HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

A hybrid network node for ad-hoc and mobile cellular network simulations using PeerfactSim.KOM

Bachelor Thesis

by

Kevin Kalagi

born in
Düsseldorf

submitted to

Technology of Social Networks Lab
Jun.Prof. Dr.-Ing. Kalman Graffi
Heinrich-Heine-Universität Düsseldorf

Juni 2017

Supervisor:
Raphael Bialon, M. Sc.

Abstract

Topic of this thesis is the implementation of a hybrid node for ad-hoc and mobile cellular networks simulations into the already existing network simulator PeerfactSim.KOM [pee]. There are currently two different *Mobility Models* already implemented into the simulator but none of them offers a possibility for a mobile cellular network simulation.

For the mobile part I decided to implement cell towers, which are able to represent different mobile standards. That way the nodes will check for the tower with the highest mobile standard within range in order to send or receive their messages. The implementation offers the possibility of generating random cell towers, as well as user specific cell towers through a configuration file.

A realistic scenario is given, involving the university campus, which focuses on the strength of this implementation and what results we could expect from this scenario.

In conclusion we can assume this implementation is a step into the direction of more realistic mobility simulations and evaluations.

Acknowledgments

First and foremost I would like to thank my brother Eric Kalagi, for all the help and time he contributed to me.

Also a huge thanks to my advisor Raphael Bialon, who helped me throughout the thesis with a lot of advice.

Additionally I would like to thank my mother Dunja Kalagi and Vladimir Tassev Belyashki for being helpful and supportive during the months I wrote my thesis.

I would also like to thank Jun.-Prof. Dr. Kalman Graffi for giving me the opportunity to write this thesis.

Finally I would like to thank all my friends, who were supportive the last months.

Contents

List of Figures

Listings

Chapter 1

Introduction

1.1 Motivation

In today's world smart phones play a major role. Almost everyone has one and is constantly connected to the Internet via different cell phone providers. These smart phones are mobile hybrid nodes as they are able to use both the mobile cellular network, as well as the ad-hoc network. Field tests considering, for example, the movement behavior of nodes or different routing protocols for more than up to 100.000 nodes [tso] would simply be way too expensive. So instead we use simulations in order to make such researches possible.

Being able to use the mobile cellular network as well as the ad-hoc network has a lot of benefits. While the cellular network is mainly designed for voice traffic, the ad-hoc network is designed to meet the best effort data traffic requirements. If we can choose between two connections it makes it more reliable, because if one connection is not working we can switch to the other. This means with the possibility of switching to the ad-hoc network we do not rely on mobile service providers.

1.2 Overview

The upcoming chapter 2 discusses the related work, as well as some of the problems and differences. The next chapter 3 focuses on what I planned in order to make this implementation. Afterwards in chapter 4 I present the implementation itself. In chapter 5 I present a possible scenario for this implementation and the expected results for it. Finally in chapter 6 I sum up this thesis and present ideas for future work.

Chapter 2

Related work

In this chapter I will briefly talk about the already existing simulator PeerfactSim.KOM [pee](2.1) and how it works. I will also give a short overview on the two already implemented Mobility Models in the sections 2.2.1 and 2.2.2 and how they work. Also I will briefly talk about tethering as alternative example for hybrid communication.

2.1 PeerfactSim.KOM

As mentioned before, PeerfactSim.KOM [pee] is an open source event-based simulator written in Java, with the main focus on peer-to-peer research projects. The simulator was originally developed at the Technical University (TU) Darmstadt [tud]. It was then further extended at the University of Paderborn (UPB) [upb] and the University of Düsseldorf (HHU) [hhu], where the simulator is currently being maintained and extended.

The simulator is using a layered architecture as shown in figure 2.1. Every peer is represented by a host, while the different layers for each host are being created, based on declarations specified in the *XML configuration file*. This way the host knows all of its layers and components and vice versa - every layer and component has access to its host.

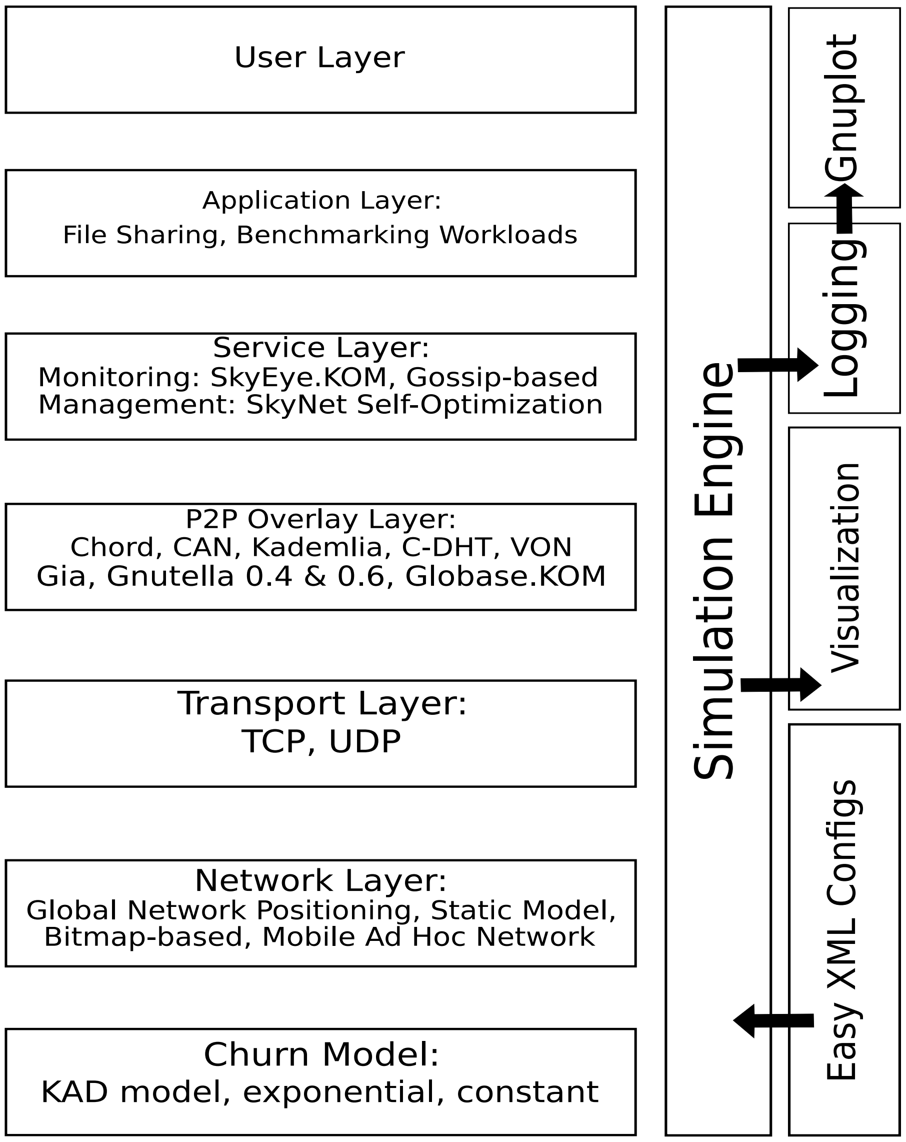
User Layer
Application Layer:
File Sharing, Benchmarking Workloads
Service Layer:
Monitoring: SkyEye.KOM, Gossip-based
Management: SkyNet Self-Optimization
P2P Overlay Layer:
Chord, CAN, Kademlia, C-DHT, VON
Gia, Gnutella 0.4 & 0.6, Globase.KOM
Transport Layer:
TCP, UDP
Network Layer:
Global Network Positioning, Static Model,
Bitmap-based, Mobile Ad Hoc Network
Churn Model:
KAD model, exponential, constant
Simulation Engine
Gnuplot
Logging
Visualization
Easy XML Configs

Figure 2.1: The architecture of PeerfactSim.KOM [arc]

The simulations run event-based, this means that every event, for example a host sending a message, is scheduled for a specific time. That way the message is only passed to the receiver the moment the event is due. This also means that there is a strict order for the events to happen as they follow a time line. An example of how this works can be seen in figure 2.2.

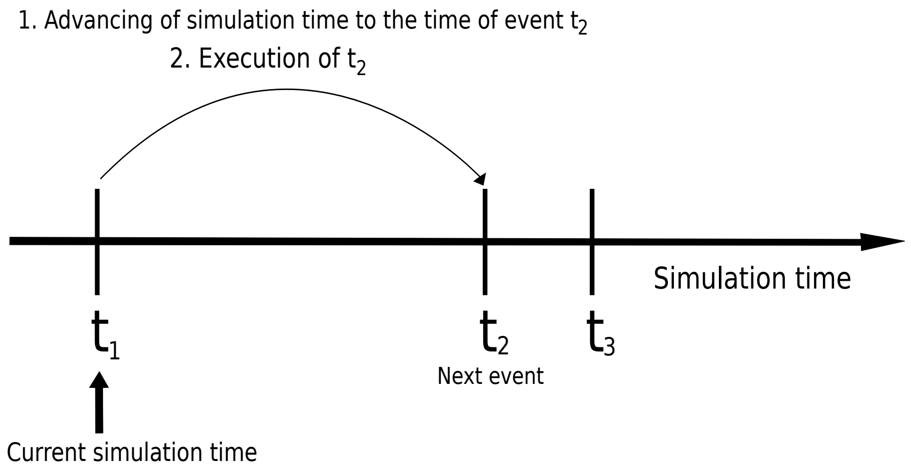

Figure 2.2: Advancing of simulation time in event-based simulations [doc]

2.2 Mobility Models

Prior to this thesis there were already two different Mobility Models implemented into the simulator PeerfactSim.KOM [pee].

The first implementation was developed by Carsten Snider throughout his bachelor thesis *"Mobility Aware Peer-to-Peer Networking"* [cas] at the Technical University (TU) Darmstadt [tud] in 2009.

The second thesis with an implementation of a Mobility Model was developed by Tobias Korfmacher during his bachelor thesis *"Implementation and Evaluation of a Mobility Model in a Peer-to-Peer Simulator"* [tko] at the University of Düsseldorf (HHU) [hhu] in 2015.

2.2.1 Mobility Aware Peer-to-Peer Networking

The implementation of Carsten Snider finds a path of nodes between the sender node and the receiver node. The delay to send the message is then calculated by the number of hops the path found from the sender to the receiver. To make it easier finding the path between sender and receiver, Carsten Snider decided to divide the simulation area into squares (called *Quadrants*), with each Quadrant having a list of nodes that it contains. For the movement of the nodes, he used the *RandomWayPointModel*.

What makes this implementation unrealistic is, that if there was no path found between the sender and the receiver, the sender node is simply set offline for the remaining duration of the simulation. Also, the path found is most of the time not the shortest path (with the least hops) between sender and receiver, because the path is created by finding the next nearest hop. This results in the path being made of the nearest neighbor nodes and therefore not the shortest path with the least hops between the sending node and the receiving node.

2.2.2 Implementation and Evaluation of a Mobility Model in a Peer-to-Peer Simulator

Tobias Korfmacher decided to give the position information of the nodes in real time rather than only every minute, as it was implemented in the model of Carsten Snider.

He also implemented different models, such as the position-based *PlacementModel* which makes it possible for nodes to have to move back to their original location, in case their battery drops low and they have to recharge it. All processes, such as sending or moving, reduce the battery of a node. This way it can happen, that a node might not be able to send a message, as the battery is already too low and must be recharged first. Another example for a model he implemented, is the *ObstacleModel*, which can create figures of different shapes, in the world that a node has to pass.

SoftWorld is the implementation of a world interface, which understands the world as a globe, rather than a map and thus it is possible for nodes, to move over the edges of the implementation area and even continue to send and receive their messages.

A huge difference to the model used by Carsten Snider is also that he got rid of the approach with the Quadrants and finding a path between the sending node and the receiving node. Instead he would use his implemented real time positions and the method *getDistance()*, to get the euclidean distance between the sender and the receiver. To calculate the latency for sending and receiving a message, he used the euclidean distance between sender and receiver and divided it by the velocity of light, which is 29979245800 *mm/sec* [vol].

2.3 Tethering

Another example for hybrid communication is the tethering. Tethering means to share the internet connection of a mobile device with other wireless devices in order to give them a connection to the Internet via the mobile network. The mobile device works then as a modem for the wireless device, the connection between those two devices can be achieved over WLAN, over Bluetooth or by a physical connection using for example USB. Note that if the connection is established over WLAN the mobile device becomes a mobile hotspot in which case it serves as a portable router.

2.4 Conclusion

In this chapter I gave a short overview on the simulator PeerfactSim.KOM [pee], which I will be using for my implementation. I also described the two already implemented Mobility Models and some of their major differences. I also gave an example for another hybrid communication.

Chapter 3

Preparation

In this chapter I will provide an insight on my approach to implement a hybrid node for ad-hoc and mobile cellular network simulations into the already existing project PeerfactSim.KOM [pee].

3.1 Hybrid Cell Tower Map

At the beginning it was planned to first implement the mobile cellular part and when it is working to add the ad-hoc part to it. For the mobile part I oriented myself towards the already implemented Mobility Model by Carsten Snider. Instead of dividing the simulation area into equal squares (called *Quadrants*) like Carsten Snider did, I planned to implement a *HybridCellTowerMap*. This HybridCellTowerMap would create circles on the simulation area, which would each represent the range of a cell tower. Each tower would have the following attributes: radius, x-coordinate, y-coordinate, bitrate-up and bitrate-down. For the first version it was planned to create these towers at random, while the attributes for them are determined by the mobile standard they represent. For the final version it was planned that the user would be able to modify the attributes and the number of towers in the *XML-configuration-file*.

In his Mobility Model Carsten Snider finds a path of nodes between the sender and receiver by always selecting the nearest node. He then calculates the latency for sending and receiving the message with the hopcount of the found path and a default bandwidth. In my model I planned to find the best cell tower for the sender and the best cell tower for the receiver, and then calculate the latency for the message by using the upload-rate of the best tower for the sender and the download-rate of the best tower for the receiver. This way the calculation for the latency would look like shown in figure 3.1. As the towers represent different mobile standards, the nodes need to be able to select the tower (or one of the towers) with the best performance for forwarding or receiving their messages. In order to make this work it is necessary that either the nodes have a list of the towers or each tower has a list of the nodes within its range.

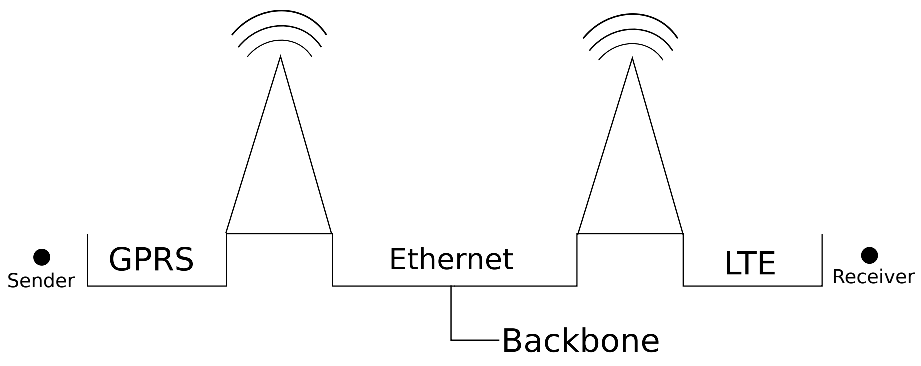

Figure 3.1: Connection between two nodes

3.2 Network Layer and Client

Nodes which are not within the range of a cell tower are not able to send or receive any messages, at least not by using a cell tower. It was planned that when trying to send a message it would work the following way: First step is to put the package into the queue. Second step is to try to send the first package in the queue. If this succeeds, it will check if the queue is empty and if it is it will finish the process. If the queue is not empty, it will go back to the start of the second step and start over. In case that the encounter trying to send the first package in the queue fails, it will check if the timeout has run out and if it has, it will discard the package. If the timeout has not run out it will go back to the start of step two with a delay (for example 0.5 seconds). The *DEA*, deterministic finite automaton, for this is shown in figure 3.2.

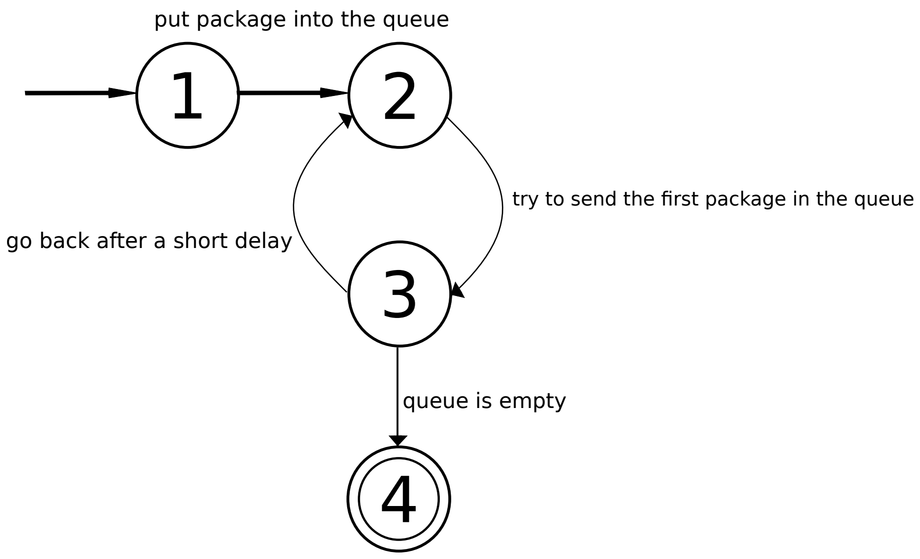
put package into the queue
1
2
try to send the first package in the queue
go back after a short delay
3
queue is empty
4

Figure 3.2: DEA for the package handling.

In order to send and receive messages I planned to create a *Network Layer* for every node and then send the messages from the sender Network Layer to the receiver Network Layer through the *Subnet*, which abstracts and represents the Internet. A good example for this can be seen in figure 4.3 in one of the later chapters.

In conclusion - going into this thesis it was planned to first add the mobile part and then later add the ad-hoc part to it. I would create cell towers on the simulation area. In order to calculate the latency, the nodes would use a method to find the tower with the best mobile standard to get the best possible upload-rate or download-rate for receiving or sending their messages. If a node is not inside the radius of any cell tower it has no mobile connection. Packages to send would be put into a queue, so the messages are delivered in the right order and if a package is not able to be send, it would try to send it again after a short delay. To achieve this I planned to use the Network Layer, so it would send the messages from the sender Network Layer to the receiver Network Layer through the Subnet.

Chapter 4

Implementation

This chapter focuses on the implementation of the hybrid network node for ad-hoc and mobile cellular network simulations, which was to be developed during this thesis for the simulator PeerfactSim.KOM [pee]. In section 4.2 I will explain the new implemented cell towers and how they work. The following section 4.3 presents the possibility to generate towers at random. Afterwards in section 4.4 I present the values for the different mobile standards I found during my research. Section 4.5 explains the method, nodes use to select the best tower for sending or receiving their messages, if they are within the range of more than one tower. The following section 4.6 mentions the *Movement Model* we are using. Section 4.7 is about the *XML configuration file* and how to set up cell towers for a simulation. In the final section 4.8 I explain how I had to adjust from a *Network Layer* and *Subnet* to a *Routing Layer*.

The implementation can be found in the folder: *src/org/peerfact/impl/network/hybrid*

4.1 Code of the Mobility Part

Currently there are problems with the code of the mobility part of the simulator. I ran into some flawed code when I was trying to run my implementation using parts of the mobility part, for example in the *RoutingFactory.java*, where in code-line 78 it states:

```
int x = 1/0;
```

As the already implemented mobility part is not part of my thesis, it was not in my hands to fix this.

4.2 Hybrid Cell Tower Map

There is already a Mobility Model implemented into the simulator by Carsten Snider, which divides the simulation area into equal squares called *Quadrants*. It will use those squares to check for the nearest nodes, in order to find a path of nodes between the sender and the receiver. For the implementation of the hybrid network node I got rid of this approach. Instead I decided to use mobile cellular towers which brings the simulation a lot closer to reality. This way the latency is not calculated by the amount of hops of the path found, but rather by the mobile standard of the tower with the best performance within the range of the node. Each tower has the following attributes: radius, x-coordinate, y-coordinate, bitrate-up and bitrate-down.

The x-coordinate and the y-coordinate determine the position on the simulation area of the mobile cell tower, as well as the center of the tower. The circle space is then created with the radius, which also defines the size of the circle. The bitrates represent the different mobile standards, as each mobile standard has different bitrates for the upload and the download. It also determines how fast a message between two nodes is delivered and if they are used by nodes for forwarding their message. Each tower has a list of the nodes within its range at the current time, which is updated after every movement phase. Every node also has a list of all the towers in reach. So when the event occurs that a node is supposed to send or receive a message, it will check which tower offers the best connection, in order to achieve the fastest receival or delivery of the message, and then choose this tower. If more than one tower offers the best performance to send or receive a message it does not matter, which of those towers the node will choose. In this case the nodes choose out from the best towers the one which they found first, which will be the tower with the lowest index.

4.3 Generating Random Towers

If the user decided not to set any towers in the configuration file *CellTowers.xml*, which can be found under *config/scenarios/wireless/*, manually or for some reason wants to generate random towers for a simulation, there is a method implemented, which will generate random towers. This method is invoked when the parse did not catch any towers and is currently set to generate five (can be changed in the *HybridMobileMovementManager.java* file) random towers. The towers will then get the value for each attribute (i.e. uploadrate, radius, position etc.) from an *enum-file* called *MobileStandard.java*. The user can simply delete all CellTower elements in the file. At present the CellTower.xml file contains five default towers to give a general idea on how the creation of towers work.

4.4 Mobile Standard

The *enum-file MobileStandard.java* contains five different standards I decided to include, which each tower is able to represent: *GPRS, EDGE, UTMS 3G, 3G HSDPA* and *LTE*. For each standard it has a set value for bitrate-up, bitrate-down and the radius.

To have those values as close to reality as possible, I researched the up- and down-bitrate values for those standards on the websites *3GPP* [3gpa] and *ETSI* [ets]. On the 3GPP site I was able to find the bitrates for LTE: 75Mbit/s in the uplink, the downlink can reach up to 300Mbit/s [3lt]. As bitrates for GPRS is stated it reaches 14kbit/s in the uplink and 40kbit/s in the downlink [3gpb]. The bitrates for the other three mobile standards I am using were not specified on the site. For the last three mobile standards I used the values from the website *elektronik-kompendium* [ekm], as their values for LTE were the same as on the 3GPP site [3gpa] and the closest to their GPRS values from all the other sites I found. So for EDGE I chose 236.8 kbit/s downlink and 118.4 kbit/s uplink, for UTMS 384 kbit/s downlink and 128 kbit/s uplink and for HSDPA 7.2 Mbit/s downlink and 384 kbit/s uplink as stated on the site [ekv].

Finding a value for the radius of the different mobile standards was difficult, as most sources would contradict each other and only state approximate or up to values, instead of exact values. This is probably because the mobile service providers want to keep this a secret. Most sources also stated, that there is no fixed value for the radius of a cell tower, as it would depend on different factors (see [lte], [mbs], [cel]), like the area it covers (for example in a city, it would have a lower radius, than in a rural area), the size of the antenna or how many obstacles like buildings or even mountains are in the way, just to name a few examples. So for testing I decided to go with 100 metre radius for LTE, 200 metres for HSDPA, 300 metres for UTMS, 800 metres for EDGE and 2400 metres for GPRS, which seemed pretty plausible for a scenario in a city area.

All those values are of course not set in stone and the ones in the *enum file Mobile-Standard.java* are only used to generate the random cell towers, if the user did not specify the towers himself in the *XML-file HybridMobilePeerfact.xml*. Any user can feel free to use his own values he found, even for the randomly generated towers.

4.5 Best Performance

As mentioned in one of the previous sections, the implementation contains a method called *bestPerformance()*, which is used to determine how the nodes will choose the cell tower in order to send or receive a message. Figure 4.1 shows an example how the implemented *HybridCellTowerMap* looks like and how the method *bestPerformance()* works.

In this example towers one, two and three represent a GPRS tower, tower four represents a LTE tower and tower five represents a 3G HSDPA tower. Assuming node four wants to send a message to node two. In order to calculate the time needed to send and receive the message, the nodes have to select the tower which offers the best possible performance. Each tower has a list of nodes, which are within its range. So node four can choose between tower one and tower two. Since they offer the same performance it does not matter which tower node four chooses to forward its message to node two. In this case the tower with the lowest index in the list is selected as the best performing tower (in this case tower one). Node two however can choose between tower five, four and three. To determine which tower offers the best performance to receive the message from node four, the method *bestPerformance()* is called, which iterates over the tower list of the node (in this case node two) and chooses the tower, which offers the highest download-rate (in this case tower four). So the time to send and receive the message is calculated with the upload-rate of tower one and the download-rate of tower four.

As another example let us assume node three wants to send a message to node one. Currently node one is only in the list of tower zero, which represents all pending nodes, that are currently not in range of any cell tower. Node three now has to wait and queue the package, since node one has at this moment in time no connection. After the movement update (shown by the arrows in figure 4.1), node one is within the range of tower one and now the message from node three can be delivered and received, as node three is still within the range of tower five.

4.6 Movement Model

There are two types of *Mobility Models*, the *Entity Mobility Models* [emm], where the nodes move independently and the *Group Mobility Models* [gmm], where nodes move in groups on a predefined path. For the movement of the nodes we are using the *Random Waypoint Model*, which was implemented by the model of Carsten Snider 2.2.1. The Random Waypoint Model is one of the most commonly used Mobility Models for network simulations. The way it works is the following: the nodes will at the start of the simulation select a random destination in the simulation area. The nodes then travel

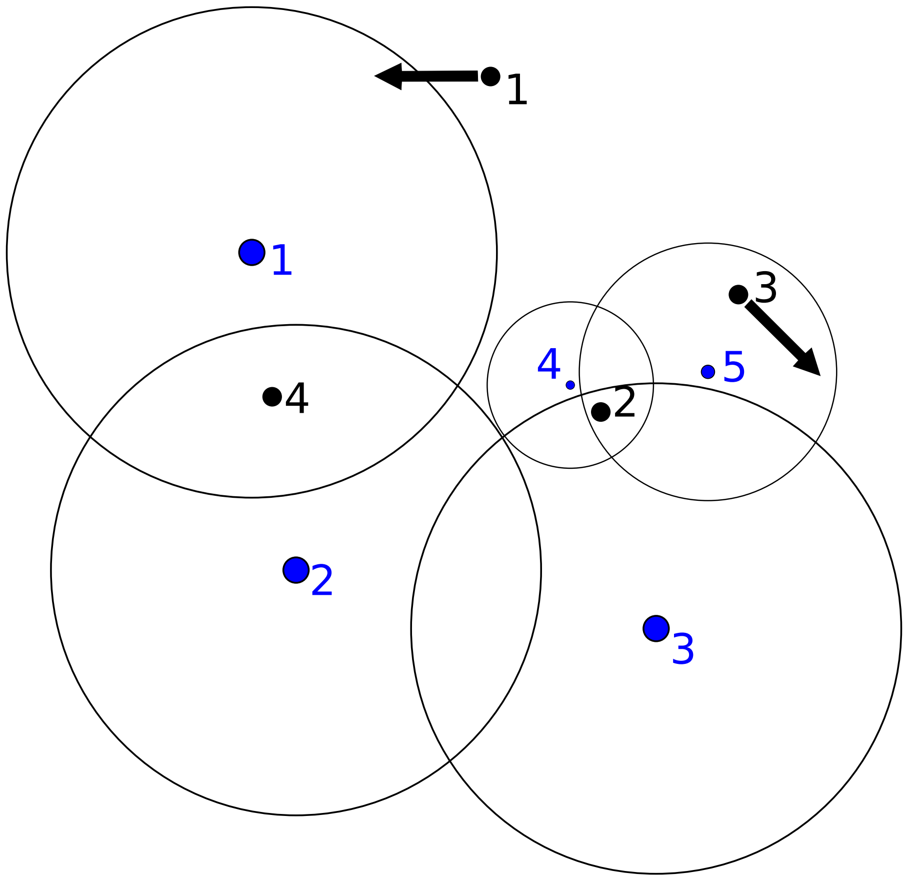

Figure 4.1: An example for five cell towers and four nodes

towards their destination with a velocity chosen randomly from an given interval. Once they reach their destination they will again select a random destination within the simulation area and the process starts over again until the simulation is finished. To give you a graphical example of the movement of a single node in the Random Waypoint Model you can have a look at figure 4.2. There are a lot of other Mobility Models which could be used for different kind of simulations. Therefore it is easy to exchange the MovementModel to be used for the implementation, by changing it in the *HybridMobileFactory.java*, provided that such a model has been implemented.

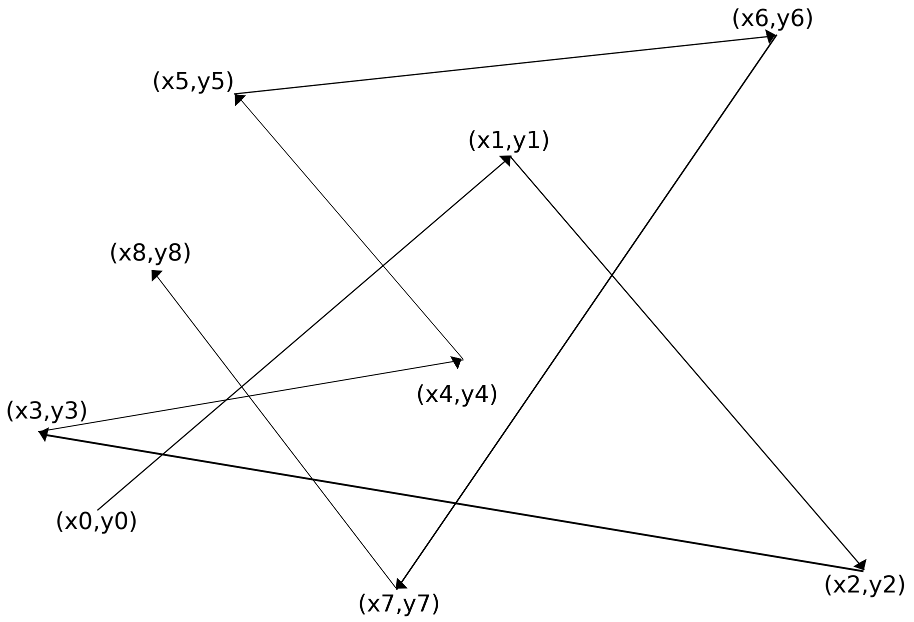

Figure 4.2: Random Waypoint Model [emm]: (section 2.1)

4.7 XML-based Configuration File and the Configuration for the Cell Towers

In order to determine which implementations and configurations are used on which layer, a XML-based configuration file is used. The XML file also contains the action file to be used. The XML file I am using for my implementation can be found under *config/scenarios/wireless/HybridMobilePeerfact.xml* .

The *Default* part in the XML file will be visible and adjustable by the user in the GUI of the simulator before starting a simulation. The user can adjust attributes like *size*, which will determine how many nodes will be created for the simulation or the *finishtime*, which determines the length of the simulation time. The values in the XML file are the default values for the simulation but can be, as already mentioned, changed by the user in the GUI before starting the simulation, in order to configure the simulation the user wants to run. In the listing 4.1 you can see all attributes that are included in the Default part.

Listing 4.1: Default XML structure

```
<Default>
      <Variable name="seed" value="1" />
      <Variable name="size" value ="5 "/>
      <Variable name="startTime" value="0m" />
      <Variable name="world_X" value="400" />
      <Variable name="world_Y" value="400" />
      <Variable name="finishTime" value="100m" />
      <Variable name="actions" value="config/scenarios/
          wireless/hybridmobile-actions.dat" />
</Default>
```

Next up is the *SimulationCore*, which loads the simulator itself and in our case provides it with the seed and the finishtime of the simulation. Those two values are taken from the Default part.

The next part of the XML file contains the different components we are using for the simulation. In our case these components all represent the different layers, which are instantiated by the different factories. The *Link Layer* includes the file *CellTower.xml*, which can be found under *config/scenarios/wireless/*. It contains the number and the attributes for each cell tower. To give you an example on how to set up the towers in the configuration file, you can look at 4.2. In this file the user can choose how many cell towers he wants to create simply by creating the right amount of towers in the file. Currently the attributes *name* and *standard* are only placeholders and are not used by my implementation, although it would be possible as they are parsed together with the other attributes. This is due to the fact that the mobile standard is defined by the upload-rate and download-rate, and the tower name by the index it gets for the list of towers. The other attributes speak for themselves, *radius* in metres, *x* and *y* for the coordinates, which define the position of the tower and *up* and *down* for the upload-rate and download-rate the tower offers. As mentioned before in section 4.3 the user can select to leave the file CellTower.xml without any CellTower element, in which case five towers will be generated randomly, with the help of default values for the radius, upload-rate and download-rate, given in the MobileStandard.java enum-file.

Next we specify the *HostBuilder*, which composes the set amount of hosts with the pre-

Listing 4.2: Example cell tower configuration

```
<CellTower>
    <name>tower1 </name>
    <radius >100</radius >
    <x >0.5 </x>
    <y >0.5 </y>
    <standard >LTE</standard >
    <up >75000</up>
    <down >300000</down>
</CellTower>
```

defined layers. The number of hosts is taken from the *size* value of the Default section and the predefined layers are the ones listed in the component section.

Last but not least we have a part for the *Scenario*, which contains the action-file to be used. This action-file in turn determines the actions to be taken for the hosts, as well as the scheduling for those actions.

4.8 Hybrid Mobile Routing Layer

As already mentioned in the chapter Preparation 3, it was originally planned to send and receive the messages to/from the nodes with the *Network Layer*, which exchanges the messages with the above lying *Transport Layer* through the *Subnet* as shown in 4.3.

As my Mobility Model works on the *Link Layer* and the layer above the Link Layer is the Routing Layer, it turned out, that I need a Routing Layer. The way the Routing Layer was implemented in the Mobility Part of the simulator makes it impossible to have a Routing Layer and a Network Layer work at the same time. To get an idea how the layers are implemented into the simulator see figure 4.4. As visible from the figure, the Routing Layer and the Network Layer operate at the same level and therefore are not able to coexist.

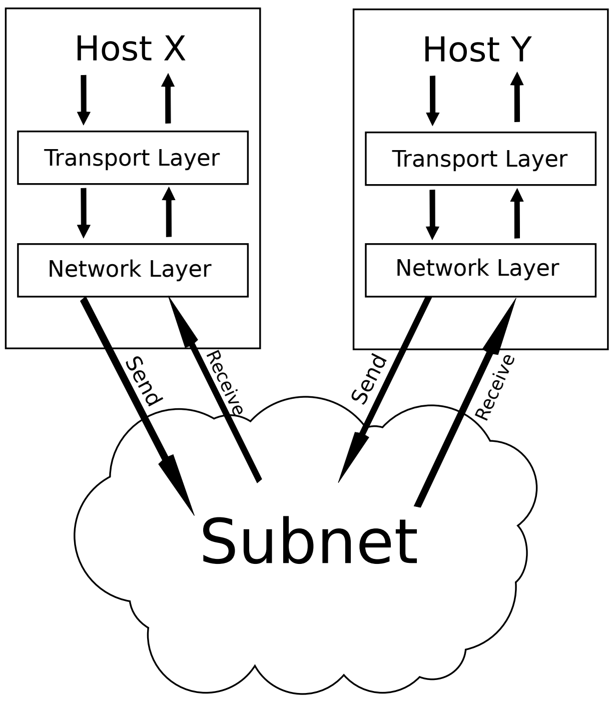

Figure 4.3: Communication between Network Layers through the Subnet. [sub]

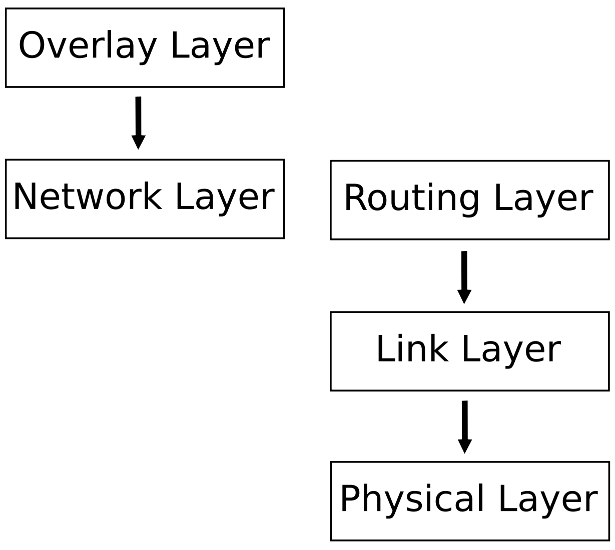

Figure 4.4: The layer architecture of the simulator PeerfactSim.KOM [pee]

For that reason I had to adjust my original plan with Network Layer and the Subnet and go from the Network Layer to the Routing Layer. I decided to get completely rid of the Subnet, as it would only work with a Network Layer and not with a Routing Layer. Instead of sending the messages via the Network Layer and through the Subnet, I decided to directly send the messages from the Routing Layer of the sender to the Routing Layer of the receiver, while keeping the communication with the Transport Layer above. To give a graphical example on how the new implemented Routing Layer works, in order to receive and send messages between the nodes, see figure 4.5. To make this work I reimplemented my originally developed Network Layer into a Routing Layer, which also handles the events, handled by the Subnet before that.

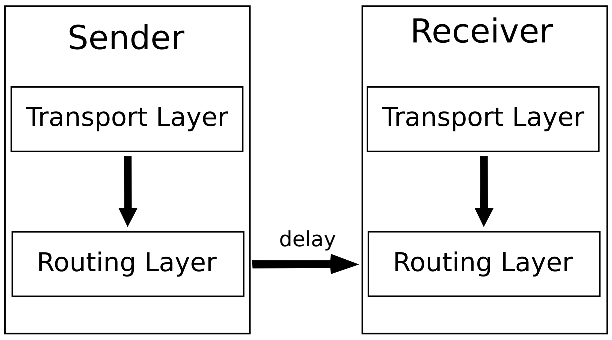

Figure 4.5: The updated communication model

Chapter 5

Evaluation

In this chapter I describe a scenario in 5.1, in order to evaluate my implementation. Afterwards I discuss the expected result of this scenario in section 5.2.

5.1 Metrics

The metrics, which are the most interesting to this implementation, are the positioning of the cell towers in combination with the choice of the mobile standard and the broadcast radius, and how these affect realistic scenarios. As already mentioned in 4.4 the values for the radius can vary, even for the the same mobile standards, as they depend on different factors.

For the realistic scenario I chose the campus of the University of Düsseldorf (HHU) [hhu]. The placement of the cell towers for the campus can be seen on the site of the *Bundesnetzagentur* [bna]. To find the campus of the University of Düsseldorf (HHU) one can simply enter „Universitäts Straße 1“ into the address field *Straße* and „40225“ into the postcode field *PLZ* and zoom in onto the found address.

An overview on how the campus map, with focus on the four cell towers, could look like see figure 5.1.

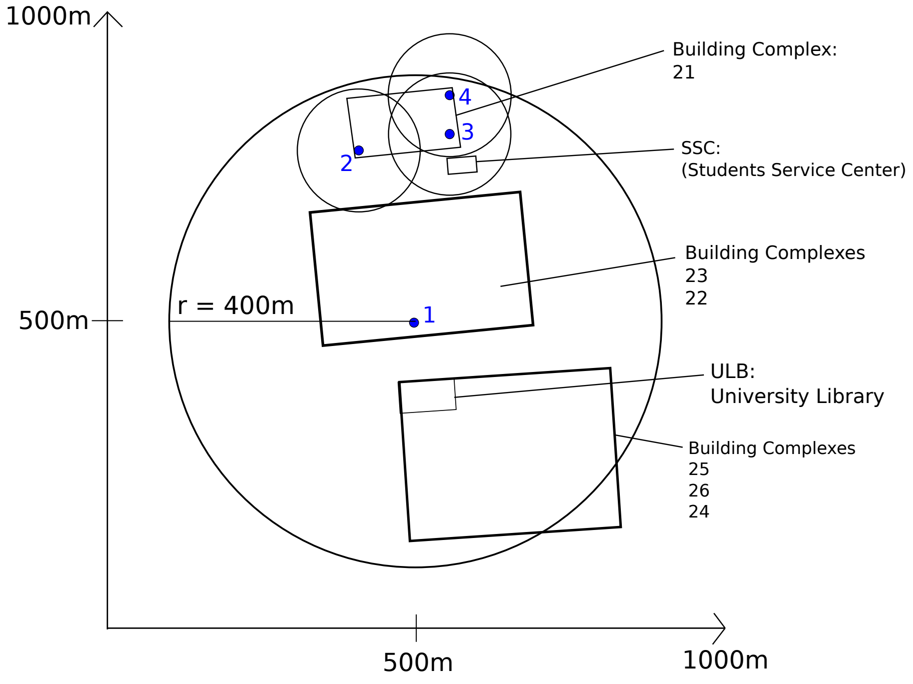

Figure 5.1: Possible scenario for the university campus

First I created a simulation area of 1000x1000 metres and put the centre of the cell tower in the middle of the campus, as the centre of the simulation area (500, 500). With the help of the scale on the website *Bundesnetzagentur* [bna] I positioned the centres of the other cell towers accordingly. Afterwards I created, again according to the scale, the obstacles, in this case the different buildings and building complexes. My thought process then was, that tower one probably wants to cover most of the university campus and decided, that for my set mobile standards the UTMS one, with a radius of 400 metres, would be the most fitting. Apart from a little corner of the building complex 26, it covers almost every building up to the cell towers in the north. For the towers two, three and four, I chose a radius of 100 metres, which reflects a LTE cell tower. As they cover all the buildings in the north - two student dorms, a kindergarten and the canteen, as well as almost the end of the simulation area.

To create this simulation we first have to set the simulation area to 1000x1000 metres, we achieve this by changing the values of *world_X* and *world_Y* in the XML-configuration file or in the GUI, before starting the simulation. For the towers I would have chosen the values in listing 5.1. I would have started the scenario with two hosts so the value for *size* would be two. The first host is always created in the middle of the simulation area, while the other host would be generated at a random position. For the simulation time I would have chosen 120 minutes, that corresponds to setting the *finishTime* value to 120m.

5.2 Results

For reasons stated earlier I was not able to test the scenario described in section 5.1. Therefore I will talk here about the expected results from this particular scenario instead. As we have two different types of cell towers in this scenario, I would expect a maximum of four different latency calculations, assuming the size of the message would be the same. For the following calculations I used 100.000kbit for the size of the message.

a.) Both nodes are only within the range of the UTMS cell tower, therefore we would calculate the latency as follows:

$$\text{latency} = \frac{100000kbit}{128kbit/s} + \frac{100000kbit}{384kbit/s} \approx 1041,67sec \rightarrow \frac{1041,67sec}{60} \approx 17,36min$$

b.) Both nodes are within the range of one of the LTE towers, therefore we would calculate the latency as follows:

$$\text{latency} = \frac{100000kbit}{75000bit/s} + \frac{100000kbit}{300000kbit/s} \approx 1,67sec$$

c.) The sender node is only within the range of the UTMS tower, but the receiver node is within the range of a one of the LTE towers, therefore we would calculate the latency as follows:

$$\text{latency} = \frac{100000kbit}{128kbit/s} + \frac{100000kbit}{300000kbit/s} \approx 781,58sec \rightarrow \frac{781,58sec}{60} \approx 13,03min$$

Listing 5.1: Scenario cell tower configuration

```
<CellTower>
    <name>tower1 </name>
    <radius >400</radius >
    <x>500</x>
    <y>500</y>
    <standard >UTMS</standard >
    <up>128</up>
    <down>384</down>
</CellTower>
<CellTower>
    <name>tower2 </name>
    <radius >100</radius >
    <x>406</x>
    <y>774</y>
    <standard >LTE</standard >
    <up>75000</up>
    <down>300000</down>
</CellTower>
<CellTower>
    <name>tower3 </name>
    <radius >100</radius >
    <x>554</x>
    <y>800</y>
    <standard >LTE</standard >
    <up>75000</up>
    <down>300000</down>
</CellTower>
<CellTower>
    <name>tower4 </name>
    <radius >100</radius >
    <x>554</x>
    <y>865</y>
    <standard >LTE</standard >
    <up>75000</up>
    <down>300000</down>
</CellTower>
```

d.) The sender is within the range of one of the LTE towers, but the receiver is only within the range of the UTMS tower, therefore we would calculate the latency as follows:

$$\text{latency} = \frac{100000kbit}{75000kbit/s} + \frac{100000kbit}{384kbit/s} \approx 261,75sec \rightarrow \frac{261,75sec}{60} \approx 4,36min$$

The simulation area is:

$$1000m \cdot 1000m = 1.000.000m^2.$$

Tower one covers:

$$400m^2 \cdot \pi \approx 502.654,82m^2$$

The towers two, three and four only cover an area of:

$$100m^2 \cdot \pi \approx 31.415,93m^2$$

Of course the towers two, three and four all overlap and they all also overlap with tower four as well. If we stay with a constant message size of 100.000kbit for the calculation of the latency, I would expect that most messages will have the biggest possible latency of 17,36min, as tower one covers the biggest part of the simulation area. Most unlikely is a latency of only 1,67sec, as the three LTE towers cover the smallest portion of the simulation area and even less, because they overlap each other.

As the possible a outcome for this scenario I would expect that everything works accordingly. The nodes would not send or receive any messages if they are not within the range of a cell tower. For sending and receiving the nodes would always choose the best performing tower if they are within the range of more than one tower. This means we should be able to run realistic mobile cellular network simulations excluding the movement behavior of the nodes, which I will talk about in the section 6.2.

Chapter 6

Summary

In this chapter I briefly summarise and draw a conclusion for my implementation in section 6.1 and then in 6.2 propose some ideas for possible future work with the simulator *PeerfactSim.KOM* [pee] and in particular my own implementation.

6.1 Conclusion

During this thesis a hybrid network node for ad-hoc and mobile cellular network simulations was to be developed for the simulator PeerfactSim.KOM [pee]. At first I considered related work in chapter 2, in particular both Mobility Models by Carsten Snider 2.2.1 and by Tobias Korfmacher 2.2.2, in order to plan my own Model. In chapter 3 I then discussed everything I planned for this implementation and how to make these work. In the following chapter 4, I presented the *HybridCellTowerMap* 4.2 I developed. I also mentioned the enum file *MobileStandard.java* 4.4 which contains the values for the different mobile standards I had researched and what purpose it serves. Speaking of its purpose, I presented the method of generating cell towers with a randomly chosen mobile standard and a randomly chosen position in the section 4.3, with the help of the enum file. I then talked about the method *bestPerformance()* 4.5, which determines how the nodes always choose the tower with the best bitrates, in order to receive or send their messages, if they are within reach of more than one tower. I also briefly talked about the *Movement Model* 4.6 we are using, as well as the *XML Configuration File* 4.7 and how to configure your own cell towers. Finally in this chapter I talked about how I ran into a problem using the original planned Network Layer together with a Subnet and how I had to readjust using a Routing Layer 4.8 instead. In chapter 5 I then presented a possible scenario testing my implementation, using the campus of the University of Düsseldorf (HHU) [hhu] and the results I would have expected from running this specific scenario.

This implementation enables the user to run realistic mobile network simulations by placing his own towers. This way it is possible to run simulations in order to find the best placement and mobile standards for cell towers in different areas. With the addition of the ad-hoc part to the implementation a routing decision could be added, as the nodes could choose between two different networks for sending or receiving their messages.

6.2 Future work

In this section I present some ideas for future work with this Hybrid Mobility Model and the simulator PeerfactSim.KOM [pee].

6.2.1 Reference Point Group Mobility Model

As already mentioned in 4.6 there are different Mobility Models. One of the *Group Mobility Models* is the *Reference Point Group Mobility Model* [gmm]. In this Model nodes build groups, each group has a node that is the so called *logical center*. The rest of the nodes follow the motions of this logical center - its destination, velocity, direction, etc., while each node still has its own reference point. The nodes then follow a predefined path consisting of checkpoints they have to reach within a time limit. This Model would work pretty well for the scenario I described in 5.1. As the nodes could represent groups of students moving together to the next lecture, the canteen, etc., which then again could be represented by the checkpoints the nodes are moving towards. This would not only be a good use for the university campus scenario, but also for scenarios in cities, in order to find the best placement, mobile standard and radius for cell towers.

6.2.2 Manhattan Grid Mobility Model

Speaking of scenarios in cities, the addition of a *Manhattan Grid Mobility Model* [man] would be great, as it uses a *grid road topology*. This means, the nodes are only allowed to move on the grid of horizontal and vertical streets on the map. On crossroads a probabilistic approach is employed, in order to determine the next movement of the node. The node will continue to move forward on the same street with a probability of 0.5, it will turn left with a probability of 0.25 and turn right with a probability of 0.25.

6.2.3 Routing decision

When the possibility of two different networks is given, the nodes need to be able to choose which suits their purposes best. For example one way to solve this could be always trying to first send the messages via the mobile network and if this is not working, because both or one of the nodes is not within the range of a cell tower, try to go via the ad-hoc network. So future work could include finding the best or one of the best routing decisions for the hybrid nodes and implementing it.

Bibliography

[3gpa] *3GPP - A Global Initiative*. http://www.3gpp.org/, . Accessed: 2017-06-04

[3gpb] *3GPP - GPRS and EDGE*. http://www.3gpp.org/technologies/keywords-acronyms/102-gprs-edge, . Accessed: 2017-06-04

[3lt] *3GPP - LTE*. http://www.3gpp.org/technologies/keywords-acronyms/98-lte, . Accessed: 2017-06-04

[arc] *Peerfactsim.KOM - Community Edition - Simulator Details*. peerfact.com/simulator-details/, . Accessed: 2017-06-03

[bna] *Bundesnetzagentur - EMF-Datenbank*. emf3.bundesnetzagentur.de/karte/Default.aspx, . Accessed: 2017-06-10

[cas] *Carsten Snider, KOM-S-0296, 2009, "Mobility Aware Peer-to-Peer Networking"*.

[cel] *Cell site*. https://en.wikipedia.org/wiki/Cell_site, . Accessed: 2017-06-14

[doc] *Peerfactsim.KOM-2011-Documentation.pdf (Page 6)*. peerfact.com/documentation/, . Accessed: 2017-06-03

[ekm] *Elektronik Kompendium*. https://www.elektronik-kompendium.de, . Accessed: 2017-06-04

[ekv] *Elektronik Kompendium - Bitrates for Mobile Standards*. `https://www.elektronik-kompendium.de/sites/kom/0910141.htm`, . Accessed: 2017-06-04

[emm] *A Survey of Mobility Models*. `https://www.cise.ufl.edu/~helmy/papers/Survey-Mobility-Chapter-1.pdf`, . Accessed: 2017-06-09

[ets] *ETSI - The European Telecommunications Standards Institute*. `http://www.etsi.org/`, . Accessed: 2017-06-04

[gmm] *A Group Mobility Model for Ad Hoc Wireless Networks*. `https://pdfs.semanticscholar.org/db5a/a7a856b6788b1e99d10069fb34eda7b84e7b.pdf`, . Accessed: 2017-06-09

[hhu] *University of Düsseldorf (HHU)*. `https://www.uni-duesseldorf.de`, . Accessed: 2017-06-03

[lte] *LTE*. `http://www.lte-infos.de/lte-ratgeber/wie-hoch-ist-bei-lte-die-reichweite/`, . Accessed: 2017-06-14

[man] *Manhattan Mobility Model*. `https://www.researchgate.net/figure/258841673_fig7_Figure-7-Manhattan-Mobility-Model`, . Accessed: 2017-06-11

[mbs] *Mobile Base Stations*. `www.mobilenetworkguide.com.au/mobile_base_stations.html`, . Accessed: 2017-06-14

[pee] *Peerfactsim.KOM - Community Edition*. `peerfact.com`, . Accessed: 2017-05-22

[sub] *Peerfactsim.KOM-2011-Documentation.pdf (Page 13).* peerfact.com/wp-content/uploads/2015/03/PeerfactSim.KOM-2011-Documentation.pdf, . Accessed: 2017-06-03

[tko] *Implementation and Evaluation of a Mobility Model in a Peer-to-Peer Simulator.* https://wwwcn.cs.uni-duesseldorf.de/publications/publications/library/Korfmacher2015a.pdf, . Accessed: 2017-06-14

[tso] *Tutorial Slides on PeerfactSim.KOM - Slide 8.* peerfact.com/wp-content/uploads/2015/03/PeerfactSim.KOM-2011-UPB-PG-Nodes-P2P-Simulations.pdf, . Accessed: 2017-06-14

[tud] *Technical University (TU) Darmstadt.* https://www.tu-darmstadt.de, . Accessed: 2017-06-03

[upb] *University of Paderborn (UPB).* https://www.uni-paderborn.de, . Accessed: 2017-06-03

[vol] *Speed of light.* https://en.wikipedia.org/wiki/Speed_of_light, . Accessed: 2017-06-14

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Bachelorarbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt zu haben. Alle Stellen, die aus den Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Düsseldorf, 16.Juni 2017 Kevin Kalagi

Please add here

the DVD holding sheet

This DVD contains:

- A *pdf* version of this bachelor thesis
- All LaTeXand grafic files that have been used, as well as the corresponding scripts
- The source code of the software that was created during the bachelor thesis
- The referenced websites and papers