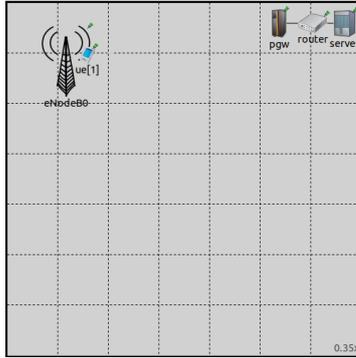


Figure 5.3: A position close to eNodeB



Figure 5.4: last position far from eNodeB

- Hypotheses:

-	Number	Height	Antenna gain	Speed	TX power	Moving
UE	1	1.5m	0	15m/s	26 dBm	✓
Macro EnodeB	1	25m	18	0m/s	46 dBm	✗

Table 5.1: Experiment specifications

- Expectation: As the UE approaches eNodeB, we anticipate higher SINR, and as it goes farther from eNodeB, we expect the received signal at UE is weaker so that the SINR will decrease.
- Result: From two curves below, we observe that measured results for both distance and SINR meet our expectations. So at the beginning of the experiment, UE gets close to static eNodeB smoothly (shorter distance) while the SINR approaches its top value. The closer UE to eNodeB, the higher SINR until around time=22 seconds, then by going UE far from eNodeB, a downturn in SINR starts and continues until the end of the experiment at time=120 seconds.

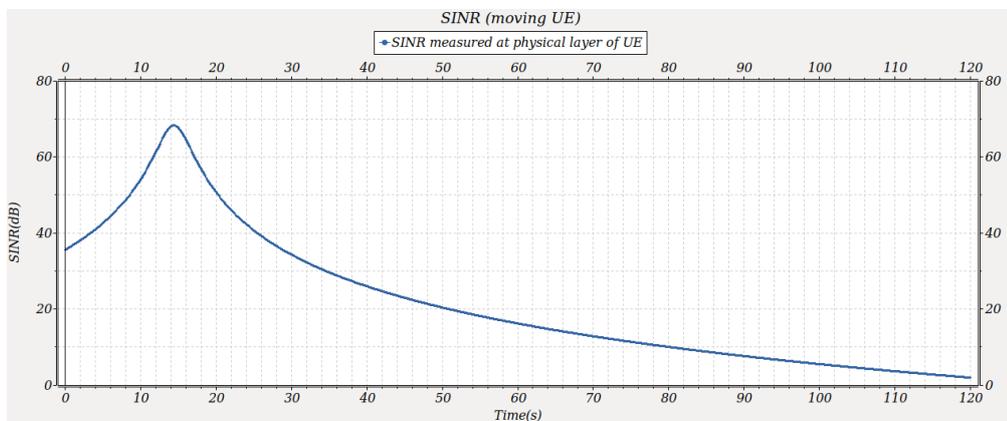


Figure 5.5: SINR vs time for Moving UE

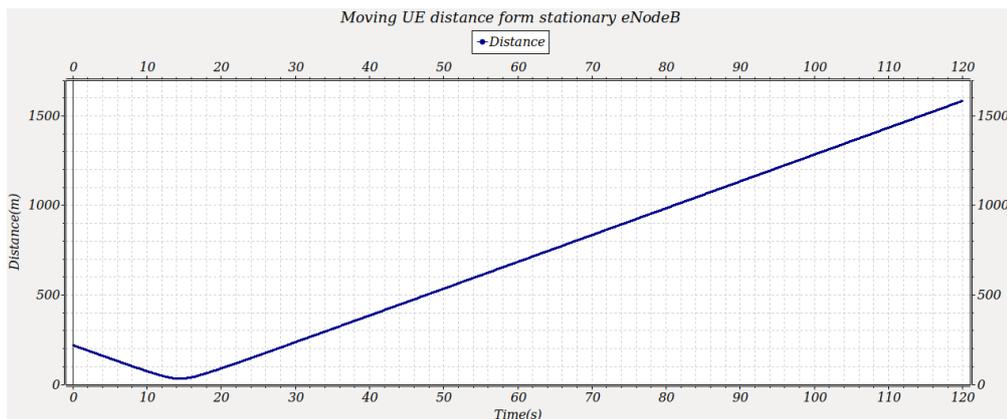


Figure 5.6: Moving UE distance from stationary eNodeB vs time

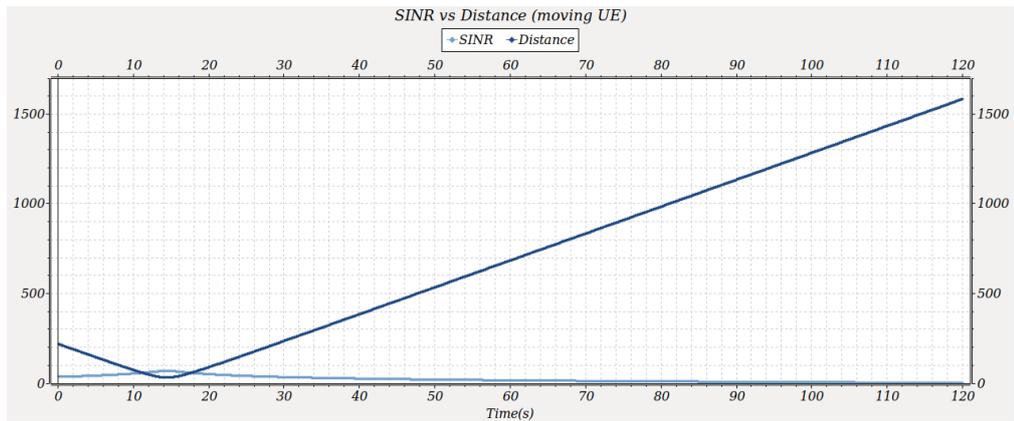


Figure 5.7: SINR vs Distance

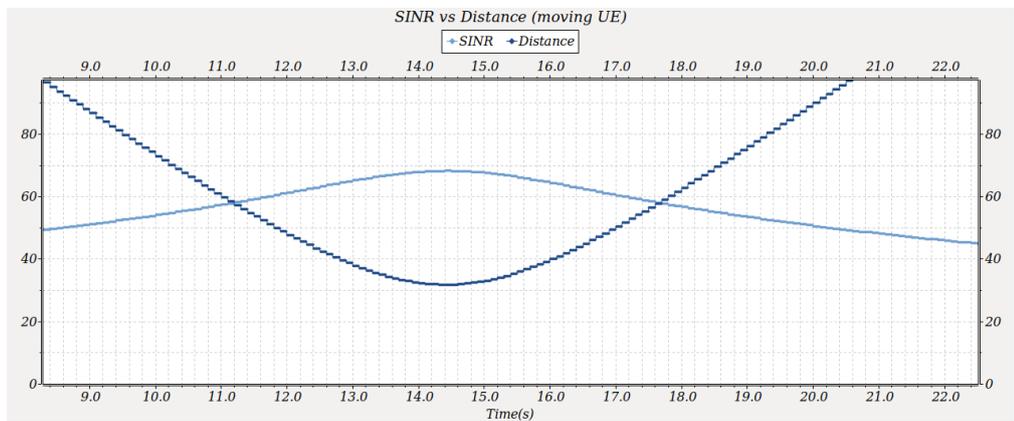


Figure 5.8: SINR vs Distance (zoomed)

- Second setup: eNodeB moves, UE does not move.
 - Goal: Examination of eNodeB movement.
 - Description: we set up the scenario precisely opposite of the previous scenario in terms of positions. In this experiment, instead, we fix UE's position on the place that eNodeB was in the previous experiment for the whole experiment duration, and we move eNodeB on the same path and speed of UE in the previous experiment. Please note that moving eNodeB goes close to UE diagonally-straight and then gets far from it.
 - Hypotheses: the same as previous experiment except moving part which is vice-versa.

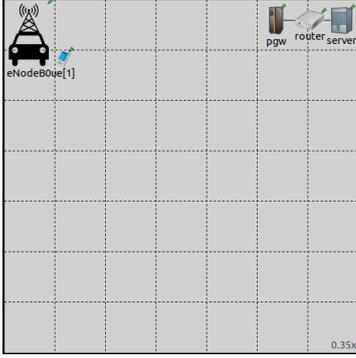


Figure 5.9: initial position of Moving eNodeB

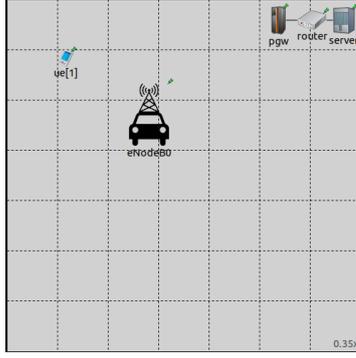


Figure 5.10: A position close to fixed UE



Figure 5.11: last position far from fixed UE

-	Number	Height	Antenna gain	Speed	TX power	Moving
UE	1	1.5m	0	0m/s	26 dBm	✗
Macro EnodeB	1	25m	18	15m/s	46 dBm	✓

Table 5.2: Experiment specifications

- Expectation: We expect that by making the distance between UE and eNodeB larger, we witness a lower SINR, and more importantly, we anticipate getting almost identical results compared to the first setup.
- Result: Depicted results show that measured SINR and distance values are identical to the first setup. So the results are in line with our expectations.

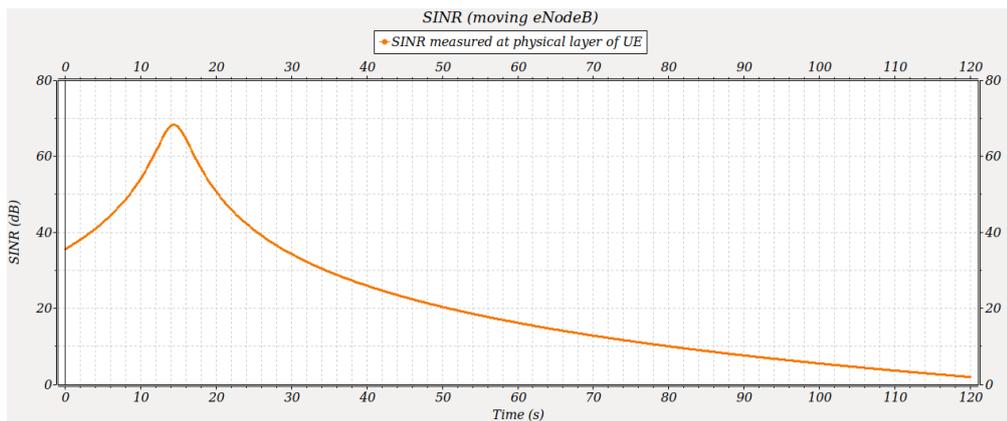


Figure 5.12: SINR vs time for moving eNodeB

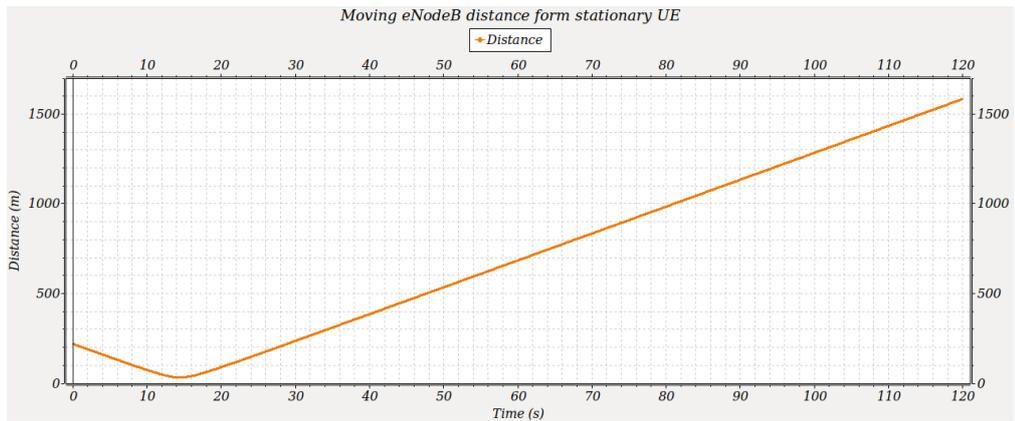


Figure 5.13: UE distance from moving eNodeB vs time

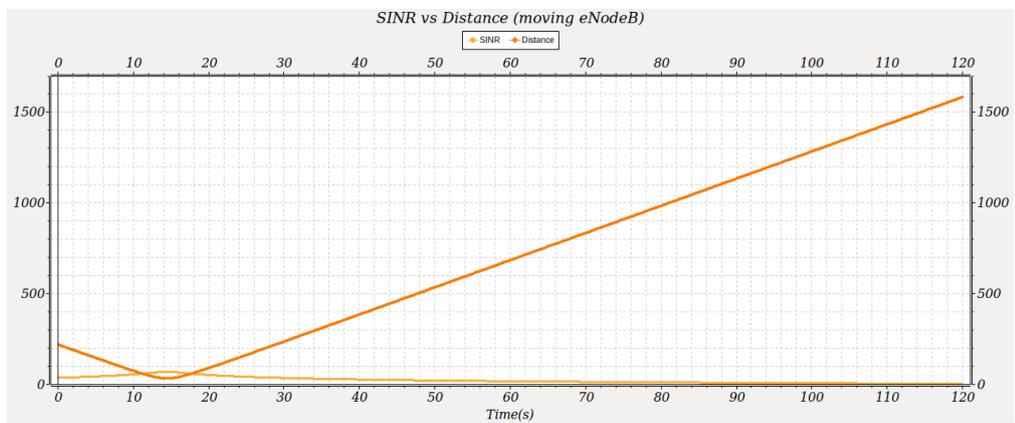


Figure 5.14: SINR vs Distance

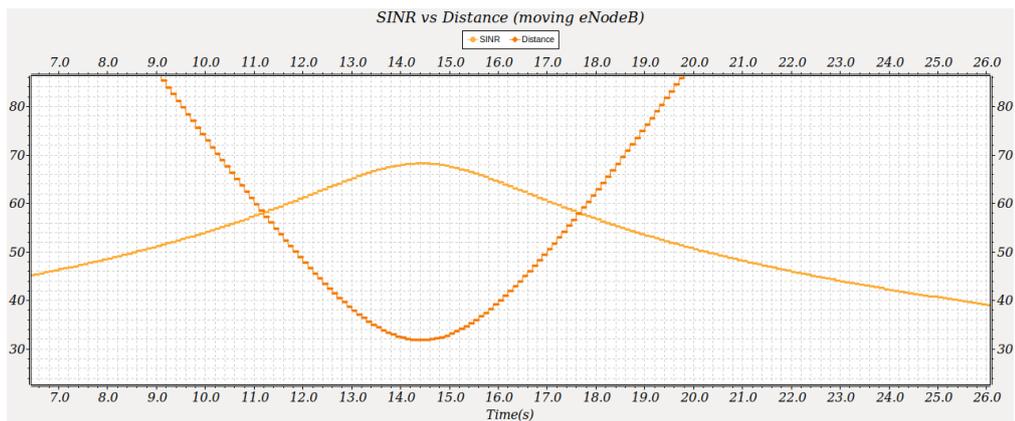


Figure 5.15: SINR vs Distance (zoomed)

5.1.2 eNodeB On and off experiment

This experiment aims to validate the developed framework in this thesis through a turn on then a shutdown that we give to a moving eNodeB. This way, we ensure that the mounted eNodeB on top of entering the car to the scenario is operative, and it will be expelled functionally from the scenario when its carrying car leaves. Since cars enter from one side of the setup scenario and leave from the other side, the mounted eNodeB on the corresponding car which comes to scenario and leaves must be turned on and off at the time which that car enters and leaves the scenario respectively.

- Description: The moving eNodeB will arrive in the scenario from the left side of the playground after 20 seconds passed from the starting time of simulation (Figure 5.16), this means moving eNodeB should be turned on at $t=20$. There is a UE in the middle of the playground at position (716,716), which is going to start to communicate with moving eNodeB once the eNodeB enters ($t=20$ s, turned on). As time passes moving, eNodeB gets closer to UE. After reaching the closest point (Figure 5.17), it moves away gradually until it vanishes ($t=100$, turned off)(Figure 5.18).



Figure 5.16: Moving eNodeB is entering to playground

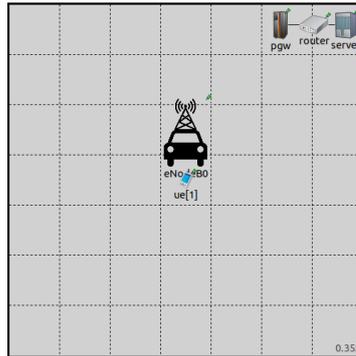


Figure 5.17: moving eNodeB position closest to fixed UE

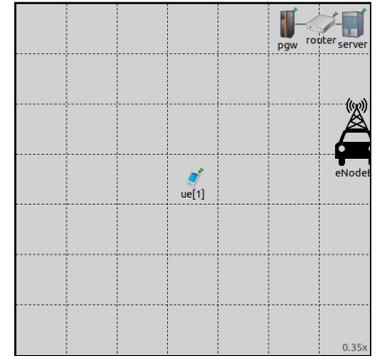


Figure 5.18: Moving eNodeB is about to leave the playground

- Hypotheses: Table 5.3

-	Number	Height	Antenna gain	Speed	TX power	Moving
UE	1	1.5m	0	0m/s	26 dBm	✗
Macro EnodeB	1	25m	18	10m/s	46 dBm	✓

Table 5.3: Experiment specifications

- **Expectation:** we look forward to seeing no SINR computed for the $t < 20$ s because there is no eNodeB, therefore no communication between UE and eNodeB before $t = 20$ s then we have to see computed SINR from $t = 20$ s, and by passing the time it creates a bell shape until it goes out of the playground because when eNodeB goes further, it approaches closer to UE thus higher SINR and when it departs away we expect lower SINR and when eNodeB disappears, no SINR.
- **Results:** Both graphs shown in Figure 5.19 and Figure 5.20 do not show any value before time=20s and after time=100 for SINR and distance, respectively, which means eNodeB has been successfully deactivated before time=20 and after time=100. eNodeB is activated exactly at time=20, then it remains active until time=100s and just right after time=100 it is off. We observe the highest amount of SINR around time=60 in Figure 5.19, which translates to the minimum distance of eNodeB around time=60 in Figure 5.20. results match our expectations.

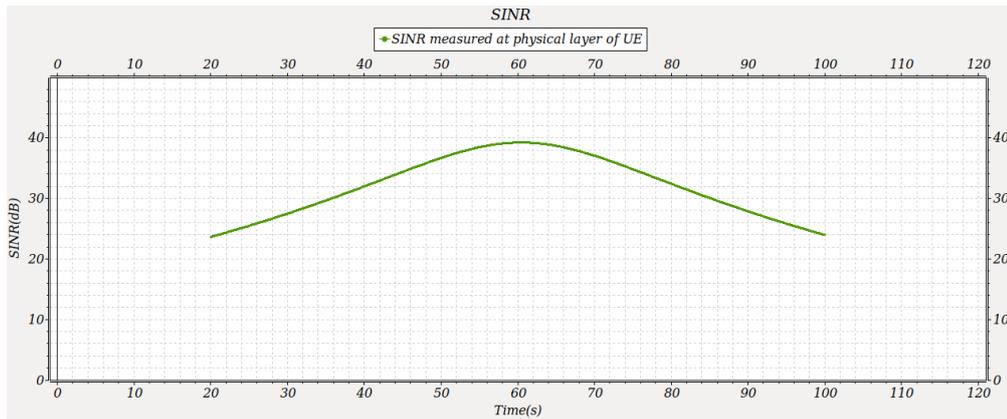


Figure 5.19: SINR vs time for moving eNodeB

5.2 Performance test experiment

In this section, the aim we envision is to evaluate and compare throughput and delay at packet reception of each UEs, which is its physical layer, when they are served by macro and micro eNodeBs in two different scenarios: static micro eNodeBs and moving micro eNodeBs. Please keep in mind two important cases:

- **Case 1:** in all scenarios and all different seeds, we put a macro eNodeB in the middle of the playground which its coverage is over total of 200 UEs present in

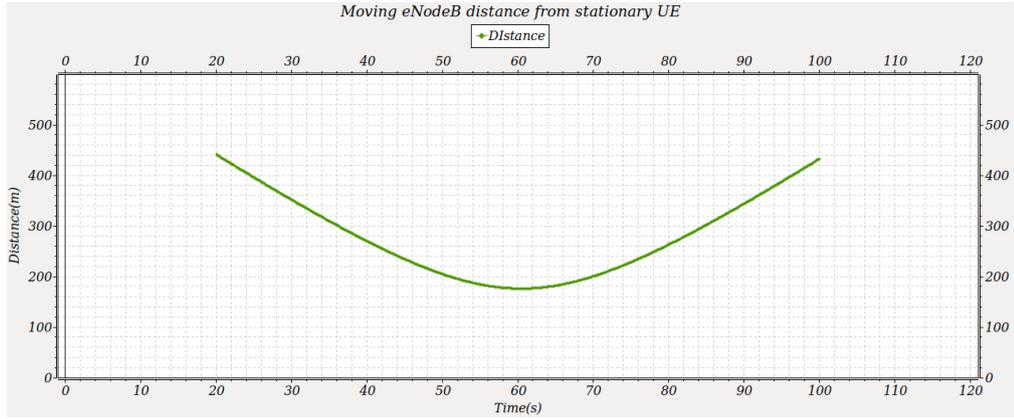


Figure 5.20: UE distance from moving eNodeB vs time

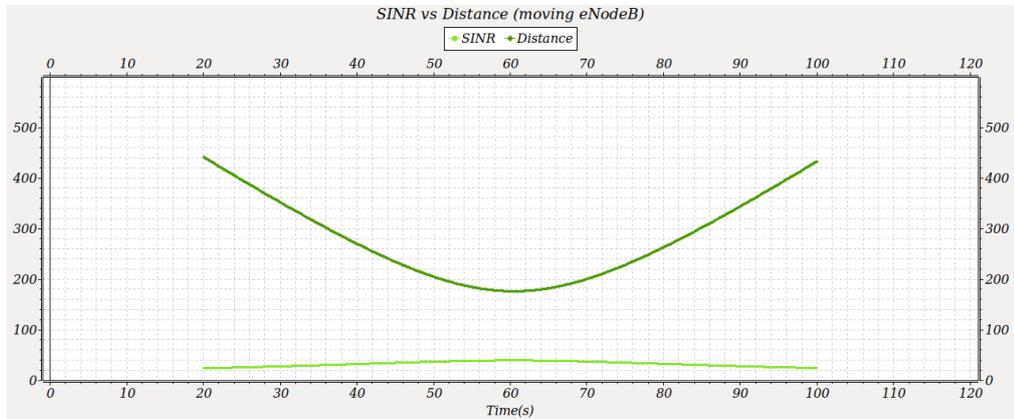


Figure 5.21: SINR vs Distance

the playground in all scenarios. This means in the absence of micro eNodeBs in a part of the playground, the corresponding UEs present in that part are under the coverage of macro eNodeB.

- Case 2: we did a preliminary experiment to compare the performance when we have
 - Only a macro eNodeB in the middle.
 - A macro eNodeB in the middle and six micro ones randomly positioned in the playground.

Results showed, on average, 73% increase in throughput when six micro eNodeBs come to assist macro eNodeB. the reasons why we chose six eNodeB was firstly, to have a low computation complexity to have lower run time for

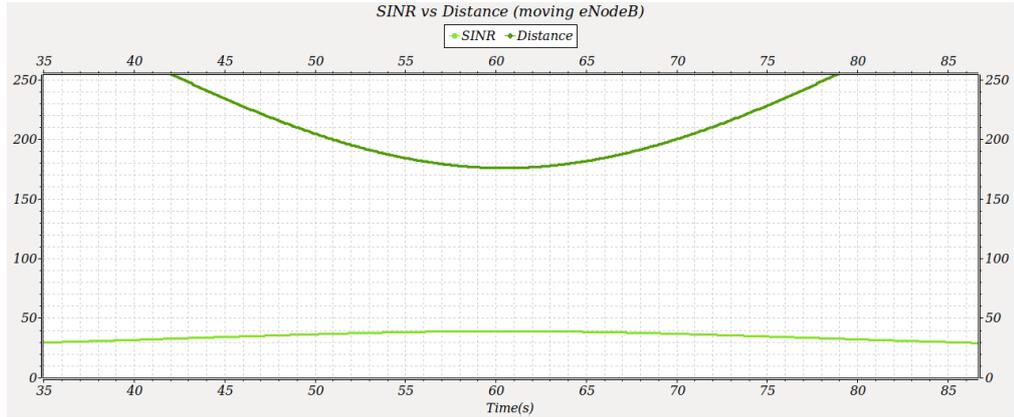


Figure 5.22: SINR vs Distance (zoomed)

simulating each experiment, and secondly, to ensure that for all UEs, there is almost the same chance to get served by micro eNodeBs when we throw micro eNodeBs in the playground uniformly at random.

Now we are going to see two comparative setups' specifications and results:

- First setup: One macro eNodeB in the middle and six stationary micro eNodeBs are randomly positioned.
 - Goal: Study throughput and delay.
 - Description: one macro eNodeB is fixed in the middle of a grid similar to Manhattan streets with 7×7 blocks, each block size is $200m \times 200m$. There are six micro eNodeBs positioned uniformly at random in the grid playground for five different random seeds. There will also be 200 UEs that are randomly positioned uniformly in the playground for each run. The server sends packets towards all UEs (downlink) with a speed of 100 packets per second for a total of 120 seconds simulation run time. Throughput and delay are measured at the physical layer of each eNodeB.
 - Hypotheses: Table 5.4
 - Expectations: As each micro eNodeB randomly gets a position, we expect that for those runs in which micro eNodeBs are in close neighboring, throughput gets lower value due to the impact of interference of closely located micro eNodeBs.

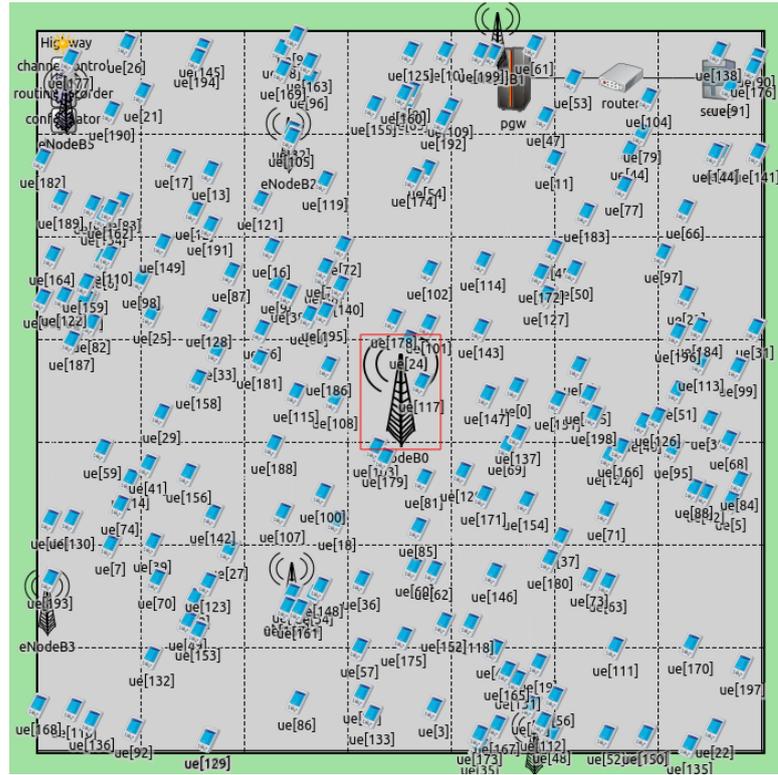


Figure 5.23: one of five figures of this setup

Characteristic	Value	Characteristic	Value
Number of macro eNodeBs	1	Micro TxPower	30dBm
Number of micro eNodeBs	6	Number of sent packets/s	100
Number of UEs	200	Macro eNodeB height	25m
Bandwidth	20MHz	Micro eNodeB height	2.5m
Macro TxPower	46dBm	Simulated time (seconds)	120
Moving macro eNodeB	No	Moving micro eNodeB	No
packet size (at application layer)	40B	Connection Type	UDP

Table 5.4: Experiment specifications

- Results: Depicted positions in Table 5.5 show that whenever micro eNodeBs are getting close to each other or macro eNodeB, the interference increases and causes low throughput. This is more visible in the second run, where three micro eNodeBs are close to each other, and one is close to macro eNodeB, which causes the lowest throughput among all runs.

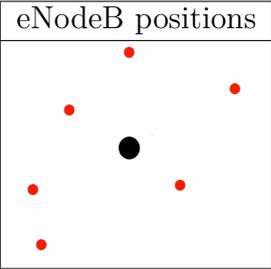
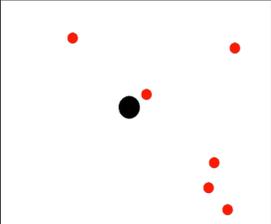
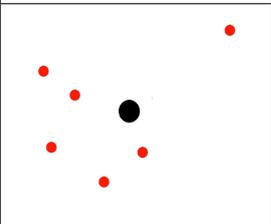
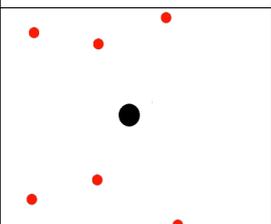
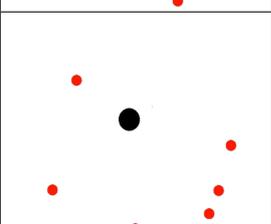
Runs	DL avg throughtput	DL avg delay	eNodeB positions
1	7214.35	0.001s	
2	6069.13	0.001s	
3	6865.39	0.001s	
4	7286.68	0.001s	
5	6770.72	0.001s	
average:	6841.25	0.001	-
95% CI	(6239.55 , 7442.96)	(0.001-1.53e-10,0.001+1.53e-10)	-
99% CI	(5843.46 , 7839.05)	(0.001-2.55e-10,0.001+2.55e-10)	-

Table 5.5: Experiment Results, DL: downlink, avg: average, CI: Confidence Interval, Throughput (Bytes/s) is measured at physical layer , black dot : macro eNodeB position, red dots: micro eNodeB positions

On the contrary, in the fourth run, we observe that not only micro eNodeBs are far from each other, but they are nearer to the playground's

borders. This nice fall-apart distribution of micro eNodeBs leads to the best throughput amongst all. The average throughput of overall runs is 6841.25 Bytes/s.

- Second setup: One macro eNodeB in the middle and six moving micro eNodeBs circulate in the streets of pseudo-Manhattan playground according to random paths.
 - Goal: Study throughput and delay
 - Description: all the description for this setup is identical to the first setup except that now micro eNodeBs are no longer stationary. They move based on a random path assigned to each of them.

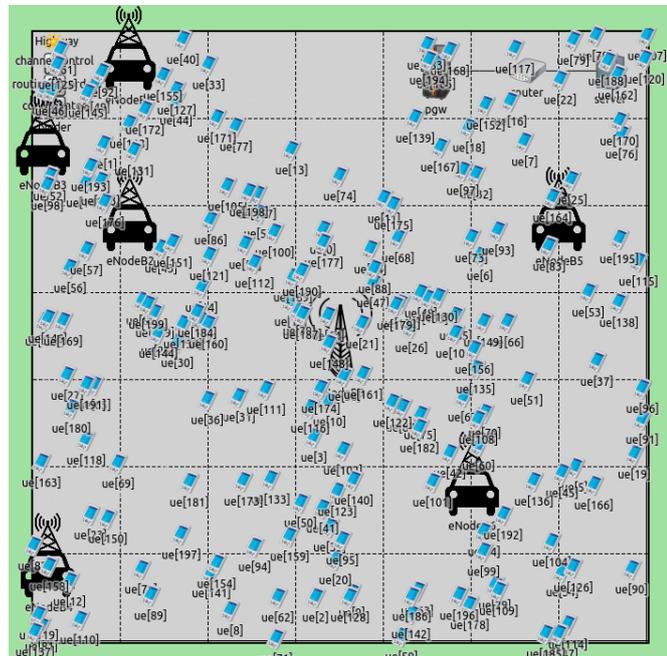


Figure 5.24: one of five figures of this setup

- Hypotheses: Table 5.6
- Expectations: Since we will have 6 moving micro eNodeBs circulating in the playground on their own path, we expect to see many handovers in the scenario. the impact of handovers on performance should not be so much because we set handover latency to be one millisecond.

Characteristic	Value	Characteristic	Value
Number of macro eNodeBs	1	Micro TxPower	30dBm
Number of micro eNodeBs	6	Number of sent packets/s	100
Number of UEs	200	Macro eNodeB height	25m
Bandwidth	20MHz	Micro eNodeB height	2.5m
Macro TxPower	46dBm	Simulated time(seconds)	120
Moving macro eNodeB	No	Moving micro eNodeB	Yes
packet size (at application layer)	40B	Handover latency	0.001s
connection Type	UDP	-	-

Table 5.6: Experiment specifications

Runs	DL avg throughtput	DL avg delay	Total handovers
1	6732.72	0.001s	544
2	7147.92	0.001s	481
3	6380.26	0.001s	391
4	7263.35	0.001s	495
5	7283.34	0.001s	470
average:	6961.51	0.001s	476
95% CI	(6472.82 , 7450.21)	(0.001-1.23e-10,0.001+1.23e-10)	-
99% CI	(6151.13 , 7771.91)	(0.001-2.31e-10,0.001+2.31e-10)	-

Table 5.7: Experiment Results, DL: downlink, avg: average,CI: Confidence Interval, Throughput is measured Bytes/s

- Results: Even though we have many handovers in this scenario, we see that the average throughput of all runs for this scenario is 6961.51, which is almost two percent higher than the static scenario. Even if we do not have improvement in throughput and result of stationary scenario is equal to moving scenario, this is interesting because this moving micro eNodeB results show that with lower price (CAPEX) we are getting almost the same result as stationary micro eNodeB scenario.
- Third setup: repetition of the second setup with zero handovers latency. In this experiment, we repeat the second setup with all its specifications but with different handover latency. In the beginning, we tried to set the handover latency to zero, but OMNeT++ issues an out of range error because of its constraints on my machine for processing intermediate instructions within handover. After analyze and experiment, we realized that the minimum possible latency for the handover process is 10 microseconds. So we repeat the second setup with handover process latency equal to 10 microseconds.

As all other parameters are exactly equivalent to the second setup, the goal, description, and hypotheses in this setup are the same as previous, so we just mention the results.

Runs	DL avg throughtput	DL avg delay	Total handovers
1	6755.81	0.001s	544
2	7166.61	0.001s	481
3	6397.46	0.001s	391
4	7283.15	0.001s	495
5	7299.54	0.001s	470
average:	6980.51	0.001s	476
95% CI	(6491.92 , 7470.40)	(0.001-1.23e-10,0.001+1.23e-10)	-
99% CI	(6169.77 , 7792.85)	(0.001-2.31e-10,0.001+2.31e-10)	-

Table 5.8: Experiment Results, DL: downlink, avg: average, CI: Confidence Interval, Throughput is measured Bytes/s

-Results: What we observe from the table tells us that although we decreased the handover latency to 10 microseconds, there is no significant change in throughput, which is reasonable because we are sending 100 packets per second at the application layer. Each packet size at application layer is 40B and we have 73B air frame (at physical layer = 40+5(GTP)+20(IP)+8(UDP)), so it means that during the moment in which handover is processing in the previous setup, which lasts one millisecond, there will be at most 7 bytes reception less for those nodes which are involving in handover. By reducing handover latency, we can say in this setup nodes that execute the handover process in 10 microseconds instead of one-millisecond receive at most 7 bytes more after each handover at their physical layer with respect to previous setup. By doing simple math, it is quite rational that we have a little higher average throughput in the third setup.

what we learn from all three setups of these experiments is that, By comparing static and moving scenario throughput and delay results, we realize that even if we have six moving eNodeBs that move non stop for 120 seconds around our playground and there are, on average, two handovers per UE and interference is higher when two cars passing by each other, still throughput and delay are more or less the same. On average moving scenario (third setup) gives throughput 2% more than the static scenario.

5.3 Conclusion

After developing and running many simulations to test the simulator itself and measuring performance parameters in both static and moving eNodeB scenarios,

- The first achievement in this thesis is a simulation tool that has developed SimuLTE capabilities to run and test new simulations based on moving base stations. It also collects new statistics such as:
 - Per-user physical layer throughput and delay.
 - Per-user number of handovers.
 - Per-user received SINR in uplink direction at eNodeB.
 - Distance of each node from the base station.

There are also some functions (C++ methods) are created or developed in this thesis, such as:

- `getPhyByMacNodeId()`, which provides access to the physical layer of a node by having its Mac Id. This is a handy function that can be used by other developers whenever they need to have access to the physical layer of a UE or car in different scenarios.
 - `computeUrbanMacro()` for computing attenuation for both micro and macro eNodeBs, according to Winner II, which before was only computing attenuation for macro eNodeBs according to Winner II recipe.
 - `computeUIStatistics()`, this function is created at the node's physical layer to receive parameters from eNodeB, then computes statistics according to that parameters. For future works, this function can be used as a parameter receiver from eNodeB when the direction of data is uplink.
- The second achievement of this thesis is what we obtained from the last experiment. The results of moving micro eNodeBs scenario reflect the fact that "very likely" not only movement of base stations with interference and attenuation and fading, but also handovers up to 1-millisecond latency in process, do not reduce the performance of cellular network comparing to the scenario in which all micro eNodeBs are stationary and positioned at random. To prove the second achievement, doing more math and experiments and comparing theoretical and simulation results is required.

Appendix A

All the functions or piece of codes with light blue background are fully written in this thesis and codes with gray background are original codes that are manipulated for different purposes of this thesis. In order to be easily visible, all the added code lines are specified by comments starting and ending with five stars like this style: `/** added code in this thesis */`.

```
1 cModule* getPhyByMacNodeId(MacNodeId nodeId)
2 {
3     int id = getBinder()->getOmnetId(nodeId);
4     if (id == 0){
5         return NULL;
6     }
7     return (getSimulation()->getModule(getBinder()->getOmnetId(
8         nodeId))->getSubmodule("lteNic")->getSubmodule("phy"));
9 }
```

Listing A.1: C++ method that returns access to physical layer of device which pointer object needs to point to. (lteCommon.cc)

```
1 double LteRealisticChannelModel::computeUrbanMacro(double d,
2     MacNodeId eNbId, bool los)
3 {
4     if (d < 10)
5         d = 10;
6     switch (eNbId)
7     {
8     case 1:
9         hNodeB_ = 25; //Macro eNodeB height(meter)
10        break;
11     case 2:
12         hNodeB_ = 2.5; //micro eNodeB height(meter)
13        break;
14     case 3:
15         hNodeB_ = 2.5;
16        break;
17 }
```

```

16 case 4:
17     hNodeB_ = 2.5;
18     break;
19 case 5:
20     hNodeB_ = 2.5;
21     break;
22 default:
23     hNodeB_ = 25;
24 }
25 double dbp = 4 * (hNodeB_ - 1) * (hUe_ - 1)
26             * ((carrierFrequency_ * 1000000000) /
27 SPEED_OF_LIGHT);
28 if (los)
29 {
30     if (d > 5000){
31         if(tolerateMaxDistViolation_)
32             return ATT_MAXDISTVIOLATED;
33         else
34             throw cRuntimeError("Error LOS urban macrocell path
35 loss model is valid for d<5000 m");
36     }
37     if (d < dbp)
38         return 22 * log10(d) + 28 + 20 * log10(
39 carrierFrequency_);
40     else
41         return 40 * log10(d) + 7.8 - 18 * log10(hNodeB_ - 1)
42         - 18 * log10(hUe_ - 1) + 2 * log10(carrierFrequency_);
43 }
44 if (d < 10)
45     throw cRuntimeError("Error NLOS urban macrocell path loss
46 model is valid for 10m < d ");
47 if (d > 5000){
48     if(tolerateMaxDistViolation_)
49         return ATT_MAXDISTVIOLATED;
50     else
51         throw cRuntimeError("Error NLOS urban macrocell path
52 loss model is valid for d <5000 m");
53 }
54 if (eNbId==1)
55 {
56     double att = 161.04 - 7.1 * log10(wStreet_) + 7.5 * log10(
57 hBuilding_)
58             - (24.37 - 3.7 * pow(hBuilding_ / hNodeB_, 2)) * log10(
59 hNodeB_)
60             + (43.42 - 3.1 * log10(hNodeB_)) * (log10(d) - 3)
61             + 20 * log10(carrierFrequency_)
62             - (3.2 * (pow(log10(11.75 * hUe_), 2)) - 4.97);
63     return att;
64 }

```

```

58     else
59     {
60         return 36.7 * log10(d) + 22.7 + 26 * log10(
        carrierFrequency_);
61     }
62 }

```

Listing A.2: C++ method for attenuation computation according to winner II formula for different eNodeB heights (here for simplicity we assume 1 macro and 4 micro eNodeBs)

```

1     if(result)// result is a Boolean, if it is true we accept the
        packet, otherwise drop. decision is made based on SINR level at
        packet reception
2     {
3         unsigned int size = pkt->getByteLength();
4         totalRcvdBytesDl_ += size;
5         double tputSample = (double)totalRcvdBytesDl_ / (NOW -
        getSimulation()->getWarmupPeriod());
6         emit(phyThroughputDl_, tputSample);
7         //Delay:
8         emit(phyDelayDl_, (NOW - pkt->getSendingTime()).dbl());
9     }
10    double distance = getCoord().distance(lteInfo->getCoord());
11    emit(Dist_, distance);

```

Listing A.3: C++ piece of code used to collect downlink throughput and delay plus distance of UE from eNodeB (LtePhyUe.cc)

```

1 if(result)
2 {
3     phyUe_ = check_and_cast<LtePhyUe*>(getPhyByMacNodeId(lteInfo->
        getSourceId()));
4     unsigned int size = pkt->getByteLength();
5     double delay= (NOW - pkt->getCreationTime()).dbl();
6     double RcvdTime = (NOW - getSimulation()->getWarmupPeriod()).
        dbl();
7     double distance = getCoord().distance(lteInfo->getCoord());
8     phyUe_->computeUlStatistics(size, delay, RcvdTime);
9 }

```

Listing A.4: C++ piece of code used to collect uplink throughput and delay plus distance of UE from eNodeB (LtePhyEnb.cc)

```

1 void LtePhyUe::computeUlStatistics(unsigned int size, double delay
        , double RcvdTime)
2 {
3     totalRcvdBytesUl_ += size;

```

```

4     double tputSample = (double)totalRcvdBytesUl_/RcvdTime;
5     emit(phyThroughputUl_,tputSample);
6     emit(phyDelayUl_,delay);
7 }

```

Listing A.5: C++ computeUlStatistic method for record statistics (LtePhyUe.cc)

```

1     double ti = simTime().dbl();
2     FILE*fp=fopen("outputPosition.movements","a");
3     std::string str = std::to_string(ti) + " " + std::to_string(p.x)
4     + " " + std::to_string(p.y) + " ";
5     fputs(str.c_str(),fp);
6     fclose(fp);

```

Listing A.6: C++ code to extract car trajectory from sumo API at OMNeT++(TraCISecarioManager.cc)

```

1 #Nodesmobility #####
2 *.eNodeB*.mobility.constraintAreaMaxX = 1432m
3 *.eNodeB*.mobility.constraintAreaMaxY = 1432m
4 *.eNodeB*.mobility.constraintAreaMaxZ = 0m
5 *.eNodeB*.mobility.constraintAreaMinX = 0m
6 *.eNodeB*.mobility.constraintAreaMinY = 0m
7 *.eNodeB*.mobility.constraintAreaMinZ = 0m
8 *.eNodeB*.mobility.typename = "BonnMotionMobility"
9 *.eNodeB*.mobility.is3D = false
10 *.eNodeB*.mobility.nodeId = -1
11
12 *.ue[*].mobility.constraintAreaMaxX = 1432m
13 *.ue[*].mobility.constraintAreaMaxY = 1432m
14 *.ue[*].mobility.constraintAreaMaxZ = 0m
15 *.ue[*].mobility.constraintAreaMinX = 0m
16 *.ue[*].mobility.constraintAreaMinY = 0m
17 *.ue[*].mobility.constraintAreaMinZ = 0m
18 *.ue[*].mobility.typename = "BonnMotionMobility"
19 *.ue[*].mobility.traceFile = "stationary.movements"
20 *.ue[*].mobility.is3D = false
21 *.ue[*].mobility.nodeId = -1
22 #end of Nodes mobility#####
23 #CellId,NodeId #####
24 **.eNodeB*.macCellId = index+1
25 **.eNodeB*.macNodeId = index+1
26 #end of CellId,NodeId #####
27 [Config Cbr-UL]## for app layer#####
28 *.server.numUdpApps = 1
29 *.server.udpApp[*].typename = "CbrReceiver"
30 *.server.udpApp[*].localPort = 3000 + ancestorIndex(0)
31 *.ue[*].numUdpApps = 1
32 *.ue[*].udpApp[0].typename = "CbrSender"

```

```
33 *.ue[*].udpApp[0].destAddress = "server"
34 *.ue[*].udpApp[0].destPort = 3000 + ancestorIndex(1)
35 #enodeB0#####
36 *.eNodeB0.x2App[0].client.connectAddress = "eNodeB1%x2ppp0"
37 *.eNodeB0.x2App[1].client.connectAddress = "eNodeB2%x2ppp0"
38 *.eNodeB0.x2App[2].client.connectAddress = "eNodeB3%x2ppp0"
39 *.eNodeB0.x2App[3].client.connectAddress = "eNodeB4%x2ppp0"
40 *.eNodeB0.x2App[4].client.connectAddress = "eNodeB5%x2ppp0"
41 *.eNodeB0.x2App[5].client.connectAddress = "eNodeB6%x2ppp0"
42 *.eNodeB0.x2App[6].client.connectAddress = "eNodeB7%x2ppp0"
43 *.eNodeB0.x2App[7].client.connectAddress = "eNodeB8%x2ppp0"
44 *.eNodeB0.x2App[8].client.connectAddress = "eNodeB9%x2ppp0"
45 *.eNodeB0.x2App[9].client.connectAddress = "eNodeB10%x2ppp0"
46 *.eNodeB0.x2App[10].client.connectAddress = "eNodeB11%x2ppp0"
47 *.eNodeB0.x2App[11].client.connectAddress = "eNodeB12%x2ppp0"
48 *.eNodeB0.x2App[12].client.connectAddress = "eNodeB13%x2ppp0"
49 *.eNodeB0.x2App[13].client.connectAddress = "eNodeB14%x2ppp0"
50 *.eNodeB0.x2App[14].client.connectAddress = "eNodeB15%x2ppp0"
51 *.eNodeB0.x2App[15].client.connectAddress = "eNodeB16%x2ppp0"
52 *.eNodeB0.x2App[16].client.connectAddress = "eNodeB17%x2ppp0"
53 *.eNodeB0.x2App[17].client.connectAddress = "eNodeB18%x2ppp0"
54 *.eNodeB0.x2App[18].client.connectAddress = "eNodeB19%x2ppp0"
55 *.eNodeB0.x2App[19].client.connectAddress = "eNodeB20%x2ppp0"
56 *.eNodeB0.x2App[20].client.connectAddress = "eNodeB21%x2ppp0"
57 *.eNodeB0.x2App[21].client.connectAddress = "eNodeB22%x2ppp0"
58 *.eNodeB0.x2App[22].client.connectAddress = "eNodeB23%x2ppp0"
59 #enodeB1#####
60 *.eNodeB1.x2App[0].client.connectAddress = "eNodeB0%x2ppp0"
61 *.eNodeB1.x2App[1].client.connectAddress = "eNodeB2%x2ppp1"
62 *.eNodeB1.x2App[2].client.connectAddress = "eNodeB3%x2ppp1"
63 *.eNodeB1.x2App[3].client.connectAddress = "eNodeB4%x2ppp1"
64 *.eNodeB1.x2App[4].client.connectAddress = "eNodeB5%x2ppp1"
65 *.eNodeB1.x2App[5].client.connectAddress = "eNodeB6%x2ppp1"
66 *.eNodeB1.x2App[6].client.connectAddress = "eNodeB7%x2ppp1"
67 *.eNodeB1.x2App[7].client.connectAddress = "eNodeB8%x2ppp1"
68 *.eNodeB1.x2App[8].client.connectAddress = "eNodeB9%x2ppp1"
69 *.eNodeB1.x2App[9].client.connectAddress = "eNodeB10%x2ppp1"
70 *.eNodeB1.x2App[10].client.connectAddress = "eNodeB11%x2ppp1"
71 *.eNodeB1.x2App[11].client.connectAddress = "eNodeB12%x2ppp1"
72 *.eNodeB1.x2App[12].client.connectAddress = "eNodeB13%x2ppp1"
73 *.eNodeB1.x2App[13].client.connectAddress = "eNodeB14%x2ppp1"
74 *.eNodeB1.x2App[14].client.connectAddress = "eNodeB15%x2ppp1"
75 *.eNodeB1.x2App[15].client.connectAddress = "eNodeB16%x2ppp1"
76 *.eNodeB1.x2App[16].client.connectAddress = "eNodeB17%x2ppp1"
77 *.eNodeB1.x2App[17].client.connectAddress = "eNodeB18%x2ppp1"
78 *.eNodeB1.x2App[18].client.connectAddress = "eNodeB19%x2ppp1"
79 *.eNodeB1.x2App[19].client.connectAddress = "eNodeB20%x2ppp1"
80 *.eNodeB1.x2App[20].client.connectAddress = "eNodeB21%x2ppp1"
81 *.eNodeB1.x2App[21].client.connectAddress = "eNodeB22%x2ppp1"
```

```
82 *.eNodeB1.x2App[22].client.connectAddress = "eNodeB23%x2ppp1 "  
83 #enodeB2#####  
84 *.eNodeB2.x2App[0].client.connectAddress = "eNodeB0%x2ppp1 "  
85 *.eNodeB2.x2App[1].client.connectAddress = "eNodeB1%x2ppp1 "  
86 *.eNodeB2.x2App[2].client.connectAddress = "eNodeB3%x2ppp2 "  
87 *.eNodeB2.x2App[3].client.connectAddress = "eNodeB4%x2ppp2 "  
88 *.eNodeB2.x2App[4].client.connectAddress = "eNodeB5%x2ppp2 "  
89 *.eNodeB2.x2App[5].client.connectAddress = "eNodeB6%x2ppp2 "  
90 *.eNodeB2.x2App[6].client.connectAddress = "eNodeB7%x2ppp2 "  
91 *.eNodeB2.x2App[7].client.connectAddress = "eNodeB8%x2ppp2 "  
92 *.eNodeB2.x2App[8].client.connectAddress = "eNodeB9%x2ppp2 "  
93 *.eNodeB2.x2App[9].client.connectAddress = "eNodeB10%x2ppp2 "  
94 *.eNodeB2.x2App[10].client.connectAddress = "eNodeB11%x2ppp2 "  
95 *.eNodeB2.x2App[11].client.connectAddress = "eNodeB12%x2ppp2 "  
96 *.eNodeB2.x2App[12].client.connectAddress = "eNodeB13%x2ppp2 "  
97 *.eNodeB2.x2App[13].client.connectAddress = "eNodeB14%x2ppp2 "  
98 *.eNodeB2.x2App[14].client.connectAddress = "eNodeB15%x2ppp2 "  
99 *.eNodeB2.x2App[15].client.connectAddress = "eNodeB16%x2ppp2 "  
100 *.eNodeB2.x2App[16].client.connectAddress = "eNodeB17%x2ppp2 "  
101 *.eNodeB2.x2App[17].client.connectAddress = "eNodeB18%x2ppp2 "  
102 *.eNodeB2.x2App[18].client.connectAddress = "eNodeB19%x2ppp2 "  
103 *.eNodeB2.x2App[19].client.connectAddress = "eNodeB20%x2ppp2 "  
104 *.eNodeB2.x2App[20].client.connectAddress = "eNodeB21%x2ppp2 "  
105 *.eNodeB2.x2App[21].client.connectAddress = "eNodeB22%x2ppp2 "  
106 *.eNodeB2.x2App[22].client.connectAddress = "eNodeB23%x2ppp2 "  
107 #enodeB3#####  
108 *.eNodeB3.x2App[0].client.connectAddress = "eNodeB0%x2ppp2 "  
109 *.eNodeB3.x2App[1].client.connectAddress = "eNodeB1%x2ppp2 "  
110 *.eNodeB3.x2App[2].client.connectAddress = "eNodeB2%x2ppp2 "  
111 *.eNodeB3.x2App[3].client.connectAddress = "eNodeB4%x2ppp3 "  
112 *.eNodeB3.x2App[4].client.connectAddress = "eNodeB5%x2ppp3 "  
113 *.eNodeB3.x2App[5].client.connectAddress = "eNodeB6%x2ppp3 "  
114 *.eNodeB3.x2App[6].client.connectAddress = "eNodeB7%x2ppp3 "  
115 *.eNodeB3.x2App[7].client.connectAddress = "eNodeB8%x2ppp3 "  
116 *.eNodeB3.x2App[8].client.connectAddress = "eNodeB9%x2ppp3 "  
117 *.eNodeB3.x2App[9].client.connectAddress = "eNodeB10%x2ppp3 "  
118 *.eNodeB3.x2App[10].client.connectAddress = "eNodeB11%x2ppp3 "  
119 *.eNodeB3.x2App[11].client.connectAddress = "eNodeB12%x2ppp3 "  
120 *.eNodeB3.x2App[12].client.connectAddress = "eNodeB13%x2ppp3 "  
121 *.eNodeB3.x2App[13].client.connectAddress = "eNodeB14%x2ppp3 "  
122 *.eNodeB3.x2App[14].client.connectAddress = "eNodeB15%x2ppp3 "  
123 *.eNodeB3.x2App[15].client.connectAddress = "eNodeB16%x2ppp3 "  
124 *.eNodeB3.x2App[16].client.connectAddress = "eNodeB17%x2ppp3 "  
125 *.eNodeB3.x2App[17].client.connectAddress = "eNodeB18%x2ppp3 "  
126 *.eNodeB3.x2App[18].client.connectAddress = "eNodeB19%x2ppp3 "  
127 *.eNodeB3.x2App[19].client.connectAddress = "eNodeB20%x2ppp3 "  
128 *.eNodeB3.x2App[20].client.connectAddress = "eNodeB21%x2ppp3 "  
129 *.eNodeB3.x2App[21].client.connectAddress = "eNodeB22%x2ppp3 "  
130 *.eNodeB3.x2App[22].client.connectAddress = "eNodeB23%x2ppp3 "
```

```
131 #enodeB4#####
132 *.eNodeB4.x2App[0].client.connectAddress = "eNodeB0%x2ppp3"
133 *.eNodeB4.x2App[1].client.connectAddress = "eNodeB1%x2ppp3"
134 *.eNodeB4.x2App[2].client.connectAddress = "eNodeB2%x2ppp3"
135 *.eNodeB4.x2App[3].client.connectAddress = "eNodeB3%x2ppp3"
136 *.eNodeB4.x2App[4].client.connectAddress = "eNodeB5%x2ppp4"
137 *.eNodeB4.x2App[5].client.connectAddress = "eNodeB6%x2ppp4"
138 *.eNodeB4.x2App[6].client.connectAddress = "eNodeB7%x2ppp4"
139 *.eNodeB4.x2App[7].client.connectAddress = "eNodeB8%x2ppp4"
140 *.eNodeB4.x2App[8].client.connectAddress = "eNodeB9%x2ppp4"
141 *.eNodeB4.x2App[9].client.connectAddress = "eNodeB10%x2ppp4"
142 *.eNodeB4.x2App[10].client.connectAddress = "eNodeB11%x2ppp4"
143 *.eNodeB4.x2App[11].client.connectAddress = "eNodeB12%x2ppp4"
144 *.eNodeB4.x2App[12].client.connectAddress = "eNodeB13%x2ppp4"
145 *.eNodeB4.x2App[13].client.connectAddress = "eNodeB14%x2ppp4"
146 *.eNodeB4.x2App[14].client.connectAddress = "eNodeB15%x2ppp4"
147 *.eNodeB4.x2App[15].client.connectAddress = "eNodeB16%x2ppp4"
148 *.eNodeB4.x2App[16].client.connectAddress = "eNodeB17%x2ppp4"
149 *.eNodeB4.x2App[17].client.connectAddress = "eNodeB18%x2ppp4"
150 *.eNodeB4.x2App[18].client.connectAddress = "eNodeB19%x2ppp4"
151 *.eNodeB4.x2App[19].client.connectAddress = "eNodeB20%x2ppp4"
152 *.eNodeB4.x2App[20].client.connectAddress = "eNodeB21%x2ppp4"
153 *.eNodeB4.x2App[21].client.connectAddress = "eNodeB22%x2ppp4"
154 *.eNodeB4.x2App[22].client.connectAddress = "eNodeB23%x2ppp4"
155 #enodeB5#####
156 *.eNodeB5.x2App[0].client.connectAddress = "eNodeB0%x2ppp4"
157 *.eNodeB5.x2App[1].client.connectAddress = "eNodeB1%x2ppp4"
158 *.eNodeB5.x2App[2].client.connectAddress = "eNodeB2%x2ppp4"
159 *.eNodeB5.x2App[3].client.connectAddress = "eNodeB3%x2ppp4"
160 *.eNodeB5.x2App[4].client.connectAddress = "eNodeB4%x2ppp4"
161 *.eNodeB5.x2App[5].client.connectAddress = "eNodeB6%x2ppp5"
162 *.eNodeB5.x2App[6].client.connectAddress = "eNodeB7%x2ppp5"
163 *.eNodeB5.x2App[7].client.connectAddress = "eNodeB8%x2ppp5"
164 *.eNodeB5.x2App[8].client.connectAddress = "eNodeB9%x2ppp5"
165 *.eNodeB5.x2App[9].client.connectAddress = "eNodeB10%x2ppp5"
166 *.eNodeB5.x2App[10].client.connectAddress = "eNodeB11%x2ppp5"
167 *.eNodeB5.x2App[11].client.connectAddress = "eNodeB12%x2ppp5"
168 *.eNodeB5.x2App[12].client.connectAddress = "eNodeB13%x2ppp5"
169 *.eNodeB5.x2App[13].client.connectAddress = "eNodeB14%x2ppp5"
170 *.eNodeB5.x2App[14].client.connectAddress = "eNodeB15%x2ppp5"
171 *.eNodeB5.x2App[15].client.connectAddress = "eNodeB16%x2ppp5"
172 *.eNodeB5.x2App[16].client.connectAddress = "eNodeB17%x2ppp5"
173 *.eNodeB5.x2App[17].client.connectAddress = "eNodeB18%x2ppp5"
174 *.eNodeB5.x2App[18].client.connectAddress = "eNodeB19%x2ppp5"
175 *.eNodeB5.x2App[19].client.connectAddress = "eNodeB20%x2ppp5"
176 *.eNodeB5.x2App[20].client.connectAddress = "eNodeB21%x2ppp5"
177 *.eNodeB5.x2App[21].client.connectAddress = "eNodeB22%x2ppp5"
178 *.eNodeB5.x2App[22].client.connectAddress = "eNodeB23%x2ppp5"
179 #enodeB6#####
```

```
180 *.eNodeB6.x2App[0].client.connectAddress = "eNodeB0%x2ppp5" *.
      eNodeB6.x2App[1].client.connectAddress = "eNodeB1%x2ppp5"
181 *.eNodeB6.x2App[2].client.connectAddress = "eNodeB2%x2ppp5"
182 *.eNodeB6.x2App[3].client.connectAddress = "eNodeB3%x2ppp5"
183 *.eNodeB6.x2App[4].client.connectAddress = "eNodeB4%x2ppp5"
184 *.eNodeB6.x2App[5].client.connectAddress = "eNodeB5%x2ppp5"
185 *.eNodeB6.x2App[6].client.connectAddress = "eNodeB7%x2ppp6"
186 *.eNodeB6.x2App[7].client.connectAddress = "eNodeB8%x2ppp6"
187 *.eNodeB6.x2App[8].client.connectAddress = "eNodeB9%x2ppp6"
188 *.eNodeB6.x2App[9].client.connectAddress = "eNodeB10%x2ppp6"
189 *.eNodeB6.x2App[10].client.connectAddress = "eNodeB11%x2ppp6"
190 *.eNodeB6.x2App[11].client.connectAddress = "eNodeB12%x2ppp6"
191 *.eNodeB6.x2App[12].client.connectAddress = "eNodeB13%x2ppp6"
192 *.eNodeB6.x2App[13].client.connectAddress = "eNodeB14%x2ppp6"
193 *.eNodeB6.x2App[14].client.connectAddress = "eNodeB15%x2ppp6"
194 *.eNodeB6.x2App[15].client.connectAddress = "eNodeB16%x2ppp6"
195 *.eNodeB6.x2App[16].client.connectAddress = "eNodeB17%x2ppp6"
196 *.eNodeB6.x2App[17].client.connectAddress = "eNodeB18%x2ppp6"
197 *.eNodeB6.x2App[18].client.connectAddress = "eNodeB19%x2ppp6"
198 *.eNodeB6.x2App[19].client.connectAddress = "eNodeB20%x2ppp6"
199 *.eNodeB6.x2App[20].client.connectAddress = "eNodeB21%x2ppp6"
200 *.eNodeB6.x2App[21].client.connectAddress = "eNodeB22%x2ppp6"
201 *.eNodeB6.x2App[22].client.connectAddress = "eNodeB23%x2ppp6"
202 #enodeB7#####
203 *.eNodeB7.x2App[0].client.connectAddress = "eNodeB0%x2ppp6"
204 *.eNodeB7.x2App[1].client.connectAddress = "eNodeB1%x2ppp6"
205 *.eNodeB7.x2App[2].client.connectAddress = "eNodeB2%x2ppp6"
206 *.eNodeB7.x2App[3].client.connectAddress = "eNodeB3%x2ppp6"
207 *.eNodeB7.x2App[4].client.connectAddress = "eNodeB4%x2ppp6"
208 *.eNodeB7.x2App[5].client.connectAddress = "eNodeB5%x2ppp6"
209 *.eNodeB7.x2App[6].client.connectAddress = "eNodeB6%x2ppp6"
210 *.eNodeB7.x2App[7].client.connectAddress = "eNodeB8%x2ppp7"
211 *.eNodeB7.x2App[8].client.connectAddress = "eNodeB9%x2ppp7"
212 *.eNodeB7.x2App[9].client.connectAddress = "eNodeB10%x2ppp7"
213 *.eNodeB7.x2App[10].client.connectAddress = "eNodeB11%x2ppp7"
214 *.eNodeB7.x2App[11].client.connectAddress = "eNodeB12%x2ppp7"
215 *.eNodeB7.x2App[12].client.connectAddress = "eNodeB13%x2ppp7"
216 *.eNodeB7.x2App[13].client.connectAddress = "eNodeB14%x2ppp7"
217 *.eNodeB7.x2App[14].client.connectAddress = "eNodeB15%x2ppp7"
218 *.eNodeB7.x2App[15].client.connectAddress = "eNodeB16%x2ppp7"
219 *.eNodeB7.x2App[16].client.connectAddress = "eNodeB17%x2ppp7"
220 *.eNodeB7.x2App[17].client.connectAddress = "eNodeB18%x2ppp7"
221 *.eNodeB7.x2App[18].client.connectAddress = "eNodeB19%x2ppp7"
222 *.eNodeB7.x2App[19].client.connectAddress = "eNodeB20%x2ppp7"
223 *.eNodeB7.x2App[20].client.connectAddress = "eNodeB21%x2ppp7"
224 *.eNodeB7.x2App[21].client.connectAddress = "eNodeB22%x2ppp7"
225 *.eNodeB7.x2App[22].client.connectAddress = "eNodeB23%x2ppp7"
226 #enodeB8#####
227 *.eNodeB8.x2App[0].client.connectAddress = "eNodeB0%x2ppp7"
```

```
228 *.eNodeB8.x2App[1].client.connectAddress = "eNodeB1%x2ppp7"
229 *.eNodeB8.x2App[2].client.connectAddress = "eNodeB2%x2ppp7"
230 *.eNodeB8.x2App[3].client.connectAddress = "eNodeB3%x2ppp7"
231 *.eNodeB8.x2App[4].client.connectAddress = "eNodeB4%x2ppp7"
232 *.eNodeB8.x2App[5].client.connectAddress = "eNodeB5%x2ppp7"
233 *.eNodeB8.x2App[6].client.connectAddress = "eNodeB6%x2ppp7"
234 *.eNodeB8.x2App[7].client.connectAddress = "eNodeB7%x2ppp7"
235 *.eNodeB8.x2App[8].client.connectAddress = "eNodeB9%x2ppp8"
236 *.eNodeB8.x2App[9].client.connectAddress = "eNodeB10%x2ppp8"
237 *.eNodeB8.x2App[10].client.connectAddress = "eNodeB11%x2ppp8"
238 *.eNodeB8.x2App[11].client.connectAddress = "eNodeB12%x2ppp8"
239 *.eNodeB8.x2App[12].client.connectAddress = "eNodeB13%x2ppp8"
240 *.eNodeB8.x2App[13].client.connectAddress = "eNodeB14%x2ppp8"
241 *.eNodeB8.x2App[14].client.connectAddress = "eNodeB15%x2ppp8"
242 *.eNodeB8.x2App[15].client.connectAddress = "eNodeB16%x2ppp8"
243 *.eNodeB8.x2App[16].client.connectAddress = "eNodeB17%x2ppp8"
244 *.eNodeB8.x2App[17].client.connectAddress = "eNodeB18%x2ppp8"
245 *.eNodeB8.x2App[18].client.connectAddress = "eNodeB19%x2ppp8"
246 *.eNodeB8.x2App[19].client.connectAddress = "eNodeB20%x2ppp8"
247 *.eNodeB8.x2App[20].client.connectAddress = "eNodeB21%x2ppp8"
248 *.eNodeB8.x2App[21].client.connectAddress = "eNodeB22%x2ppp8"
249 *.eNodeB8.x2App[22].client.connectAddress = "eNodeB23%x2ppp8"
250 #enodeB9#####
251 *.eNodeB9.x2App[0].client.connectAddress = "eNodeB0%x2ppp8"
252 *.eNodeB9.x2App[1].client.connectAddress = "eNodeB1%x2ppp8"
253 *.eNodeB9.x2App[2].client.connectAddress = "eNodeB2%x2ppp8"
254 *.eNodeB9.x2App[3].client.connectAddress = "eNodeB3%x2ppp8"
255 *.eNodeB9.x2App[4].client.connectAddress = "eNodeB4%x2ppp8"
256 *.eNodeB9.x2App[5].client.connectAddress = "eNodeB5%x2ppp8"
257 *.eNodeB9.x2App[6].client.connectAddress = "eNodeB6%x2ppp8"
258 *.eNodeB9.x2App[7].client.connectAddress = "eNodeB7%x2ppp8"
259 *.eNodeB9.x2App[8].client.connectAddress = "eNodeB8%x2ppp8"
260 *.eNodeB9.x2App[9].client.connectAddress = "eNodeB10%x2ppp9"
261 *.eNodeB9.x2App[10].client.connectAddress = "eNodeB11%x2ppp9"
262 *.eNodeB9.x2App[11].client.connectAddress = "eNodeB12%x2ppp9"
263 *.eNodeB9.x2App[12].client.connectAddress = "eNodeB13%x2ppp9"
264 *.eNodeB9.x2App[13].client.connectAddress = "eNodeB14%x2ppp9"
265 *.eNodeB9.x2App[14].client.connectAddress = "eNodeB15%x2ppp9"
266 *.eNodeB9.x2App[15].client.connectAddress = "eNodeB16%x2ppp9"
267 *.eNodeB9.x2App[16].client.connectAddress = "eNodeB17%x2ppp9"
268 *.eNodeB9.x2App[17].client.connectAddress = "eNodeB18%x2ppp9"
269 *.eNodeB9.x2App[18].client.connectAddress = "eNodeB19%x2ppp9"
270 *.eNodeB9.x2App[19].client.connectAddress = "eNodeB20%x2ppp9"
271 *.eNodeB9.x2App[20].client.connectAddress = "eNodeB21%x2ppp9"
272 *.eNodeB9.x2App[21].client.connectAddress = "eNodeB22%x2ppp9"
273 *.eNodeB9.x2App[22].client.connectAddress = "eNodeB23%x2ppp9"
274 #enodeB10#####
275 *.eNodeB10.x2App[0].client.connectAddress = "eNodeB0%x2ppp9"
276 *.eNodeB10.x2App[1].client.connectAddress = "eNodeB1%x2ppp9"
```

```
277 *.eNodeB10.x2App[2].client.connectAddress = "eNodeB2%x2ppp9"
278 *.eNodeB10.x2App[3].client.connectAddress = "eNodeB3%x2ppp9"
279 *.eNodeB10.x2App[4].client.connectAddress = "eNodeB4%x2ppp9"
280 *.eNodeB10.x2App[5].client.connectAddress = "eNodeB5%x2ppp9"
281 *.eNodeB10.x2App[6].client.connectAddress = "eNodeB6%x2ppp9"
282 *.eNodeB10.x2App[7].client.connectAddress = "eNodeB7%x2ppp9"
283 *.eNodeB10.x2App[8].client.connectAddress = "eNodeB8%x2ppp9"
284 *.eNodeB10.x2App[9].client.connectAddress = "eNodeB9%x2ppp9"
285 *.eNodeB10.x2App[10].client.connectAddress = "eNodeB11%x2ppp10"
286 *.eNodeB10.x2App[11].client.connectAddress = "eNodeB12%x2ppp10"
287 *.eNodeB10.x2App[12].client.connectAddress = "eNodeB13%x2ppp10"
288 *.eNodeB10.x2App[13].client.connectAddress = "eNodeB14%x2ppp10"
289 *.eNodeB10.x2App[14].client.connectAddress = "eNodeB15%x2ppp10"
290 *.eNodeB10.x2App[15].client.connectAddress = "eNodeB16%x2ppp10"
291 *.eNodeB10.x2App[16].client.connectAddress = "eNodeB17%x2ppp10"
292 *.eNodeB10.x2App[17].client.connectAddress = "eNodeB18%x2ppp10"
293 *.eNodeB10.x2App[18].client.connectAddress = "eNodeB19%x2ppp10"
294 *.eNodeB10.x2App[19].client.connectAddress = "eNodeB20%x2ppp10"
295 *.eNodeB10.x2App[20].client.connectAddress = "eNodeB21%x2ppp10"
296 *.eNodeB10.x2App[21].client.connectAddress = "eNodeB22%x2ppp10"
297 *.eNodeB10.x2App[22].client.connectAddress = "eNodeB23%x2ppp10"
298 #enodeB11#####
299 *.eNodeB11.x2App[0].client.connectAddress = "eNodeB0%x2ppp10"
300 *.eNodeB11.x2App[1].client.connectAddress = "eNodeB1%x2ppp10"
301 *.eNodeB11.x2App[2].client.connectAddress = "eNodeB2%x2ppp10"
302 *.eNodeB11.x2App[3].client.connectAddress = "eNodeB3%x2ppp10"
303 *.eNodeB11.x2App[4].client.connectAddress = "eNodeB4%x2ppp10"
304 *.eNodeB11.x2App[5].client.connectAddress = "eNodeB5%x2ppp10"
305 *.eNodeB11.x2App[6].client.connectAddress = "eNodeB6%x2ppp10"
306 *.eNodeB11.x2App[7].client.connectAddress = "eNodeB7%x2ppp10"
307 *.eNodeB11.x2App[8].client.connectAddress = "eNodeB8%x2ppp10"
308 *.eNodeB11.x2App[9].client.connectAddress = "eNodeB9%x2ppp10"
309 *.eNodeB11.x2App[10].client.connectAddress = "eNodeB10%x2ppp10"
310 *.eNodeB11.x2App[11].client.connectAddress = "eNodeB12%x2ppp11"
311 *.eNodeB11.x2App[12].client.connectAddress = "eNodeB13%x2ppp11"
312 *.eNodeB11.x2App[13].client.connectAddress = "eNodeB14%x2ppp11"
313 *.eNodeB11.x2App[14].client.connectAddress = "eNodeB15%x2ppp11"
314 *.eNodeB11.x2App[15].client.connectAddress = "eNodeB16%x2ppp11"
315 *.eNodeB11.x2App[16].client.connectAddress = "eNodeB17%x2ppp11"
316 *.eNodeB11.x2App[17].client.connectAddress = "eNodeB18%x2ppp11"
317 *.eNodeB11.x2App[18].client.connectAddress = "eNodeB19%x2ppp11"
318 *.eNodeB11.x2App[19].client.connectAddress = "eNodeB20%x2ppp11"
319 *.eNodeB11.x2App[20].client.connectAddress = "eNodeB21%x2ppp11"
320 *.eNodeB11.x2App[21].client.connectAddress = "eNodeB22%x2ppp11"
321 *.eNodeB11.x2App[22].client.connectAddress = "eNodeB23%x2ppp11"
322 #enodeB12#####
323 *.eNodeB12.x2App[0].client.connectAddress = "eNodeB0%x2ppp11"
324 *.eNodeB12.x2App[1].client.connectAddress = "eNodeB1%x2ppp11"
325 *.eNodeB12.x2App[2].client.connectAddress = "eNodeB2%x2ppp11"
```

```
326 *.eNodeB12.x2App[3].client.connectAddress = "eNodeB3%x2ppp11"
327 *.eNodeB12.x2App[4].client.connectAddress = "eNodeB4%x2ppp11"
328 *.eNodeB12.x2App[5].client.connectAddress = "eNodeB5%x2ppp11"
329 *.eNodeB12.x2App[6].client.connectAddress = "eNodeB6%x2ppp11"
330 *.eNodeB12.x2App[7].client.connectAddress = "eNodeB7%x2ppp11"
331 *.eNodeB12.x2App[8].client.connectAddress = "eNodeB8%x2ppp11"
332 *.eNodeB12.x2App[9].client.connectAddress = "eNodeB9%x2ppp11"
333 *.eNodeB12.x2App[10].client.connectAddress = "eNodeB10%x2ppp11"
334 *.eNodeB12.x2App[11].client.connectAddress = "eNodeB11%x2ppp11"
335 *.eNodeB12.x2App[12].client.connectAddress = "eNodeB13%x2ppp12"
336 *.eNodeB12.x2App[13].client.connectAddress = "eNodeB14%x2ppp12"
337 *.eNodeB12.x2App[14].client.connectAddress = "eNodeB15%x2ppp12"
338 *.eNodeB12.x2App[15].client.connectAddress = "eNodeB16%x2ppp12"
339 *.eNodeB12.x2App[16].client.connectAddress = "eNodeB17%x2ppp12"
340 *.eNodeB12.x2App[17].client.connectAddress = "eNodeB18%x2ppp12"
341 *.eNodeB12.x2App[18].client.connectAddress = "eNodeB19%x2ppp12"
342 *.eNodeB12.x2App[19].client.connectAddress = "eNodeB20%x2ppp12"
343 *.eNodeB12.x2App[20].client.connectAddress = "eNodeB21%x2ppp12"
344 *.eNodeB12.x2App[21].client.connectAddress = "eNodeB22%x2ppp12"
345 *.eNodeB12.x2App[22].client.connectAddress = "eNodeB23%x2ppp12"
346 #enodeB13#####
347 *.eNodeB13.x2App[0].client.connectAddress = "eNodeB0%x2ppp12"
348 *.eNodeB13.x2App[1].client.connectAddress = "eNodeB1%x2ppp12"
349 *.eNodeB13.x2App[2].client.connectAddress = "eNodeB2%x2ppp12"
350 *.eNodeB13.x2App[3].client.connectAddress = "eNodeB3%x2ppp12"
351 *.eNodeB13.x2App[4].client.connectAddress = "eNodeB4%x2ppp12"
352 *.eNodeB13.x2App[5].client.connectAddress = "eNodeB5%x2ppp12"
353 *.eNodeB13.x2App[6].client.connectAddress = "eNodeB6%x2ppp12"
354 *.eNodeB13.x2App[7].client.connectAddress = "eNodeB7%x2ppp12"
355 *.eNodeB13.x2App[8].client.connectAddress = "eNodeB8%x2ppp12"
356 *.eNodeB13.x2App[9].client.connectAddress = "eNodeB9%x2ppp12"
357 *.eNodeB13.x2App[10].client.connectAddress = "eNodeB10%x2ppp12"
358 *.eNodeB13.x2App[11].client.connectAddress = "eNodeB11%x2ppp12"
359 *.eNodeB13.x2App[12].client.connectAddress = "eNodeB12%x2ppp12"
360 *.eNodeB13.x2App[13].client.connectAddress = "eNodeB14%x2ppp13"
361 *.eNodeB13.x2App[14].client.connectAddress = "eNodeB15%x2ppp13"
362 *.eNodeB13.x2App[15].client.connectAddress = "eNodeB16%x2ppp13"
363 *.eNodeB13.x2App[16].client.connectAddress = "eNodeB17%x2ppp13"
364 *.eNodeB13.x2App[17].client.connectAddress = "eNodeB18%x2ppp13"
365 *.eNodeB13.x2App[18].client.connectAddress = "eNodeB19%x2ppp13"
366 *.eNodeB13.x2App[19].client.connectAddress = "eNodeB20%x2ppp13"
367 *.eNodeB13.x2App[20].client.connectAddress = "eNodeB21%x2ppp13"
368 *.eNodeB13.x2App[21].client.connectAddress = "eNodeB22%x2ppp13"
369 *.eNodeB13.x2App[22].client.connectAddress = "eNodeB23%x2ppp13"
370 #enodeB14#####
371 *.eNodeB14.x2App[0].client.connectAddress = "eNodeB0%x2ppp13"
372 *.eNodeB14.x2App[1].client.connectAddress = "eNodeB1%x2ppp13"
373 *.eNodeB14.x2App[2].client.connectAddress = "eNodeB2%x2ppp13"
374 *.eNodeB14.x2App[3].client.connectAddress = "eNodeB3%x2ppp13"
```

```
375 *.eNodeB14.x2App[4].client.connectAddress = "eNodeB4%x2ppp13"
376 *.eNodeB14.x2App[5].client.connectAddress = "eNodeB5%x2ppp13"
377 *.eNodeB14.x2App[6].client.connectAddress = "eNodeB6%x2ppp13"
378 *.eNodeB14.x2App[7].client.connectAddress = "eNodeB7%x2ppp13"
379 *.eNodeB14.x2App[8].client.connectAddress = "eNodeB8%x2ppp13"
380 *.eNodeB14.x2App[9].client.connectAddress = "eNodeB9%x2ppp13"
381 *.eNodeB14.x2App[10].client.connectAddress = "eNodeB10%x2ppp13"
382 *.eNodeB14.x2App[11].client.connectAddress = "eNodeB11%x2ppp13"
383 *.eNodeB14.x2App[12].client.connectAddress = "eNodeB12%x2ppp13"
384 *.eNodeB14.x2App[13].client.connectAddress = "eNodeB13%x2ppp13"
385 *.eNodeB14.x2App[14].client.connectAddress = "eNodeB15%x2ppp14"
386 *.eNodeB14.x2App[15].client.connectAddress = "eNodeB16%x2ppp14"
387 *.eNodeB14.x2App[16].client.connectAddress = "eNodeB17%x2ppp14"
388 *.eNodeB14.x2App[17].client.connectAddress = "eNodeB18%x2ppp14"
389 *.eNodeB14.x2App[18].client.connectAddress = "eNodeB19%x2ppp14"
390 *.eNodeB14.x2App[19].client.connectAddress = "eNodeB20%x2ppp14"
391 *.eNodeB14.x2App[20].client.connectAddress = "eNodeB21%x2ppp14"
392 *.eNodeB14.x2App[21].client.connectAddress = "eNodeB22%x2ppp14"
393 *.eNodeB14.x2App[22].client.connectAddress = "eNodeB23%x2ppp14"
394 #enodeB15#####
395 *.eNodeB15.x2App[0].client.connectAddress = "eNodeB0%x2ppp14"
396 *.eNodeB15.x2App[1].client.connectAddress = "eNodeB1%x2ppp14"
397 *.eNodeB15.x2App[2].client.connectAddress = "eNodeB2%x2ppp14"
398 *.eNodeB15.x2App[3].client.connectAddress = "eNodeB3%x2ppp14"
399 *.eNodeB15.x2App[4].client.connectAddress = "eNodeB4%x2ppp14"
400 *.eNodeB15.x2App[5].client.connectAddress = "eNodeB5%x2ppp14"
401 *.eNodeB15.x2App[6].client.connectAddress = "eNodeB6%x2ppp14"
402 *.eNodeB15.x2App[7].client.connectAddress = "eNodeB7%x2ppp14"
403 *.eNodeB15.x2App[8].client.connectAddress = "eNodeB8%x2ppp14"
404 *.eNodeB15.x2App[9].client.connectAddress = "eNodeB9%x2ppp14"
405 *.eNodeB15.x2App[10].client.connectAddress = "eNodeB10%x2ppp14"
406 *.eNodeB15.x2App[11].client.connectAddress = "eNodeB11%x2ppp14"
407 *.eNodeB15.x2App[12].client.connectAddress = "eNodeB12%x2ppp14"
408 *.eNodeB15.x2App[13].client.connectAddress = "eNodeB13%x2ppp14"
409 *.eNodeB15.x2App[14].client.connectAddress = "eNodeB14%x2ppp14"
410 *.eNodeB15.x2App[15].client.connectAddress = "eNodeB16%x2ppp15"
411 *.eNodeB15.x2App[16].client.connectAddress = "eNodeB17%x2ppp15"
412 *.eNodeB15.x2App[17].client.connectAddress = "eNodeB18%x2ppp15"
413 *.eNodeB15.x2App[18].client.connectAddress = "eNodeB19%x2ppp15"
414 *.eNodeB15.x2App[19].client.connectAddress = "eNodeB20%x2ppp15"
415 *.eNodeB15.x2App[20].client.connectAddress = "eNodeB21%x2ppp15"
416 *.eNodeB15.x2App[21].client.connectAddress = "eNodeB22%x2ppp15"
417 *.eNodeB15.x2App[22].client.connectAddress = "eNodeB23%x2ppp15"
418 #enodeB16#####
419 *.eNodeB16.x2App[0].client.connectAddress = "eNodeB0%x2ppp15"
420 *.eNodeB16.x2App[1].client.connectAddress = "eNodeB1%x2ppp15"
421 *.eNodeB16.x2App[2].client.connectAddress = "eNodeB2%x2ppp15"
422 *.eNodeB16.x2App[3].client.connectAddress = "eNodeB3%x2ppp15"
423 *.eNodeB16.x2App[4].client.connectAddress = "eNodeB4%x2ppp15"
```

```
424 *.eNodeB16.x2App[5].client.connectAddress = "eNodeB5%x2ppp15"
425 *.eNodeB16.x2App[6].client.connectAddress = "eNodeB6%x2ppp15"
426 *.eNodeB16.x2App[7].client.connectAddress = "eNodeB7%x2ppp15"
427 *.eNodeB16.x2App[8].client.connectAddress = "eNodeB8%x2ppp15"
428 *.eNodeB16.x2App[9].client.connectAddress = "eNodeB9%x2ppp15"
429 *.eNodeB16.x2App[10].client.connectAddress = "eNodeB10%x2ppp15"
430 *.eNodeB16.x2App[11].client.connectAddress = "eNodeB11%x2ppp15"
431 *.eNodeB16.x2App[12].client.connectAddress = "eNodeB12%x2ppp15"
432 *.eNodeB16.x2App[13].client.connectAddress = "eNodeB13%x2ppp15"
433 *.eNodeB16.x2App[14].client.connectAddress = "eNodeB14%x2ppp15"
434 *.eNodeB16.x2App[15].client.connectAddress = "eNodeB15%x2ppp15"
435 *.eNodeB16.x2App[16].client.connectAddress = "eNodeB17%x2ppp15"
436 *.eNodeB16.x2App[17].client.connectAddress = "eNodeB18%x2ppp16"
437 *.eNodeB16.x2App[18].client.connectAddress = "eNodeB19%x2ppp16"
438 *.eNodeB16.x2App[19].client.connectAddress = "eNodeB20%x2ppp16"
439 *.eNodeB16.x2App[20].client.connectAddress = "eNodeB21%x2ppp16"
440 *.eNodeB16.x2App[21].client.connectAddress = "eNodeB22%x2ppp16"
441 *.eNodeB16.x2App[22].client.connectAddress = "eNodeB23%x2ppp16"
442 #enodeB17#####
443 *.eNodeB17.x2App[0].client.connectAddress = "eNodeB0%x2ppp16"
444 *.eNodeB17.x2App[1].client.connectAddress = "eNodeB1%x2ppp16"
445 *.eNodeB17.x2App[2].client.connectAddress = "eNodeB2%x2ppp16"
446 *.eNodeB17.x2App[3].client.connectAddress = "eNodeB3%x2ppp16"
447 *.eNodeB17.x2App[4].client.connectAddress = "eNodeB4%x2ppp16"
448 *.eNodeB17.x2App[5].client.connectAddress = "eNodeB5%x2ppp16"
449 *.eNodeB17.x2App[6].client.connectAddress = "eNodeB6%x2ppp16"
450 *.eNodeB17.x2App[7].client.connectAddress = "eNodeB7%x2ppp16"
451 *.eNodeB17.x2App[8].client.connectAddress = "eNodeB8%x2ppp16"
452 *.eNodeB17.x2App[9].client.connectAddress = "eNodeB9%x2ppp16"
453 *.eNodeB17.x2App[10].client.connectAddress = "eNodeB10%x2ppp16"
454 *.eNodeB17.x2App[11].client.connectAddress = "eNodeB11%x2ppp16"
455 *.eNodeB17.x2App[12].client.connectAddress = "eNodeB12%x2ppp16"
456 *.eNodeB17.x2App[13].client.connectAddress = "eNodeB13%x2ppp16"
457 *.eNodeB17.x2App[14].client.connectAddress = "eNodeB14%x2ppp16"
458 *.eNodeB17.x2App[15].client.connectAddress = "eNodeB15%x2ppp16"
459 *.eNodeB17.x2App[16].client.connectAddress = "eNodeB16%x2ppp16"
460 *.eNodeB17.x2App[17].client.connectAddress = "eNodeB18%x2ppp17"
461 *.eNodeB17.x2App[18].client.connectAddress = "eNodeB19%x2ppp17"
462 *.eNodeB17.x2App[19].client.connectAddress = "eNodeB20%x2ppp17"
463 *.eNodeB17.x2App[20].client.connectAddress = "eNodeB21%x2ppp17"
464 *.eNodeB17.x2App[21].client.connectAddress = "eNodeB22%x2ppp17"
465 *.eNodeB17.x2App[22].client.connectAddress = "eNodeB23%x2ppp17"
466 #enodeB18#####
467 *.eNodeB18.x2App[0].client.connectAddress = "eNodeB0%x2ppp17"
468 *.eNodeB18.x2App[1].client.connectAddress = "eNodeB1%x2ppp17"
469 *.eNodeB18.x2App[2].client.connectAddress = "eNodeB2%x2ppp17"
470 *.eNodeB18.x2App[3].client.connectAddress = "eNodeB3%x2ppp17"
471 *.eNodeB18.x2App[4].client.connectAddress = "eNodeB4%x2ppp17"
472 *.eNodeB18.x2App[5].client.connectAddress = "eNodeB5%x2ppp17"
```

```
473 *.eNodeB18.x2App[6].client.connectAddress = "eNodeB6%x2ppp17"
474 *.eNodeB18.x2App[7].client.connectAddress = "eNodeB7%x2ppp17"
475 *.eNodeB18.x2App[8].client.connectAddress = "eNodeB8%x2ppp17"
476 *.eNodeB18.x2App[9].client.connectAddress = "eNodeB9%x2ppp17"
477 *.eNodeB18.x2App[10].client.connectAddress = "eNodeB10%x2ppp17"
478 *.eNodeB18.x2App[11].client.connectAddress = "eNodeB11%x2ppp17"
479 *.eNodeB18.x2App[12].client.connectAddress = "eNodeB12%x2ppp17"
480 *.eNodeB18.x2App[13].client.connectAddress = "eNodeB13%x2ppp17"
481 *.eNodeB18.x2App[14].client.connectAddress = "eNodeB14%x2ppp17"
482 *.eNodeB18.x2App[15].client.connectAddress = "eNodeB15%x2ppp17"
483 *.eNodeB18.x2App[16].client.connectAddress = "eNodeB16%x2ppp17"
484 *.eNodeB18.x2App[17].client.connectAddress = "eNodeB17%x2ppp17"
485 *.eNodeB18.x2App[18].client.connectAddress = "eNodeB19%x2ppp18"
486 *.eNodeB18.x2App[19].client.connectAddress = "eNodeB20%x2ppp18"
487 *.eNodeB18.x2App[20].client.connectAddress = "eNodeB21%x2ppp18"
488 *.eNodeB18.x2App[21].client.connectAddress = "eNodeB22%x2ppp18"
489 *.eNodeB18.x2App[22].client.connectAddress = "eNodeB23%x2ppp18"
490 #enodeB19#####
491 *.eNodeB19.x2App[0].client.connectAddress = "eNodeB0%x2ppp18"
492 *.eNodeB19.x2App[1].client.connectAddress = "eNodeB1%x2ppp18"
493 *.eNodeB19.x2App[2].client.connectAddress = "eNodeB2%x2ppp18"
494 *.eNodeB19.x2App[3].client.connectAddress = "eNodeB3%x2ppp18"
495 *.eNodeB19.x2App[4].client.connectAddress = "eNodeB4%x2ppp18"
496 *.eNodeB19.x2App[5].client.connectAddress = "eNodeB5%x2ppp18"
497 *.eNodeB19.x2App[6].client.connectAddress = "eNodeB6%x2ppp18"
498 *.eNodeB19.x2App[7].client.connectAddress = "eNodeB7%x2ppp18"
499 *.eNodeB19.x2App[8].client.connectAddress = "eNodeB8%x2ppp18"
500 *.eNodeB19.x2App[9].client.connectAddress = "eNodeB9%x2ppp18"
501 *.eNodeB19.x2App[10].client.connectAddress = "eNodeB10%x2ppp18"
502 *.eNodeB19.x2App[11].client.connectAddress = "eNodeB11%x2ppp18"
503 *.eNodeB19.x2App[12].client.connectAddress = "eNodeB12%x2ppp18"
504 *.eNodeB19.x2App[13].client.connectAddress = "eNodeB13%x2ppp18"
505 *.eNodeB19.x2App[14].client.connectAddress = "eNodeB14%x2ppp18"
506 *.eNodeB19.x2App[15].client.connectAddress = "eNodeB15%x2ppp18"
507 *.eNodeB19.x2App[16].client.connectAddress = "eNodeB16%x2ppp18"
508 *.eNodeB19.x2App[17].client.connectAddress = "eNodeB17%x2ppp18"
509 *.eNodeB19.x2App[18].client.connectAddress = "eNodeB18%x2ppp18"
510 *.eNodeB19.x2App[19].client.connectAddress = "eNodeB20%x2ppp19"
511 *.eNodeB19.x2App[20].client.connectAddress = "eNodeB21%x2ppp19"
512 *.eNodeB19.x2App[21].client.connectAddress = "eNodeB22%x2ppp19"
513 *.eNodeB19.x2App[22].client.connectAddress = "eNodeB23%x2ppp19"
514 #enodeB20#####
515 *.eNodeB20.x2App[0].client.connectAddress = "eNodeB0%x2ppp19"
516 *.eNodeB20.x2App[1].client.connectAddress = "eNodeB1%x2ppp19"
517 *.eNodeB20.x2App[2].client.connectAddress = "eNodeB2%x2ppp19"
518 *.eNodeB20.x2App[3].client.connectAddress = "eNodeB3%x2ppp19"
519 *.eNodeB20.x2App[4].client.connectAddress = "eNodeB4%x2ppp19"
520 *.eNodeB20.x2App[5].client.connectAddress = "eNodeB5%x2ppp19"
521 *.eNodeB20.x2App[6].client.connectAddress = "eNodeB6%x2ppp19"
```

```
522 *.eNodeB20.x2App[7].client.connectAddress = "eNodeB7%x2ppp19"
523 *.eNodeB20.x2App[8].client.connectAddress = "eNodeB8%x2ppp19"
524 *.eNodeB20.x2App[9].client.connectAddress = "eNodeB9%x2ppp19"
525 *.eNodeB20.x2App[10].client.connectAddress = "eNodeB10%x2ppp19"
526 *.eNodeB20.x2App[11].client.connectAddress = "eNodeB11%x2ppp19"
527 *.eNodeB20.x2App[12].client.connectAddress = "eNodeB12%x2ppp19"
528 *.eNodeB20.x2App[13].client.connectAddress = "eNodeB13%x2ppp19"
529 *.eNodeB20.x2App[14].client.connectAddress = "eNodeB14%x2ppp19"
530 *.eNodeB20.x2App[15].client.connectAddress = "eNodeB15%x2ppp19"
531 *.eNodeB20.x2App[16].client.connectAddress = "eNodeB16%x2ppp19"
532 *.eNodeB20.x2App[17].client.connectAddress = "eNodeB17%x2ppp19"
533 *.eNodeB20.x2App[18].client.connectAddress = "eNodeB18%x2ppp19"
534 *.eNodeB20.x2App[19].client.connectAddress = "eNodeB19%x2ppp19"
535 *.eNodeB20.x2App[20].client.connectAddress = "eNodeB21%x2ppp20"
536 *.eNodeB20.x2App[21].client.connectAddress = "eNodeB22%x2ppp20"
537 *.eNodeB20.x2App[22].client.connectAddress = "eNodeB23%x2ppp20"
538 #enodeB21#####
539 *.eNodeB21.x2App[0].client.connectAddress = "eNodeB0%x2ppp20"
540 *.eNodeB21.x2App[1].client.connectAddress = "eNodeB1%x2ppp20"
541 *.eNodeB21.x2App[2].client.connectAddress = "eNodeB2%x2ppp20"
542 *.eNodeB21.x2App[3].client.connectAddress = "eNodeB3%x2ppp20"
543 *.eNodeB21.x2App[4].client.connectAddress = "eNodeB4%x2ppp20"
544 *.eNodeB21.x2App[5].client.connectAddress = "eNodeB5%x2ppp20"
545 *.eNodeB21.x2App[6].client.connectAddress = "eNodeB6%x2ppp20"
546 *.eNodeB21.x2App[7].client.connectAddress = "eNodeB7%x2ppp20"
547 *.eNodeB21.x2App[8].client.connectAddress = "eNodeB8%x2ppp20"
548 *.eNodeB21.x2App[9].client.connectAddress = "eNodeB9%x2ppp20"
549 *.eNodeB21.x2App[10].client.connectAddress = "eNodeB10%x2ppp20"
550 *.eNodeB21.x2App[11].client.connectAddress = "eNodeB11%x2ppp20"
551 *.eNodeB21.x2App[12].client.connectAddress = "eNodeB12%x2ppp20"
552 *.eNodeB21.x2App[13].client.connectAddress = "eNodeB13%x2ppp20"
553 *.eNodeB21.x2App[14].client.connectAddress = "eNodeB14%x2ppp20"
554 *.eNodeB21.x2App[15].client.connectAddress = "eNodeB15%x2ppp20"
555 *.eNodeB21.x2App[16].client.connectAddress = "eNodeB16%x2ppp20"
556 *.eNodeB21.x2App[17].client.connectAddress = "eNodeB17%x2ppp20"
557 *.eNodeB21.x2App[18].client.connectAddress = "eNodeB18%x2ppp20"
558 *.eNodeB21.x2App[19].client.connectAddress = "eNodeB19%x2ppp20"
559 *.eNodeB21.x2App[20].client.connectAddress = "eNodeB20%x2ppp20"
560 *.eNodeB21.x2App[21].client.connectAddress = "eNodeB22%x2ppp21"
561 *.eNodeB21.x2App[22].client.connectAddress = "eNodeB23%x2ppp21"
562 #enodeB22#####
563 *.eNodeB22.x2App[0].client.connectAddress = "eNodeB0%x2ppp21"
564 *.eNodeB22.x2App[1].client.connectAddress = "eNodeB1%x2ppp21"
565 *.eNodeB22.x2App[2].client.connectAddress = "eNodeB2%x2ppp21"
566 *.eNodeB22.x2App[3].client.connectAddress = "eNodeB3%x2ppp21"
567 *.eNodeB22.x2App[4].client.connectAddress = "eNodeB4%x2ppp21"
568 *.eNodeB22.x2App[5].client.connectAddress = "eNodeB5%x2ppp21"
569 *.eNodeB22.x2App[6].client.connectAddress = "eNodeB6%x2ppp21"
570 *.eNodeB22.x2App[7].client.connectAddress = "eNodeB7%x2ppp21"
```

```

571 *.eNodeB22.x2App[8].client.connectAddress = "eNodeB8%x2ppp21"
572 *.eNodeB22.x2App[9].client.connectAddress = "eNodeB9%x2ppp21"
573 *.eNodeB22.x2App[10].client.connectAddress = "eNodeB10%x2ppp21"
574 *.eNodeB22.x2App[11].client.connectAddress = "eNodeB11%x2ppp21"
575 *.eNodeB22.x2App[12].client.connectAddress = "eNodeB12%x2ppp21"
576 *.eNodeB22.x2App[13].client.connectAddress = "eNodeB13%x2ppp21"
577 *.eNodeB22.x2App[14].client.connectAddress = "eNodeB14%x2ppp21"
578 *.eNodeB22.x2App[15].client.connectAddress = "eNodeB15%x2ppp21"
579 *.eNodeB22.x2App[16].client.connectAddress = "eNodeB16%x2ppp21"
580 *.eNodeB22.x2App[17].client.connectAddress = "eNodeB17%x2ppp21"
581 *.eNodeB22.x2App[18].client.connectAddress = "eNodeB18%x2ppp21"
582 *.eNodeB22.x2App[19].client.connectAddress = "eNodeB19%x2ppp21"
583 *.eNodeB22.x2App[20].client.connectAddress = "eNodeB20%x2ppp21"
584 *.eNodeB22.x2App[21].client.connectAddress = "eNodeB21%x2ppp21"
585 *.eNodeB22.x2App[22].client.connectAddress = "eNodeB23%x2ppp22"
586 #enodeB23#####
587 *.eNodeB23.x2App[0].client.connectAddress = "eNodeB0%x2ppp22"
588 *.eNodeB23.x2App[1].client.connectAddress = "eNodeB1%x2ppp22"
589 *.eNodeB23.x2App[2].client.connectAddress = "eNodeB2%x2ppp22"
590 *.eNodeB23.x2App[3].client.connectAddress = "eNodeB3%x2ppp22"
591 *.eNodeB23.x2App[4].client.connectAddress = "eNodeB4%x2ppp22"
592 *.eNodeB23.x2App[5].client.connectAddress = "eNodeB5%x2ppp22"
593 *.eNodeB23.x2App[6].client.connectAddress = "eNodeB6%x2ppp22"
594 *.eNodeB23.x2App[7].client.connectAddress = "eNodeB7%x2ppp22"
595 *.eNodeB23.x2App[8].client.connectAddress = "eNodeB8%x2ppp22"
596 *.eNodeB23.x2App[9].client.connectAddress = "eNodeB9%x2ppp22"
597 *.eNodeB23.x2App[10].client.connectAddress = "eNodeB10%x2ppp22"
598 *.eNodeB23.x2App[11].client.connectAddress = "eNodeB11%x2ppp22"
599 *.eNodeB23.x2App[12].client.connectAddress = "eNodeB12%x2ppp22"
600 *.eNodeB23.x2App[13].client.connectAddress = "eNodeB13%x2ppp22"
601 *.eNodeB23.x2App[14].client.connectAddress = "eNodeB14%x2ppp22"
602 *.eNodeB23.x2App[15].client.connectAddress = "eNodeB15%x2ppp22"
603 *.eNodeB23.x2App[16].client.connectAddress = "eNodeB16%x2ppp22"
604 *.eNodeB23.x2App[17].client.connectAddress = "eNodeB17%x2ppp22"
605 *.eNodeB23.x2App[18].client.connectAddress = "eNodeB18%x2ppp22"
606 *.eNodeB23.x2App[19].client.connectAddress = "eNodeB19%x2ppp22"
607 *.eNodeB23.x2App[20].client.connectAddress = "eNodeB20%x2ppp22"
608 *.eNodeB23.x2App[21].client.connectAddress = "eNodeB21%x2ppp22"
609 *.eNodeB23.x2App[22].client.connectAddress = "eNodeB22%x2ppp22"

```

Listing A.7: Codes written in Omnetpp.ini file

```

1 parameters:
2     double playgroundSizeX @unit(m) = 1432m; // x size of the
   area the nodes are in (in meters)
3     double playgroundSizeY @unit(m) = 1432m; // y size of the
   area the nodes are in (in meters)
4     double playgroundSizeZ @unit(m) = 50m; // z size of the
   area the nodes are in (in meters)

```

```

5      @display("bgb=1432,1432;bgg=1432,7,black");
6
7      volatile int ueX = intuniform (0,1432);
8      volatile int ueY = intuniform (0,1432);
9
10     volatile int EnbX = intuniform (0,1432);
11     volatile int EnbY = intuniform (0,1432);
12 submodules:
13     eNodeB0: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
14     eNodeB1: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
15     eNodeB2: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
16     eNodeB3: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
17     eNodeB4: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
18     eNodeB5: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
19     eNodeB6: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
20     eNodeB7: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
21     eNodeB8: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
22     eNodeB9: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
23     eNodeB10: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
24     eNodeB11: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
25     eNodeB12: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
26     eNodeB13: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
27     eNodeB14: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
28     eNodeB15: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
29     eNodeB16: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
30     eNodeB17: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
31     eNodeB18: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
32     eNodeB19: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
33     eNodeB20: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
34     eNodeB21: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
35     eNodeB22: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
36     eNodeB23: eNodeB {@display("p=$EnbX,$EnbY;is=1");}
37     ue[200]: Ue {
38         parameters:
39             @display("p=$ueX,$ueY;is=s");
40     }
41 connections allowunconnected:
42     pgw.pppg++ <--> Eth10G <--> eNodeB0.ppp;
43     pgw.pppg++ <--> Eth10G <--> eNodeB1.ppp;
44     pgw.pppg++ <--> Eth10G <--> eNodeB2.ppp;
45     pgw.pppg++ <--> Eth10G <--> eNodeB3.ppp;
46     pgw.pppg++ <--> Eth10G <--> eNodeB4.ppp;
47     pgw.pppg++ <--> Eth10G <--> eNodeB5.ppp;
48     pgw.pppg++ <--> Eth10G <--> eNodeB6.ppp;
49     pgw.pppg++ <--> Eth10G <--> eNodeB7.ppp;
50     pgw.pppg++ <--> Eth10G <--> eNodeB8.ppp;
51     pgw.pppg++ <--> Eth10G <--> eNodeB9.ppp;
52     pgw.pppg++ <--> Eth10G <--> eNodeB10.ppp;
53     pgw.pppg++ <--> Eth10G <--> eNodeB11.ppp;

```

```
54 pgw.pppg++ <--> Eth10G <--> eNodeB12.ppp;
55 pgw.pppg++ <--> Eth10G <--> eNodeB13.ppp;
56 pgw.pppg++ <--> Eth10G <--> eNodeB14.ppp;
57 pgw.pppg++ <--> Eth10G <--> eNodeB15.ppp;
58 pgw.pppg++ <--> Eth10G <--> eNodeB16.ppp;
59 pgw.pppg++ <--> Eth10G <--> eNodeB17.ppp;
60 pgw.pppg++ <--> Eth10G <--> eNodeB18.ppp;
61 pgw.pppg++ <--> Eth10G <--> eNodeB19.ppp;
62 pgw.pppg++ <--> Eth10G <--> eNodeB20.ppp;
63 pgw.pppg++ <--> Eth10G <--> eNodeB21.ppp;
64 pgw.pppg++ <--> Eth10G <--> eNodeB22.ppp;
65 pgw.pppg++ <--> Eth10G <--> eNodeB23.ppp;
66 //eNodeB0////////////////////////////////////
67 eNodeB0.x2++ <--> Eth10G <--> eNodeB1.x2++;
68 eNodeB0.x2++ <--> Eth10G <--> eNodeB2.x2++;
69 eNodeB0.x2++ <--> Eth10G <--> eNodeB3.x2++;
70 eNodeB0.x2++ <--> Eth10G <--> eNodeB4.x2++;
71 eNodeB0.x2++ <--> Eth10G <--> eNodeB5.x2++;
72 eNodeB0.x2++ <--> Eth10G <--> eNodeB6.x2++;
73 eNodeB0.x2++ <--> Eth10G <--> eNodeB7.x2++;
74 eNodeB0.x2++ <--> Eth10G <--> eNodeB8.x2++;
75 eNodeB0.x2++ <--> Eth10G <--> eNodeB9.x2++;
76 eNodeB0.x2++ <--> Eth10G <--> eNodeB10.x2++;
77 eNodeB0.x2++ <--> Eth10G <--> eNodeB11.x2++;
78 eNodeB0.x2++ <--> Eth10G <--> eNodeB12.x2++;
79 eNodeB0.x2++ <--> Eth10G <--> eNodeB13.x2++;
80 eNodeB0.x2++ <--> Eth10G <--> eNodeB14.x2++;
81 eNodeB0.x2++ <--> Eth10G <--> eNodeB15.x2++;
82 eNodeB0.x2++ <--> Eth10G <--> eNodeB16.x2++;
83 eNodeB0.x2++ <--> Eth10G <--> eNodeB17.x2++;
84 eNodeB0.x2++ <--> Eth10G <--> eNodeB18.x2++;
85 eNodeB0.x2++ <--> Eth10G <--> eNodeB19.x2++;
86 eNodeB0.x2++ <--> Eth10G <--> eNodeB20.x2++;
87 eNodeB0.x2++ <--> Eth10G <--> eNodeB21.x2++;
88 eNodeB0.x2++ <--> Eth10G <--> eNodeB22.x2++;
89 eNodeB0.x2++ <--> Eth10G <--> eNodeB23.x2++;
90 //eNodeB1////////////////////////////////////
91 eNodeB1.x2++ <--> Eth10G <--> eNodeB2.x2++;
92 eNodeB1.x2++ <--> Eth10G <--> eNodeB3.x2++;
93 eNodeB1.x2++ <--> Eth10G <--> eNodeB4.x2++;
94 eNodeB1.x2++ <--> Eth10G <--> eNodeB5.x2++;
95 eNodeB1.x2++ <--> Eth10G <--> eNodeB6.x2++;
96 eNodeB1.x2++ <--> Eth10G <--> eNodeB7.x2++;
97 eNodeB1.x2++ <--> Eth10G <--> eNodeB8.x2++;
98 eNodeB1.x2++ <--> Eth10G <--> eNodeB9.x2++;
99 eNodeB1.x2++ <--> Eth10G <--> eNodeB10.x2++;
100 eNodeB1.x2++ <--> Eth10G <--> eNodeB11.x2++;
101 eNodeB1.x2++ <--> Eth10G <--> eNodeB12.x2++;
102 eNodeB1.x2++ <--> Eth10G <--> eNodeB13.x2++;
```

```
103 eNodeB1.x2++ <--> Eth10G <--> eNodeB14.x2++;
104 eNodeB1.x2++ <--> Eth10G <--> eNodeB15.x2++;
105 eNodeB1.x2++ <--> Eth10G <--> eNodeB16.x2++;
106 eNodeB1.x2++ <--> Eth10G <--> eNodeB17.x2++;
107 eNodeB1.x2++ <--> Eth10G <--> eNodeB18.x2++;
108 eNodeB1.x2++ <--> Eth10G <--> eNodeB19.x2++;
109 eNodeB1.x2++ <--> Eth10G <--> eNodeB20.x2++;
110 eNodeB1.x2++ <--> Eth10G <--> eNodeB21.x2++;
111 eNodeB1.x2++ <--> Eth10G <--> eNodeB22.x2++;
112 eNodeB1.x2++ <--> Eth10G <--> eNodeB23.x2++;
113 //eNodeB2////////////////////////////////////
114 eNodeB2.x2++ <--> Eth10G <--> eNodeB3.x2++;
115 eNodeB2.x2++ <--> Eth10G <--> eNodeB4.x2++;
116 eNodeB2.x2++ <--> Eth10G <--> eNodeB5.x2++;
117 eNodeB2.x2++ <--> Eth10G <--> eNodeB6.x2++;
118 eNodeB2.x2++ <--> Eth10G <--> eNodeB7.x2++;
119 eNodeB2.x2++ <--> Eth10G <--> eNodeB8.x2++;
120 eNodeB2.x2++ <--> Eth10G <--> eNodeB9.x2++;
121 eNodeB2.x2++ <--> Eth10G <--> eNodeB10.x2++;
122 eNodeB2.x2++ <--> Eth10G <--> eNodeB11.x2++;
123 eNodeB2.x2++ <--> Eth10G <--> eNodeB12.x2++;
124 eNodeB2.x2++ <--> Eth10G <--> eNodeB13.x2++;
125 eNodeB2.x2++ <--> Eth10G <--> eNodeB14.x2++;
126 eNodeB2.x2++ <--> Eth10G <--> eNodeB15.x2++;
127 eNodeB2.x2++ <--> Eth10G <--> eNodeB16.x2++;
128 eNodeB2.x2++ <--> Eth10G <--> eNodeB17.x2++;
129 eNodeB2.x2++ <--> Eth10G <--> eNodeB18.x2++;
130 eNodeB2.x2++ <--> Eth10G <--> eNodeB19.x2++;
131 eNodeB2.x2++ <--> Eth10G <--> eNodeB20.x2++;
132 eNodeB2.x2++ <--> Eth10G <--> eNodeB21.x2++;
133 eNodeB2.x2++ <--> Eth10G <--> eNodeB22.x2++;
134 eNodeB2.x2++ <--> Eth10G <--> eNodeB23.x2++;
135 //eNodeB3////////////////////////////////////
136 eNodeB3.x2++ <--> Eth10G <--> eNodeB4.x2++;
137 eNodeB3.x2++ <--> Eth10G <--> eNodeB5.x2++;
138 eNodeB3.x2++ <--> Eth10G <--> eNodeB6.x2++;
139 eNodeB3.x2++ <--> Eth10G <--> eNodeB7.x2++;
140 eNodeB3.x2++ <--> Eth10G <--> eNodeB8.x2++;
141 eNodeB3.x2++ <--> Eth10G <--> eNodeB9.x2++;
142 eNodeB3.x2++ <--> Eth10G <--> eNodeB10.x2++;
143 eNodeB3.x2++ <--> Eth10G <--> eNodeB11.x2++;
144 eNodeB3.x2++ <--> Eth10G <--> eNodeB12.x2++;
145 eNodeB3.x2++ <--> Eth10G <--> eNodeB13.x2++;
146 eNodeB3.x2++ <--> Eth10G <--> eNodeB14.x2++;
147 eNodeB3.x2++ <--> Eth10G <--> eNodeB15.x2++;
148 eNodeB3.x2++ <--> Eth10G <--> eNodeB16.x2++;
149 eNodeB3.x2++ <--> Eth10G <--> eNodeB17.x2++;
150 eNodeB3.x2++ <--> Eth10G <--> eNodeB18.x2++;
151 eNodeB3.x2++ <--> Eth10G <--> eNodeB19.x2++;
```

```
152 eNodeB3.x2++ <--> Eth10G <--> eNodeB20.x2++;
153 eNodeB3.x2++ <--> Eth10G <--> eNodeB21.x2++;
154 eNodeB3.x2++ <--> Eth10G <--> eNodeB22.x2++;
155 eNodeB3.x2++ <--> Eth10G <--> eNodeB23.x2++;
156 //eNodeB4////////////////////////////////////
157 eNodeB4.x2++ <--> Eth10G <--> eNodeB5.x2++;
158 eNodeB4.x2++ <--> Eth10G <--> eNodeB6.x2++;
159 eNodeB4.x2++ <--> Eth10G <--> eNodeB7.x2++;
160 eNodeB4.x2++ <--> Eth10G <--> eNodeB8.x2++;
161 eNodeB4.x2++ <--> Eth10G <--> eNodeB9.x2++;
162 eNodeB4.x2++ <--> Eth10G <--> eNodeB10.x2++;
163 eNodeB4.x2++ <--> Eth10G <--> eNodeB11.x2++;
164 eNodeB4.x2++ <--> Eth10G <--> eNodeB12.x2++;
165 eNodeB4.x2++ <--> Eth10G <--> eNodeB13.x2++;
166 eNodeB4.x2++ <--> Eth10G <--> eNodeB14.x2++;
167 eNodeB4.x2++ <--> Eth10G <--> eNodeB15.x2++;
168 eNodeB4.x2++ <--> Eth10G <--> eNodeB16.x2++;
169 eNodeB4.x2++ <--> Eth10G <--> eNodeB17.x2++;
170 eNodeB4.x2++ <--> Eth10G <--> eNodeB18.x2++;
171 eNodeB4.x2++ <--> Eth10G <--> eNodeB19.x2++;
172 eNodeB4.x2++ <--> Eth10G <--> eNodeB20.x2++;
173 eNodeB4.x2++ <--> Eth10G <--> eNodeB21.x2++;
174 eNodeB4.x2++ <--> Eth10G <--> eNodeB22.x2++;
175 eNodeB4.x2++ <--> Eth10G <--> eNodeB23.x2++;
176 //eNodeB5////////////////////////////////////
177 eNodeB5.x2++ <--> Eth10G <--> eNodeB6.x2++;
178 eNodeB5.x2++ <--> Eth10G <--> eNodeB7.x2++;
179 eNodeB5.x2++ <--> Eth10G <--> eNodeB8.x2++;
180 eNodeB5.x2++ <--> Eth10G <--> eNodeB9.x2++;
181 eNodeB5.x2++ <--> Eth10G <--> eNodeB10.x2++;
182 eNodeB5.x2++ <--> Eth10G <--> eNodeB11.x2++;
183 eNodeB5.x2++ <--> Eth10G <--> eNodeB12.x2++;
184 eNodeB5.x2++ <--> Eth10G <--> eNodeB13.x2++;
185 eNodeB5.x2++ <--> Eth10G <--> eNodeB14.x2++;
186 eNodeB5.x2++ <--> Eth10G <--> eNodeB15.x2++;
187 eNodeB5.x2++ <--> Eth10G <--> eNodeB16.x2++;
188 eNodeB5.x2++ <--> Eth10G <--> eNodeB17.x2++;
189 eNodeB5.x2++ <--> Eth10G <--> eNodeB18.x2++;
190 eNodeB5.x2++ <--> Eth10G <--> eNodeB19.x2++;
191 eNodeB5.x2++ <--> Eth10G <--> eNodeB20.x2++;
192 eNodeB5.x2++ <--> Eth10G <--> eNodeB21.x2++;
193 eNodeB5.x2++ <--> Eth10G <--> eNodeB22.x2++;
194 eNodeB5.x2++ <--> Eth10G <--> eNodeB23.x2++;
195 //eNodeB6////////////////////////////////////
196 eNodeB6.x2++ <--> Eth10G <--> eNodeB7.x2++;
197 eNodeB6.x2++ <--> Eth10G <--> eNodeB8.x2++;
198 eNodeB6.x2++ <--> Eth10G <--> eNodeB9.x2++;
199 eNodeB6.x2++ <--> Eth10G <--> eNodeB10.x2++;
200 eNodeB6.x2++ <--> Eth10G <--> eNodeB11.x2++;
```

```
201 eNodeB6.x2++ <--> Eth10G <--> eNodeB12.x2++;
202 eNodeB6.x2++ <--> Eth10G <--> eNodeB13.x2++;
203 eNodeB6.x2++ <--> Eth10G <--> eNodeB14.x2++;
204 eNodeB6.x2++ <--> Eth10G <--> eNodeB15.x2++;
205 eNodeB6.x2++ <--> Eth10G <--> eNodeB16.x2++;
206 eNodeB6.x2++ <--> Eth10G <--> eNodeB17.x2++;
207 eNodeB6.x2++ <--> Eth10G <--> eNodeB18.x2++;
208 eNodeB6.x2++ <--> Eth10G <--> eNodeB19.x2++;
209 eNodeB6.x2++ <--> Eth10G <--> eNodeB20.x2++;
210 eNodeB6.x2++ <--> Eth10G <--> eNodeB21.x2++;
211 eNodeB6.x2++ <--> Eth10G <--> eNodeB22.x2++;
212 eNodeB6.x2++ <--> Eth10G <--> eNodeB23.x2++;
213 //eNodeB7////////////////////////////////////
214 eNodeB7.x2++ <--> Eth10G <--> eNodeB8.x2++;
215 eNodeB7.x2++ <--> Eth10G <--> eNodeB9.x2++;
216 eNodeB7.x2++ <--> Eth10G <--> eNodeB10.x2++;
217 eNodeB7.x2++ <--> Eth10G <--> eNodeB11.x2++;
218 eNodeB7.x2++ <--> Eth10G <--> eNodeB12.x2++;
219 eNodeB7.x2++ <--> Eth10G <--> eNodeB13.x2++;
220 eNodeB7.x2++ <--> Eth10G <--> eNodeB14.x2++;
221 eNodeB7.x2++ <--> Eth10G <--> eNodeB15.x2++;
222 eNodeB7.x2++ <--> Eth10G <--> eNodeB16.x2++;
223 eNodeB7.x2++ <--> Eth10G <--> eNodeB17.x2++;
224 eNodeB7.x2++ <--> Eth10G <--> eNodeB18.x2++;
225 eNodeB7.x2++ <--> Eth10G <--> eNodeB19.x2++;
226 eNodeB7.x2++ <--> Eth10G <--> eNodeB20.x2++;
227 eNodeB7.x2++ <--> Eth10G <--> eNodeB21.x2++;
228 eNodeB7.x2++ <--> Eth10G <--> eNodeB22.x2++;
229 eNodeB7.x2++ <--> Eth10G <--> eNodeB23.x2++;
230 //eNodeB8////////////////////////////////////
231 eNodeB8.x2++ <--> Eth10G <--> eNodeB9.x2++;
232 eNodeB8.x2++ <--> Eth10G <--> eNodeB10.x2++;
233 eNodeB8.x2++ <--> Eth10G <--> eNodeB11.x2++;
234 eNodeB8.x2++ <--> Eth10G <--> eNodeB12.x2++;
235 eNodeB8.x2++ <--> Eth10G <--> eNodeB13.x2++;
236 eNodeB8.x2++ <--> Eth10G <--> eNodeB14.x2++;
237 eNodeB8.x2++ <--> Eth10G <--> eNodeB15.x2++;
238 eNodeB8.x2++ <--> Eth10G <--> eNodeB16.x2++;
239 eNodeB8.x2++ <--> Eth10G <--> eNodeB17.x2++;
240 eNodeB8.x2++ <--> Eth10G <--> eNodeB18.x2++;
241 eNodeB8.x2++ <--> Eth10G <--> eNodeB19.x2++;
242 eNodeB8.x2++ <--> Eth10G <--> eNodeB20.x2++;
243 eNodeB8.x2++ <--> Eth10G <--> eNodeB21.x2++;
244 eNodeB8.x2++ <--> Eth10G <--> eNodeB22.x2++;
245 eNodeB8.x2++ <--> Eth10G <--> eNodeB23.x2++;
246 //eNodeB9////////////////////////////////////
247 eNodeB9.x2++ <--> Eth10G <--> eNodeB10.x2++;
248 eNodeB9.x2++ <--> Eth10G <--> eNodeB11.x2++;
249 eNodeB9.x2++ <--> Eth10G <--> eNodeB12.x2++;
```

```
250 eNodeB9.x2++ <--> Eth10G <--> eNodeB13.x2++;
251 eNodeB9.x2++ <--> Eth10G <--> eNodeB14.x2++;
252 eNodeB9.x2++ <--> Eth10G <--> eNodeB15.x2++;
253 eNodeB9.x2++ <--> Eth10G <--> eNodeB16.x2++;
254 eNodeB9.x2++ <--> Eth10G <--> eNodeB17.x2++;
255 eNodeB9.x2++ <--> Eth10G <--> eNodeB18.x2++;
256 eNodeB9.x2++ <--> Eth10G <--> eNodeB19.x2++;
257 eNodeB9.x2++ <--> Eth10G <--> eNodeB20.x2++;
258 eNodeB9.x2++ <--> Eth10G <--> eNodeB21.x2++;
259 eNodeB9.x2++ <--> Eth10G <--> eNodeB22.x2++;
260 eNodeB9.x2++ <--> Eth10G <--> eNodeB23.x2++;
261 //eNodeB10////////////////////////////////////
262 eNodeB10.x2++ <--> Eth10G <--> eNodeB11.x2++;
263 eNodeB10.x2++ <--> Eth10G <--> eNodeB12.x2++;
264 eNodeB10.x2++ <--> Eth10G <--> eNodeB13.x2++;
265 eNodeB10.x2++ <--> Eth10G <--> eNodeB14.x2++;
266 eNodeB10.x2++ <--> Eth10G <--> eNodeB15.x2++;
267 eNodeB10.x2++ <--> Eth10G <--> eNodeB16.x2++;
268 eNodeB10.x2++ <--> Eth10G <--> eNodeB17.x2++;
269 eNodeB10.x2++ <--> Eth10G <--> eNodeB18.x2++;
270 eNodeB10.x2++ <--> Eth10G <--> eNodeB19.x2++;
271 eNodeB10.x2++ <--> Eth10G <--> eNodeB20.x2++;
272 eNodeB10.x2++ <--> Eth10G <--> eNodeB21.x2++;
273 eNodeB10.x2++ <--> Eth10G <--> eNodeB22.x2++;
274 eNodeB10.x2++ <--> Eth10G <--> eNodeB23.x2++;
275 //eNodeB11////////////////////////////////////
276 eNodeB11.x2++ <--> Eth10G <--> eNodeB12.x2++;
277 eNodeB11.x2++ <--> Eth10G <--> eNodeB13.x2++;
278 eNodeB11.x2++ <--> Eth10G <--> eNodeB14.x2++;
279 eNodeB11.x2++ <--> Eth10G <--> eNodeB15.x2++;
280 eNodeB11.x2++ <--> Eth10G <--> eNodeB16.x2++;
281 eNodeB11.x2++ <--> Eth10G <--> eNodeB17.x2++;
282 eNodeB11.x2++ <--> Eth10G <--> eNodeB18.x2++;
283 eNodeB11.x2++ <--> Eth10G <--> eNodeB19.x2++;
284 eNodeB11.x2++ <--> Eth10G <--> eNodeB20.x2++;
285 eNodeB11.x2++ <--> Eth10G <--> eNodeB21.x2++;
286 eNodeB11.x2++ <--> Eth10G <--> eNodeB22.x2++;
287 eNodeB11.x2++ <--> Eth10G <--> eNodeB23.x2++;
288 //eNodeB12////////////////////////////////////
289 eNodeB12.x2++ <--> Eth10G <--> eNodeB13.x2++;
290 eNodeB12.x2++ <--> Eth10G <--> eNodeB14.x2++;
291 eNodeB12.x2++ <--> Eth10G <--> eNodeB15.x2++;
292 eNodeB12.x2++ <--> Eth10G <--> eNodeB16.x2++;
293 eNodeB12.x2++ <--> Eth10G <--> eNodeB17.x2++;
294 eNodeB12.x2++ <--> Eth10G <--> eNodeB18.x2++;
295 eNodeB12.x2++ <--> Eth10G <--> eNodeB19.x2++;
296 eNodeB12.x2++ <--> Eth10G <--> eNodeB20.x2++;
297 eNodeB12.x2++ <--> Eth10G <--> eNodeB21.x2++;
298 eNodeB12.x2++ <--> Eth10G <--> eNodeB22.x2++;
```

```
299 eNodeB12.x2++ <--> Eth10G <--> eNodeB23.x2++;
300 //eNodeB13////////////////////////////////////
301 eNodeB13.x2++ <--> Eth10G <--> eNodeB14.x2++;
302 eNodeB13.x2++ <--> Eth10G <--> eNodeB15.x2++;
303 eNodeB13.x2++ <--> Eth10G <--> eNodeB16.x2++;
304 eNodeB13.x2++ <--> Eth10G <--> eNodeB17.x2++;
305 eNodeB13.x2++ <--> Eth10G <--> eNodeB18.x2++;
306 eNodeB13.x2++ <--> Eth10G <--> eNodeB19.x2++;
307 eNodeB13.x2++ <--> Eth10G <--> eNodeB20.x2++;
308 eNodeB13.x2++ <--> Eth10G <--> eNodeB21.x2++;
309 eNodeB13.x2++ <--> Eth10G <--> eNodeB22.x2++;
310 eNodeB13.x2++ <--> Eth10G <--> eNodeB23.x2++;
311 //eNodeB14////////////////////////////////////
312 eNodeB14.x2++ <--> Eth10G <--> eNodeB15.x2++;
313 eNodeB14.x2++ <--> Eth10G <--> eNodeB16.x2++;
314 eNodeB14.x2++ <--> Eth10G <--> eNodeB17.x2++;
315 eNodeB14.x2++ <--> Eth10G <--> eNodeB18.x2++;
316 eNodeB14.x2++ <--> Eth10G <--> eNodeB19.x2++;
317 eNodeB14.x2++ <--> Eth10G <--> eNodeB20.x2++;
318 eNodeB14.x2++ <--> Eth10G <--> eNodeB21.x2++;
319 eNodeB14.x2++ <--> Eth10G <--> eNodeB22.x2++;
320 eNodeB14.x2++ <--> Eth10G <--> eNodeB23.x2++;
321 //eNodeB15////////////////////////////////////
322 eNodeB15.x2++ <--> Eth10G <--> eNodeB16.x2++;
323 eNodeB15.x2++ <--> Eth10G <--> eNodeB17.x2++;
324 eNodeB15.x2++ <--> Eth10G <--> eNodeB18.x2++;
325 eNodeB15.x2++ <--> Eth10G <--> eNodeB19.x2++;
326 eNodeB15.x2++ <--> Eth10G <--> eNodeB20.x2++;
327 eNodeB15.x2++ <--> Eth10G <--> eNodeB21.x2++;
328 eNodeB15.x2++ <--> Eth10G <--> eNodeB22.x2++;
329 eNodeB15.x2++ <--> Eth10G <--> eNodeB23.x2++;
330 //eNodeB16////////////////////////////////////
331 eNodeB16.x2++ <--> Eth10G <--> eNodeB17.x2++;
332 eNodeB16.x2++ <--> Eth10G <--> eNodeB18.x2++;
333 eNodeB16.x2++ <--> Eth10G <--> eNodeB19.x2++;
334 eNodeB16.x2++ <--> Eth10G <--> eNodeB20.x2++;
335 eNodeB16.x2++ <--> Eth10G <--> eNodeB21.x2++;
336 eNodeB16.x2++ <--> Eth10G <--> eNodeB22.x2++;
337 eNodeB16.x2++ <--> Eth10G <--> eNodeB23.x2++;
338 //eNodeB17////////////////////////////////////
339 eNodeB17.x2++ <--> Eth10G <--> eNodeB18.x2++;
340 eNodeB17.x2++ <--> Eth10G <--> eNodeB19.x2++;
341 eNodeB17.x2++ <--> Eth10G <--> eNodeB20.x2++;
342 eNodeB17.x2++ <--> Eth10G <--> eNodeB21.x2++;
343 eNodeB17.x2++ <--> Eth10G <--> eNodeB22.x2++;
344 eNodeB17.x2++ <--> Eth10G <--> eNodeB23.x2++;
345 //eNodeB18////////////////////////////////////
346 eNodeB18.x2++ <--> Eth10G <--> eNodeB19.x2++;
347 eNodeB18.x2++ <--> Eth10G <--> eNodeB20.x2++;
```

```

348 eNodeB18.x2++ <--> Eth10G <--> eNodeB21.x2++;
349 eNodeB18.x2++ <--> Eth10G <--> eNodeB22.x2++;
350 eNodeB18.x2++ <--> Eth10G <--> eNodeB23.x2++;
351 //eNodeB19////////////////////////////////////
352 eNodeB19.x2++ <--> Eth10G <--> eNodeB20.x2++;
353 eNodeB19.x2++ <--> Eth10G <--> eNodeB21.x2++;
354 eNodeB19.x2++ <--> Eth10G <--> eNodeB22.x2++;
355 eNodeB19.x2++ <--> Eth10G <--> eNodeB23.x2++;
356 //eNodeB20////////////////////////////////////
357 eNodeB20.x2++ <--> Eth10G <--> eNodeB21.x2++;
358 eNodeB20.x2++ <--> Eth10G <--> eNodeB22.x2++;
359 eNodeB20.x2++ <--> Eth10G <--> eNodeB23.x2++;
360 //eNodeB21////////////////////////////////////
361 eNodeB21.x2++ <--> Eth10G <--> eNodeB22.x2++;
362 eNodeB21.x2++ <--> Eth10G <--> eNodeB23.x2++;
363 //eNodeB22////////////////////////////////////
364 eNodeB22.x2++ <--> Eth10G <--> eNodeB23.x2++;
365 //eNodeB23////////////////////////////////////

```

Listing A.8: NED code for whole playground scenario for 24 eNodeBs and 200 UEs

```

1 void LtePhyUe::doHandover()
2 {
3     // Delete Old Buffers
4     deleteOldBuffers(masterId_);
5     // amc calls
6     LteAmc *oldAmc = getAmcModule(masterId_);
7     LteAmc *newAmc = getAmcModule(candidateMasterId_);
8     assert(newAmc != NULL);
9     oldAmc->detachUser(nodeId_, UL);
10    oldAmc->detachUser(nodeId_, DL);
11    newAmc->attachUser(nodeId_, UL);
12    newAmc->attachUser(nodeId_, DL);
13    // binder calls
14    binder->unregisterNextHop(masterId_, nodeId_);
15    binder->registerNextHop(candidateMasterId_, nodeId_);
16    binder->updateUeInfoCellId(nodeId_, candidateMasterId_);
17    das->setMasterRuSet(candidateMasterId_);
18    // change masterId and notify handover to the MAC layer
19    MacNodeId oldMaster = masterId_;
20    masterId_ = candidateMasterId_;
21    mac->doHandover(candidateMasterId_); // do MAC operations
22    for handover
23        currentMasterRssi_ = candidateMasterRssi_;
24        hysteresisTh_ = updateHysteresisTh(currentMasterRssi_);
25    // update cellInfo

```

```

25  LteMacEnb* newMacEnb = check_and_cast<LteMacEnb*>(
    getSimulation()->getModule(binder_->getOmnetId(
    candidateMasterId_))->getSubmodule("lteNic")->getSubmodule("mac
    "));
26  LteCellInfo* newCellInfo = newMacEnb->getCellInfo();
27  cellInfo_->detachUser(nodeId_);
28  newCellInfo->attachUser(nodeId_);
29  cellInfo_ = newCellInfo;
30  // update DL feedback generator
31  LteDlFeedbackGenerator* fbGen = check_and_cast<
    LteDlFeedbackGenerator*>(getParentModule()->getSubmodule("
    dlFbGen"));
32  fbGen->handleHandover(masterId_);
33  // collect stat
34  emit(servingCell_, (long)masterId_);
35
36  //*****collecting per user number of handovers Developed in
    this thesis (begin)
37  totalHandovers++;
38  emit(numHandovers_,totalHandovers);
39  if (getEnvir()->isGUI())
40      getParentModule()->getParentModule()->bubble("Handover
    complete!");//show the handover completion on top of each UE or
    car
41  //*****collecting per user number of handovers Developed in
    this thesis (end)
42
43  EV << NOW << " LtePhyUe::doHandover - UE " << nodeId_ << " has
    completed handover to eNB " << masterId_ << "... " << endl;
44  binder_->removeUeHandoverTriggered(nodeId_);
45  // inform the UE's IP2lte module to forward held packets
46  IP2lte* ip2lte = check_and_cast<IP2lte*>(getParentModule()->
    getSubmodule("ip2lte"));
47  ip2lte->signalHandoverCompleteUe();
48  // inform the eNB's IP2lte module to forward data to the
    target eNB
49  IP2lte* enbIp2lte = check_and_cast<IP2lte*>(getSimulation()->
    getModule(binder_->getOmnetId(masterId_))->getSubmodule("lteNic
    ")->getSubmodule("ip2lte"));
50  enbIp2lte->signalHandoverCompleteTarget(nodeId_,oldMaster);
51 }

```

Listing A.9: per use number of handovers collection lines 44 to 48(LtePhyUe.cc)

```

1  void PPP::handleMessage(cMessage *msg)
2  {
3      //***** turning the ppp link off or no for 1 base station as a
        sample for time<10s (befor car carrying eNodeB arrival and
        time>80s after car carrying eNodeB departure(begin)

```

```

4   if ( (simTime()<=10 || simTime()>=80) && strcmp(
getParentModule()->getParentModule()->getFullName(),"eNodeB0")
==0)
5   {
6       isOperational=false;
7   }
8   else
9       isOperational=true;
10  //***** turning the ppp link off or on for 1 base station as a
sample for time<10s (befor car carrying eNodeB arrival and
time>80s after car carrying eNodeB departure(end)
11  if (!isOperational) {
12      handleMessageWhenDown(msg);
13      return;
14  }
15  if (msg == endTransmissionEvent) {
16      // Transmission finished, we can start next one.
17      EV_INFO << "Transmission successfully completed.\n";
18      emit(txStateSignal, 0L);
19      // fire notification
20      notifDetails.setPacket(nullptr);
21      emit(NF_PP_TX_END, &notifDetails);
22      if (!txQueue.isEmpty()) {
23          cPacket *pk = (cPacket *)txQueue.pop();
24          startTransmitting(pk);
25      }
26      else if (queueModule && 0 == queueModule->
getNumPendingRequests()) {
27          queueModule->requestPacket();
28      }
29  }
30  else if (msg->arrivedOn("phys$i")) {
31      EV_INFO << "Received " << msg << " from network.\n";
32      notifDetails.setPacket(PK(msg));
33      emit(NF_PP_RX_END, &notifDetails);
34      emit(packetReceivedFromLowerSignal, msg);
35      // check for bit errors
36      if (PK(msg)->hasBitError()) {
37          EV_WARN << "Bit error in " << msg << endl;
38          emit(dropPkBitErrorSignal, msg);
39          numBitErr++;
40          delete msg;
41      }
42      else {
43          // pass up payload
44          PPPFrame *pppFrame = check_and_cast<PPPFrame *>(msg);
45          emit(rxPkOkSignal, pppFrame);
46          cPacket *payload = decapsulate(pppFrame);
47          numRcvdOK++;

```

```

48     emit(packetSentToUpperSignal, payload);
49     EV_INFO << "Sending " << payload << " to upper layer.\n";
50     send(payload, "netwOut");
51 }
52 }
53 else { // arrived on gate "netwIn"
54     EV_INFO << "Received " << msg << " from upper layer.\n";
55     if (datarateChannel == nullptr) {
56         EV_WARN << "Interface is not connected, dropping
packet " << msg << endl;
57         numDroppedIfaceDown++;
58         emit(dropPkIfaceDownSignal, msg);
59         delete msg;
60         if (queueModule && 0 == queueModule->
getNumPendingRequests())
61             queueModule->requestPacket();
62     }
63     else {
64         emit(packetReceivedFromUpperSignal, msg);
65
66         if (endTransmissionEvent->isScheduled()) {
67             // We are currently busy, so just queue up the
packet.
68             EV_DETAIL << "Received " << msg << " for
transmission but transmitter busy, queueing.\n";
69             if (txQueueLimit && txQueue.getLength() >
txQueueLimit)
70                 throw cRuntimeError("txQueue length exceeds %d
-- this is probably due to "
71                                     "a bogus app model
generating excessive traffic "
72                                     "(or if this is normal,
increase txQueueLimit!)",
73                                     txQueueLimit);
74             txQueue.insert(msg);
75         }
76         else {
77             // We are idle, so we can start transmitting right
away.
78             startTransmitting(PK(msg));
79         }
80     }
81 }
82 }

```

Listing A.10: turn on and off ppp of base station at lines 3 to 10 (inet/linklayer/ppp.cc)

```

1 void LtePhyEnb::handleAirFrame(cMessage* msg)
2 {
3     UserControlInfo* lteInfo = check_and_cast<UserControlInfo*>(
4     msg->removeControlInfo());
5     if (!lteInfo )
6     {
7         return;
8     }
9     /****** turning the radio link off or no for 1 base station as a
10    sample for time<10s (befor car carrying eNodeB arrival and
11    time>80s after car carrying eNodeB departure(begin)
12    if((simTime()<=10 || simTime()>=80) && strcmp(getParentModule
13    ()->getParentModule()->getFullName(),"eNodeB0")==0
14    {
15        txPower_=0;
16        return;
17    }
18    else
19        txPower_= eNodeBtxPower_;
20    /****** turning the ppp link off or no for 1 base station as a
21    sample for time<10s (befor car carrying eNodeB arrival and
22    time>80s after car carrying eNodeB departure(begin)
23    LteAirFrame* frame = static_cast<LteAirFrame*>(msg);
24    EV << "LtePhy: received new LteAirFrame with ID " << frame->
25    getId() << " from channel" << endl;
26    if ( lteInfo->getUserTxParams() != NULL )
27    {
28        double cqi = lteInfo->getUserTxParams()->readCqiVector()[
29    lteInfo->getCw()];
30    // handle broadcast packet sent by another eNB
31    if (lteInfo->getFrameType() == HANDOVERPKT)
32    {
33        EV << "LtePhyEnb::handleAirFrame - received handover
34    packet from another eNodeB. Ignore it." << endl;
35        delete lteInfo;
36        delete frame;
37        return;
38    }
39    if (binder_->getNextHop(lteInfo->getSourceId()) != nodeId_)
40    {
41        EV << "WARNING: frame from a UE that is leaving this cell
42    (handover): deleted " << endl;
43        EV << "Source MacNodeId: " << lteInfo->getSourceId() <<
44    endl;
45        EV << "Master MacNodeId: " << nodeId_ << endl;
46        delete lteInfo;
47        delete frame;

```

```

38     return;
39 }
40 connectedNodeId_ = lteInfo->getSourceId();
41 int sourceId = getBinder()->getOmnnetId(connectedNodeId_);
42 int senderId = getBinder()->getOmnnetId(lteInfo->getDestId());
43 if(sourceId == 0 || senderId == 0)
44 {
45     // either source or destination have left the simulation
46     delete msg;
47     return;
48 }
49 //handle all control pkt
50 if (handleControlPkt(lteInfo, frame))
51     return; // If frame contain a control pkt no further
action is needed
52 bool result = true;
53 RemoteSet r = lteInfo->getUserTxParams()->readAntennaSet();
54 if (r.size() > 1)
55 {
56     // Use DAS
57     // Message from ue
58     for (RemoteSet::iterator it = r.begin(); it != r.end(); it
59 ++)
60     {
61         EV << "LtePhy: Receiving Packet from antenna " << (*it
62 ) << "\n";
63         RemoteUnitPhyData data;
64         data.txPower = lteInfo->getTxPower();
65         data.m = getRadioPosition();
66         frame->addRemoteUnitPhyDataVector(data);
67     }
68     result = channelModel_->isCorruptedDas(frame, lteInfo);
69 }
70 else
71 {
72     result = channelModel_->isCorrupted(frame, lteInfo);
73 }
74 if (result)
75     numAirFrameReceived_++;
76 else
77     numAirFrameNotReceived_++;
78 EV << "Handled LteAirframe with ID " << frame->getId() << "
with result "
79 << (result ? "RECEIVED" : "NOT RECEIVED") << endl;
80 cPacket* pkt = frame->decapsulate();
81 // here frame has to be destroyed since it is no more useful
82 delete frame;
// attach the decider result to the packet as control info
lteInfo->setDeciderResult(result);

```

```

83     pkt->setControlInfo(lteInfo);
84     // send decapsulated message along with result control info to
      upperGateOut_
85     send(pkt, upperGateOut_);
86     if (getEnvir()->isGUI())
87         updateDisplayString();
88 }

```

Listing A.11: turn on and off radio of base station at lines 3 to 10 (inet/linklayer/ppp.cc)

```

1  double DistanceBasedConflictGraph::getDbmFromDistance(double
      distance)
2  {
3      // get the reference to the channel model of the eNB
4      LteChannelModel* channelModel = phyEnb_->getChannelModel();
5      //*****In order to sent eNodeB id to computePathloss method(
      begin)
6      MacNodeId eNbId= phyEnb_->getId();
7      //*****In order to sent eNodeB id to computePathloss method(
      end)
8      // obtain path loss in dBm
9      bool los = false;    // TODO make it configurable
10     double dbp = 0;
11     double pLoss = channelModel->computePathLoss(distance, eNbId,
      dbp, los); //Masoud
12     return pLoss;
13 }

```

Listing A.12: modified version of getDbmFromDistance Method

Bibliography

- [1] M. Ajmone Marsan, F. Mohammadnia, C. Vitale, M. Fiore, and V. Mancuso. «Towards mobile radio access infrastructures for mobile users». In: 14 (Mar. 2019), pp. 1–4 (cit. on pp. 3–5).
- [2] <http://www.dael.com/en/telecom/cell-on-wheels>. In: () (cit. on p. 3).
- [3] F. Mohammadnia, M. Fiore, and M.A. Marsan. «Adaptive densification of mobile networks: Exploring correlations in vehicular and telecom traffic». In: *The 17th Annual Mediterranean Ad Hoc Networking Workshop*. Capri, Italy, 2018, pp. 20–22 (cit. on p. 4).
- [4] <https://www.physiome.org/jsim/> (cit. on p. 9).
- [5] <https://www.nsnam.org/> (cit. on p. 9).
- [6] M. Saidallah, A. El Fergougui, and A. Elbelrhiti Elalaoui. «A Comparative Study of Urban Road Traffic Simulators». In: 6 (2016), p. 2 (cit. on pp. 11, 12).
- [7] G. Kotusevski and K.A. Hawick. «A Review of Traffic Simulation Software». In: 20 (2009), p. 2 (cit. on pp. 12, 13).
- [8] <http://veins.car2x.org/> (cit. on p. 13).
- [9] <http://veins-lte.car2x.org/> (cit. on p. 14).
- [10] G. Nardini, A. Viridis, and G. Stea. «Simulating cellular communications in vehicular networks: making SimuLTE interoperable with Veins». In: 4 (2016), p. 1 (cit. on p. 14).
- [11] A. Viridis, G. Nardini, and G. Stea. «Simulating LTE/LTE-Advanced networks with SimuLTE». In: 18 (Jan. 2016), p. 1 (cit. on p. 14).
- [12] <https://inet.omnetpp.org/docs/showcases/mobility/basic/doc/index.html>. In: () (cit. on p. 36).