Realistic Simulation of Vehicular Communication and Vehicle-2-X Applications

Björn Schünemann, Kay Massow, Ilja Radusch
Technische Universität Berlin
Ernst-Reuter-Platz 7, 10587 Berlin, Germany
{bjoern.schuenemann, kay.massow, ilja.radusch}@dcaiti.com

ABSTRACT

In future intelligent transport systems, wireless vehicular communication will provide the basis for new applications to enhance safety, traffic efficiency, and provide infotainment services. In the near future, field tests are to be carried out to verify the improvements that could be achieved by these new Vehicle-2-X applications. However, the realisation of such field tests is very complex and expensive. Therefore, detailed simulations are necessary to prepare the tests in the real world and reduce their costs. Current simulation tools do not support all aspects necessary for Vehicle-2-X applications. In this paper, we present an integrated software simulation environment that fulfils the special requirements of Vehicle-2-X applications. Furthermore, we introduce our testbed architecture that allows simulating vehicular communication under real physical conditions.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design—*Wireless communication*;
I.6.7 [**Simulation and Modeling**]: Simulation Support Systems—*Environments*

General Terms

Design, Experimentation, Performance

Keywords

Simulator, Vehicular Ad-hoc Network, VANET, Vehicle-2-X Communication

1. INTRODUCTION

An important aim of the scientific and industrial automotive research is the development of applications that enhance safety, traffic efficiency, and provide infotainment services. Vehicular communication based on wireless short-ranged networks (Vehicle-2-X Communication) provides the foundation for such applications. For example, an obstacle warning application can inform other vehicles in its neighbourhood about this dangerous situation before the obstacle is visible for them. Field tests are carried out to evaluate the improvements in traffic safety and efficiency that could be achieved by these new applications.

However, detailed simulations have to precede the tests in the real world. Currently, different kinds of simulators are necessary for the simulation of vehicular communication. Traffic simulators are used to generate the movement of vehicles. They can, in general, be classified into macroscopic and microscopic simulators [13]. A macroscopic simulator considers global measures, i.e. traffic density and traffic flow, to compute road capacity and distribution of traffic in the road net. In contrast, microscopic simulators determine the movement of each vehicle that participates in road traffic. In general, traffic simulators have no or only rudimental functionality for the simulation of direct and multi-hop vehicular communication. Hence, network simulators are used to simulate the communication between vehicles. They simulate all aspects of the behaviour of a wireless network, such as medium access control, signal strength, and propagation delays.

One key requirement is the interaction at runtime of the simulation between network simulator, traffic simulator, and the application. Thereby, modifications of traffic parameters, like movements of vehicles and characteristics of roads, need to be made at runtime. For example, when an obstacle warning application of a vehicle detects a dangerous situation, this vehicle sends a warning message to vehicles in its neighbourhood using the network simulator. As a result, receivers of this warning could, then, change their routes, which has to be fed into the traffic simulator.

Our architecture couples both traffic and network simulator interactively. Moreover, an environment generator provides real road maps and an application interface simulator allows the integration of applications designed for real vehicles.

1.1 Related Work

Several tools exist, e.g. MOVE [5] and traceExporter[1], which use a traffic simulator for the movement generation of the vehicles and, then, convert the movement files into a format that can be interpreted by the network simulator. The advantage of this combination of simulators is that realistic traffic scenarios are generated by the traffic simulator and,

[1]TraceExporter.
http://sumo.sourceforge.net/wiki/index.php/TraceExporter

SIMUTools, March 03 - 07, 2008, Marseille, France.

then, used by the network simulator to generate the wireless communication between the vehicles. However, these tools are not suitable for applications where vehicle-2-X communication is used to change the route of vehicles. Since the movements of the vehicles are generated before the network simulation starts, vehicles cannot respond to warning messages and, thus, change their routes.

TraNS [15] is an Open Source simulation environment that integrates both traffic and network simulators. The incorporation of the traffic simulator SUMO [7] and the network simulator ns-2[2] allows the generation of realistic mobile traces used by the network simulator. It is planned to implement a *feedback loop*, i.e. a TCP communication between ns-2 and SUMO, that enables a direct interaction of both simulators. The current version 0.21 of TraNS does not implement the *feedback loop* and only allows the generation of mobility traces.

The Multiple Simulator Interlinking Environment for C2CC in VANETs [9] was developed at the University of Duesseldorf within the Network on Wheels (NoW) project[3] in cooperation with Volkswagen AG. It integrates the following simulators: ns-2 for network simulation, VISSIM[4] for traffic simulation, Matlab/Simulink[5] for application development and simulation, and Click! [6] for routing protocol simulations. The simulation environment consists of a centralized architecture where the controlling instance is provided by the ns-2 simulator. It is possible to influence the simulation procedure, i.e. to stop one or more cars, influence the driver behaviour, or to reroute cars. A disadvantage of [9] is that the whole simulation process is controlled by the network simulator. In the real world, applications initiate the necessary activities themselves.

The event-based simulator GrooveNet [11] supports multiple vehicle, trip and mobility models over a variety of network link and physical layer models. It includes simple car-following, traffic lights, lane changing, and simulated GPS models. Three types of simulated nodes are supported: vehicles which are capable of multi-hopping data over one or more DSRC channels, fixed infrastructure nodes, and mobile gateways capable of vehicle-to-vehicle and vehicle-to-infrastructure communication. Furthermore, GrooveNet supports multiple message types such as beacons, which are broadcast periodically to inform neighbours of the current position of a vehicle, and vehicle emergency and warning event messages with priorities. Moreover, communication between simulated vehicles and real vehicles on the road is possible. A disadvantage of GrooveNet is that only few communication protocols are supported. In fact, this limits the accuracy of the simulation of Vehicle-2-X applications and avoids the realistic evaluation of network metrics, such as packet delivery ratio or packet end-to-end delay.

1.2 Outline

This paper is organised as follows. Section 2 gives a short overview about Vehicle-2-X applications and presents several use case examples. In Section 3, we analyse the requirements and introduce our new software simulation architecture. Our real world testbed is described in Section 4. Finally, section 5 concludes this paper and outlines future work.

2. SCENARIOS AND USE CASES

The aim of our simulation architecture is to evaluate the effectiveness of Vehicle-2-X applications for enhancing safety, traffic efficiency, and providing infotainment services. The Car2Car Communication Consortium Manifesto[6] defines several scenarios and use cases for such Vehicle-2-X applications. In the following sections, we give a short overview about a selection of them.

2.1 Safety

Safety use cases are characterised by vehicular communication used to mitigate dangerous situations and accidents. In general, they interpret information received from vehicles in the neighbourhood and react if a dangerous situation could occur. The following three use cases illustrate ways of increasing traffic safety:

Cooperative Forward Collision Warning: During driving, vehicles share relevant information, i.e. position, speed, and direction, with vehicles in their neighbourhood and monitor the activities of their own drivers. To prevent rear-end collisions, a vehicle warns its driver if it detects another vehicle in its critical proximity.

Pre-Crash Sensing/Warning: Similar to the Cooperative Forward Collision Warning, all vehicles periodically share information from neighbouring vehicles to predict a collision. If a collision is not avoidable, the involved vehicles exchange more detailed information, e.g. vehicle size, to enable an optimized use of air bags, motorized seat belt tensioners, and extendable bumpers.

Hazardous Location V2V Notification: In contrast to the examples above, this use case utilizes the network of vehicles to share information about dangerous locations on the roadway, e.g. slippery roads or potholes. Additionally, information from external service providers can be accessed via a roadside unit and propagated through the vehicular ad hoc network. The received information is used by the vehicles to either inform the driver or automatically optimize the safety systems.

2.2 Traffic Efficiency

Traffic efficiency use cases improve the efficiency of the transportation network by providing information either to the owners of the transportation network or to the drivers on the network. Vehicular communication is used to create and share traffic related information in a way that is not possible without this communication technology.

Enhanced Route Guidance and Navigation: In this use case, the infrastructure owner collects data for predicting traffic congestion on roadways. As a result, information on current and expected traffic conditions is sent to the drivers via roadside units. The vehicles

[2]The network simualtor ns-2.
http://www.isi.edu/nsnam/ns/
[3]Network on Wheels (NoW) project.
http://www.network-on-wheels.de/
[4]VISSIM.
http://www.ptv-vision.com/cgi-bin/traffic/traf_vissim.pl
[5]Matlab/Simulink. http://www.mathworks.com/

[6]Car2Car Communication Consortium Manifesto, September 2007. http://www.car-2-car.org/index.php?id=570

inform their drivers about expected delays and better routes that may exist due to the traffic conditions.

Green Light Optimal Speed Advisory: The idea is to provide a smoother driving and avoid stopping by receiving information regarding the locations of the intersections and their signal timings. As a result, vehicles can calculate their optimal speed to arrive at an intersection when the signal is green.

V2V Merging Assistance: With the help of the V2V Merging Assistance, merging vehicles can join the flowing traffic without disrupting the traffic flow. When a vehicle enters an on-ramp to a limited access roadway, this vehicle communicates with the traffic participants of the roadway in order to merge into the regular traffic in a non-disruptive and safe way.

2.3 Infotainment and Remote Diagnostics

This category contains the remaining use cases which are not directed at Safety or Traffic Efficiency. In general, they offer entertainment and other information on a regular basis or provide additional services, such as diagnostic information for a more efficient service at a garage.

Internet Access in Vehicle: This use case provides common IP based services in vehicles. So, different vehicles establish a multi-hop route to a roadside unit that acts as an Internet gateway.

Point of Interest Notification: Here, roadside units broadcast information regarding local businesses, tourist attractions, or other points of interest to vehicles in the vicinity. The huge amount of information is filtered by the vehicles and only the appropriate pieces are presented to the driver. For instance, if the fuel level is low, the vehicle informs the driver about locations and prices of filling stations in its neighbourhood.

Remote Diagnostics: Remote Diagnostics is to be used to reduce the amount of time and costs necessary for a service in a garage. When a vehicle enters the neighbourhood of a service garage, the service garage requests diagnostic information related to the problem reported by the customer. Furthermore, if software updates are required, the system can install these updates.

3. SOFTWARE SIMULATION ARCHITECTURE

To simulate all aspects of the use cases introduced in section 2, a simulation environment is necessary that integrates traffic and network simulator. Both simulators have to run simultaneously and interact at runtime. In addition, it should be possible to integrate future applications of real vehicles, e.g. an electronic brake light application, into the simulation environment. For this reason, an application interface simulator contains virtual machines to run these applications. The application interface simulator has to provide interfaces for sending/receiving messages and controlling the movement of the vehicles. Since traffic and application interface simulators use the same road map data, an environment generator is to be used that includes road map data and location-based information. Location-based

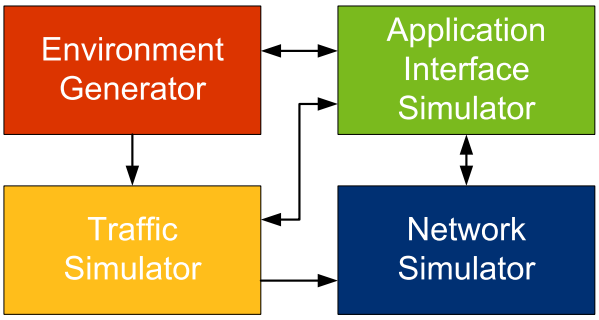

Figure 1: Components of the Software Simulation Architecture

information can be divided into static information, e.g. maximum velocity of a road section, and temporary information, e.g. the location of an obstacle or black ice. The temporary location-based information also includes data provided by the traffic management centre.

We decided to integrate only open source tools in our architecture thus keeping the flexibility to modify interfaces for realising the interaction between the simulators. The second advantage is that everybody can use our simulation environment for free.

The components of our architecture and the data flow between them is depicted in Figure 1. In the following subsections, we present the requirements that our architecture fulfils and give a detailed overview about all components.

3.1 Requirements

As a result of our analysis, the following main requirements have to be fulfilled to simulate Vehicle-2-X applications:

- Interaction between traffic, network, and application interface simulators during runtime of the simulation.
- Each particular vehicle, simulated by the traffic simulator, has to be addressable by network and application interface simulators, e.g. for modification of movement.
- The application interface simulator has to be able to use the network simulator for sending and receiving messages via the vehicular ad-hoc network.
- Road map related data, e.g. the throughput of a road when obstacles restrict the traffic flow, has to be modifiable at runtime of the simulation.

3.2 Interactions

This section explains the required interactions between the three simulators and the database for the simulation of the use cases introduced in chapter 2. For every use case, the initial interaction is caused by the application interface simulator. Here, the applications integrated in the vehicles and the overall infrastructure, e.g. Roadside Units (RSUs) and traffic management centre, are implemented.

3.2.1 Interaction between traffic and application interface simulator

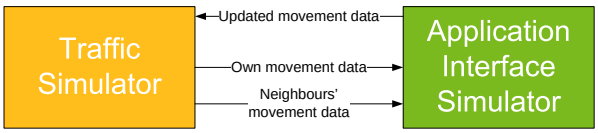

Figure 2: Interaction between traffic and application interface simulator

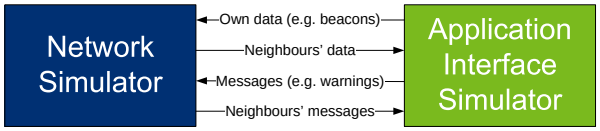

Figure 3: Interaction between network and application interface simulator

The following movement data of vehicles, depicted in Figure 2, are exchanged between traffic and application interface simulator:

- Each vehicle, periodically, receives information on its own position, velocity, and direction from the traffic simulator. This is necessary to simulate the detection of the own location, done by GPS in reality.
- From the traffic simulator, each vehicle obtains position, velocity, and direction of its neighbours within its range of vision. In reality, this data are detected by the own sensors of a vehicle, e.g. realised by radar.
- A vehicle sends its updated data to the traffic simulator if it changes its own position, velocity, or direction in response to an exceptional situation, e.g. to circumnavigate congestion.

3.2.2 *Interaction between network and application interface simulator*

Wireless communication of vehicles is simulated by the network simulator. Therefore, the following data flow between network and application interface simulator is necessary (see Figure 3):

- Each vehicle, periodically, sends Beacons, short messages with its own position and other relevant data, via the network simulator to its neighbours and reachable Roadside Units. Furthermore, information about important circumstances and exceptional situations, e.g. debris on the road, detected by own sensors or received from neighbours is also sent to relevant vehicles and Roadside Units in the neighbourhood.
- Each vehicle, periodically, receives Beacons and other relevant data, e.g. warnings, service messages delivered by the traffic management or other services, from its neighbours and reachable Roadside Units via the network simulator.

3.2.3 *Interaction between environment generator and application interface simulator*

Since all location-based information is provided by the environment generator, Vehicle-2-X applications need access to the following data:

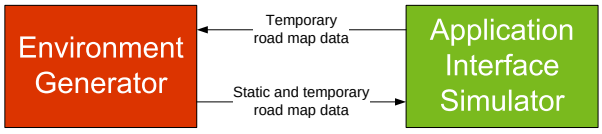

Figure 4: Interaction between environment generator and application interface simulator

Figure 5: Interaction between environment generator and traffic simulator

- Information about exceptional situations, e.g. about obstacles, road works, or congestions, is provided by the environment generator. Vehicles obtain this temporary location-based data directly from the environment generator if the detection by own sensors is to be simulated, i.e. the exceptional situations occur at close range of the vehicle. Moreover, static road map data, e.g. the maximum velocity of a road section, are received by the applications from the environment generator if necessary.
- Applications can change temporary road map data, e.g. the traffic management centre sends information about congestions to the environment generator.

Figure 4 depicts the interactions between environment generator and application interface simulator.

3.2.4 *Interaction between environment generator and traffic simulator*

The traffic simulator obtains static road map data during the initialization of the simulation. Temporary data, e.g. the reduced throughput of a road section due to an obstacle, is updated periodically. Figure 5 depicts this interaction.

3.2.5 *Interaction between traffic and network simulator*

The network simulator, periodically, receives the current positions of the vehicles generated by the traffic simulator (see Figure 6).

3.3 The Traffic Simulator

To generate the movements of the vehicles, a traffic simulator is needed. We evaluated the three open source simulators SUMO [7], VanetMobiSim [4], and FreeSim [12]. For

Figure 6: Interaction between traffic and network simulator

our simulation architecture, SUMO and VanetMobiSim has proved to be the best choice.

SUMO is an microscopic traffic simulation package which was developed by the Institute of Transportation Research at the German Aerospace Centre. It is designed to handle large road networks. Each vehicle has an own route and is simulated individually. To simulate the movements of the vehicles on the network, a model is used that uses discrete time steps of 1s. About 100,000 to 200,000 vehicles can be simulated in real time on a desktop PC, including the simulation of traffic lights, right-of-way rules, and lane changing. Road maps are created by importing other formats, such as ESRI Shapefile[7] or TIGER-maps[8]; or by generating abstract, geometrical networks. Furthermore, vehicle routes, based on different routing paradigms, can be computed.

The SUMO developers are currently working on a socket interface, called TraCI [17], that allows controlling SUMO from an outside application. This interface has already been integrated in the SVN path and is to be an integral part of the next stable release of SUMO. We use this interface to allow the application interface simulator to change the movements of vehicles. Moreover, the environment simulator sends updated temporary road map data, e.g. the reduced throughput of a road section due to an obstacle, to the traffic simulator using this interface.

The other traffic simulator, used by our simulation environment, is VanetMobiSim. This tool is an extension of the CANU Mobility Simulation Environment[9], a framework for user mobility modelling. CanuMobiSim is JAVA-based and can generate movement traces in different formats supporting different simulation tools for mobile networks. The VanetMobiSim extension focuses on vehicular mobility and features automotive motion models at both macroscopic and microscopic levels. At macroscopic level, VanetMobiSim can import maps or randomly generate them. Furthermore, it adds support for multi-lane roads, separate directional flows, differentiated speed constraints, and traffic signs at intersections. At microscopic level, VanetMobiSim implements mobility models providing Vehicle-2-X interaction. According to these models, vehicles regulate their speed depending on the behaviour of nearby vehicles, overtake each other, and act according to traffic signs at intersections.

In contrast to SUMO, VanetMobiSim is rather simple and supports few features only. The advantage of this tool, however, is its clear programming structure and good documentation. An extension module allows the easy integration of new features. Hence, we were able to integrate interfaces for the interaction with both the network and application interface simulator.

3.4 The Network Simulator

ns-2[10], JiST/SWANS [2], GloMoSim [1], and OMNeT++ [16] are open source products that are primarily used for the simulation of communication networks. We decided to integrate JiST/SWANS into our simulation environment. JiST is a discrete event simulation engine that runs over a standard Java virtual machine. Simulation code that runs on JiST does not need to be written in a domain-specific language. Instead, JiST converts an existing virtual machine into a simulation platform by embedding simulation time semantics at the byte-code level. Thus, JiST simulations are written in Java, compiled using a regular Java compiler, and run over a standard, unmodified virtual machine. SWANS is a scalable wireless network simulator built atop the JiST platform. It is organized as an independent software component that can be composed to form complete wireless network or sensor network configurations. The capabilities of SWANS are similar to those of ns-2 and GloMoSim but SWANS has a better performance on the simulation of large networks. The simulator uses the JiST design to run standard Java network applications over simulated networks.

Moreover, our simulation environment provides interfaces for ns-2. As a result, it is possible to integrate this network simulator into a future version.

3.5 The Application Interface Simulator

The application interface simulator is used to integrate applications of real vehicles into the simulation environment. The main components of the application interface simulator are virtual nodes and a virtual node manager. A virtual node provides the environment for a simulated application. The virtual node manager deploys and starts virtual nodes, separated from each other, and assigns resources and unique network addresses to them so that a communication between a virtual node and the network simulator becomes possible. Furthermore, it synchronizes the clocks of the virtual nodes with those of the other simulators. After a simulation is finished, all resources used by virtual nodes are freed again by this manager.

3.6 The Environment Generator

All map related data are managed by the environment generator. In addition, the conventional road map data are supplemented with some additional pieces of information, necessary for the use cases as introduced in section 2:

- Speed limits, car capacity, and the number of lanes of all roads are the static parts, which are necessary for the safety and traffic use cases. In general, this data does not change at runtime of the simulation.
- Congestions, accidents, and obstacles are temporary objects, stored by the environment generator, which are used to simulate a living world with changing conditions.
- Infotainment use cases require additional data, e.g. Points of Interest. These types of temporary data are updated at runtime by several services.

We decided to use the environment generator eWorld[11] to manage all road map related data. eWorld is a framework that imports road map data from providers, such as OpenStreetMap.org (OSM)[12], visualize it, edit and enrich it with events or additional attributes. Moreover, it passes

[7] ESRI Shapefile Technical Description. http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf

[8] TIGER - Topologically Integrated Geographic Encoding and Referencing system. http://www.census.gov/geo/www/tiger/

[9] CANU Mobility Simulation Environment (CanuMobiSim). http://canu.informatik.uni-stuttgart.de/mobisim/

[10] The network simualtor ns-2. http://www.isi.edu/nsnam/ns/

[11] eWorld. http://eworld.sourceforge.net/

[12] OpenStreetMap. http://openstreetmap.org/

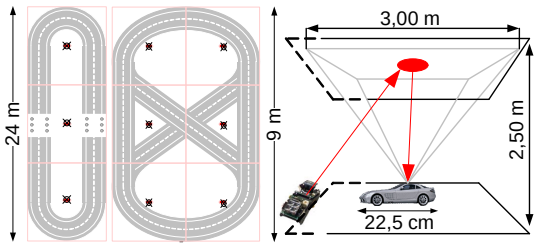

Figure 7: The Road Maps and the Positioning System of the Testbed

the managed data to traffic simulators, such as SUMO or VanetMobiSim.

OpenStreetMap.org (OSM) is a road map data service, which allows everybody to view, edit, and use geographical data in a collaborative way. Similar to Wikipedia, mistakes and errors are supposed to be corrected by users who examine changes committed by others on a voluntary basis. The entire content is free and can be used without restrictions. The OSM system allows precise requests of features, e.g. features in a specified area, only features with certain attributes, or features related to a given feature. Besides serving requests for a certain area, OSM provides all of its currently available data in a file called planet.osm, which contains compressed over 11 GB road map related information. Currently, OSM provides detailed material for densely populated areas like bigger cities. But, the amount of data grows quickly so that detailed data of sparsely populated areas should be available soon.

4. THE REAL WORLD TESTBED

The development of intervening driver assistance systems, mentioned in section 2, demands a sub microscopic test environment which differs considerably from the simulator tools mentioned in the former chapters. The influences of physical environment parameters on the behaviour of a vehicle are very complex in nature. Due to this complexity, it is not possible to entirely simulate these parameters by employing software simulations. Hence, it is inevitable to substitute parts of the simulation by real world experiments. Combining real world vehicles and software simulations provides a promising approach for designing such a test environment. In [3], the advantages of such a combination for testing Advanced Driver Assistance Systems (ADAS) [10] are shown.

For the reasons above, we have developed a real world Vehicel-2-X communication testbed. This testbed, using remote controlled model cars, is less cost-intensive and easier to scale than test vehicles in the real world. Therefore, we propose an architecture, which allows a hybrid simulation combining real and simulated entities. The testbed is designed to be coupled with the simulation architecture, introduced section 3, by replacing the Application Interface Simulator. In order to perform tailored test procedures for application evaluation, developers are able to create scenarios using the model cars. The scope of the testbed comprises applications for intervening and warning driver assistance systems as well as traffic efficiency enhancing systems.

Our testbed is an indoor system with an area size of $100m^2$. To enable the optimal space utilization, we use remote controlled model cars on a scale of 1:18. These are

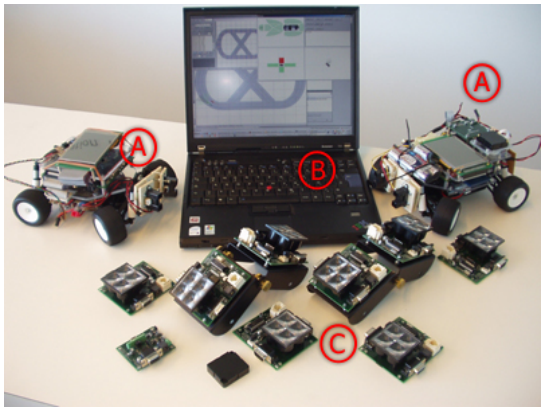

Figure 8: Hardware of the Testbed

the smallest model cars able to be equipped with the necessary embedded computer and sensors for obstacle detection. Accordingly, the test area corresponds with a real area of $32400m^2$. The model cars either drive autonomously or are remote controlled by a Control Centre deployed on a PC using WLAN for the communication. We realised the computer controlled coordination of the cars and the simulation of GPS data with the help of a Northstar[13] positioning system. This system uses infrared light spots that are projected onto the ceiling. Detectors, mounted on top of the cars, are able to localize these spots. Thus, the positions of the cars are calculated. Figure 7 depicts the Northstar positioning implementation and our road maps. One projector is able to cover an area of 3x3 m with position accuracy between 1 and 4 cm. We implemented two road maps by combining these 3x3 m cells. The first is an oblong arrangement which allows driving the models cars along a straight course at relative high speeds, in order to set up highway scenarios. The second is an oval track which enables testing of e.g. intersection and merging scenarios.

The hardware associated to the testbed is depicted in Figure 8: A indicates our model car prototypes, B shows the Control Centre deployed on a notebook, and C is the equipment of the Northstar system. As a showcase and proof of concept, we implemented the scenario Emergency Lane Changing. Accordingly, Vehicle-2-X communication is used to perform cooperative collision avoidance on a two-lane road.

In the following sections, we motivate the underlying architecture. Moreover, the realization of our testbed and an implemented scenario are described there.

4.1 The Hybrid Simulation Approach

To clarify the general structure of the architecture, we present some preliminary considerations concerning hybrid simulation aspects.

The testbed facilitates hybrid simulation by providing two compatible runtime environments: the hardware runtime environment integrated in the model cars and a software emulator engine. A realistic interaction among applications running in both environments is necessary to emulate the communication of real cars [8]. Therefore, the architecture

[13]NorthStar. Evolution Robitcs. http://www.evolution.com/products/northstar/. 2007

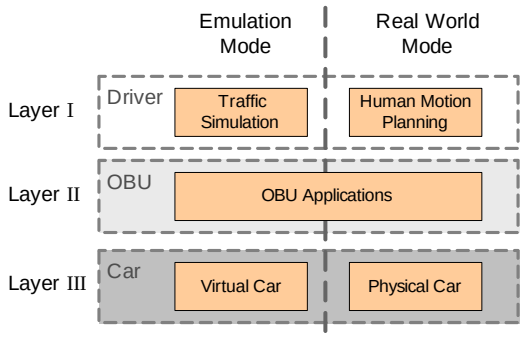

Figure 9: Architectural Arrangement

of the testbed is designed to allow the execution of several instances of the same application in both environments. The software emulation is distinguished from the earlier mentioned simulation by providing a runtime environment for the applications. From the point of view of the application, this environment has to act exactly like the hardware environment. However, the simulation is necessary to imitate the realistic behaviour of the overall system i.e. a large amount of cars in a large area. Hence, our architecture provides two compatible modes for real world and emulation purpose: Real World Mode and Emulation Mode. In the Emulation Mode, the overall missing hardware is simulated. In contrast, only the missing driver commands are generated in the Real World Mode.

To run the applications in software emulated cars, model cars, and, subsequently, cars in the real world, we developed an architecture distinguishing three layers and the two mentioned modes. We arranged the components of our architecture, according to the these modes and layers, as depicted in Figure 9.

- Layer I represents the driver's controlling behaviour. In Real World Mode, a software module performs the motion planning, i.e. the movement of the cars depending on their waypoint routes inside the testbed area. In Emulation Mode, the controlling of the cars is done by the traffic simulator.
- Layer II represents the On Board Unit OBU, i.e. all components needed to run the applications in the car, independent of vendor specific hardware. For the Real World Mode, this includes a dedicated embedded computer; for Emulation Mode, a compatible software emulation module is realised.
- Layer III represents the car, its sensors, and its actuators. In Real World Mode, a model car provides sensor data, e.g. concerning obstacle detection, as well as actuators. Actuators are used by the OBU applications to influence the state of a car. In Emulation Mode, the OBU is provided with simulated sensor data. Control commands, coming from the OBU, influence the state of the software simulated car.

4.2 The Architecture of the Testbed

In this section, we introduce our real world testbed architecture. Functional building blocks compose the architecture according to the three layers described in section 4.1.

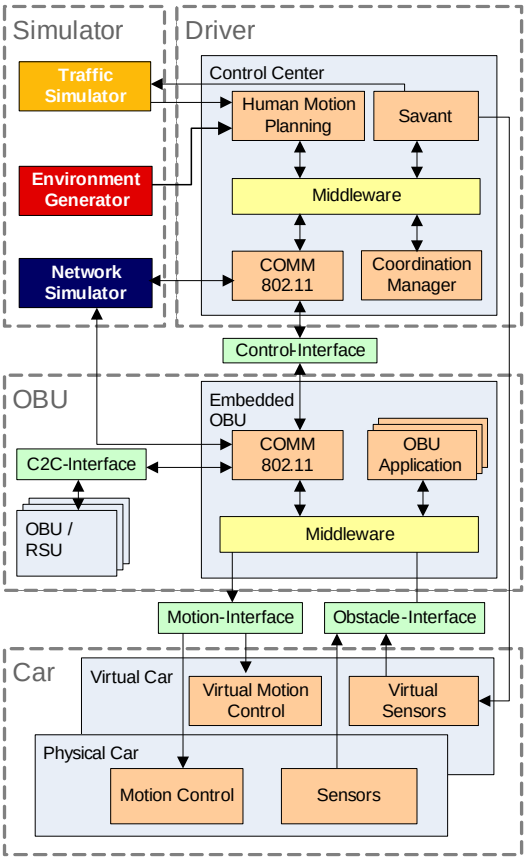

Figure 10: The Architecture of the Testbed

The whole architecture is illustrated in Figure 10. The components OBU and Car are deployed on a model car in Real World Mode while the components Driver and Simulator are deployed on a PC. The communication between model car and PC is realized by an IEEE 802.11a connection. However, an IEEE 802.11p communication can also be integrated later. In Emulation Mode, all components are deployed on a PC. Instead of the Application Interface Simulator described in section 3.5, the testbed architecture can be coupled to the software simulation environment.

Driver Layer: The Control Centre is used to control the cars. In Real World Mode, developers are able to define the routes cars are to move. This is done by specifying time coupled waypoints inside the testbed area with the help of the Coordination Manager. In that way, the driver's input is imitated and the coordination of the cars for simulating concrete scenarios is achieved. For example, a concrete scenario is the simulation of a situation that leads to an accident. In order to prevent the accident, the ADAS, to be tested with the help of this scenario, is forced to intervene and override the predefined routes. The Human Motion Planning component generates control commands at runtime in order to drive the cars. These commands are used to emulate human driving behaviour. The Savant component tracks all processes and car positions. Furthermore, it transmits the car positions to the Traffic Simulator. In Emulation Mode, the waypoints are

not generated by developers but by the Traffic Simulator. All components are able to interact with each other via a dedicated communication Middleware. The connection to the counterpart of the Middleware at the OBU Layer is established by the COMM module. In Real World Mode, this module is realized by a standard IEEE 802.11[14] component which is part of a network connecting all model cars and the Control Center deployed on PC. The link to the Network Simulator is necessary to realize the communication with emulated cars. In Emulation Mode the communication is done completely by the Network Simulator.

OBU Layer: Applications evaluated in the testbed are called OBU Applications. They are deployed in the Embedded OBU container. In Real World Mode, this container represents an embedded computer, which is installed on the model car. In contrast, the Embedded OBU container represents a software emulation module in Emulation Mode. This module is a runtime environment imitating the embedded computer. The sensors deliver information to the OBU Applications about obstacles in the neighbourhood of a car, by leveraging the Middleware. Moreover, steering and longitudinal control commandos are sent to the car in order to override the driver's control commands. Furthermore, the communication to the Control Center as well as to other cars and Road Side Units is realized by the Middleware via the COMM module.

Simulator: As described earlier, the Simulator is coupled with the testbed architecture. In order to control the movement of the emulated cars, the Traffic Simulator delivers their current positions. Depending on location based data delivered by the Environment Generator, the positions of the cars are processed by the Human Motion Planning component at runtime. The positions are updated and sent back by the Savant component.

Car Layer : The Physical Car represents a model car in the Real World Mode, whereas the Virtual Car stands for a software component that substitutes a model car in Emulator Mode. Both are able to receive control commands from the OBU through the Motion Interface. The Motion Control is the interface to the motor speed and steering control of a model car. In Emulator Mode, the Virtual Motion Control computes position updates, caused by the received control commands, and commits them to the Savant. Obstacles, detected by the Sensors, are sent to the OBU through the Obstacle Interface. In Emulator Mode, this task is performed by the Virtual Sensors with the help of position information coming from the Savant.

4.3 Scenario: Emergency Lane Changing

As a showcase and proof of concept for our testbed, we have implemented a Emergency Lane Changing application. This application addresses the effects of Vehicle-2-X communication concerning autonomous vehicle collision avoidance. The procedure of this scenario is described in the following and depicted in Figure 11.

[14]IEEE Standard for Wireless LAN-Medium Access Control and Physical Layer Specification. IEEE, 1990

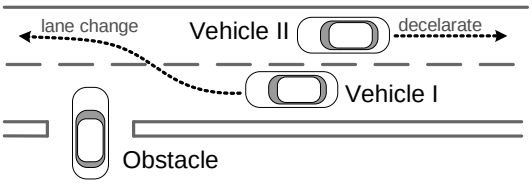

Figure 11: Lane Change Manoeuvre to Avoid a Collision

Two vehicles drive side by side on a two-lane road. Vehicle I is ahead about half a car length. Unforeseeable, an obstacle appears on the lane of Vehicle I. The distance to the obstacle is so close that the driver's response time is too long for an emergency stop. Furthermore, an abrupt stop involves the danger of rear end collisions by following vehicles. Lane II is blocked by Vehicle II. A lane change manoeuvre of Vehicle I would require an immediate reaction of Vehicle II to avoid a crash of both vehicles. The Advanced Driver Assistance System, installed on both vehicles, recognizes the hazardous situation. Vehicle I informs Vehicle II about its intent to change the lane. As a result, Vehicle II brakes autonomously and allows Vehicle I to change the lane. Moreover, all other following vehicles get informed of the incident. Hence, they can adapt to the dangerous situation. The merging of the following vehicles to lane II can be performed in sufficient time before the obstacle is reached. Furthermore, the influence to the traffic flow is minimised.

To evaluate the above scenario in our testbed, the following workflow is necessary: The developers define the desired routes of the two vehicles inside the testbed. Time coupled waypoints are used to define the side by side driving of the vehicles and the position of the obstacle. At runtime, the Control Center navigates the model cars along the defined routes. The application, implementing the Advanced Driver Assistance System, is running on the embedded computer attached to the model car. During the drive, the application analyses the data delivered by the sensors and the state of the car. When the sensors of the model car detect the obstacle, the application assumes the control of the car. As a result, the control commands of the Control Center are overridden. The application contacts the other model car that reacts as described above. For both vehicles, adapted routes are computed and driven. After passing the obstacle, the control of the cars is given back to the Control Center.

Figure 12 depicts the visualisation of the testbed at the Control Center running on a PC. The predefined routes of the model cars inside the test area are shown. The grey model car leaves its lane in order to bypass the obstacle. The screen depicted in the upper left corner visualizes the sensor data of the model car. The screen in the upper right corner shows information concerning Vehicle-2-X communication with the neighbourhood.

5. CONCLUSION AND FUTURE WORK

In this paper, we present an integrated software simulation architecture and a testbed for the simulation of Vehicle-2-X applications enhancing safety, traffic efficiency, and providing infotainment services. Since wireless communication between driving vehicles as well as the modification of movements as a result of this communication is a strong re-

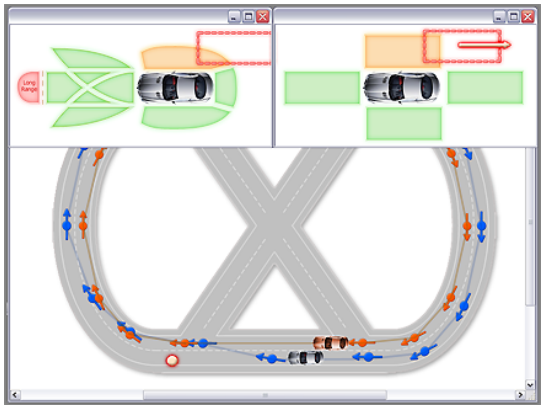

Figure 12: Visualisation Tools of the Control Center

quirement for the simulation of Vehicle-2-X applications, an interaction between traffic and network simulator at runtime is necessary. Furthermore, our architecture integrates an application interface simulator, used to implement real Vehicle-2-X applications, and an environment generator for the management of all location-based information necessary for traffic simulation and Vehicle-2-X applications.

We use our software simulation environment to further investigate the use cases introduced in section 2. The intention of our work is to limit the costs of expensive field tests in the real world as far as possible. Thereby, our testbed is used to emulate real physical conditions. One aim of these simulations is to evaluate the balance between network load and information exchange of the traffic participants. Since the bandwidth of wireless networks is limited, a high packet sending rate can cause network congestion and result in a delay or loss of transmitted packets. Hence, it is necessary to restrict the transmission of messages. On the other hand, a key requirement for the realisation of the Vehicle-2-X applications is the periodical information exchange between vehicles, e.g. traffic participants have to send their current movements to their neighbours. As a result, our simulations help to detect a tradeoff between network load and information exchange.

6. ACKNOWLEDGMENTS

The authors would like to thank Tobias Queck for his invaluable contributions.

7. REFERENCES

[1] L. Bajaj, M. Takai, R. Ahuja, R. Bagrodia, and M. Gerla. Glomosim: A scalable network simulation environment. Technical Report 990027, 13, 1999.

[2] R. Barr, Z. J. Haas, and R. van Renesse. Jist: an efficient approach to simulation using virtual machines: Research articles. *Softw. Pract. Exper.*, 35(6):539–576, 2005.

[3] O. Gietelink, J. Ploeg, B. De Schutter, and M. Verhaegen. Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations. *Vehicle System Dynamics*, 44(7):569–590, July 2006.

[4] J. Härri, F. Filali, C. Bonnet, and M. Fiore. Vanetmobisim: generating realistic mobility patterns for vanets. In W. Holfelder, D. B. Johnson, H. Hartenstein, and V. Bahl, editors, *Vehicular Ad Hoc Networks*, pages 96–97. ACM, 2006.

[5] F. K. Karnadi, Z. H. Mo, and K. Lan. Rapid generation of realistic mobility models for vanet. *Wireless Communications and Networking Conference, 2007.WCNC 2007. IEEE*, pages 2506–2511, March 2007.

[6] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek. The click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, August 2000.

[7] D. Krajzewicz, M. Bonert, and P. Wagner. The open source traffic simulation package sumo. 2006.

[8] S.-B. Lee, G. Pan, J.-S. Park, M. Gerla, and S. Lu. Secure incentives for commercial ad dissemination in vehicular networks. In *MobiHoc '07: Proceedings of the 8th ACM international symposium on Mobile ad hoc networking and computing*, pages 150–159, New York, NY, USA, 2007. ACM Press.

[9] C. Lochert, M. Caliskan, B. Scheuermann, A. Barthels, A. Cervantes, and M. Mauve. Multiple Simulator Interlinking Environment for Inter Vehicle Communication. In *VANET 2005: Proceedings of the Second ACM International Workshop on Vehicular Ad Hoc Networks*, pages 87–88, Sept. 2005.

[10] W. Louwerse and S. Hoogendoorn. Adas safety impacts on rural and urban highways. *Intelligent Vehicles Symposium*, pages 887–890, 2004.

[11] R. Mangharam, D. Weller, R. Rajkumar, P. Mudalige, and F. Bai. Groovenet: A hybrid simulator for vehicle-to-vehicle networks. *Mobile and Ubiquitous Systems: Networking & Services, 2006 Third Annual International Conference on*, pages 1–8, July 2006.

[12] E. Miller, Jeffrey; Horowitz. Freesim – a free real-time freeway traffic simulator. *Intelligent Transportation Systems Conference, 2007. ITSC 2007. IEEE*, pages 18–23, Sept. 30 2007-Oct. 3 2007.

[13] V. Naumov, R. Baumann, and T. R. Gross. An evaluation of inter-vehicle ad hoc networks based on realistic vehicular traces. In Palazzo et al. [14], pages 108–119.

[14] S. Palazzo, M. Conti, and R. Sivakumar, editors. *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2006, Florence, Italy, May 22-25, 2006.* ACM, 2006.

[15] M. Piorkowski, M. Raya, A. Lugo, P. Papadimitratos, M. Grossglauser, and J.-P. Hubaux. TraNS: Realistic Joint Traffic and Network Simulator for VANETs. *ACM MC2R.*

[16] A. Varga. The omnet++ discrete event simulation system. In *Proceedings of the European Simulation Multiconference*, pages 319–324, Prague, Czech Republic, June 2001. SCS – European Publishing House.

[17] A. Wegener, M. Piorkowski, M. Raya, H. Hellbrück, S. Fischer, and J.-P. Hubaux. TraCI: An Interface for Coupling Road Traffic and Network Simulators.