HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Coupled simulation of road traffic, V2X-applications and V2X-communication via mobile cellular networks

Master's Thesis

by

Raphael Bialon

born in
Velbert

presented to

Professorship for Computer Networks
Prof. Dr. Martin Mauve
Heinrich-Heine-Universität Düsseldorf

August 2015

Supervisor:
Norbert Goebel, M. Sc.

Abstract

V2X applications are an important current and future part of road traffic safety. But, testing and evaluating these applications in real world scenarios is expensive, time consuming and even small application errors can invalidate whole scenarios. Thus, simulation opens up new possibilities for the testing and evaluation of such applications and use cases. The V2X Simulation Runtime Infrastructure (VSimRTI) provides an open framework for the coupled simulation of different processes. While it contains many simulation models for 802.11 wireless networks in both mobile and stationary situations, mobile cellular networks are more complex to simulate and not included in most network simulators.

The existing Trace Based UMTS Simulation (TBUS) model enables a fast, easy, reliable and precise simulation of mobile cellular networks with regards to selected and measured simulation scenarios. This thesis ports the TBUS model into the network simulator OMNeT++, providing an additional simulation model to be used for mobile cellular network use cases. Therefore, parts of the VSimRTI have been extended, modified and fixed to fully support the TBUS. A verification of the ported TBUS model to the original implementation demonstrates full functionality of the ported version. Extensions to process cellular network interferences and simulation input data from different measurements were defined and simplified implementations of those show their possibilities. A base framework for V2X applications using TBUS functionalities is introduced.

The ported simulation model and new extensions are evaluated using real-world recorded data and demonstrate the suitability of the TBUS model used in coupled simulation. Simulation results are compared to European Telecommunications Standards Institute (ETSI) standards and definitions. An emergency vehicle warning application has been developed to test the coupled simulations' functionality. The final evaluation verifies the capability of coupled simulation to compete with ETSI standards for a sufficient probability and proposes directions for adapted standards to be used in mobile cellular networks.

Acknowledgements

A lot of people supported me during my work on this thesis to whom I wish to express my gratitude.

First of all, I would like to thank my supervisor, Norbert Goebel, for all the time he took, the discussions on ideas, and the support during my whole thesis. It was a great, motivated and enjoyable work. His mentorship allowed a well-rounded scientific work with a great outcome. I also would like to thank Adrian Skuballa for the processed input data that has been used for my simulations and the input and discussion on implemented processes.

The DCAITI and FOKUS provided insight into parts of the sources of the V2X Simulation Runtime Infrastructure, enabling me to introduce an optimized port of the TBUS model into their software. I would like to thank Robert Protzmann, Michalis Adamidis, Bernard Ladenthin and Dr. Björn Schünemann for their technical support, documentations and the access to VSimRTI.

The professorship for Computer Networks provided a great amount of hardware and technical support. I would like to thank Prof. Dr. Martin Mauve for the working possibilities and Thomas Spitzlei for the interesting talks on IT and the provided hardware and support.

My friends, Andre Ippisch, Tobias Krauthoff and Daniel Neugbauer backed me during the work on my thesis, provided new input and proofread parts of my work. Thank you all for this great time.

And last, but not least, I would like to thank my family. You have always been there for me, regardless of the situation. Thank you for supporting me on my way.

Contents

List of Figures

List of Tables

List of Listings

Chapter 1

Introduction

Computer simulators are highly sophisticated programs that help understanding real world processes by emulating them in software. If high simulation precision is needed, simulation models become more and more complex. Highly specialized simulators are capable of estimating real-world processes within a defined error range. They allow the calculation of new components or small changes to existing ones in a fast, cheap, repeatable, reproducable and easy manner. These simulators exist for various use cases, there are network simulators, traffic simulators, environment simulators, application simulators, simulators for biological and chemical reactions, and many more. Simulators are given input data, on which they calculate the simulation outcome. This outcome can then be analyzed or used as input for other simulators.

While this process of forwarding simulation results into other simulators is serialised, many use cases require parallel feedback between simulators of different real-world processes. One of this use case is the development of Vehicle-to-X (V2X) applications and communication models. Vehicles have a high and complex mobility, mobile networks between vehicles add even more complexity. Extension of specialized simulators leads to convoluted code, unmaintainable structures and require additional work on in-depth simulation of extension processes. For example, network simulators and frameworks like Objective Modular Network Testbed in C++ (OMNeT++) and INET framework for Mobile Ad-Hoc Networks (INETMANET), which are specialized on network traffic simulation, include mobility extensions. But these extensions are by far not as precise as mobility simulators, such as the output of the traffic simulator Simulation of Urban Mobility (SUMO). Mobility models are often pre-defined and inflexible, whereas SUMO allows input data changes at runtime and is extremely flexible in terms of simulation control. To combine the advantages of highly specialized simulators and enable the parallel and interactive usage of these, the Daimler Center for Automotive Information Technology Innovations (DCAITI) [DCA] and Fraunhofer Institut für offene Kommunikationssysteme/Fraunhofer Institute for Open Communication Systems (FOKUS) [FOK] introduced the V2X Simulation Runtime Infrastructure (VSimRTI) in [Sch12]. This simulation infrastructure connects different simulators and enables the direct usage of simulation outcomes as the parallel input for other simulators. This thesis investigates the combined simulation of road traffic, mobile cellular networks and V2X applications with the corresponding simulators SUMO, OMNeT++ and application_NT. In addition, a the simulation model introduced in [GKMG14] is ported to the network simulator OMNeT++.

Therefore, chapter 2 outlines the existing research on coupled simulation of wireless mobile and cellular networks with road traffic and other simulators, and presents unique features of this thesis' implementation. In chapter 3, the used simulators and simulation techniques are introduced. The ported simulation model is defined in chapter 4, where all new components and changes to existing simulators are presented. Then, the combined simulation with the ported model is utilized to generate results of simulation scenarios according to the European Telecommunications Standards Institute (ETSI). Finally, the ported simulation model together with the combined simulators and the calculated results is summarized and analyzed in chapter 6, which also concludes future work on this topic.

1.1 Terms and Definitions

Terms, that can be found exactly as written in this thesis, are set in a `typewriter` font. Definitions of new terms are *emphasized* on their first occurrence and used as regular words in the following context. All acronyms and definitions can found in the glossary on page 65. Mathematical definitions will either be introduced before the equation containing them is placed, or in the sentence directly after the equation is given.

Chapter 2

Related work

This chapter gives an overview on related work and other simulators achieving similar goals. It also outlines the difference between the presented concepts and the Trace Based UMTS Simulation (TBUS) model.

2.1 TBUS

Introduced in [GKMG14], the TBUS forms the base for the simulation model used by this thesis. A verified implementation of this simulation model already exists and has been used for simulations presented in the defining paper. It simulates mobile cellular network conditions by utilizing real world network characteristics. The simulation model is further explained in 3.2.1, as many parts of this thesis' simulation model are based on the TBUS model. Implementing the TBUS model within the VSimRTI enables the coupling of this model with other simulators. This thesis further extends TBUS by introducing a cell-share model to simulate the influence of different mobile nodes within the same mobile cellular network cell. It also enables the usage of different real world network characteristic measurements within the same simulation scenario.

2.2 CCMSim

Similar to this thesis, C2X Channel Model Simulation (CCMSim) utilizes the VSimRTI to add another simulation model within existing simulators. CCMSim is a MATLAB[1]-based expansion to OMNeT++. It uses OMNeT++ as the network simulator for upper layers and places lower layer calculations into a MATLAB extension. Presented in [PMOR12], it introduces an interface between OMNeT++ and MATLAB within the VSimRTI. CCMSim operates on real world measured channel characteristic data, obtained via specialized hardware. It utilizes this data to model the channel for mobile, wireless network transmission via 802.11p[2]. Using empirical simulation methods and data, CCMSim is able to simulate a node on every position inside the measured simulation area. The TBUS implementation used for this thesis uses a different approach and processes mobile cellular network data. Network characteristics are measured for discrete positions in the simulation area, positions in between receive optimized chosen network characteristics. Physical channel characteristics are not used for this thesis, as all needed mobile cellular network characteristics have already been gathered from measurement drives. The TBUS model adapted by this thesis utilizes data that can be gathered with simpler methods than the hardware and techniques CCMSim requires. Only roads that the simulation takes place on have to be measured, while the measurement drive is an ordinary ride through regular traffic. The overall complexity of the TBUS model compared to the CCMSim is lower.

2.3 MiXiM

Mixed Simulation (MiXiM) is a combination of various simulation models and frameworks in OMNeT++ and aims at simulating mobile wireless networks. In [KSW+08], the authors describe MiXiM in detail and also explain the simulated layers in wireless 802.11 networks. MiXiM offers an in-depth simulation framework which also uses the *Mobility Framework* (see [Mob]) for node movements with predefined or random routes.

[1] `http://www.mathworks.com/products/matlab/`
[2] `https://www.standards.its.dot.gov/Factsheets/Factsheet/80`

Wireless communication is calculated precisely and can be thoroughly configured. The TBUS simulation model does not make any use of the calculation models available in OMNeT++ and its frameworks. Instead, it utilizes its own simulation model and calculations, which have a lower complexity but maintain precise simulation results. There is no interface to couple MiXiM with any other simulator, let alone a traffic simulator. While MiXiM also implements precise calculation and a realistic projection of how wireless communication in 802.11 networks works in reality, TBUS cannot make use of these modules and instead introduces its own.

2.3.1 Veins

Vehicles in Network Simulation (Veins) combines OMNeT++ and MiXiM with the road traffic simulator SUMO. As presented in [SGD11], Veins provides a bidirectional coupling with possible feedback and reactions on simulated events and values. To obtain this strong coupling, Veins uses Traffic Control Interface (TraCI) (see [WPR$^+$08]) as the control protocol. TraCI offers a broad interface to read all available SUMO values and change some values by e.g. requesting a vehicle to brake or change its route. This enables Veins to integrate vehicular traffic data and send feedback to SUMO for possible changes in how the vehicles behave, but restricts its use to only a bidirectional coupling between SUMO and itself. Veins is not easily extendable to include other simulators, because no general standardized protocol for communication between simulators is defined.

2.4 VSimRTI_Cell

VSimRTI includes an integrated cellular network simulator, namely VSimRTI_Cell since version 0.13.0. In the current version 0.14.0, VSimRTI_Cell offers a configurable delay simulation. Based on *delay regions* and a given *delay type* (constant delay, random delay with simple and gamma distributions, and a random delay with a gamma distribution including the current node speed), the network's transport delay is calculated. VSimRTI_Cell acts on probabilistic models, thus the simulation area has to be measured and divided into delay regions. Depending on the size of delay regions and quality of measured or estimated data, the simulation outcome might differ significantly from reality. Packet ordering is not preserved and the backbone delay is assumed as a constant value and thus not simulated. TBUS, on the other hand, preserves packet ordering and takes backbone delay of the measured mobile cellular network provider's infrastructure into account. It also allows a more precise simulation of the measured network characteristics while maintaining a low complexity.

Chapter 3

Fundamentals

This chapter presents the fundamentals of this thesis. Showing the differences between trace-based Global Positioning System (GPS) and graph simulation and the methods used for each case. By outlining the used and modified software and libraries, it introduces the technologies used for this thesis.

3.1 Terminology

This thesis uses the term V2X and provides a framework suitable for every kind of vehicle that can be simulated using the traffic simulator SUMO coupled with VSimRTI. Vehicle-specific functions and methods can be modelled on top of it, so that Car-to-X (C2X) applications, belonging to a superset of vehicle-specific applications, can take full advantage of functions and methods provided by TBUS. Simulations utilize position based *network characteristics*, for this thesis defined as bit rate, loss probability and backbone delay.

3.2 Trace-based simulation model

Trace-based simulation, as introduced in [GKMG14], uses data gathered in real world scenarios rather than data generated by probabilistic models. This ensures a result closer to reality than probabilistic models because the gathered data, although it might have been processed, represents a snapshot of reality. The next sections explain the Trace Based UMTS Simulation (TBUS) simulation in section 3.2.1 and compare two possibilities of trace-based simulation in sections 3.2.2 and 3.2.3.

3.2.1 Trace Based UMTS Simulation

TBUS simulation is based on the paper *Trace-based Simulation of C2X-Communication using Cellular Networks* by Norbert Goebel, Dr. Markus Koegel, Prof. Dr. Martin Mauve and Jun.-Prof. Dr. Kálmán Graffi (see [GKMG14]). While regular network simulators for wireless and cellular networks use mathematical probabilistic models to estimate network characteristics, the TBUS simulator uses real world measured position based network characteristics to model the mobile cellular network. Mobile cellular network providers do not offer lots of information about their networks. Although network protocols are standardized and physical parameters are known, exact mathematical models of mobile cellular networks are very complex. In addition, the exactness of their results varies with the quality of input data such as environmental interference, network interference and bandwidth. Parameters of every antenna like available bandwidth ranges, transmit and receive power, alignment and many more have to be gathered as input for the mathematical model of the simulation. Small changes of input data can have a huge impact on simulation results and have to be verified before their usage.

Simulations using traces, on the other hand, allow an easy and realistic measurement of the simulation area. Multiple measurements can be merged using various methods and can be selected for only parts of the simulation area. This idea is investigated further in [Sku15]. Additional research on how concurrent nodes in one cell influence each other's network characteristics is done in [Kar15].

Information on mobile cellular network characteristics is collected by using the Rate Measurement Framework (RMF) as presented in [Wil12]. An additional measuring algorithm is discussed in [Kra14]. Measurement can either be done on x86 hardware with GPS- and Universal Mobile Telecommunications System (UMTS)-modules, as explained in [Lan13], or on Android[3]devices, as described in [Olf13].

3.2.2 Simulating with GPS-based traces

By default, node movement is simulated by using two dimensional Cartesian coordinates on an even plane. When applying measured traces onto Cartesian coordinates, GPS inaccuracy has to be taken into consideration: Network characteristics measured and positioned by a GPS system are inaccurate by $\pm 6\,\mathrm{m}$ in every direction, explained in [DG08]. Measured traces do not follow roads exactly but are merely distributed within a specific range around the real world coordinate, influenced by environmental and other interferences.

They can be mapped onto streets and lanes, but as they belong to a point in a two dimensional plane with x and y axes, it can eventually happen that two distinct network characteristics, even measured from different vehicle headings, are mapped onto the same point in space and cannot be differentiated during simulation. While this error is unlikely to occur, another problem arises: If network characteristics from one measurement drive, visiting a road on different directions, have been measured and mapped onto one road, simulation methods may only differentiate between both directions if the heading is saved in addition to the vehicle's position and can be calculated within the simulator later on. The exact definition of this problem and the solution are presented in the next section.

[3]`http://www.android.com/`

3.2.3 Simulating with graph-based traces

As proposed by Norbert Goebel[4], using directed graph-based traces enables easier position to data matching. Although graph-based traces contain a tuple of data just as GPS-based data, the informational content is greater. A GPS tuple (x,y) contains information on a point in a two-dimensional plane. No additional information can be extracted from a single tuple and there might be network characteristics from measurement drives along different directions mapped onto one tuple.

In contrast, a graph tuple (r,l), with r being the current road id and l being a continuous position on the edge represented by r, contains more information than just the tuple's elements. Let $P = \{\,(r,l) \mid r \in E, l \in \mathbb{R}^+\,\}$, with E being the graph's edge set, be the set of all graph-based positions. Then I is our information space – a three dimensional space where every point is associated with its three rotation angles:

$$I = \left\{ (c,\varphi) \,\middle|\, c = (x,y,z) \in \mathbb{R}^3,\ \varphi = (\alpha,\beta,\gamma) \in [0,2\pi[^3 \right\} \tag{3.1}$$

Using P and the underlying graph $G = (V,E)$ with V being the graph's vertices set, the following function provides more information than there is contained in P by mapping P to I:

$$f_G : P \to I, \quad (r,l) \mapsto (c,\varphi) \tag{3.2}$$

While this mapping is not bijective, it is injective and as such, it enables the distinct mapping to a six dimensional information from a two dimensional input space – and it could be even more, if I is expanded with more information mapped on G.

For this thesis, the exact position of a vehicle is now given through the road id and lane position tuple. There is no need to map this tuple onto a greater information space and then retrieve network characteristics, and so the information space is exchanged with the following:

$$I_{\text{TBUS}} = \left\{ (r_{\text{bit}},\, r_{\text{drop}},\, d,\, c) \,\middle|\, r_{\text{bit}} \in \mathbb{R}^+, \quad r_{\text{drop}} \in [0,1], \quad d \in \mathbb{R}^+, \quad c \in C \right\} \tag{3.3}$$

where r_{bit} is the bit rate in ns, r_{drop} is the drop rate, d the delay of the packet travelling

[4]goebel@cs.uni-duesseldorf.de

through the mobile cellular network provider's backbone network and c the id of the currently connected cell with C representing all available cell ids. The mapping of P, extended with a simulation time variable t used to differentiate between multiple measurement inputs, onto I_{TBUS} with the underlying graph G is realised by a database query outlined in section 4.3.7, represented by

$$f_G^{\text{TBUS}} : P \times \mathbb{N}^+ \to I_{\text{TBUS}}, \quad (r,l,t) \mapsto (r_{\text{bit}}, r_{\text{drop}}, d, c). \tag{3.4}$$

Having these prepared data, an accurate simulation of mobile cellular data depending on position and direction towards the network's antennas is possible. The mapping of vehicle positions to network characteristics, belonging to the vehicle's position and heading, has been reduced from input data of dimension six with six continuous and finite components to input data of dimension three with data of all components being finite, two of these having continuous data and one component, the set E, being discrete.

Using graph-based simulation requires special data preparation. This *raw* data has to be map-matched to the used route, as the GPS positions from measured traces are inaccurate, and the graph used for the graph-based simulation does not represent the road network exactly – corners and other shapes can only be approximated. Using map-matching, the measured network characteristics have to be mapped onto the graph's corresponding edge and distance on the edge. Mapping recorded data to edges and logging additional information is presented in [Sku15].

3.3 The VSimRTI framework

The VSimRTI, as explained in [Sch12], is used as the base framework for simulator coupling. It provides a High Level Architecture (HLA), which is standardized in [IEE10] and described in [KDW00], with already predefined ambassador and federate modules for

Traffic simulators: VISSIM and SUMO
Communication simulators: ns-3, OMNeT++, JiST/SWANS and VSimRTI_cell

Application simulator: application_NT

VSimRTI does not interact with connected simulators directly, instead a pre-defined interface for communication with a simulator's ambassador is used. This ambassador connects to the simulator's federate and finally the federate translates all given commands into simulator-native actions. This can be seen in figure 3.1. This section gives

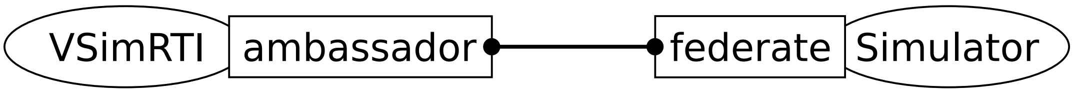


Figure 3.1: Interaction between VSimRTI and a simulator through an ambassador and federate structure

an overview of VSimRTI and the three simulators and frameworks used for coupled simulation – namely OMNeT++ and INETMANET in section 3.3.2 as well as SUMO in section 3.3.3.

3.3.1 VSimRTI

VSimRTI exchanged the default OMNeT++ scenario manager with its own, enabling a synchronized simulation time advance with other simulators coupled with VSimRTI. Also, dynamic node creation and removal at runtime as well as node movement given by other simulators is integrated into the OMNeT++ simulation core. Applications simulated with the application simulator application_NT can send messages through VSimRTI which are forwarded to OMNeT++ either via their length or content, depending on needed information and whether a faster or slower but more in-depth simulation is demanded. Sending nodes can add headers and information to messages before the physical layer routes and simulates network characteristics on them. Receiving nodes are able to inspect and modify incoming messages before they are delegated to the corresponding application simulated by applicationNT.

3.3.2 OMNeT++

OMNeT++ is a network simulation framework originally written and still maintained by András Varga at the Department of Telecommunications of the Technical University of Budapest. It consists of a library of C++ modules and a simulation manager, forming the foundation for network simulations. Network structures are defined in so-called Network Description (NED) files which can also be hierarchically ordered and support inheritance, and simulation settings are placed in an Initialization (INI) file. This enables a quick and easy change of network characteristics without the need of compilation. Parts of the network such as Network Interface Cards (NICs), network layers as User Datagram Protocol (UDP) or Transmission Control Protocol (TCP), or even wholly equipped nodes can be modelled in a reusable manner. Some modules are interchangeable which allows a simple comparison of the simulation results of two components within the same scenario. Because OMNeT++ is written in C++, the memory usage is low and the compiler can take full advantage of Central Processing Unit (CPU) extensions to speed up simulation execution. Then again, single-threaded execution is given by the event-based simulation management and OMNeT++ without multi-threading extensions does not scale with the number of used cores. In addition to these modular simulation tools, OMNeT++ offers various aggregation methods and visualization of statistical data collected through every simulation run.

The INETMANET framework

The INETMANET framework is a collection of community-developed modules written for OMNeT++ (see [INEa]). It contains an implementation of the internet stack, Mobile Ad-Hoc Network (MANET) protocols, physical link protocols, application models and network devices. Basing on OMNeT++'s modular structure, INETMANET allows an easy construction of new devices and protocols while relying on existing parts of the framework. It is thoroughly tested and under constant development with new protocols and features added with every version. Part of this framework is the *Mobility Framework* [Mob], which enables the usage of moving nodes. Various movement models such as random movement, pre-defined movement or following models are implemented and allow the simulation of mobile networks with changing network characteristics and setups.

3.3.3 SUMO

SUMO is a microscopic traffic simulator. Based on a two-dimensional road graph, SUMO simulates public transport, all kinds of vehicles and traffic lights. Vehicle types can be defined in detail – external vehicle parameters such as the vehicle's length and its minimum gap towards the next vehicle ahead and internal parameters like acceleration and deceleration, its maximum speed and car-following models are only a few examples of the many options SUMO offers – and are simulated with in-depth output, too. Simulated vehicles follow pre-defined routes given as a list of graph edges, but it is also possible to change routes dynamically during runtime by interacting with SUMO. The protocol used for simulation control, besides configuration values, is the Traffic Control Interface (TraCI). It can be used to control the simulation flow or to interact with vehicles, traffic lights and any other configurable and changeable items simulated by SUMO.

Road graphs can be generated using SUMO's *netconvert*. It supports many map file formats with *OpenStreetMap*[5]being the most popular one. Mapping the three-dimensional real world simulation onto a two-dimensional plane is achieved by SUMO using the proj library[6], a cartographic projection library provided by the Open Source Geospatial Foundation (OSGeo). By default, SUMO configures libproj to use a Universal Transversal Mercator (UTM) projection on the standardized WGS84 ellipsoid (see [Nat00]). Calculations can then be done on the two-dimensional plane and are mostly vector-based, there is no need for taking the earth curvature into consideration. This speeds up calculations while maintaining a reasonable low error towards the exact three-dimensional coordinates.

Within VSimRTI, SUMO takes a special position as it has no need for a federate but is directly controlled via TraCI. The ambassador abstraction layer does not include every command TraCI supports, but VSimRTI offers the possibility to directly interact

[5] `https://www.openstreetmap.org/`
[6] `https://github.com/OSGeo/proj.4`

with SUMO by using so-called `SumoTraciBinaryMessages` which support the full TraCI. SUMO is also no event-based simulator and as such would not require *time advance* messages, because it returns simulated data if given a timestamp until which it simulates to. VSimRTI sets a fixed interval for SUMO updates which provides SUMO's resolution and can be freely configured.

Chapter 4

Implementing TBUS for VSimRTI

This chapter explains the changes made to existing VSimRTI components and introduces new components necessary for TBUS simulations. While VSimRTI offers simulator coupling through time synchronization between simulators and exchange of simulation data on demand, the introduction of TBUS for OMNeT++ required additional data for a more precise simulation. The adapted simulation model is verified against the original implementation. Finally, an installer script is introduced to provide an easy installation method.

4.1 Extending VSimRTI

VSimRTI is distributed and licensed by the DCAITI and FOKUS. The infrastructure core and simulators like application_NT and VSimRTI_Cell are closed-source, while the framework structure and a well-documented interface for the integration of new simulators into the infrastructure are open-source and enable the use of TBUS with a broad variety of different simulators. VSimRTI offers information needed for graph-based simulation delivered by SUMO, but only in application_NT as part of a vehicle's information. To obtain this information, the OMNeT++ ambassador and federate have been extended to submit additional data needed for graph-based simulation. This can be seen in 4.1. Some parts of the INETMANET needed alterations to comply with the TBUS implementation. Those changes are explained in section 4.2. One of the greater parts of this thesis, the implementation of TBUS in the VSimRTI, is outlined in section 4.3

Edge value retrieval With friendly support of the DCAITI and FOKUS, graph-based trace simulation is implemented in VSimRTI's OMNeT++ ambassador and federate. The existing protocol is extended in such a way that backward compatibility is given. *Extended information* refers to the new position model based on directed graph-based positions and introduced with TBUS for VSimRTI as described in 3.2.3. *Regular information* identifies the commonly used position model based on Cartesian coordinates on a two-dimensional plane. Already existing projects are not affected by this extension, as only a new base class called `TbusMobileNode` receives extended information updates. Nodes with extended information can co-exist with regular information nodes in the same simulation – this enables the parallel usage of INETMANET modules with TBUS modules (although there are some restrictions, see section 4.2.1).

It has to be noted that occasionally vehicles provide an empty road id but correct lane position. This is due to the algorithm SUMO uses to simulate vehicle movements: Move-

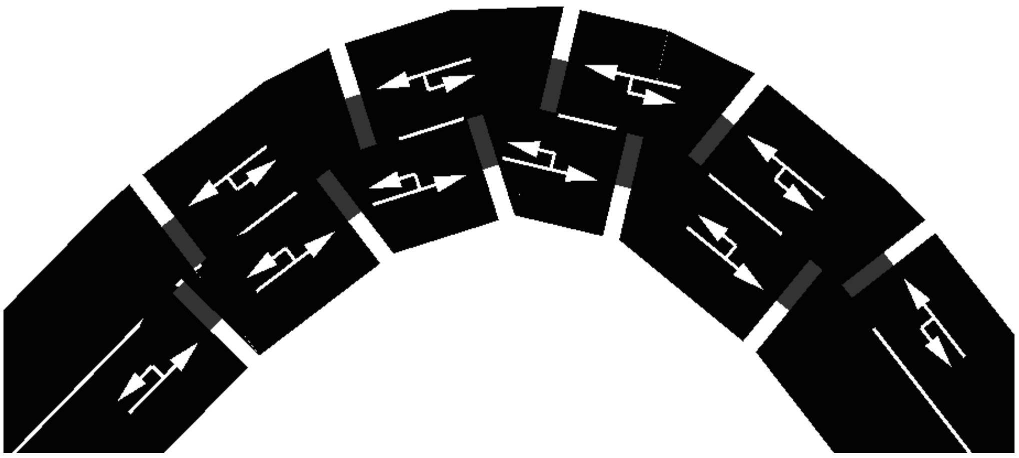


Figure 4.1: Graphical representation of SUMO edges

ments around corners or other special shapes for example across intersections are simulated on *internal edges* which are not represented in the graph – see figure 4.1 for an example, spaces between edges are each filled with an internal edge. Also, if a vehicle joins the simulation but its starting point is blocked by another vehicle already being there, the vehicle is joined with an empty road id and a lane position value of 0 until the starting point becomes available.

4.2 Patching the INETMANET framework

OMNeT++ provides the simulation platform, while the INETMANET framework introduces implementations of various network components, protocols and tools. It enables OMNeT++ to extend nodes with mobility and movement capabilities, includes a publish-subscribe-messaging module and many more. Most of its components are incorporated into TBUS as they are, but two of them required more attention and patching. The following sections 4.2.1 and 4.2.2 introduce patches for altered implementations of INETMANET modules, which still preserve their functionality.

4.2.1 ARP

Because OMNeT++ and parts of the INETMANET framework are designed for predefined wired and wireless networks, most of its components do not handle dynamic node adding and removing at runtime, as well as moving nodes, the right way. Especially the Address Resolution Protocol (ARP) module crashed when used in combination with dynamic node creation. ARP is not necessary when using the TBUS simulation model, the connection between a mobile node and a base station is merely a point-to-point-connection and addressing is carried out by the corresponding provider. Further delays caused by ARP in the providers backend or other networks are included in the measured delay values. Thus, the TBUS implementation in VSimRTI makes use of INETMANET's so-called *global ARP* (see [INEb, line 84]), enabling instantaneous address resolution and disabling ARP delay.

Attention had also be given to the destructor of the global ARP cache. Upon removing entries, this module checked each Internet Protocol (IP) address for its existence to prevent logical errors within protocols. VSimRTI divides a part of the IPv4 address space from `0.0.0.0` to `3.255.255.255` into four parts as follows in table 4.1. The first vehicle for example is assigned with the IP `0.0.0.0`, the second with `0.0.0.1` and so on. An overflow in a low byte results into an increment of the next higher byte, excluding the highest byte which is only used for categorization of the modules. This way, VSimRTI can assign IP addresses to $255^3 = 16\,581\,375$ nodes in each category. The IP address

Address range	Category
`0.0.0.0` to `0.255.255.255`	Vehicle
`1.0.0.0` to `1.255.255.255`	Road Side Unit (RSU)
`2.0.0.0` to `2.255.255.255`	Traffic light
`3.0.0.0` to `3.255.255.255`	Charging station

Table 4.1: IP address distribution from VSimRTI's `IPResolver`

assigned to the first vehicle leads to an error with INETMANET's IP and ARP implementations upon removing IP addresses on vehicle removal. The IP address `0.0.0.0` is internally defined as *<unspecified>* and is therefore not recognized as a valid IP address. Even if compared to the same instance, the comparison operator returns false and leads to the logical assumption of this IP address not being assigned to any vehicle at all. This results in an exception thrown by the INETMANET framework and crashing the simulation at the point at which the first vehicle left the simulation. The VSimRTI developers have been informed of this problem and will provide a fix by restructuring the IP address distribution within the next releases.

4.2.2 ChannelAccess and ChannelControl

Connections between nodes in OMNeT++ are usually designed as *connections* between *gates*, that means physically available one-to-one connections. Because the wireless interface offers a one-to-many connection, INETMANET uses a different and loosely coupled design approach. Vehicles equipped with wireless interfaces use a physical layer derived from `ChannelAccess` (or `ChannelAccessExtended` respectively, if extended functionality is required) which interacts with `ChannelControl` (or `ChannelControlExtended` respectively), acting as the physical medium. To comply with VSimRTI's Wireless Locale Area Network (WLAN) equipped vehicles design, the TBUS implementation extends INETMANET's `ChannelAccessExtended` by `TbusMobilePHY` and `ChannelControlExtended` by `TbusChannelControl`. Because `ChannelControlExtended` lacks some of the functionality available in `ChannelControl`, especially null-pointer checks, this functionality was introduced into `ChannelControlExtended` as a patch. More recent versions than the one VSimRTI requires have already worked out this error. This patch enables the switch between WLAN- or TBUS-equipped vehicles by only changing OMNeT++'s configuration file.

4.3 Implementing TBUS in OMNeT++

Having all necessary extensions and adaptions to the VSimRTI, the following sections introduce the most important modules and classes developed for the TBUS implementation in VSimRTI. All of these classes are written in C++ and modules are designed in OMNeT++'s NED language. The TBUS extension is then compiled as a library which is linked into VSimRTI's OMNeT++ adaption. Figure 4.2 ranges the TBUS extension for OMNeT++ in VSimRTI into the whole infrastructure.

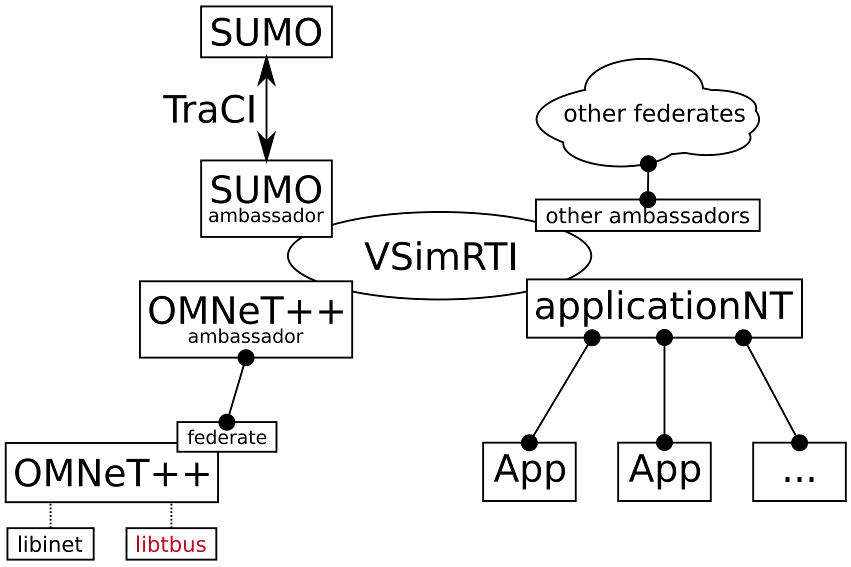


Figure 4.2: Overview of VSimRTI components

4.3.1 TBUS mobile node

A mobile node using TBUS methods is modelled similar to VSimRTI's `Vehicle` module. To comply with the International Organization for Standardization/Open Systems Interconnection (ISO/OSI) model as described in [Zim80], an analogous layer structure for the network stack is implemented. A combination of INETMANET, TBUS and VSimRTI modules are used to model the corresponding layers as can be seen in table 4.2. The resulting module is designed as a compound module in OMNeT++ and is presented in figure 4.3, with Medium Access Control (MAC) and Physical layer (PHY) layers integrated into the TBUS radio `TbusMobileRadio` displayed in figure 4.4. The structure of a `TbusMobileNode` is kept simple, as the logic and more complex structures are encapsulated in the `TbusMobileRadio` module, which is explained in section 4.3.3. Therefore, the `TbusMobileNode` is merely a compound module combining and connecting all necessary modules.

Layer		Module	Source
5	Application	`VSimRTIUnreliableApp`	VSimRTI
4	Transport	`UDP`	INETMANET
3	Network	`NetworkLayer`	INETMANET
2	Data link	`TbusMobileMAC`	TBUS
1	Physical	`TbusMobilePHY`	TBUS

Table 4.2: TBUS layer concept

4.3.2 TBUS inet node

The `TbusInetNode` represents a wired server that is able to connect to mobile nodes via the `TbusChannelControl`. It also implements the network stack as specified in the ISO/OSI model, but has no delays or packet loss on its own connections. These delays are already included in the measured backbone delays and need no additional simulation. The inet node allows applications to run on top of it via the application_NT simulator and provides lower network layers as well as an theoretically unlimited bandwidth access.

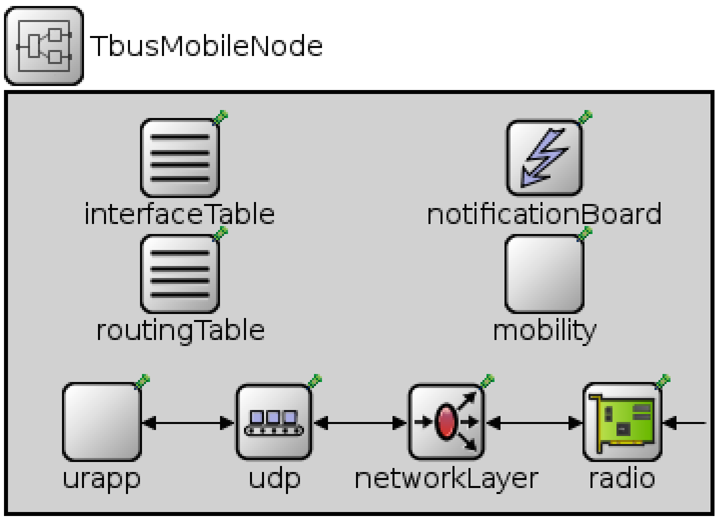


Figure 4.3: `TbusMobileNode` NED structure

4.3.3 TBUS mobile radio

The TBUS radio module `TbusMobileRadio` represents the connecting layer between a node and the physical transmission medium. It therefore combines the MAC and PHY modules `TbusMobileMAC` and `TbusMobilePHY` via four TBUS simulation queues and one `TbusQueueControl` module. MAC and PHY layers are further explained in section 4.3.4 and the simulation queues and queue control are introduced in section 4.3.5. The `TbusMobileRadio` module is meant to be exchangeable with other radio modules, so that mobile nodes can be equipped with different radios for different purposes.

4.3.4 TBUS MAC and PHY layers

Connecting the network layer with the physical access medium, these layers play an important role during simulation. The MAC module `TbusMobileMAC` allows upper network layers to send a packet correctly addressed into the network, the PHY layer module `TbusMobilePHY` connects the mobile node with the air interface represented by `TbusChannelControl`, which is introduced in section 4.3.6.

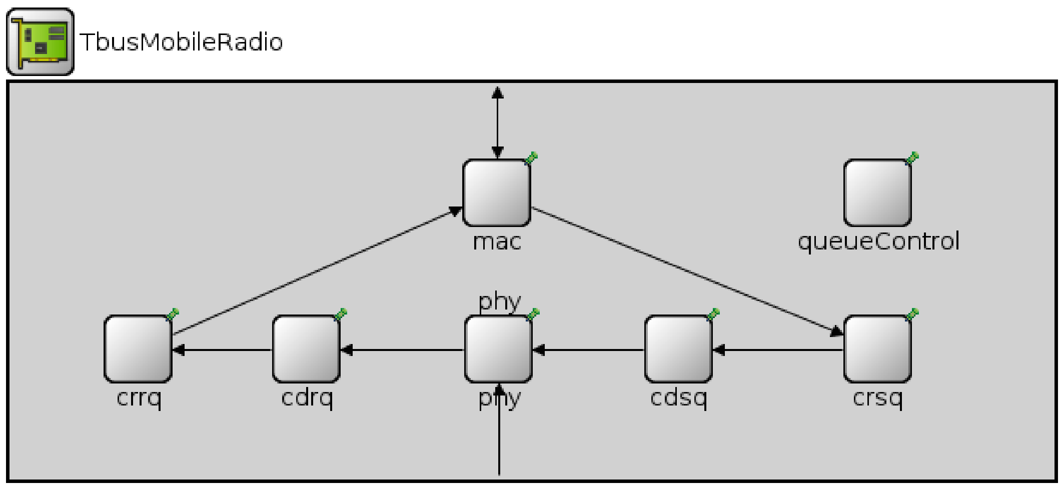


Figure 4.4: `TbusMobileRadio` NED structure

Following INETMANET's concept, the TBUS MAC layer module `TbusMobileMAC` initialises the NIC with a simulation-wide unique MAC address as well as standard values for the options listed in table 4.3. To assign calculated delays or timestamps used for

Option	Value
Name	`tbus0`
Multicast	`true`
Broadcast	`true`
Down	`false`
MTU	`1500`
Data rate	`100.0`[a]

[a]This value is only used as a dummy value and is overwritten by TBUS values during simulation

Table 4.3: TBUS NIC options

further calculations, a *control info* is appended to each packet before entering queues and removed after leaving all queues in one node. This is done for each packet because the delays differ depending on network characteristics, packet size and the time of a packet entering the simulation. Control infos are OMNeT++'s way of storing information on a packet readable for every layer or module accessing it. This `TbusControlInfo` contains three time values:

`queueArrival`	Simulation time of packet arrival in queue

`earliestDelivery` Simulation time of earliest queue dispatch
`headOfQueue` Simulation time of when the packet became head of queue

While `queueArrival` and `headOfQueue` timestamps are used for the calculation of queue delays, `earliestDelivery` is given to OMNeT++ as the event time of queue dispatch.

4.3.5 TBUS queue design

As introduced and explained in detail in [GKMG14], network characteristics in the TBUS simulation model are realised with the use of queues and events. The TBUS model utilizes four queues on each node, their setup is shown in figure 4.5. Queues exist for calculations on the down- and upstream side of bit rate and loss, and backbone delay. Those calculations result in delays representing the packets' transmission time over the air interface or through the network's backbone, or in a loss probability for the packet they are calculated for. Because nodes influence each other while connected to the same mobile cellular network, network characteristics are processed to meet the current networks' state by the cell-share model introduced in section 4.3.8. The C++ implementation makes heavy use of inheritance and template classes, as large parts of the queue's code relay on the same structure. This leads to a clean and easy-to-maintain code base. Network characteristics are assigned for each queue with an on-demand push and pull model, which is explained in the following.

Push model For the push model, a queue's state can maintain one of the following three states:

INACTIVE The queue has no packets enqueued and does not need to store any network characteristics. Used by all queues.
ACTIVE The queue has packets enqueued and stores assigned network characteristics. Used by backbone delay queues (CDRQ and CDSQ).
CELL_ACTIVE Same as *ACTIVE*, but the queue's and as a result the node's bit rate usage influences other node's behaviour. See section 4.3.8. Used by bit rate delay

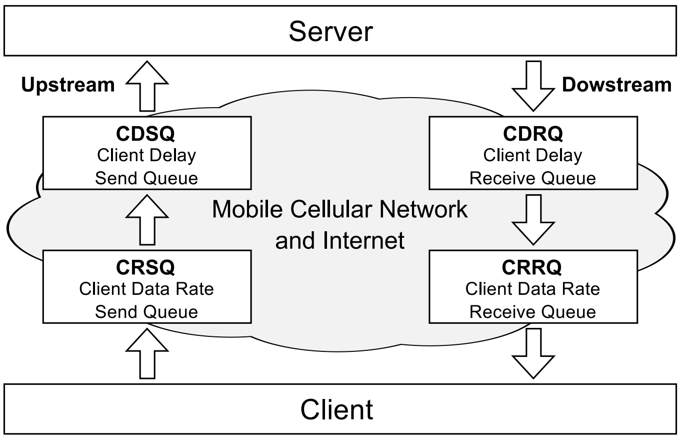


Figure 4.5: TBUS queue connections, taken from [GKMG14]

and packet loss queues (CRRQ and CRSQ).

Active queues are now defined as queues either in the *ACTIVE* or *CELL_ACTIVE* state. To improve performance for the push-model and prohibit unneeded network characteristics requests to the data source, only active queues receive push value updates at update triggers. This update triggers is identified as node movement, network characteristics for the new position are retrieved from the data source. The network characteristics are then processed by the cell-share model and assigned to active queues. In addition, an update trigger used by the cell-share model, is the activity change from *INACTIVE* to *CELL_ACTIVE* state. It results in an update of the queues' processed network characteristics, but not the network characteristics from the data source as no node movement has taken place.

Pull model Pull updates only occur on a queue's activity change from *INACTIVE* Queues pull new network characteristics when a packet enters the queue and no previous network characteristic has been stored because the queue has previously been in an *INACTIVE* state. This pull results in an update from the data source, additional updates triggered by the cell-share model might also occur afterwards.

Processed network characteristics are then used to calculate transmission and backbone delays and loss probability for each packet. Calculated delays are stored in a packet's control info, introduced in section 4.3.4.

Each queue, as well as every other component of the TBUS C++ implementation, was thoroughly tested and is equipped with assertion clauses to warn of runtime errors. Assertions are compiled into OMNeT++ and the TBUS extension and have to be explicitly disabled by using the compilation flag `-D NDEBUG`. The C++ implementation from this thesis has been compared with the original Java implementation from [GKMG14] and evaluated with the original simulation model in section 4.4.

4.3.6 TBUS Channel Control

To coordinate the physical medium access on the air interface, a singleton instance of a *Channel Control* module based on INETMANET's `ChannelControlExtended` is introduced. It has an omnipotent view of all nodes and their interfaces and thus controls message routing between mobile nodes as well as inet nodes. Supporting unicast and broadcast to all nodes within the mobile cellular network, it provides basic functions to maintain a reliable network structure.

4.3.7 Data source

For an easy retrieval of position based network characteristics to be used in the simulation, a data source interface is defined in the TBUS implementation. A SQLite[7] implementation is included with this thesis, though the interface allows implementations of other data sources to be used in the TBUS implementation. The corresponding Structured Query Language (SQL) table structure is shown in appendix A.3. As OMNeT++ is an event-based simulator, the database is only accessed sequentially, thus no precautions on threaded and concurrent access have been taken. The query, constituted from the mapping in equation 3.4, works as follows. The road id r is mapped one-to-one onto the corresponding road id in the database column. This works because road ids are discrete values from a finite set, whereas the lane position l and simulation time t are continuous values from a theoretically infinite space, which is restricted by its digital representation. Also because of the measurement resolution, network characteristics are only available

[7]`https://www.sqlite.org/`

for discrete distances on the road and timestamps. The selection of the right network characteristics for this input is explained in the following.

When taking measurement drives for a simulation area, some nodes can be measured more than once, resulting in value groups with different network characteristics for the same edge. A simplified example measurement drive is shown in figure 4.6, where the lower middle edge is visited two times and has two value groups, whereas every other edge only has one value group. Each set of network characteristics belonging to one value group is identified by a *group id*. If a network characteristics is only chosen by the

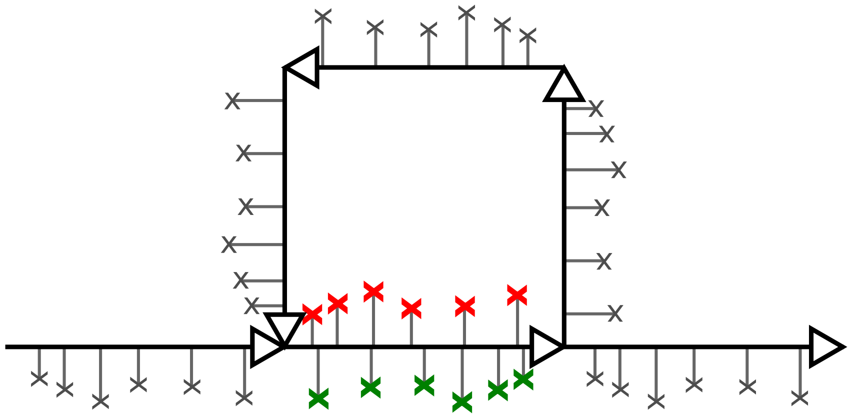


Figure 4.6: Network characteristics mapped onto a simplified road graph

shortest distance to the current simulated position, the data source would return network characteristics from either value group in an oscillating way. Thus, taking simulation time as a strictly monotonically and continuous increasing unit into account, the id of a value group to be used is chosen depending on the current simulation time as follows. The group id is chosen as the group id of a network characteristic whose timestamp is not placed in the future, relative to the current simulation time, and whose time gap between the current simulation time and the network characteristics' timestamp is the overall minimum. If no such entry exists, the group id of an overall network characteristic with a timestamp closest to the current simulation time is chosen. This identifies a group id g, so that

$$\min\{\|t-\tau\|\}: \quad \tau \in \mathrm{T}_g, \quad t \geq \tau, \quad g \in G_r, \tag{4.1}$$

with $\|\ldots\|$ as the euclidean distance, T_g the finite set of measurement timestamps for group g and G_r the set of all group ids for edge r. If no such g can be found, the restriction

$t \geq \tau$ is removed so that at least one group id can be used if any have been mapped onto the edge. This way, if there are network characteristics mapped onto an edge, there is always at least one value group returned, defined as network characteristics measured within the group id g. It is then to be further restricted by the position on the edge as shown in equation 4.2. As only the first value group is accessed by timestamps from before the group measurement started, no invalid network characteristics are chosen. Without having this relaxation of equation 4.1, there would be no network characteristics for some timestamps and with this relaxation only the beginning timestamp of when the first value group's network characteristics are valid is corrected to the simulation start time.

Having the group id g, choosing the correct network characteristic for a given lane position l is different: Network characteristics with the given group id g are chosen by the smallest distance towards the given lane position l. With g as the group id of the measured value group and λ being a measured lane position within Λ_g, the set of all measured lane positions of group g, the following condition has to be met:

$$\min\{\|l-\lambda\|\}: \quad \lambda \in \Lambda_g. \tag{4.2}$$

With the resulting values r, λ and g the correct network characteristic can be chosen from the data source. The queries for retrieving the group id with and without the restriction given in equation 4.1 are shown in pseudo-SQL in listings 4.1 and 4.2, while listing 4.3 shows the pseudo-SQL query for network characteristics retrieval.

```
SELECT groupid, (t−τ) AS diff FROM table WHERE roadid = r AND τ <= t
  ↪ ORDER BY diff ASC LIMIT 1;
```

Listing 4.1: Pseudo-SQL restricted query for the group id

```
SELECT groupid, (t−τ) AS diff FROM table WHERE roadid = r ORDER BY
  ↪ diff ASC LIMIT 1;
```

Listing 4.2: Pseudo-SQL unrestricted query for the group id

```
SELECT *, abs(λ−l) AS dist FROM table WHERE roadid = r AND groupid =
  ↪  g ORDER BY dist ASC LIMIT 1;
```

Listing 4.3: Pseudo-SQL query for network characteristics

4.3.8 Cell-share simulation

First, the term *cell* used for this thesis' cell-share model has to be defined. It is merely a real mobile cellular network cell but a set of points having the same cell properties recorded as the simulation input. The placement of real network cells is not taken into account for this value, only the measured data is used. This enables the grouping of network characteristics from the same cellular characteristic, and, as a result, enables the grouping of simulated nodes using the same cellular characteristics.

Network characteristics gathered from measurement drives represent real data between the measurement node and the mobile cellular network provider's cell. During simulations, multiple nodes can be situated in the measured cell. Assigning every simulated node the same bandwidth would result in a large error, because the mobile cellular network cell's maximum network characteristics can be overestimated. Thus, a cell-share model was introduced, distributing the measured network characteristics according to network cell's loads. To create a cell-share model, situations that trigger a change in bit rate distribution are identified as node movement and node activity change (start/stop of sending/receiving). While node movement is triggered centrally by VSimRTI's *simulation manager*, node activity change is triggered by each node' bit rate queues individually. These two triggers result in a *cell change*, which is a change in the situation of simulated nodes within the measured network cells as shown in figure 4.8. A cell change affects all nodes situated in changing cells and every active queue in each affected node. Thus, a new omnipotent management class `TbusWorldView`, which coordinates the cell-share models' actions, was introduced. The calculation of bit rate distribution with

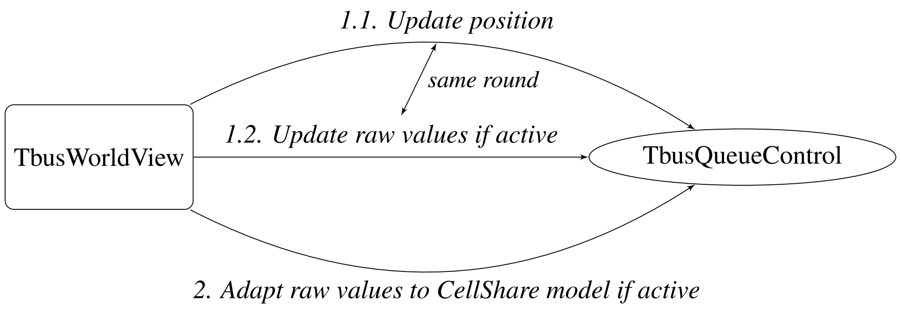
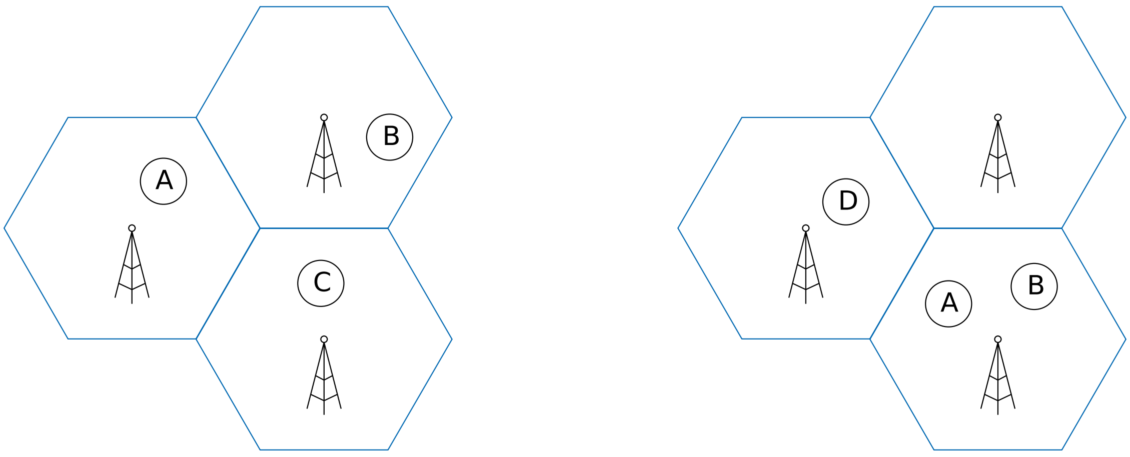

1.1. Update position
same round
1.2. Update raw values if active
TbusWorldView
TbusQueueControl
2. Adapt raw values to CellShare model if active

Figure 4.7: Round-based cell-share model

the cell-share model is round-based: While nodes can move and update their raw network characteristics from a data source independently, they need the final amount of active nodes in the same cell (see figure 4.8) to let the cell-share model calculate their share of the current network characteristics. These leads to two update rounds. The `Tbus-`



(a) Node distribution before movement (b) Node distribution after movement

Figure 4.8: Node movement in different cells

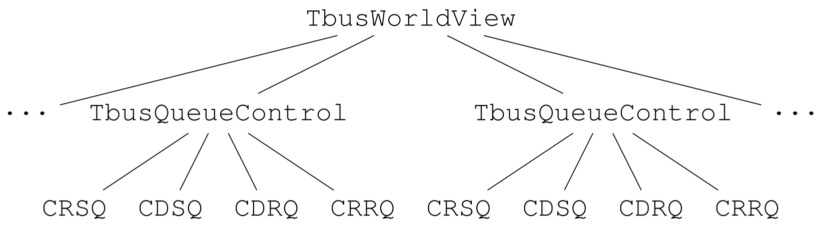


Figure 4.9: Hierarchical structure of TBUS

`WorldView` module, ranged into the TBUS structure as shown in figure 4.9, knows the amount of present nodes and receives and forwards position updates to them. It coordinates the update process, which is displayed in figure 4.7 and consists of the following: During the first round, active nodes receive their updated position and query the data source for unprocessed network characteristics and the updated position's cell id. The cell-share model is informed on nodes changing their cell id and keeps track of cell belonging for each node. Eventually, every node has updated its position and informed the cell-share model of a possible cell change. The cell-share model now has a complete view on the division of nodes into cells and the actual cell-share algorithm can be used in the next round. In round two, every node updates its processed network characteristic, that was adapted by the cell-share model according to the implemented algorithm.

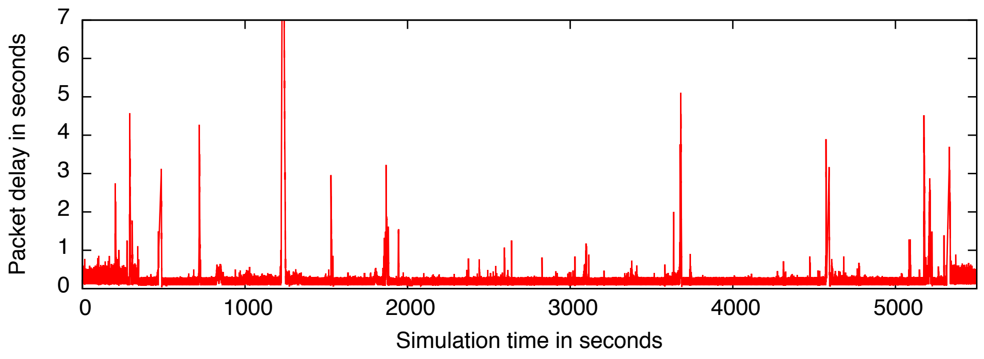
The cell-share model is implemented against a simple interface included in appendix A.5. For this thesis, a simplified cell-share implementation is already included. It only adapts the available bit rate, as backbone delays should be independent from the amount of nodes per cell. Loss probability division is not taken into account and the loss probabilities remain unchanged. The cell-share interface allows other implementations to adapt all network characteristics. With *N* being the number of all active nodes in the cell of the node for which the cell-share distribution is calculated for, and *b* as the bit rate retrieved from the data source, equation 4.3 outlines the implemented cell-share division:

$$\mathrm{CS}(N,b) = \begin{cases} b & \text{if } N = 1 \\ 110\,\% \cdot \frac{b}{N} & \text{otherwise} \end{cases} \tag{4.3}$$

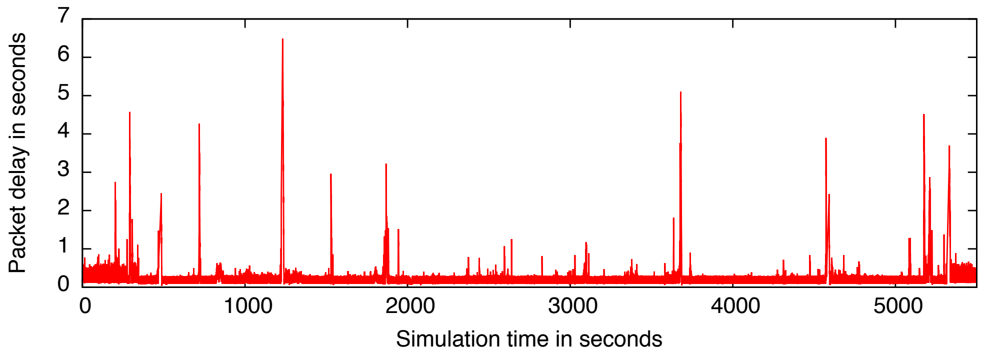
This simplified model approximates an ideal division given by scrambling and spreading codes. As the research on this division is done in [Kar15], this implementation only gives an ideal share for each node. Thus, the 110 % were chosen for this thesis.

4.4 Verification of the adapted TBUS model

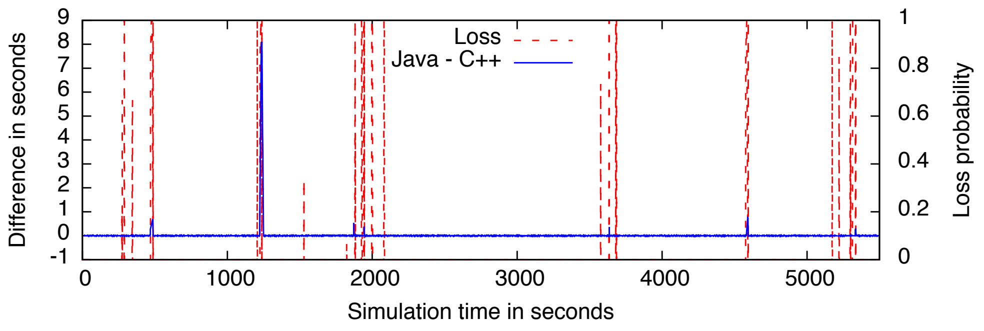
The implementation of the TBUS model for OMNeT++ in VSimRTI is redesigned from scratch because some paradigms and calculations could not be transferred from Norbert Goebel's Java implementation. Therefore, efficiency and an easy-to-integrate modularity is the goal for TBUS in VSimRTI, but calculations and simulation outcomes had to be close or equal to the already verified and extensively tested Java implementation. To compare both models, the same situations are tested with the same input data on both implementations. The `TbusChannelControl` implementation therefore provides a testing mode to compare the network C++ implementation with the original Java version based on simulation time updates only. This testing mode can be enabled via the compile flag `-D TBUS_DEBUG`. Using similar sets of input data from *Tram 2* of [GKMG14] with both implementations, the difference between simulation outcomes is presented in figure 4.10 as the simulated delay of packets over the simulation time. The input data used



(a) Java implementation



(b) C++ implementation



(c) Implementation difference

Figure 4.10: Implementations and difference

by the Java implementation was processed with an error in the backbone delay during times with high packet loss after times with packet loss above 0 % and below 100 %. The graphs in 4.10a and 4.10b also do not show an one-to-one comparison between packets, this is due to the loss probability combined with different random number generators.

By taking a closer look at the difference between simulated packet delays, they can be explained with diverse sources of error. First of all, it has to be noted that more than 98 %

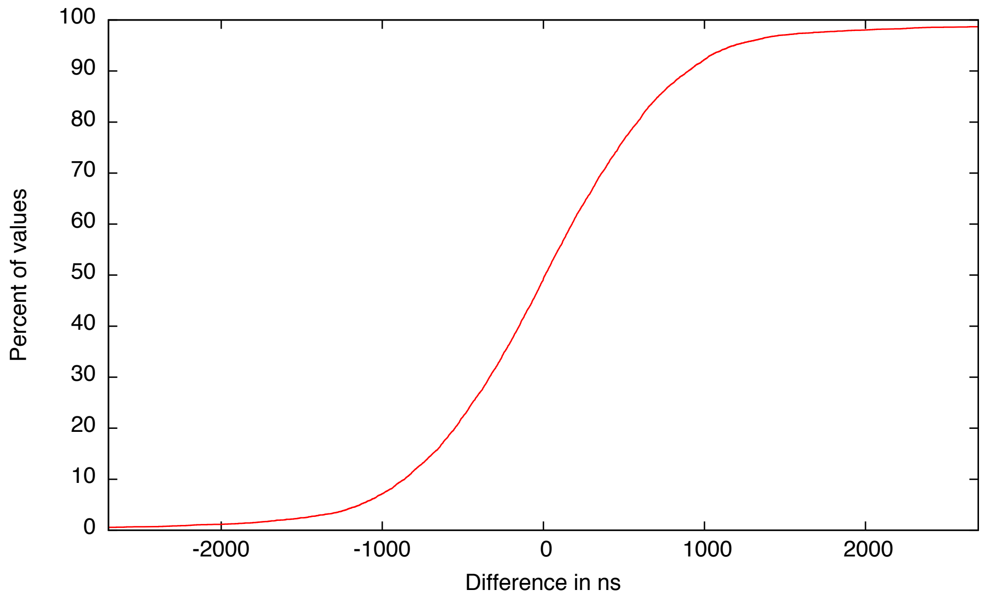


Figure 4.11: CDF of implementation differences from figure 4.10c

of all errors are within the range of 2700 ns = 2.7 µs. With UMTS/High Speed Downlink Packet Access (HSDPA) bit rates at 7.2 Mbit/s maximum (see [ETS15, table 5.1a]), a single bit has an airtime of

$$\frac{1\,\text{bit}}{7.2\,\text{Mbit/s}} = 139\,\text{ns}.$$

This explains smaller errors in both directions: Different rounding functions have been used and, because the maximum bit rate is only available at good network conditions and not on average, the average airtime of a single bit is longer than 139 ns. As can be seen in the Cumulative Distribution Function (CDF) in figure 4.11, the amount of negative and positive errors is equally distributed as 50 % of all errors are smaller than or equal zero and the other 50 % are above or equal zero. This, once again, is due to rounding errors between both implementations.

Inspecting the larger errors, positive errors can be greater by the order of some magnitudes than negative errors. The smallest negative error is $-270\,000\,\mathrm{ns} = -270\,\mu\mathrm{s}$, whereas the highest positive error has a value of $8.104\,55 \cdot 10^9\,\mathrm{ns} \approx 8.104\,\mathrm{s}$. The difference between both implementations has been calculated as $j_t - c_t$ where j_t is the simulated delay of the Java implementation at time t and c_t the simulated delay of the C++ implementation at time t. Negative errors occur when the C++ implementation has greater simulated packet delays than the Java implementation. With the maximum error in this direction at 270 µs, this is still due to rounding errors with bad network conditions and thus high delays.

For the positive errors, where the Java implementation had larger delays than the C++ implementation, the greater errors can be traced back to the processed input data. As mentioned above, the C++ implementation used differently processed input data. The data processing adapts bit rates and backbone delays during loss intervals to compensate bit rate values for high loss probabilities. Because the loss probability is high does not mean that the data rate was low, resulting in an extrapolated data rate during these events. While this is correct for the bit rate, the same had also been done to the backbone delay. Because the backbone delay is independent of the mobile cellular networks' loss probability, this is incorrect and has been corrected. Whereas the Java implementation used adapted backbone delay values, the C++ implementation does not.

Figure 4.10c shows that the spikes in delay differences take place when the loss probability is at 100 % and has been above 0 % shortly before. Right there, the wrongly adapted values result in larger delays from the Java implementation, whereas the C++ implementation utilizes the correct backbone delay and simulates lower delays. In conclusion, the two implementations operate the same within the error range given by different rounding methods and random number generators.

4.5 The TBUS installer

For an easy installation of all needed components and patches and to enable the direct usage of test scenarios used in this thesis, the TBUS installer, a bash[8] script packed with

[8]Bourne Again SHell, see `http://www.gnu.org/software/bash/`

installation archives of necessary sources and binary files, is introduced. It covers the installation of a vanilla VSimRTI as of version 0.14.0, SUMO 0.21.0 and OMNeT++ 4.4. The INETMANET framework as well as VSimRTI's OMNeT++ patch and project are included, too. TBUS patches are applied and sources installed, scenarios copied into the corresponding folders and a virtual environment starter is generated. Featuring dependency checks, usage of all processor cores for compilation, a freely configurable installation folder and a virtual environment script to keep the original paths and environments clean, the installer has been tested on Debian Wheezy (7.8) and Jessie (8.1)[9]. The only dependencies are a reasonably state-of-the-art machine, internet access for downloading additional dependency packages, a bash shell with root privileges (direct or via *sudo*) and the *apt-get* package installer with the default Debian lists.

[9] `https://www.debian.org`

Chapter 5

Simulations and Results

This chapter introduces a simulation scenario with regard to ETSI standards and specifications. A use case suited for the TBUS model is chosen accordingly, an example application following the use case is defined in section 5.2. Message routing for mobile cellular networks based on geographical positions is outlined in section 5.3. The simulation scenario used for the evaluation is then presented in section 5.5, information on how such a simulation scenario is created can be found in section 5.4.

All applications introduced in this chapter are written for the application_NT simulator of VSimRTI 0.14.0. They are all configurable via JavaScript Object Notation (JSON) files on three layers: Every application has default values for configurable variables which allow the direct usage of applications with a correct behaviour. These default values can be overridden by a simulation-wide configuration, which then again can be overridden by per-vehicle configurations.

5.1 ETSI Standards and definitions

The ETSI specifies V2X communication in terms of message standardization, use cases and applications, network structure and node design, and many more. Relevant definitions for this thesis are message format, use cases and applications. Network structure and node design are abstracted by the simulators. The two main message types, the Co-operative Awareness Messages (CAMs) and Decentralized Environmental Notification Messages (DENMs), are used for position propagation and message broadcasting in this thesis.

CAMs are defined in [ETS11a] and offer information on node presence, position and additional data which can be of interest for neighbouring nodes. This type of message should be send periodically all the time, either when vehicle data changed above a threshold or if a timeout occurs. Receiving nodes can act upon these CAMs and calculate their own conclusion from what they received. DENMs, on the other hand, contain information on environmental events or predictions like traffic jams, passing emergency vehicles or emergency brakes. The message format and usage is specified in [ETS10]. DENMs are only send if a certain event occurs and are repeated periodically until the event is over. Receiving nodes are instructed on which actions to take as for every event a corresponding reaction is defined.

With Ad-Hoc networks, DENMs can simply be broadcasted locally and no actions on message routing have to be taken into account. In mobile cellular networks, message routing has to be used because local broadcasts are not possible with conventional mobile cellular networks. This message routing is introduced with a GeoServer in section 5.3.

5.2 Choosing an application

Given the use cases provided by the ETSI in [ETS12], the use case C.1.2.1, Emergency Vehicle Warning, was chosen. From all available use cases, it differs by having a low complexity but huge impact on everyday emergency situations. It requires no specific environmental circumstances, resulting in a simple and reasonable simulation scenario creation which can easily be reproduced with few resources. The size of the simulation area has a low significance which leads to easier trace measurement of the simulation area.

The EmergencyWarningApp According to ETSI use case C.1.2.1, the Emergency Vehicle Warning application was implemented using DENMs over mobile cellular networks. It is configurable via a JSON file in means of emergency vehicle status, the DENM sending interval, message timeout, application start offset, the maximum distance ahead of the emergency vehicle in which nodes should be warned, the speed nodes should slow down to and the duration of the slow down. The configured emergency vehicles send their DENMs periodically after the application start offset until they leave the simulation. Other vehicles slow down to the given speed for the given duration as soon as they receive a DENM, the send interval and message timeout should be chosen in a way so that receiving nodes get messages within the timeout window with a high probability. The right of way, as it is implemented in SUMO, is not changed by the application, as there is no possibility to do so with TraCI or other VSimRTI functionalities. Non-emergency vehicles should stop far enough from intersections and driveways so that the emergency vehicle does not consider the other vehicle as one with the right of way. This enforces that an emergency vehicle has the right of way while being in an active emergency state during the simulation. The default values assure this behaviour.

5.3 Message routing

To enable applications running on moving and stationary nodes to broadcast messages to a set of vehicles positioned in a specific area, two types of *GeoServers* and *-Clients* have been implemented. The GeoServer is a singular instance of a service which coordinates message routing, GeoClients connect to this instance via its known and static address.

The GeoServer therefore maintains a database of every registered vehicle's position and metadata as, for example, a last seen timestamp. This is realised by GeoClients sending periodic position updates containing all necessary information with CAMs to the well-known IP of the GeoServer. Whenever a GeoClient wants to send a broadcast message, it is sent as a DENM to the GeoServer, which duplicates the message as often as needed for every node present in the affected broadcast area. This reduces the network load on the mobile cellular network and saves bandwidth on the mobile node's side. This type of message broadcast and amplification is now defined as *geo-broadcast* or *geo-*

broadcasting and referenced as such further on.

While VSimRTI offers an own implementation of a GeoServer and -Client, this implementation only utilizes GPS data for vehicle position updates and message broadcasts target areas. Though this data is still available within the TBUS extension, a new GeoServer and -Client are introduced to take advantage of graph-based positions. They can already be used as simple applications for basic position update transmission and message geo-broadcasting or, just as VSimRTI's classes, provide a base class for further inheritance to create more complex applications.

The following sections introduce the `TbusGeoServer` and `TbusGeoClient`, both of which are implemented in VSimRTI's application_NT simulator. The VSimRTI connects them with either one of the `urapp` module, representing an *unreliable application* that utilizes UDP on the transport layer, in TBUS' OMNeT++ implementation of `Tbus-MobileNode` and `TbusInetNode`.

5.3.1 TbusGeoServer

The `TbusGeoServer` provides an already usable base class of a GeoServer. It gathers vehicle position information as a pair of the current road id and lane position from regularly sent CAMs. A simplification towards the CAM and DENM specifications is made, as only the message lengths of 200 B, see [BBM13], and 40 B, defined in [ETS10, 6.2.3] respectively and not their full contents are implemented, only information processed from the `TbusGeoServer` is included in the messages. While the ETSI defines the shape of the affected geo-broadcast area as either rectangular, circular or elliptical (see [ETS11c] and [ETS11b, section 4]), the area covered by the TBUS graph structure results in a complex polygon. This suffices for all TBUS-enabled nodes and guarantees an even more precise geo-broadcast method, but differs from ETSI standards. To comply with the ETSI standard, an additional circular area with the geo-broadcasting vehicle's position and the given maximum distance on the road graph can be chosen to cover every affected node.

The `TbusGeoServer` has one configurable variable, the path to the SUMO graph file used for the simulation and is the only mandatory configuration value. The internal graph structure is generated from this SUMO net file and provides a consistent mapping between geo-broadcast messages and the edges nodes are on. More complex applications can inherit `TbusGeoServer`'s functionalities and extend or adapt to their own needs.

5.3.2 TbusGeoClient

The `TbusGeoClient` is this thesis' implementation of a GeoClient. It is capable of sending CAMs for periodic position updates as well as receiving and sending geo-broadcasts meant for nodes positioned in a specific area via DENMs. Configuration options are the message interval for CAMs, an application start offset, the possibility to disable message sending, and values for default road ids and lane positions, if none are available from the traffic simulator. Message receiving cannot be disabled, if the network conditions allow the transmission of data, a `TbusGeoClient` receives it. An extended application can inherit from `TbusGeoClient` and use its interface for geo-broadcasting. This enables further applications to send messages via geo-broadcast without knowing the GeoServer and GeoClient structure.

5.4 VSimRTI scenario creation

This section gives general instructions on how to create a new scenario. As TBUS scenarios are extended VSimRTI scenarios, consult the in-depth documentation on creating VSimRTI scenarios found in VSimRTI's manual and documentation, if needed.

Prerequisites A valid VSimRTI licence is mandatory for using the framework. Simulated applications and the vehicle types they are assigned to are selected. The simulation area has to be known and a suitable OpenStreetMap (OSM) map file of the selected simulation area is obtained via the official website[10] or other sources.

[10] `https://www.openstreetmap.org/`

Data preprocessing The map's contents have to be checked for consistence and inclusion of all necessary road segments. Measured and mapped mobile cellular network characteristics have to be checked for measurement errors; as errors in the input data can have a great impact on the simulation outcome. It then has to be exported into a data source having a suitable connector available in the TBUS library as mentioned in section 4.3.7. An example SQL database schema is given in A.3.

Importing data into VSimRTI Then, using VSimRTI's `scenario-convert`, the navigation database and SUMO files are generated. Routes can be generated using the same tool, but have to be verified and maybe edited as they are chosen randomly and might not fit the simulation's needs. If necessary, additional routes can be created but have to be re-imported into VSimRTI's navigation database. Consult `scenario-convert`'s help output for further information.

Setting up the scenario The scenario has to be set up for the given simulation in VSimRTI's configuration files and in addition in TBUS's `omnetpp.ini` file. This file, read by OMNeT++ and the TBUS implementation, is the configuration file with necessary values explained by comments in the line above them. Vehicles can then be defined, routes assigned to them and applications equipped on them. The suited log-levels for VSimRTI can be set. Please keep in mind, that verbose output generates a large overhead containing no information on the simulation outcome. This data is often not necessary for the evaluation.

Starting the simulation OMNeT++ has to be started and deployed manually, as VSimRTI currently cannot start OMNeT++ with the additional TBUS library. VSimRTI is then started given the scenario's configuration file and licence information.

5.5 Choosing a scenario

The choice of a scenario area depends on requirements defined by the measuring complexity, the geometric size and applicable scenes for the simulated application, and the simulated use case. It had to be close to the measuring person's workspace to enable simple and frequent measurement drives. Measurement drives have been conducted by Norbert Goebel and simulation data was map-matched and preprocessed by Adrian Skuballa[11].

Taubental scenario The industrial area around the *Im Taubental* street located in Neuss, Germany and shown in figure 5.1, was chosen as the simulation area for the scenario this thesis investigates. All green marked roads have been measured and are used as the foundation for simulated routes, the red arrows point to start and end points of the simulated routes. An interesting part of the simulation area is highlighted with a red ellipse, the equipped emergency vehicle takes most of its actions within this field. This area includes two four- and two three-way intersections as well as all streets navigable in both directions. With a high enough vehicle flow, the emergency vehicle obtains the right of way many times within a small time and space interval. Based on this scenario, message delays can be measured and compared to ETSI standards for message timeout. This allows a conclusion the usability of this simulation model for this specific simulation scenario and input data.

5.6 Simulation

The VSimRTI scenario was created using the whole area presented in figure 5.1. Measurement data for the simulation area was gathered on August 3rd 2015 between 9:12:29 and 9:22:31 a.m. Central European Summer Time (CEST). A total of 11 942 entries, each containing the three network characteristics, were recorded. Preprocessing has kindly been conducted by Adrian Skuballa as shown in [Sku15].

[11] Adrian.Skuballa@uni-duesseldorf.de

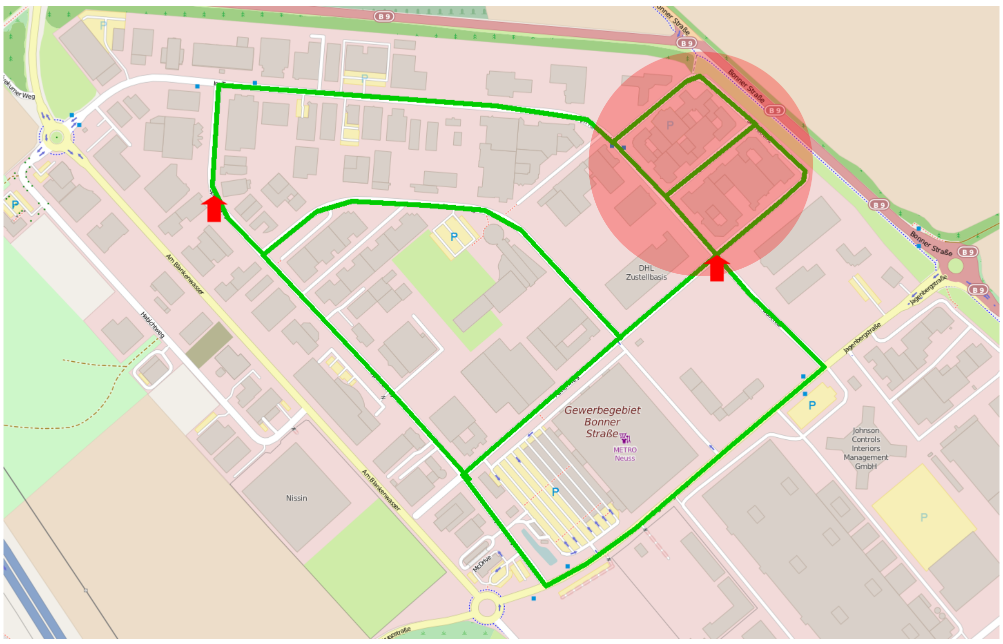


Figure 5.1: Excerpt from the simulation area (Taken from OSM)

Vehicles are equipped with `TbusGeoClient` and `EmergencyWarningApp` applications introduced in sections 5.3.2 and 5.2. The exact configuration files can be found in appendix A.2. An overview of the used simulation hardware is presented in A.1.

A RSU acts as the GeoServer and has an instance of a `TbusGeoServer` application assigned. The maximum duration of each simulation scenario was given as $5500\,\mathrm{s} \approx 90\,\mathrm{min}$. For both scenarios, vehicle 4 was chosen as the emergency vehicle. It also takes the same routes during both simulations. Other vehicle's behaviour depends on received DENMs and their reactions, as well as their obedience to traffic rules. As the simulation results heavily rely on the simulation input and settings, the following statements are not made for the simulation model in general, but explicitly for the given scenarios.

5.6.1 Metrics

As a metric, we compare simulated network loads and single node performance with regards to ETSI-defined standards. Therefore, the main requirements of ETSI use case C.1.2.1, except CAM authentication and protection as no malicious nodes are simulated,

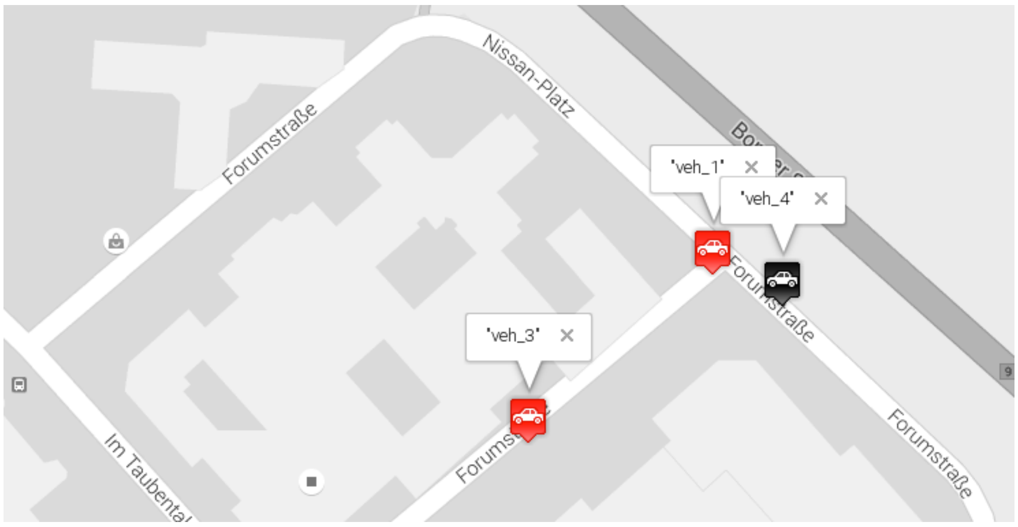


Figure 5.2: Screenshot of VSimRTI's visualizer during simulation (Image copyright by Google Maps)

have been met. All vehicles broadcast their position via CAMs with a frequency of 1 Hz. The emergency vehicle broadcasts DENMs with the minimum required frequency of 10 Hz. Because the original ETSI use case is designed for MANETs, some timeout values have to be reconsidered, as MANETs have a lower latency and perhaps a higher loss probability than mobile cellular networks. Especially, as we have mobile to server to mobile connection instead of mobile to mobile connections, the critical time requirement for the transmission of DENMs has been relaxed. The ETSI-defined value of 100 ms has been changed to 300 ms to achieve a greater amount of vehicles obeying DENMs. 300 ms were chosen, because this value was high enough as a timeout compared to delays simulated in previous scenarios not listed in this thesis. The original timeout is still compared to the measured delays. As for the simulation scenario I only one, and for scenario II only ten messages were dropped, packet loss probability is not plotted.

The delays of CAMs and DENMs have been collected for every active vehicle in both up- and download directions. Upload and download directions are from a GeoClient's point of view, because there are no simulation queues on the GeoServer. The delay of messages for the upload directions and the total delay has been recorded as follows:

Upload delay This is the application to application message delay on its way from a GeoClient to the GeoServer and has thus been recorded on the GeoServer. Upload delays were recorded for CAMs and DENMs separately to distinguish between the delays of those different sized messages.

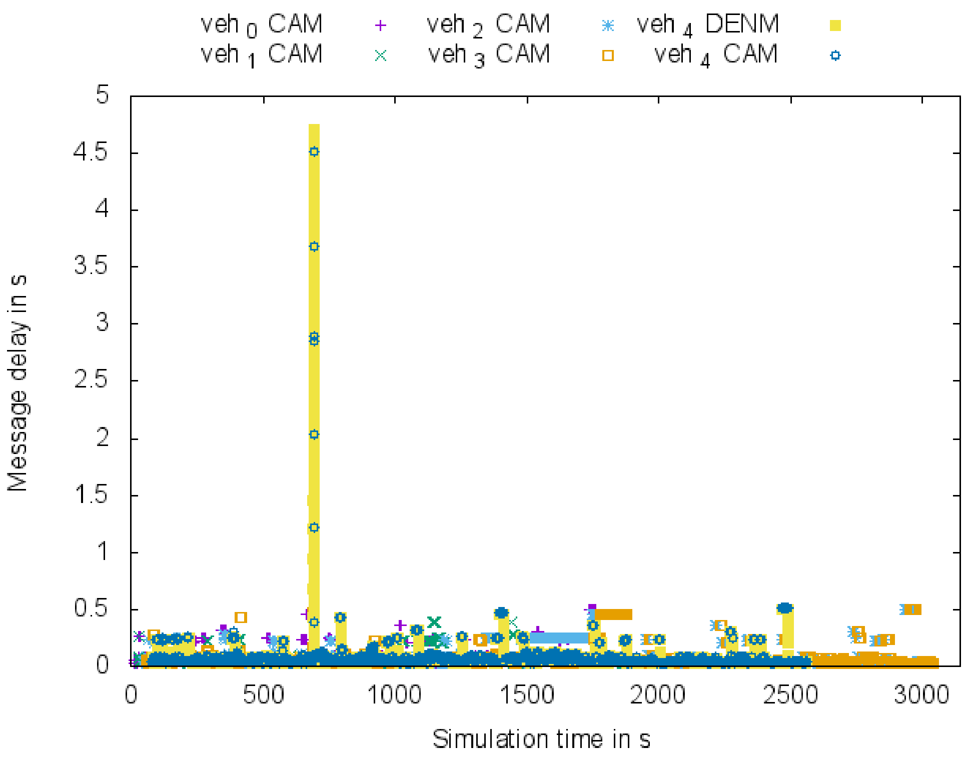
Message delay The message delay is only measured for DENMs that have been forwarded by the GeoServer to corresponding GeoClients. Thus, this delay includes the upload delay of a DENM from the sending GeoClient to the GeoServer as well as the download delay from the GeoServer to the receiving GeoClient. It is measured on each receiving GeoClient.

Both delays are then plotted against the simulation duration. Conspicuities are further explained. To enable a simple comparison to ETSI standards, the delays are also presented as CDFs. The suitability of simulation results, regarding the ETSI use case, can then be read as a percentage of all transmitted messages.

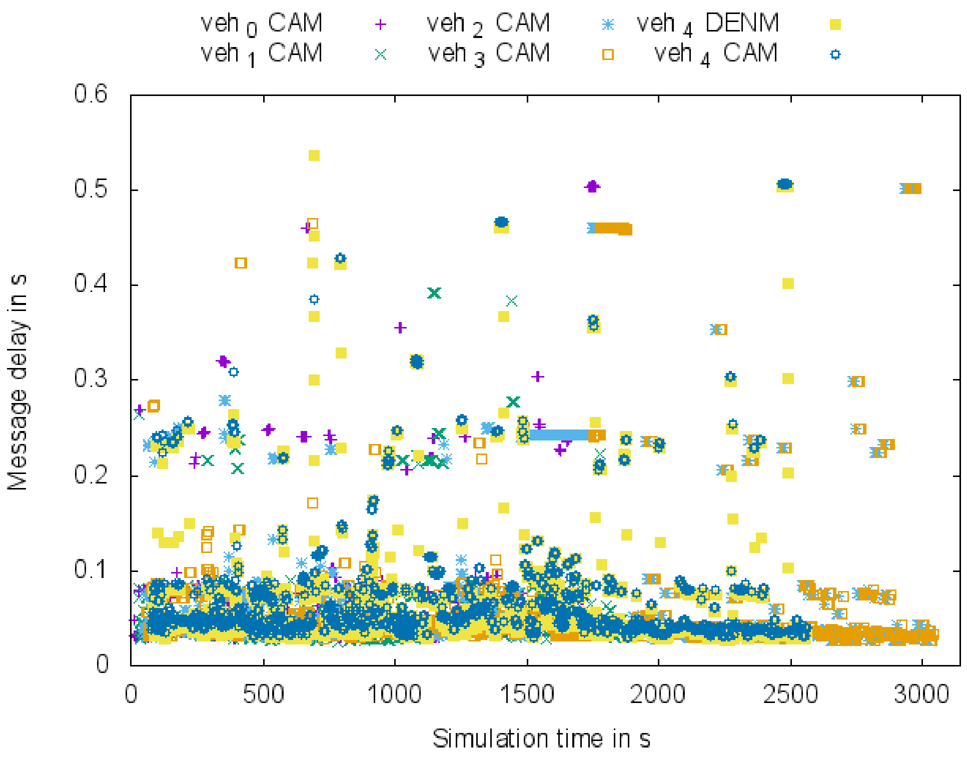
5.6.2 Scenario I: Taubental

For the small simulation scenario, five vehicles with different routes each between the same end and origin were simulated. The simulation took $1307\,\mathrm{s} \approx 22\,\mathrm{min}$ to complete and simulated $3\,042\,400\,000\,000\,\mathrm{ns} = 50\,\mathrm{min}\,42.4\,\mathrm{s}$.

Upload delay The message upload delays, as shown in figure 5.3a, mostly remain below 1 s. There is one spike between 500 to 1000 s, where the upload delays of the emergency vehicle, vehicle 4, of both CAM and DENM rise rapidly. Over the simulation time from 689 to 700 s, congestion occurs on vehicle 4's upload direction. This is due to a low sending rate of $2.029\,297\,\mathrm{kB/s}$ assigned between 27.5 to 30.5 m on the road segment 1037164_1103483136_1103483070_110348313. Figure 5.4 show the position based upload data rate of two traces along the former mentioned road segment.



(a) Message upload delays



(b) Detailed excerpt of message upload delays

Figure 5.3: Message upload delays and excerpt

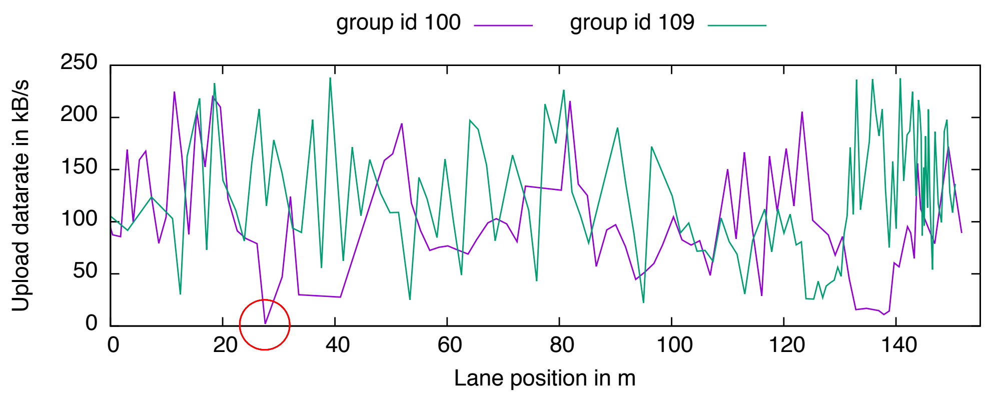


Figure 5.4: Upload data rate along road segment 1037164_1103483136_1103483070_110348313

Figure 5.3b shows message delays below 0.6 s. As can be seen, a large number of messages have a delay below 0.1 s, while also a greater number of message delays resides between 0.2 to 0.3 s and other groupings exist.

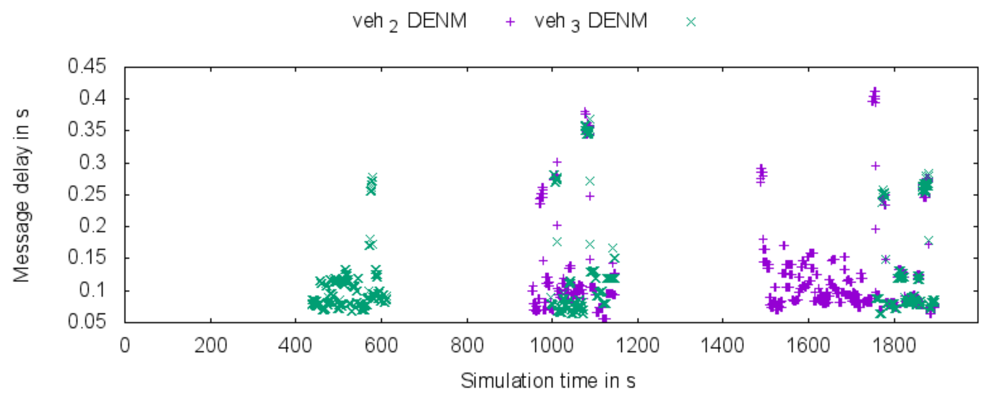
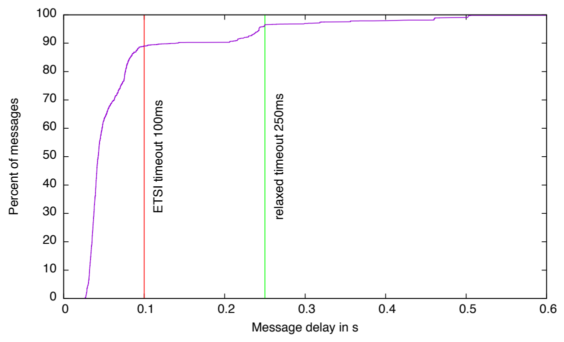
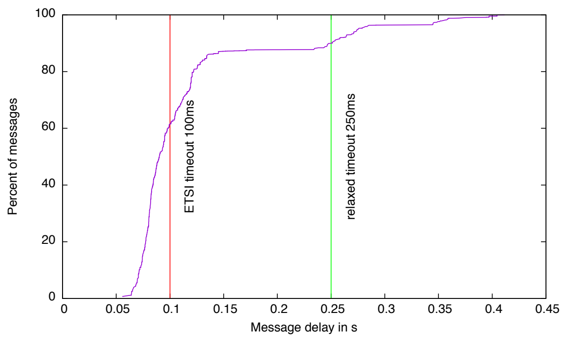


Figure 5.5: Message delays

Message delay The message delay of DENMs is shown in figure 5.5 and is overall well below 0.5 s. Because only two vehicles received DENMs and the loss during this scenario affected just one message, only two out of all four vehicles were about to cross the emergency vehicle's way at the presented times. Vehicles 2 and 3 received DENMs forwarded to them by the GeoServer. Most of the received DENMs were delivered within the ETSI specification of 100 ms.



(a) CDF of upload delays



(b) CDF of message delays

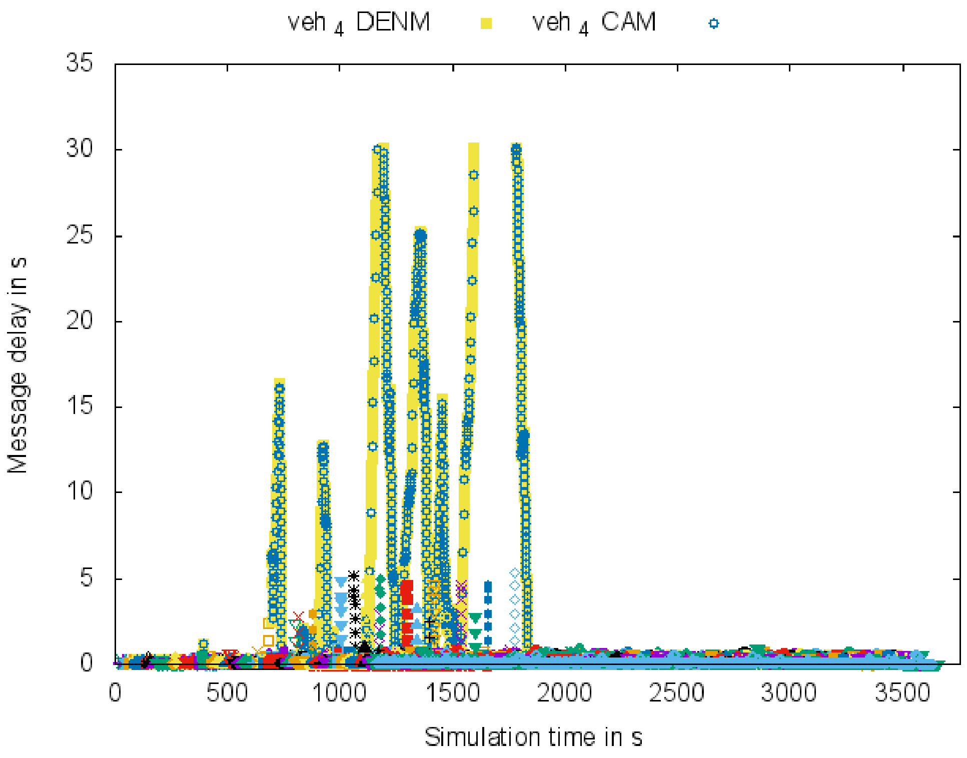
Figure 5.6: CDFs of upload and message delays with ETSI and relaxed timeout marked

By examining figures 5.6a and 5.6b, one can compare the simulated message delays to the ETSI standards given for the selected use case. On the upload direction, close to 90 % of all messages delivered to the GeoServer arrived within the ETSI-defined 100 ms timeout. By raising the timeout to 250 ms, more than 95 % of all received messages would be able to have a lower delay than the given timeout. For the message delay, the 100 ms timeout can still be met by around 60 % of all delivered messages. Again, by raising it to 250 ms, around 90 % of all received messages would match the given criteria.

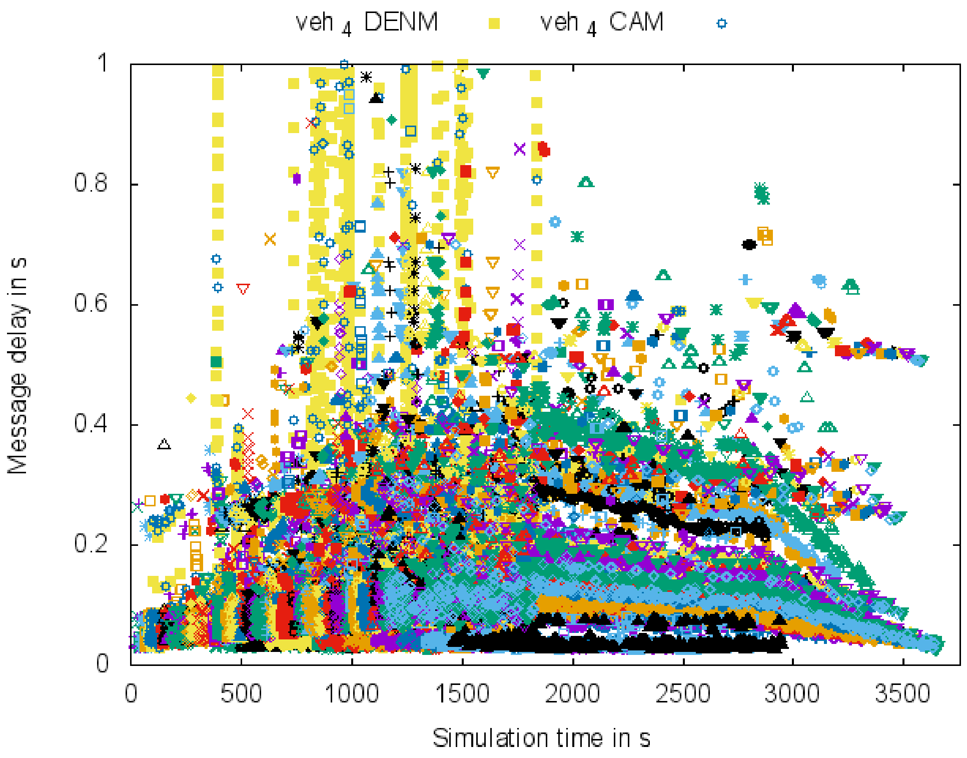
5.6.3 Scenario II: Taubental-large

For the large simulation, 50 vehicles were simulated in the same simulation area. The same route has been assigned for ten vehicles each and new vehicles spawned every two minutes per route. The simulation took 8110 s $\approx$ 2 h 15 min to complete and simulated 3 655 900 000 000 ns $=$ 1 h 55.9 s.

Upload delay Figure 5.7a shows the overall message upload delays. As one would expect, the emergency vehicle once again has a large spike from 689 to 700 s, as it took



(a) Message upload delays



(b) Detailed excerpt of message upload delays

Figure 5.7: Message upload delays and excerpt

the same route during scenario II as for scenario I and the same vehicles joined the simulation before the emergency vehicle did. It was also given the right of way if other vehicle's successfully processed DENMs. Other spikes can be traced back to a higher network load and the cell-share model dividing available data rate to all members of a cell.

What remains suspicious are gaps in between the emergency vehicle's message during delays of above 30 s. This occurs from 1166 to 1191 s and 1597 to 1783 s. At first, it seems that messages in between these intervals have been dropped by the simulation. Further investigation of message transmissions through the implemented simulation model showed, that messages during these intervals were correctly simulated by the TBUS model and not dropped. The error occurred within VSimRTI's application_NT simulator: Instead of message objects, `null` objects were delivered to the GeoServer in these timespans. Because for these `null`-messages no information other than the receiving timestamp is available and the messages cannot be identified anymore, only assumptions on this error can be made.

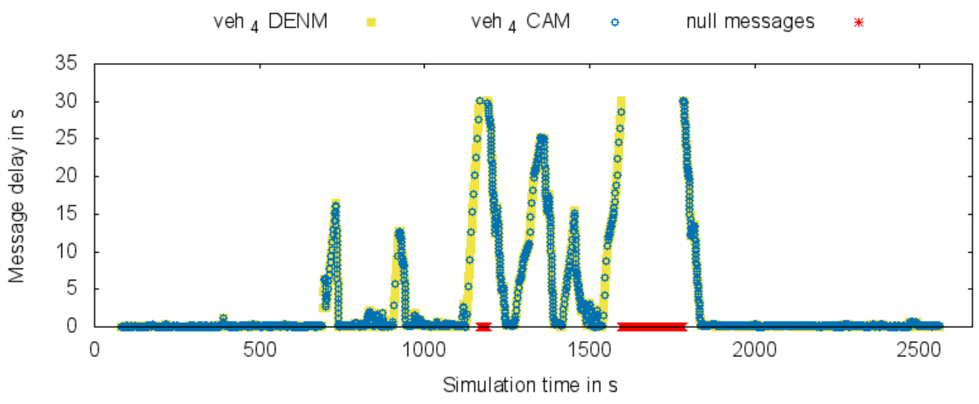


Figure 5.8: Vehicle 4 messages and `null` messages

Concluding from figure 5.8, where all received `null` messages fit seamlessly into the gaps of vehicle 4's message delays, the following statement describes the validity of the simulation results: As the implemented simulation model evidently drops no more than the correctly simulated messages, the shown figures are valid but incomplete. How message delays for vehicle 4 during phases of `null` messages behaved is neither described nor considered for the rest of this thesis. It can only be assumed, that these messages have a simulated delay above 30 s, as this is the lower delay limit for messages to be turned into `null` messages. Because this error was first encountered during the very last days of this thesis, it has not yet been further investigated or patched.

By examining figure 5.7b, the message upload delay distribution can be analyzed. Delays rise with a larger number of simulated nodes, as the cell-share model divides the available data rate according to the number of active nodes within the same cell. With a stagnating number of simulated nodes, the message upload delay becomes lower. The overall message upload delay is higher, but still a large part of the messages reach the GeoServer in less than 200 ms.

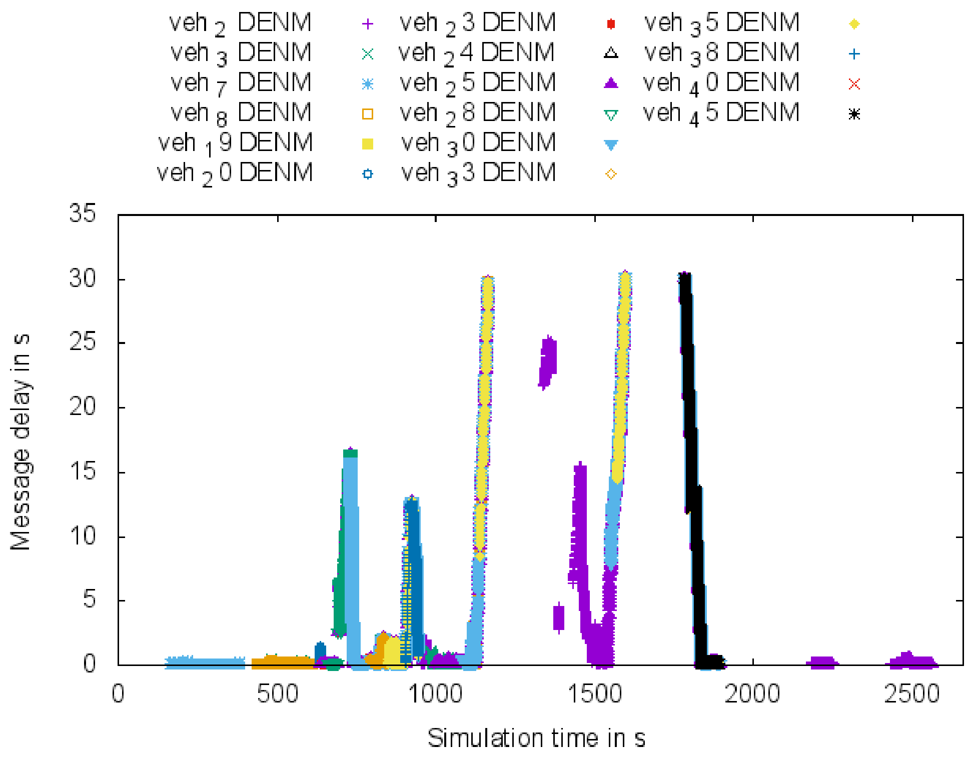
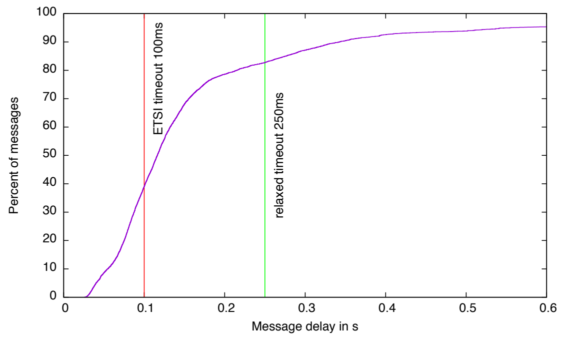
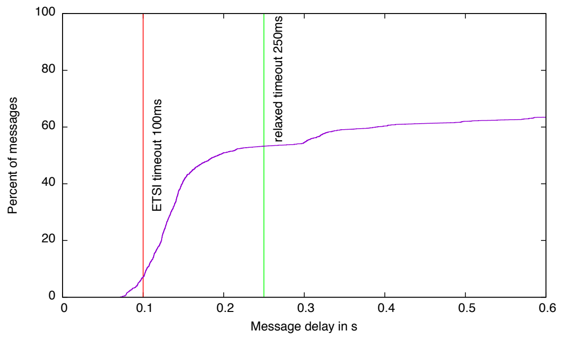


Figure 5.9: Message delays

Message delay Message delays, as shown in figure 5.9, can only be evaluated for messages other than `null` messages forwarded by the GeoServer. Because `null` messages contain no data, there were no DENMs to forward to GeoClients during these phases. Instead, only valid DENMs are investigated. In contrast to the message delays from scenario I, those in scenario II vary by a larger amount of time. Also, due to the higher number of vehicles in the scenario, more vehicles received DENMs. This lead to a higher network load, too, because DENMs were send to the GeoServer at 10 Hz and forwarded to the corresponding nodes with this frequency each.



(a) CDF of upload delays



(b) CDF of message delays

Figure 5.10: CDFs of upload and message delays with ETSI and relaxed timeout marked

The CDFs shown in figure 5.10 confirm the previously made assumptions. While still around 40 % of all messages reach the GeoServer within 0.1 s, not even 10 % of DENMs have been transmitted to GeoClients within the ETSI-defined timeout. Figure 5.10b only shows delays of up to 0.8 s as the highest increase of message percentage happens in this interval. It has to be noted, that 90 % were reached at around 20 s message delay. The network load was high and because many vehicles were simulated on the same road segments, the cell-share model only left small amounts of the originally available data rate to be used for simulated nodes. As in scenario I, raising the timeout value to 250 ms would allow around 80 % and 50 % of all transmitted messages, for upload and download respectively, to meet the timeout requirement.

5.6.4 Conclusion

The presented simulation scenarios give an impression on how the implemented simulation model behaves while coupled to other simulators and processing their input. The following is only valid for the given set of input data, but may be transferable to similar scenarios and input data.

For both scenarios, message loss was surprisingly low. Surprisingly, because 138 out of all 11 942 data entries contains a loss probability higher than 0, and 58 of those had a loss probability of 100 %. There is no great difference in the loss rates for both scenarios, as the same routes were used in both cases. Also, these routes did not take place on a road segment with high loss probability. Once again, this might be different for another simulation input or the same input but different routes.

Message delays recorded for both simulation scenarios show an expected connection between the number of simulated GeoClients and the recorded message delay. The used cell-share model reduced the overall available data rate for GeoClients situated in the same cell. Because messages are send via UDP and no precautions on how to avoid network congestion have been made, peak values of network congestion can become large. But, as soon as the congested GeoClient's data rate rises again, the congestion was relieved and lower delays were achieved.

On the application side of the simulation, the emergency vehicle was able to broadcast its DENMs, and, depending on the current network load, most of the receivers took the mandatory action and gave the emergency vehicle the right of way. This was due to the relaxation of the timeout value from 100 to 300 ms. With this value, more than 50 % of all DENMs were recognized by GeoClients. This allowed the emergency vehicle to obtain the right of way most of the time it was necessary, because the sending frequency of 10 Hz and every other message meeting the relaxed ETSI requirements informed Geo-Clients early enough to not hinder the emergency vehicle. Because the ETSI use case is not defined for mobile cellular networks, but rather suggests the message formats that should be used for these networks, timeout and frequency values have to be reconsidered. Directions on how these new values can be chosen have been presented by the conducted simulations.

Altogether, the simulation scenarios proved that the coupled simulation reaches a certain amount of reliability for the `EmergencyVehicleApp` for the given input data.

Chapter 6

Summary

Coupled simulation of road traffic, V2X-applications and V2X-communication via mobile cellular networks is a complex process. This thesis utilized the V2X simulation framework VSimRTI to transfer the TBUS model into a greater simulation infrastructure. Implementing an OMNeT++ extension and integrating it into the VSimRTI, a new possibility for the trace-based simulation of mobile cellular networks is introduced. Future users can reliably build their own V2X applications, based on the TBUS model, and simulate network traffic while being applied on mobile nodes. There is no need to consult complex mathematical simulation models and the overall entry barriers are lowered, allowing a larger group of users to utilize the coupled simulation of road traffic, V2X-applications and V2X-communication via mobile cellular networks.

6.1 Conclusion

The modification of VSimRTI and implementation of the TBUS model within OMNeT++ allows the simulation of V2X applications by utilizing data gathered from measurement drives. Additionally, a base framework for V2X applications used for TBUS simulations is introduced. These new components have then been tested and evaluated using processed real-world network characteristics within two simulation scenarios. It has been concluded, that the simulated applications meet a certain level of reliability, depending on given input network characteristics. The implementation provided by this thesis is extendable to include even more functionality; and it contains a verified core simulation model to enable reliable extensions.

Everything considered, the coupled simulation of road traffic, V2X-applications and V2X-communication via mobile cellular networks has been investigated, extended as needed to enable a general simulation process and evaluated using suitable network characteristics input.

6.2 Future work

This thesis brought the TBUS model into the VSimRTI. It provides interfaces for new functionalities, simplified implementations of those and opens up new opportunities for TBUS. In the following, some parts that require future work to be fully explored, and some suggestions concerning existing components, are listed.

6.2.1 Stronger coupling between application_NT and OMNeT++

VSimRTI provides an extensive framework for coupled simulation. Especially SUMO as a road traffic simulator is strongly coupled with the application_NT simulator. It allows V2X applications to retrieve the current state of and control the vehicle they are assigned to. A similar coupling between OMNeT++ and application_NT would allow information on the current network state to be utilized within an application's logic. Applications for this thesis assume a connected and stable network condition. They are not informed on the available network state or for example, congestion delaying packets or the type of mobile cellular network cell they are connected to. This information would allow applications to respond to a network's load, adapt to network characteristics and provide an overall better usage of the mobile cellular network. The protocol used for coupling OMNeT++ to VSimRTI has to be modified in order to achieve this extension. Other simulators would then be able to access these data and draw their own conclusion on how to react or respond.

6.2.2 `null` message investigations

As the error with `null` messages occurred only during the last simulation and within the last days of this thesis, only assumptions on how it arises and how it can be fixed are made. An error description and log files of the affected scenario will be made available to the Fraunhofer Institute. A fix of this error should be included in one of the next VSimRTI releases.

6.2.3 Extended research on the cell-share model

The cell-share model introduced with this thesis provides an interface that precise and complex cell-share models can be implemented against. A simple implementation of a cell-share model is included with this thesis, but in-depth research is required. Field measurements and evaluation of the newly gathered data have to be concluded to define a precise and more realistic cell-share model. Franz Kary currently investigates the influence of multiple nodes within the same mobile cellular network cell in his Master's Thesis [Kar15].

6.2.4 Further research on graph-based geo-routing

The graph-based geo-routing introduced for the GeoServer and GeoClients used in this thesis, enables a more directed approach on message routing. While it needs a supporting road graph structure and the mapping of vehicle's positions onto corresponding edges and position on the edge, it reduces message overhead. It also ensures message delivery only to vehicles heading in the direction relevant for the geo-broadcast. As this thesis only utilizes a simple implementation of graph-based routing, further research on the real-world usability of this technique is needed.

6.2.5 Additional simulation scenarios

Simulation creation requires a large amount of time, as a suitable simulation area has to be found, measurement drives have to be conducted and recorded data has to be processed before the simulation can be run. For further evaluations, this has to be finished before one can retrieve additional results of the coupled simulation with the TBUS model. Also, additional use cases and applications have be implemented, but can be evaluated on existing and new simulation scenarios.

Bibliography

[BBM13] Jakob Breu, Achim Brakemeier, and Michael Menth. Analysis of Cooperative Awareness Message rates in VANETs. In 13th *International Conference on ITS Telecommunications (ITST)*, pages 8–13, November 2013.

[DCA] Über uns – DCAITI – TU Berlin. `http://www.dcaiti.tu-berlin.de`. Online, last checked: August 9th 2015.

[DG08] Department of Defense and GPS Navstar. Global Positioning System Standard Positioning Service Performance Standard, 4th Edition. Technical report, Deparment of Defense (DoD), ASD(NII)/DASD (C3, Space and Spectrum), Attn: Assistant for GPS, Positioning and Navigation, 6000 Defense Pentagon, Washington, DC 20301-6000, 2008.

[ETS10] Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service. Technical Specification ETSI TS 102 637-3 V1.1.1, European Telecommunications Standards Institute, September 2010.

[ETS11a] Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service. Technical Specification ETSI TS 102 637-2 V1.2.1, European Telecommunications Standards Institute, March 2011.

[ETS11b] Intelligent Transport Systems (ITS); Vehicular Communications; Geographical Area Definition. European Standard ETSI EN 302 931 V1.1.1, European Telecommunications Standards Institute, July 2011.

[ETS11c] Intelligent Transport Systems (ITS); Vehicular communications; GeoNetworking; Part 4: Geographical addressing and forwarding for point-to-point and point-to-multipoint communications; Sub-part 1: Media-Independent Functionality. Technical Specification ETSI TS 102 636-4-1 V1.1.1, European Telecommunications Standards Institute, June 2011.

[ETS12] Intelligent Transport Systems (ITS); Framework for Public Mobile Networks in Cooperative ITS (C-ITS). Technical Report ETSI TR 102 962 V1.1.1, European Telecommunications Standards Institute, February 2012.

[ETS15] Universal Mobile Telecommunications System (UMTS); UE Radio Access capabilities. Technical Specification ETSI TS 125 306 V12.5.0, European Telecommunications Standards Institute, April 2015.

[FOK] FOKUS | Fraunhofer-Institut für Offene Kommunikationssysteme. `https://www.fokus.fraunhofer.de`. Online, last checked: August 9th 2015.

[GKMG14] Norbert Goebel, Markus Koegel, Martin Mauve, and Kálmán Graffi. Trace-based Simulation of C2X-Communication using Cellular Networks. In 11*th* *IEEE/IFIP Annual Conference on Wireless On-demand Network Systems and Services (WONS 2014) – Special Session on VANETs and ITS*, pages 108–115, April 2014.

[IEE10] IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Framework and Rules. *IEEE Std 1516-2010 (Revision of IEEE Std 1516-2000)*, pages 1–38, August 2010.

[INEa] aarizaq/inetmanet-2.0. `https://github.com/aarizaq/inetmanet-2.0`. Online, last checked: August 9th 2015.

[INEb] inetmanet-2.0/ARP.cc at 10b4f7b96496b1564ec9392c2d7241906093f06c · aarizaq/inetmanet-2.0. `https://github.com/aarizaq/inetmanet-2.0/blob/10b4f7b96496b1564ec9392c2d7241906093f06c/src/`

networklayer/arp/ARP.cc#L84. Online, last checked: August 9th 2015.

[Kar15] Franz Kary. Modelling the Interference of Multiple Participants in a Mobile Network Cell. Master's thesis, Department of Computer Science, Heinrich Heine University of Düsseldorf, December 2015.

[KDW00] Frederick Kuhl, Judith Dahmann, and Richard Weatherly. *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*. Prentice Hall PTR, 2000.

[Kra14] Tobias Krauthoff. Measurement of Position-based Network-Characteristics in Cellular Networks while Moving. Master's thesis, Department of Computer Science, Heinrich Heine University of Düsseldorf, November 2014.

[KSW$^+$08] A. Köpke, M. Swigulski, K. Wessel, D. Willkomm, P. T. Klein Haneveld, T. E. V. Parker, O. W. Visser, H. S. Lichte, and S. Valentin. Simulating Wireless and Mobile Networks in OMNeT++ the MiXiM Vision. In *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Simutools '08, pages 71:1–71:8, ICST, Brussels, Belgium, 2008. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[Lan13] Christian Simon Lange. Untersuchung der Auswirkungen paralleler Datenratenmessungen in einer Mobilfunkzelle. Master's thesis, Department of Computer Science, Heinrich Heine University of Düsseldorf, June 2013.

[Mob] Mobilty Framework for OMNeT++. http://mobility-fw.sourceforge.net. Online, last checked: August 9th 2015.

[Nat00] National Imagery and Mapping Agency (NIMA). Department of defense world geodetic system 1984 – its definition and relationships with local geodetic systems. Technical Report Technical Report 8350.2 Third Edition, National Imagery and Mapping Agency (NIMA), 2000.

[Olf13] Malte Olfen. Ende-zu-Ende-Messungen von Mobilfunkparametern mit Smartphones. Bachelor's thesis, June 2013.

[PMOR12] Robert Protzmann, Kim Mahler, Konstantin Oltmann, and Ilja Radusch. Extending the V2X simulation environment VSimRTI with advanced communication models. In 12*th* *International Conference on ITS Telecommunications (ITST)*, pages 683–688, November 2012.

[Sch12] Björn Schünemann. *The V2X Simulation Runtime Infrastructure: VSimRTI.* PhD thesis, Universität Potsdam, May 2012.

[SGD11] Christoph Sommer, Reinhard German, and Falko Dressler. Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15, January 2011.

[Sku15] Adrian Skuballa. Preprocessing of position-based cellular network measurements for trace-based v2x-simulations. Master's thesis, Department of Computer Science, Heinrich Heine University of Düsseldorf, September 2015.

[Wil12] Sebastian Wilken. Verfahren zur Datenratenmessung in Mobilfunknetzwerken. Master's thesis, Department of Computer Science, Heinrich Heine University of Düsseldorf, June 2012.

[WPR$^+$08] Axel Wegener, Michał Piórkowski, Maxim Raya, Horst Hellbrück, Stefan Fischer, and Jean-Pierre Hubaux. TraCI: An Interface for Coupling Road Traffic and Network Simulators. In *Proceedings of the* 11*th* *Communications and Networking Simulation Symposium*, CNS '08, pages 155–163, New York, NY, USA, April 2008. ACM.

[XVF] Debian – Details of package xvfb in wheezy. `https://packages.debian.org/en/wheezy/xvfb`. Online, last checked: August 9th 2015.

[Zim80] H. Zimmermann. OSI Reference Model – The ISO Model of Architecture for Open Systems Interconnection. *IEEE Transactions on Communication*, 28(4):425–432, April 1980.

Abbreviations

ARP Address Resolution Protocol. 19, 20

C2X Car-to-X. 7, *see* V2X

CAM Cooperative Awareness Message. 38–41, 44–46

CCMSim C2X Channel Model Simulation. 4

CDF Cumulative Distribution Function. xi, 34, 46, 49, 53

CEST Central European Summer Time. 43

CPU Central Processing Unit. 13

DCAITI Daimler Center for Automotive Information Technology Innovations. v, 2, 17, 18

DENM Decentralized Environmental Notification Message. 38–41, 44–46, 48, 49, 51–54

ETSI European Telecommunications Standards Institute. iii, 2, 37–40, 43–46, 49, 53, 54

FOKUS Fraunhofer Institut für offene Kommunikationssysteme/Fraunhofer Institute for Open Communication Systems. v, 2, 17, 18

GPS Global Positioning System. 7, 9–11, 40

HLA High Level Architecture. 11

HSDPA High Speed Downlink Packet Access. 34, *see* UMTS

INETMANET INET framework for Mobile Ad-Hoc Networks. 1, 12, 13, 17–22, 24, 27, 36

INI Initialization. 13

IP Internet Protocol. 19, 20, 39

ISO International Organization for Standardization. 22

JSON JavaScript Object Notation. 37, 39

l continuous position on a lane represented by an edge. 10

MAC Medium Access Control. 22–24

MANET Mobile Ad-Hoc Network. 13, 45

MiXiM Mixed Simulation. 4, 5

MTU Maximum Transmission Unit. 24

NED Network Description. 13, 21

NIC Network Interface Card. 13, 24

OMNeT++ Objective Modular Network Testbed in C++. iii, 1, 2, 4, 5, 11–13, 17–22, 24, 25, 27, 32, 36, 40, 42, 55, 56, 69

OSGeo Open Source Geospatial Foundation. 14

OSI Open Systems Interconnection. 22

OSM OpenStreetMap. 41, 44

PHY Physical layer. 22, 23

r road id of an edge. 10

RMF Rate Measurement Framework. 9

RSU Road Side Unit. 20, 44

SQL Structured Query Language. 27, 29, 42, 77

SUMO Simulation of Urban Mobility. xi, 2, 5, 7, 11, 12, 14, 15, 17, 18, 36, 39–42, 56

TBUS Trace Based UMTS Simulation. iii, v, xi, 3–8, 17–19, 21–27, 31, 32, 35–37, 40–42, 51, 55, 56, 58, 77, *see* UMTS

TCP Transmission Control Protocol. 13

TraCI Traffic Control Interface. 5, 14, 15, 39

UDP User Datagram Protocol. 13, 22, 40, 54

UMTS Universal Mobile Telecommunications System. 9, 34, *see* HSDPA

UTM Universal Transversal Mercator. 14

V2X Vehicle-to-X. iii, 1, 2, 7, 37, 55, 56

Veins Vehicles in Network Simulation. 5

VSimRTI V2X Simulation Runtime Infrastructure. iii, v, xi, xv, 2–5, 7, 11, 12, 14, 15, 17–22, 30, 32, 36, 37, 39–43, 45, 51, 55–57, 69

WLAN Wireless Locale Area Network. 21

Appendix A

Appendix

A.1 The testing server

The professorship for Computer Networks of the Heinrich-Heine-University Düsseldorf[12] provided a Linux virtual server for testing and simulation purposes. The exact specifications are shown in table A.1. To run a graphical interface and make use of OMNeT++'s and VSimRTI's visualisation options, a *virtual framebuffer* using *xvfb* (see [XVF]) accessed by VNC was installed. A modified version of VSimRTI 0.14.0

CPU	Intel® Xeon® E5-2620 v2 2.10 GHz
Number of cores	6
Memory	52 GB ECC
Linux distribution	Debian 7.8 (wheezy) 64bit
Linux kernel	3.2.0-4

Table A.1: Virtual machine specifications

was used, providing all additions and changes introduced in chapter 4. Other components of VSimRTI or the underlying system were not modified.

[12] `http://www.cn.uni-duesseldorf.de`

A.2 Taubental and Taubental-large scenarios

This section includes important configuration files of the *Taubental* and *Taubental-large* scenarios.

applicationNT

```
{
  "messageCacheTime": 30000000000,
  "behaviorActive" : "false",
  "minimalCamLength" : 1500,
  "minimalDenmLength" : 1500,
  "v2XMessageCacheSize": 2147483647
}
```

Listing A.1: Taubental/applicationNT/applicationNT_config.json

```
{
  "_isEmergencyVehicle": "",
  "isEmergencyVehicle": false,
  "_interval": "",
  "interval": 1000000000,
  "_timeout": "",
  "timeout": 300000000,
  "_offset": "",
  "offset": 4000000000,
  "_radius": "",
  "radius": 5.0,
  "_slowDownSpeed": "",
  "slowDownSpeed": 0.0,
  "_obeyTime": "",
  "obeyTime": 200
}
```

Listing A.2: Taubental/applicationNT/emergencyWarningApp.json

```
{
  "_isEmergencyVehicle": "",
  "isEmergencyVehicle": true,
  "_interval": "",
  "interval": 100000000,
  "_timeout": "",
  "timeout": 200000000,
  "_offset": "",
  "offset": 4000000000,
  "_radius": "",
  "radius": 5.0,
  "_slowDownSpeed": "",
  "slowDownSpeed": 1.0,
  "_obeyTime": "",
  "obeyTime": 2000
}
```

Listing A.3: Taubental/applicationNT/emergencyWarningApp–veh_4.json

```
{
  "_sumoNetFile": "SUMO net file path",
  "sumoNetFile": "scenarios/Taubental/sumo/Taubental.net.xml"
}
```

Listing A.4: Taubental/applicationNT/GeoServerConfig.json

```
{
  "_interval": "",
  "interval": 1000000000,
  "_offset": "",
  "offset": 5000000000,
  "_defaultRoadId": "",
  "defaultRoadId": "defaultroadid",
  "_defaultLanePos": "",
  "defaultLanePos": 0.0,
  "_shouldTransmit": "",
  "shouldTransmit": true
}
```

Listing A.5: Taubental/applicationNT/GeoClientConfig.json

mapping3

```
{
  "prototypes":[
    {
      "applications":["de.hhu.tbus.applications.nt.emergencywarning
    ↪ .EmergencyWarningApp"],
      "name":"PKW",
      "accel":1.5,
      "decel":4.5,
      "length":5.00,
      "maxSpeed":0.5,
      "minGap":2.5,
      "sigma":0.5,
      "tau":1
    },
    {
      "applications":["de.hhu.tbus.applications.nt.geoserver.edge.
    ↪ server.TbusGeoserver"],
      "name":"RSU"
    }
  ],
  "rsus":[
    {
      "lat": 51.1640075,
      "lon": 6.7527653,
      "name": "RSU"
    }
  ],
  "vehicles":[
    {
      "startingTime":6.0,
      "route":0,
      "maxNumberVehicles": 1,
      "types":[{"name":"PKW"}]
    },
    {
      "startingTime":23.0,
      "route":1,
      "maxNumberVehicles": 1,
      "types":[{"name":"PKW"}]
    },
    {
      "startingTime":40.0,
      "route":2,
      "maxNumberVehicles": 1,
      "types":[{"name":"PKW"}]
    },
    {
      "startingTime":57.0,
      "route":3,
      "maxNumberVehicles": 1,
      "types":[{"name":"PKW"}]
    },
    {
      "startingTime":74.0,
      "route":4,
      "maxNumberVehicles": 1,
      "types":[{"name":"PKW"}]
    }
  ]
}
```

Listing A.6: Taubental/mapping3/mapping_config.json

```
{
  "prototypes":[
    {
      "applications":["de.hhu.tbus.applications.nt.emergencywarning
    ↪ .EmergencyWarningApp"],
      "name":"PKW",
      "accel":1.5,
      "decel":4.5,
      "length":5.00,
      "maxSpeed":0.5,
      "minGap":2.5,
      "sigma":0.5,
      "tau":1
    },
    {
      "applications":["de.hhu.tbus.applications.nt.geoserver.edge.
    ↪ server.TbusGeoserver"],
      "name":"RSU"
    }
  ],
  "rsus":[
    {
      "lat": 51.1640075,
      "lon": 6.7527653,
      "name": "RSU"
    }
  ],
  "vehicles":[
    {
      "startingTime":6.0,
      "route":0,
      "targetDensity": 30,
      "maxNumberVehicles": 10,
      "types":[{"name":"PKW"}]
    },
    {
      "startingTime":23.0,
      "route":1,
      "targetDensity": 30,
      "maxNumberVehicles": 10,
      "types":[{"name":"PKW"}]
    },
    {
      "startingTime":40.0,
      "route":2,
      "targetDensity": 30,
      "maxNumberVehicles": 10,
      "types":[{"name":"PKW"}]
    },
    {
      "startingTime":57.0,
      "route":3,
      "targetDensity": 30,
      "maxNumberVehicles": 10,
      "types":[{"name":"PKW"}]
    },
    {
      "startingTime":74.0,
        "route":4,
        "targetDensity": 30,
        "maxNumberVehicles": 10,
        "types":[{"name":"PKW"}]
      }
    ]
}
```

Listing A.7: Taubental–large/mapping3/mapping_config.json

vsimrti

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- file version: 2015-01-22 -->
<configuration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance
    ↪ "
               xsi:noNamespaceSchemaLocation="https://www.dcaiti.tu
    ↪ -berlin.de/research/simulation/download/get/scenarios/
    ↪ scenarioname/vsimrti/vsimrti_config.xsd">
    <simulation>
        <id>Taubental</id>
        <starttime>0</starttime>
        <!-- you will want to set this to a value that covers the
    ↪ whole time frame you want to simulate -->
        <endtime>5500</endtime>
        <!-- set the center coordinates to roughly the center of
    ↪ your target area -->
        <!-- the offset can be found in the generated network file
    ↪ for your traffic simulator, e.g. the .net.xml file for sumo
    ↪ -->
  <!-- <location netOffset="-342488.08,-5670384.60" convBoundary="0
    ↪ .00,0.00,839.52,706.16" origBoundary="342488.08,5670384.
    ↪ 60,343327.60,5671090.77" projParameter="!"/> -->
        <WGS84UTMTransformConfig>
        {
            "centerCoordinates": {
                "longitude": 6.0,
                "latitude": 51.0
            },
            "cartesianOffset": {
                "x": -342488.08,
                "y": -5670384.60
            }
        }
        </WGS84UTMTransformConfig>
        <threads>1</threads>
    </simulation>
    <federates>
        <!--
            Be aware that are all network simulators (excluding cell) are exclusive.
            Do NOT set multiple to true!
        -->
        <!-- Cellular network simulator -->
        <federate id="cell" active="false"/>
        <!-- V2X (ad hoc) network simulators -->
        <federate id="sns" active="false"/>
        <federate id="swans" active="false"/>
        <federate id="omnetpp" active="true"/>
        <federate id="ns3" active="false"/>
        <!-- Traffic simulators -->
        <federate id="sumo" active="true"/>
        <federate id="vissim" active="false"/>
        <!-- Electric vehicle simulator -->
        <federate id="battery" active="false"/>
        <!-- Navigation component -->
        <federate id="navigation" active="true"/>
        <!-- Application simulator -->
        <federate id="applicationNT" active="true"/>
        <!-- Mapping -->
        <federate id="mapping3" active="true"/>
        <!-- Environment simulator -->
        <federate id="eventserver" active="false"/>
        <!-- Visualization -->
        <federate id="visualizer" active="true"/>
        <!-- Emission dispersion -->
        <federate id="hefei" active="false"/>
    </federates>
</configuration>
```

Listing A.8: Taubental/vsimrti/vsimrti_config.xml

A.3 Database structure

The following is the SQL table structure used in TBUS's SQLite database:

```sql
CREATE TABLE upload (
  id INTEGER PRIMARY KEY,
  groupid INTEGER NOT NULL DEFAULT 1,
  timestamp INTEGER NOT NULL,
  roadId TEXT,
  lanePos REAL,
  cellId INTEGER NOT NULL DEFAULT 1,
  datarate REAL,
  droprate REAL,
  delay INTEGER
);

CREATE TABLE download (
  id INTEGER PRIMARY KEY,
  groupid INTEGER NOT NULL DEFAULT 1,
  timestamp INTEGER NOT NULL,
  roadId TEXT,
  lanePos REAL,
  cellId INTEGER NOT NULL DEFAULT 1,
  datarate REAL,
  droprate REAL,
  delay INTEGER
);
```

Listing A.9: tbus_edge.sql

A.4 Database Handler interface

The following listings introduce the `TbusDatabaseHandler` interface used as the data source by this thesis:

```
/*
 * DatabaseHandler.h
 *
 *  Created on: 13.10.2014
 *      Author: bialon
 */

#ifndef DATABASEHANDLER_H_
#define DATABASEHANDLER_H_

#include "TbusQueueDatarateValue.h"
#include "TbusQueueDelayValue.h"
#include "TbusCellShareTypes.h"
#include "omnetpp.h"

/**
 * Abstract base class for all database handlers to be used in TBUS
    ↪ .
 * Provides an interface for accessing data-/droprate and delay
    ↪ from data source.
 */
class TbusDatabaseHandler {
  private:
    /**
     * Private copy constructor
     * @param
     */
    TbusDatabaseHandler(const TbusDatabaseHandler&);
    void operator=(const TbusDatabaseHandler&);

    simtime_t offset;

  protected:
    /**
     * Protected Constructor.
     * Only inheriting is possible.
     */
    TbusDatabaseHandler();

    simtime_t getSimulationTimeWithOffset(simtime_t time = simTime
    ↪ ());

public:
    /**
     * Template function for instantiation of a singleton with
    ↪ given type T.
     * This is used for an easy interface throughout the using
    ↪ classes and easy instantiation and maintenance of the
    ↪ belonging data source.
     * @return Singleton instance of type T
    */
    template<class T> static TbusDatabaseHandler* getInstance() {
      static TbusDatabaseHandler* instance = new T(); ///< static
    ↪ local variable used for singleton cleanup

      return instance;
    }

    /**
     * Destructor.
     * Database cleanup, query finalization and database closing
    ↪ should be handled here by derived classes.
     */
    virtual ~TbusDatabaseHandler() {};

    /**
     * Return the cellid at position roadId and lanePos
     * @param roadId Current road id
     * @param lanePos Lane position
     * @return Cell id
     */
    virtual cellid_t getCellId(const char* const roadId, const
    ↪ float lanePos, simtime_t time = simTime()) = 0;

    virtual uint64_t getUploadGroupId(const char* const roadId,
    ↪ simtime_t time = simTime()) = 0;
    virtual uint64_t getDownloadGroupId(const char* const roadId,
    ↪ simtime_t time = simTime()) = 0;

    virtual TbusQueueDatarateValue* getUploadDatarate(const char*
    ↪ const roadId, const float lanePos, simtime_t time = simTime
    ↪ ()) = 0;
    virtual TbusQueueDelayValue* getUploadDelay(const char* const
    ↪ roadId, const float lanePos, simtime_t time = simTime()) =
    ↪ 0;
    virtual TbusQueueDatarateValue* getDownloadDatarate(const char*
    ↪  const roadId, const float lanePos, simtime_t time = simTime
    ↪ ()) = 0;
    virtual TbusQueueDelayValue* getDownloadDelay(const char* const
    ↪  roadId, const float lanePos, simtime_t time = simTime()) =
    ↪ 0;
};

#endif /* DATABASEHANDLER_H_ */
```

Listing A.10: database/TbusDatabaseHandler.h

```
/*
 * DatabaseHandler.cc
 *
 *  Created on: 16.07.2015
 *      Author: bialon
 */

#include "TbusDatabaseHandler.h"

Register_GlobalConfigOption(CFGID_TBUS_OFFSET, "tbus-offset",
    ↪ CFG_INT, 0, "Timestamp offset used for value retrieval")

TbusDatabaseHandler::TbusDatabaseHandler() {
  offset = SimTime(ev.getConfig()->getAsInt(CFGID_TBUS_OFFSET, 0),
    ↪ SIMTIME_NS);
};


/**
 * Returns the given simulation time (or now, if none) plus offset
 * @param time Simulation time
 * @return Time plus offset
 */
simtime_t TbusDatabaseHandler::getSimulationTimeWithOffset(
    ↪ simtime_t time) {
  return time + offset;
}
```

Listing A.11: database/TbusDatabaseHandler.cc

A.5 Cell-share interface

The following listing contains the C++ interface for the cell-share model used by this thesis:

```
/*
 * TbusCellShare.h
 *
 *  Created on: 17.02.2015
 *      Author: bialon
 */

#ifndef TBUSCELLSHARE_H_
#define TBUSCELLSHARE_H_

#include "omnetpp.h"
#include "TbusQueueDatarateValue.h"
#include "TbusQueueDelayValue.h"
#include "TbusCellShareTypes.h"

#include <map>
#include <set>

/**
 * Representation of a TBUS cell share module.
 */
class TbusCellShare {
  protected:
    /**
     * Protected constructor for inheriting.
     */
    TbusCellShare() {};

    typedef std::set<TbusHost*> HostSet; ///< Set of hosts in cell
    typedef struct TbusCell {
      uint64_t numActiveHosts;
      HostSet hosts;
    };

    /**
     * Returns the number of active hosts in the given cell.
     * @param cellId Cell id of cell
     * @return Number of active hosts
     */
    uint64_t getActiveHostsInCell(cellid_t cellId) {
      return idToCell[cellId].numActiveHosts;
    }

    /**
     * Get a set of all hosts currently connected to a cell with
   ↪ cell id cellId.
     * @param cellId Cell id
     * @return Host set
     */
    HostSet& getHostsInCell(cellid_t cellId) {
      return idToCell[cellId].hosts;
    }

  private:
    TbusCellShare(const TbusCellShare&);
    void operator=(const TbusCellShare&);

    /**
     * Maps a cell id to the number of connected hosts.
     */
    std::map<cellid_t, TbusCell> idToCell;

    /**
```

```
     * Update the number of cell active hosts.
     * @param cellId
     */
    void updateNumActiveHostsInCell(cellid_t cellId) {
      uint64_t numActiveHosts = 0;
      HostSet::iterator it;
      HostSet& hosts = idToCell[cellId].hosts;

      for (it = hosts.begin(); it != hosts.end(); ++it) {
        if ((*it)->callback->getQueueStatus() == CELL_ACTIVE) {
          numActiveHosts++;
        }
      }

      idToCell[cellId].numActiveHosts = numActiveHosts;
    }

  public:
    /**
     * Singleton instantiation for every model.
     * @return A TbusCellShare of model T
     */
    template<class T> static TbusCellShare* getInstance() {
      static TbusCellShare* instance = new T(); ///< Static local
   ↪ variable used for singleton cleanup

      return instance;
    }

    /**
     * Update cell model.
     * Host host changed cells between from and to.
     * @param from Cell host was in
     * @param to Cell host is in
     * @param host Optional host reference
     */
    void hostMoved(cellid_t from, cellid_t to, TbusHost* host) {
      if (from != TBUS_INVALID_CELLID) {
        idToCell[from].hosts.erase(host);

        updateNumActiveHostsInCell(from);
      }

      if (to != TBUS_INVALID_CELLID) {
        idToCell[to].hosts.insert(host);

        updateNumActiveHostsInCell(to);
      }
    }

    /**
     * Makes every host in cell cellId adapt its queue values.
     * @param cellId Cell id
     */
    void cellActivityChanged(cellid_t cellId) {
      HostSet& hosts = idToCell[cellId].hosts;
      HostSet::iterator it;

      // First, update the number of cell active hosts
      updateNumActiveHostsInCell(cellId);

      // Second, let all hosts adapt active queue's value
      for (it = hosts.begin(); it != hosts.end(); ++it) {
        (*it)->callback->adaptQueueValues(ALL);
      }
    }

    /**
     * Adapt the given value in a new returned value to the current
    ↪  cell model.
     * The optional parameter host can be used for additional
    ↪ information.
     * @param cellId Cell to adapt on
     * @param value Given TbusQueueDatarateValue
     * @param host Optional host reference
     * @return Adapted queue datarate value
     */
    virtual TbusQueueDatarateValue* adaptDatarateValue(cellid_t
    ↪ cellId, TbusQueueDatarateValue* value, TbusHost* host = NULL
    ↪ ) = 0;

    /**
     * Adapt the given value in a new returned value to the current
    ↪  cell model.
     * The optional parameter host can be used for additional
    ↪ information.
     * @param cellId Cell to adapt on
     * @param value Given TbusQueueDelayValue
     * @param host Optional host reference
     * @return Adapted queue delay value
     */
    virtual TbusQueueDelayValue* adaptDelayValue(cellid_t cellId,
    ↪ TbusQueueDelayValue* value, TbusHost* host = NULL) = 0;
};

#endif /* TBUSCELLSHARE_H_ */
```

Listing A.12: cellshare/TbusCellShare.h

Ehrenwörtliche Erklärung

Hiermit versichere ich, die vorliegende Masterarbeit 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, 11. August 2015

Raphael Bialon

Place DVD cover

here

This DVD contains:

- a README file
- a *pdf*-version of this Master's Thesis
- the LaTeX and graphics sources of this thesis including all used scripts
- the source files of the developed software
- simulated scenarios and used data
- simulation results
- websites of the used internet sources