Multiple Simulator Interlinking Environment for IVC

Christian Lochert, Björn Scheuermann,
Andreas Barthels, Alfonso Cervantes,
and Martin Mauve
Heinrich-Heine University Düsseldorf
Düsseldorf, Germany
{lochert, scheuermann,
mauve}@cs.uni-duesseldorf.de

Murat Caliskan
Volkswagen AG
Wolfsburg, Germany
murat.caliskan@volkswagen.de

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: Experimentation, Performance

Keywords: simulator coupling, vanet

1. INTRODUCTION

The development and evaluation of applications for car-to-car communication plays an increasingly important role, both in scientific and industrial research [1, 2, 5, 7, 8]. These applications aim at enhancing traffic safety, driving comfort, and in-car entertainment. The common development cycle for these applications includes a detailed simulation step preceding the actual implementation in order to reveal hidden challenges or design flaws.

Commonly, a set of independent simulators is used to conduct these studies. In order to capture the specific characteristics of a VANET the nodes have to move in a realistic manner. The movement patterns are created by vehicular traffic simulators and are stored in traffic files. A network simulator is then used to model the algorithms employed for routing data packets through the VANET. It uses the traffic files generated by the traffic simulator to move the nodes during the simulation. Finally the application data is often approximated by some artificial stochastic or deterministic processes within the network simulator.

One key disadvantage of this approach is the inability to modify the traffic data in response to application layer events. I.e., no matter what happens at the application layer, the movement of the vehicles will remain the same. There is no way for an application to change the behavior of a vehicle during runtime. This reduces the level of realism that can be achieved for key applications like active safety which in reality influence the node's movement significantly.

To solve this problem we propose to interlink different simulators for network simulation, traffic simulation and application simulation to establish an integrated simulation environment that goes beyond existing simulator couplings like [3] or simple downstream simulators like [10]. Thus a holistic evaluation of VANET applications can be done. Two main concerns have to be dealt with: first, how to communicate between the simulators without encountering major performance losses and second even more critical, how to ensure (simulation-) time synchronization between these simulators.

VANET'05, September 2, 2005, Cologne, Germany.
ACM 1-59593-141-4/05/0009.

2. ARCHITECTURE

The aim of our architecture is to enable an holistic evaluation of VANETs with the ability to tune each 'adjusting screw' of all involved simulators. As depicted in Figure 1 the architecture comprises one simulator each for vehicular movements (VISSIM), application behavior (Matlab/Simulink), and network functionality (ns-2). A helper application (Simulation Control) is present to enable cross operating system interaction between simulators.

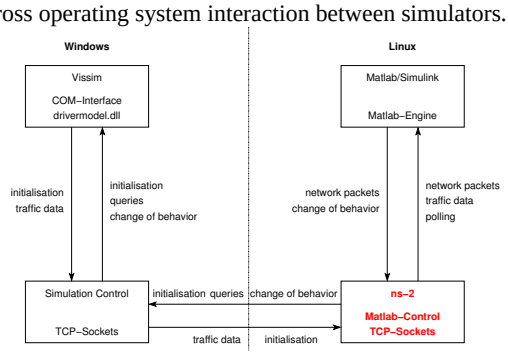


Figure 1: Representation of the simulation environment.

2.1 The Network Simulator – Central Module

In the research community multiple distinct network simulators are employed. We decided to use the freely available and widely used open-source simulator ns-2 [6].

To enable simulator interlinking, a new central class called *Synchronize* was implemented for ns-2. This class is responsible for the synchronization between all interlinked simulators. At the startup of the simulation a Tcl-object of this class is created and the connection to the traffic simulator is established. After this initialization phase the network simulator itself is instanced. Within the *Synchronize*-class a timer is set. After its expiration a request for new movement data is sent to the traffic simulator. This process is periodically repeated.

2.2 The Traffic Simulator

For the simulation of traffic data we used the simulator VISSIM [9]. This simulator is able to do microscopic simulations on the basis of realistic driver behavior. It offers two main interfaces: *i)* the Component Object Model (COM) interface and *ii)* the customizable dynamic link library (DLL) called *drivermodel.dll*. VISSIM requires a MS Windows platform.

The COM interface allows full control about most aspects of a VISSIM simulation, including the modification of attributes such as speed and acceleration of a vehicle. Since the access to this in-

terface is very heavy-weighted we used it only for those aspects that cannot be controlled via other means. This includes starting and halting the simulation process. In contrast to the COM interface the dynamic link library offers a very light-weighted access to the VISSIM simulator. This interface is originally intended to allow the customization of the driver behavior during runtime. In particular it can be used to change the behavior of the driver in response to an application-level event. Furthermore, we employ the *drivermodel.dll* interface to access data such as the positions of the vehicles in a lightweight and highly performant fashion.

Due to the fact that VISSIM requires MS Windows while the other simulators perform optimally under Unix operating systems, cross operating system communication was necessary. This task is performed by an external application (*Simulation Control*).

2.3 The Application Simulator

We decided to use the Matlab/Simulink [4] environment as an application level simulator. Simulink allows a user to create applications by providing personally adaptable drag and drop utilities. Furthermore it is possible to automatically generate C-code out of the Simulink simulation. This is regularly done to use the same code-base for simulation and real-world products.

The Matlab environment is used to run the simulations. This environment offers the possibility to operate by remote control through the Matlab engine. This engine includes a variety of library functions such as starting and quitting Matlab, exchanging data with Matlab, or requesting mathematical operations. In our environment the network simulator uses these functions to communicate with Matlab/Simulink during the simulation's runtime of a simulation.

There is one major performance issue with interlinking the network and the application simulator: handing data back and forth between the simulators is costly. One transfer from the network simulator to Matlab takes approximately 20 ms. It is therefore very important to aggregate all packets for all vehicles that need to be delivered to the application simulator within a given period of time. These packets are de-aggregated by the application simulator and are then forwarded to the simulation of the individual vehicles. From our experience it is not feasible to let one separate simulation register with the network simulator for each vehicle.

3. SIMULATIONS

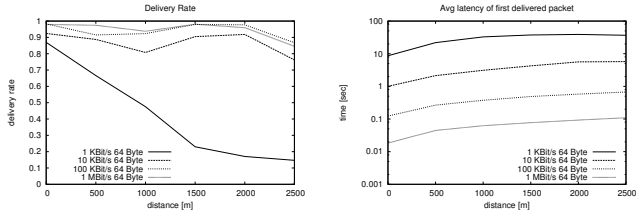
The interlinked network (ns-2.27) and traffic (VISSIM-4.0) simulators were used to investigate the transmission of emergency warnings in vehicular ad-hoc networks. Besides demonstrating the viability of interlinking the simulators, the key aim of the simulation study was to understand the impact of the single hop bandwidth on reliability and latency. The simulation area was part of a demo city. Obstacle modeling was used to ensure that only vehicles that are in direct line of sight could communicate.

For each simulation run one vehicle near the center of the simulation area was selected to simulate an accident. This triggered the stopping of the vehicle in the traffic simulator and the transmission of a 64-byte emergency warning message by this vehicle. The emergency warning was then forwarded by using simple flooding. Each vehicle that received the message forwarded it once and then decelerated using a regular braking pattern. We were interested in how reliable and how fast the emergency message would be delivered depending on the available single hop bandwidth and the distance to the accident.

For the experiments the average delivery ratio and delay of the warning messages are investigated. Vehicles are grouped according to their distance to the original sender of the emergency warning at the time this warning message is transmitted by the original sender. The first group contains all vehicles that are within a 500 m radius, the second includes all vehicles that are not in the first group but are located within a 1000 m radius and so on. This value is displayed as distance on the x-axis of Figure 2(a) and Figure 2(b).

The y-axis of Figure 2(a) shows the ratio between the number of nodes that have received a warning packet and the number of nodes that belonged to the respective group. From a reliability perspective it can be observed that a bandwidth of 10 KBit is the absolute minimum value to achieve a reliability of around 0.9 for any significant distances. It should be noted that this is influenced significantly by the topology of the radio obstacles. This seems also to be the reason why the group at 1500 m and 2000 m has a higher delivery ratio than the group at 1000 m: in the topology of our city model distances between 1500 m and 2500 m required the traversal of a long street which is unlikely to fail.

Further conclusions can be drawn by studying the average latency of the delivered warning packets (see Figure 2(b)). This latency increases dramatically as the available bandwidth decreases. To transmit a warning message early enough to warn another driver it seems likely that at least a bandwidth of 100 KBit is needed. Here the duration to transmit a packet grows from 0.1 s in a distance to 500 m up to almost 1 s in a communication range of 2500 m.



(a) Packet delivery ratio versus distance of communication partners.

(b) Avg. latency of delivered message vs. distance of comm. partners.

Figure 2: Simulation Results

4. FUTURE WORK

One focus of our work will continue to be on the simulation of vehicular emergency warning systems. In a next step we will investigate complex environments including full application level functionality, e.g., only those vehicles that are likely to be affected by the accident will break. We will then take into account that not all vehicles need to receive the warning message to increase traffic safety: if a vehicle ahead of another vehicle breaks then the other vehicle will be stopped through normal driver behavior. It is thus likely that a high reliability for the delivery of the warning message is only required very close to the accident. Also, we will use advanced flooding strategies to reduce the required bandwidth and the time until the affected vehicles will break.

5. REFERENCES

[1] The DSRC project. http://www.leearmstrong.org/DSRC/DSRCHomeset.htm.

[2] The FleetNet project. http://www.fleetnet.de.

[3] U. Hatnik. et al. Using ModelSim, Matlab/Simulink and NS for Simulation of Distributed Systems. In *PARELEC '04*, pages 114–119, Washington, DC, USA, 2004.

[4] The Matlab/Simulink application simulator. http://www.mathworks.com/products/simulink/.

[5] The Network-on-Wheels project. http://www.network-on-wheels.de.

[6] The ns-2 network simulator. http://www.isi.edu/nsnam/ns/.

[7] The PATH project. http://www.path.berkeley.edu.

[8] The PReVENT project. http://www.prevent-ip.org.

[9] The VISSIM traffic simulator. http://www.ptv.de/cgi-bin/traffic/traf_vissim.pl.

[10] Q. Xu. et al. Vehicle-to-vehicle safety messaging in DSRC. In *VANET '04*, pages 19–28, Philadelphia, PA, USA, 2004.