Observations on OMNeT++ Real-Time Behaviour

Christina Obermaier, Christian Facchi
Research Centre, Technische Hochschule Ingolstadt
Email: {christina.obermaier, christian.facchi}@thi.de

***Abstract*—*OMNeT++* is a widely used platform for all types of network simulations. The open source simulation framework *Artery* can be used to perform Vehicular Ad Hoc Network (VANET) simulations. This paper presents an approach for connecting this simulation and real-world VANET hardware to extend the test range and investigates the real-time behaviour of the simulation. As a Device Under Test (DUT) depends on real-time data to perform properly, different simulation scenarios running different hardware setups are presented. Additionally, the paper deals with the impacts of real-time losses on the test run outcomes. Most time dependant algorithms like the duplicate packet detection do not need very accurate real-time data and thus could be verified using the presented approach. Otherwise, in some cases such as testing of multi-hop communication, accurate real time is crucial.**

***Index Terms*—Vehicular Ad Hoc Network, Simulation, Hardware in the Loop**

I. Introduction

In times of increasing complexity of advanced driver assistance systems, it is crucial to enhance the environmental awareness of vehicles. Vehicles can be equipped with Vehicular Ad Hoc Network (VANET) devices, acting as a new information source besides already well known sensors.

VANETs are spontaneously created networks between road participants and Road Side Units (RSUs). They are based on IEEE 802.11p [1] and ETSI ITS G5 [2] in Europe as well as IEEE WAVE in the USA [3]. This paper focuses on the European ETSI ITS G5 standards.

VANETs and their applications were developed with the focus on enhancing traffic safety and traffic flow [4]. Especially in critical driving situations, availability of information is crucial. This leads to the question, how to test VANET communication properly. As Software in the Loop (SIL) simulation is not enough to ensure the availability, this paper evaluates if *Artery* can fulfil the real-time requirements necessary for Hardware in the Loop (HIL) testing.

In section II a overview of the related testing frameworks and hardware testbeds is given. Section III presents the state of the art and the theoretic concept of the hardware testbed. In section IV different scenarios running on different hardware setups are investigated. Section V presents the limitations of the presented approach. Finally, section VI includes a conclusion and a brief outlook.

II. Related Work

Currently there are mainly two different Inter-Vehicular-Communication (IVC) testing approaches. On the one hand, there is pure software testing with frameworks like *Artery*[1] [5] and *Veins*[2] [6] which are based on the discrete event simulator *OMNeT++*. These implement the Intelligent Transport System (ITS) G5 and the Wireless Access in Vehicular Environments (WAVE) standards, respectively. Also, different environmental circumstances and accidents can be modelled easily [7].

On the other hand, there are real-world testing approaches like the "Testfeld A9" established near Ingolstadt in Germany. Real-world testing allows for testing actual IVC hardware, but it is required to have at least two drivers in real vehicles equipped with IVC hardware. Thus, field tests are hard to reproduce and very expensive. While it is still possible to perform simple real-world scenarios, like presented in [8], it might be not feasible to do this with more complicated scenarios.

HIL tests can be used to perform hardware tests which are easy to repeat and cost effective. It is not a new approach using *OMNeT++* to connect a simulation with real-world hardware as *OMNeT++* was one of the fastest simulators in the domain of wireless networks in prior software versions [9]. In [10], a routing framework for *OMNeT++* HIL simulations is presented and the real-time behaviour of *OMNeT++* was investigated in different scenarios. They mentioned that real time will be a problem in future scenarios, especially if the node topology begins to change dynamically due to node mobility. According to [11], *OMNeT++* can also be coupled with *RoSeNet* to connect with real-hardware sensors. In [12] the performance of the *INET* framework (release 2.5.1) in emulation mode combined with an enhanced real-time scheduler was investigated. They pointed out to have performance problems. Additionally, they observed packet losses occurring in the communication between the real hardware and the simulation.

III. HiL concept

Artery is an open source framework for the discrete event simulation *OMNeT++* [5]. It allows for simulating European VANETs using *Vanetza*[3], which is an open source implementation of the ETSI ITS G5 communication stack [13]. *Veins* or *INET* provide the physical and Medium Access Control (MAC) layers. Moreover, the movement of the vehicles is simulated by the open source traffic simulator *SUMO*[4]. *SUMO* and *Artery* are coupled using the Traffic Command Interface (TraCI) [14].

[1] https://github.com/riebl/artery
[2] http://veins.car2x.org
[3] http://vanetza.org/
[4] http://sumo.dlr.de/

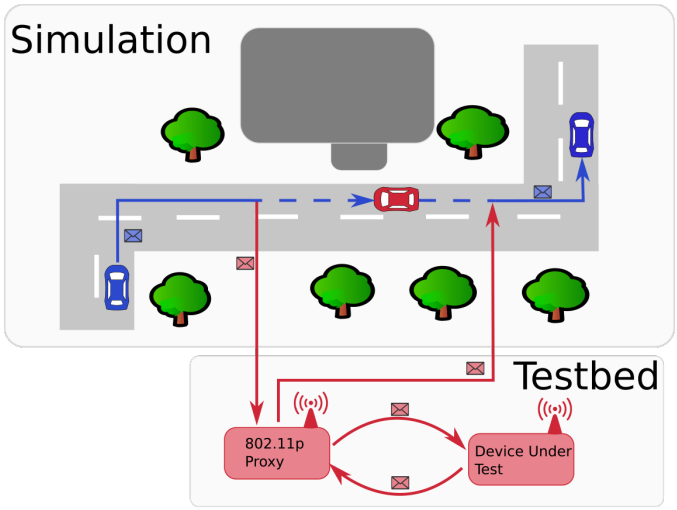


Figure 1. HiL concept overview

A. Artery Overview

Basically, each vehicle controlled by *SUMO* is represented by an *OMNeT++* compound-module called *Car*. A *Car* module is, among others, composed of a *Middleware*, a *VanetNic* and a *Mobility* submodule. The *Mobility* module is responsible for all vehicle dynamics related data and information. The *VanetNic* represents the network interface of each car. The *Middleware* module hosts all registered applications and contains an instance of *Vanetza*, taking care of routing and transport the incoming and outgoing IVC packages. Currently, the day one applications Cooperative Awareness Message (CAM) and Decentralized Environmental Notification Message (DENM) are provided by *Artery* [5], [15].

Cooperative Awareness (CA) is responsible to inform all road users in the network about basic data of all other participating vehicles. This basic data contains, among others, the network node's position, speed and heading. CAMs are generated up to ten times per second from each road participant [16]. In contrast, DENMs are only generated in case of a specific event. A DENM contains information of hazards and dangerous situtations occurring in road transport [17]. Thus, *Artery* allows for defining this dangerous situations and environmental influences like, for example, heavy rain or nearly accidents [7]. These situations are required to trigger related DENM messages.

B. HiL concept

Figure 1 shows the basic concept of a HIL simulation using *Artery* to provide test data. The red coloured vehicle is the simulated representative of a Device Under Test (DUT). For now, this car will further be called physical twin. If a message is going to be transmitted to the physical twin, the message will be transferred to a 802.11p gateway realised on software defined radio technology. This behaviour is represented by red arrows, showing the connection of the simulation to the real word and thus with the DUT. The blue dotted arrows show the current behaviour of the simulation.

Table I
HARDWARE SETUP

Component	Laptop Computer	Simulation Cluster
CPU	Intel Core i5-6300U @ 2.40GHz	Intel Xeon E7-8867 v4 @ 2.40GHz
Cores	1 x 4	4 x 18
RAM	16GB	3TB
Hard Drive	256GB SSD	450GB SAS SSD RAID 1

In contrast to other simulated cars, the physical twin has a quite limited functionality: Upper protocol layer processing is done by the DUT so the *Middleware* module as well as the *Vanetza* instance can be dropped. Otherwise, the *Mobility* module is still needed to provide Global Positioning System (GPS) data to the DUT to ensure properly working routing protocols [18]. Without appropriate GPS information, the geographic routing protocols defined in [18] would not work properly. Summarising, all functionality beginning with the MAC layer processing is stripped from the physical twin. It is only responsible for calculating message receptions and feeding received messages back on the *OMNeT++* channel.

As each kind of hardware testbed is dependent on real-time execution, the simulated environment must run in real time, too. To ensure this, the *OMNeT++* built-in scheduler is exchanged by a real-time scheduler. This scheduler is built upon *Boost ASIO* timers, ensuring asynchronous waiting. Hence, the real-time scheduler slows down the simulation, if it could run faster than real time. Additionally, it is aware of real-time losses so that it could stop the simulation if data could not be provided in real time. Also, the scheduler provides logging mechanisms to investigate the real-time behaviour of simulation runs after they are finished. A pseudocode implementation of the scheduler can be found in Appendix A.

IV. Investigations on Real-Time Behaviour

As already known from investigations described in [19] there is a quite low execution speed of *OMNeT++* VANET simulations. This leads to the question, if *Artery* is capable of reaching and holding real time to provide data for a HIL testbed while doing an online simulation.

A. Scenario Description

The chosen scenario is very simple: Three vehicles driving on a highway from north to south.

As the amount of driving vehicles seems to be the main influence on the execution speed of these simulations, the second test scenario increases the amount of driving vehicles by two. All other parameters remain the same as in the previous scenario. The setup of the radio medium is configured by *Artery* using its *INET* defaults. The in Section III-B introduced scheduler is used to evaluate the real-time behaviour. The simulation was built using *OMNeT++* version 5.1.1 and was executed on two different computers: A laptop computer and a simulation cluster. Table I presents a comrehensive hardware list of the used computers. As *OMNeT++* uses only one core

Table II
EVENT MAPPING

ID	Event name	# Events "3 vehicles"	# Events "5 vehicles"
1	TraCI Connect	1	1
2	TraCI Step	322	370
3	GeoNet packet	3870	11298
4	GeoNet data frame	3870	11298
5	txStart-0	3	5
6	endIFS	661	1189
7	configureRadioMode	1322	2378
8	transmissionTimer	661	1189
9	remove non Interfering Transmission	661	1188
10	report CL	928	1650
11	middleware update	925	1645
12	txStart-1	658	1184
13	GeoNet radio frame	1274	4460
14	reception Timer	1274	4460
	Overall events	16430	42315

in this test setup, the speed-up of the simulation running on the simulation cluster is caused by faster calculation hardware not by better parallelism.

B. Simulation Results and Scenario Comparison

Table II presents the number of events occurring in both scenarios. The ID column is used to map event IDs to the events depicted in the box plots in figure 2a and 3a. The third and the fourth columns show the number of occurring events dependent on the simulated scenario. Thus, compared to the base scenario which triggers 16430 events, the scenario with two more vehicles triggers 42315 events. Hence, adding two cars causes 2.58 times more events in this scenario setup.

Figure 2 and 3 present the evaluations of both scenarios executed on the simulation cluster and the laptop computer. The included boxplots depict the execution time of events. The indices on the x-axis correspond to the events mentioned in Table II. If we compare both figures, the time a event needs to be executed is lower while using the simulation cluster but relative event durations remain nearly the same. This behaviour is caused by the faster computing capabilities of the single cores of the simulation cluster.

The histograms 3b and 3c show the real-time misses per event while the scenarios were executed on the laptop. Histogram 2b and 2c present the same but for the execution on the simulation cluster. As the green bars indicate events executed nearly in real time, higher green bars and lower red bars indicate a nearly real-time capable simulation run.

It can be seen that there are real-time drops up to 1.5 seconds in each scenario. This is caused by the *TraCi Connect* event, which is executed as first event and takes about one second. Thus, even if the scenario itself is real-time capable, there will be always a few seconds at the beginning of the simulation which must be skipped because of the system startup. Figure 2b is a great example for a good and fast running scenario. There are only roughly 50 - 100 events which are more than one second behind the real time. About 15000 events out of the total amount of 16430 events are executed nearly in real time. This means, the simulation cluster can basically handle a small scenario with three vehicles in real time.

Figure 3b presents the same scenario executed on a laptop computer. It can be seen that there are still about 10000 events executed nearly in real time. Other 5000 events are executed only 0.1 second behind real time. Thus, only a few more events can be found in the area of about one second behind real time, but the scenario looks still quite good.

Distinct differences between execution speed of the computing hardware can be seen in Figure 2c and Figure 3c. The simulation cluster is able to handle the five car scenario with still up to 1.5 seconds real-time loss. The amount of events in the area from 1 to 1.5 seconds is significantly increased, compared to the base scenario but *OMNeT++* is still able to catch up. However, the five car scenario executed on the laptop computer exceeds the 1.5 seconds limit by far. There are many real-time losses up to three seconds. Thus, the five car scenario can only be handled by the simulation cluster.

Figure 4 admits a closer look on the real-time behaviour of the different simulation runs. Blue bars show the time needed to process all events occurring at a particular simulation time stamp. The red line depict the current gap between simulation time and real time. Thus, a higher blue bar causes a higher real-time loss peak. It seems that *Artery* tends to schedule a bunch of events every 50ms. This behaviour causes the simulation to fall behind real time every 50ms, which is indicated by the peaks of the red line.

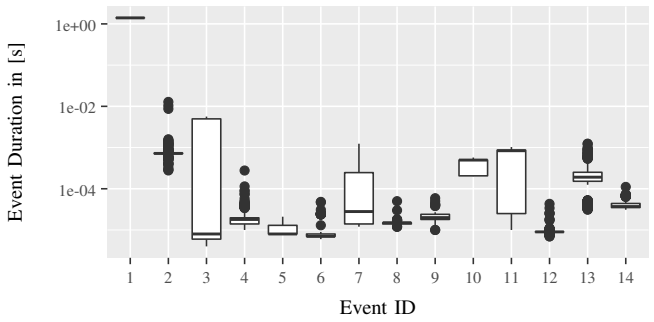
As it can be seen, the simulation cluster produces as many real-time losses as the laptop computer. But, even if the differences between the execution time of single events are not that clear, the real-time losses are much higher on the laptop computer, independent from the executed scenario. This approves that *Artery* has to execute many events at the same time because they are scheduled at nearly the same timestamp. Due to the lack of parallelism, *OMNeT++* has to execute this events in a sequence, which causes the real-time losses.

As there are many events which are closely linked together, for example the *GeoNet packet* and the *GeoNet data frame*, this behaviour could not be changed significantly. Thus, there will always be real-time losses caused by many events occurring at the same time if there is no parallelism.

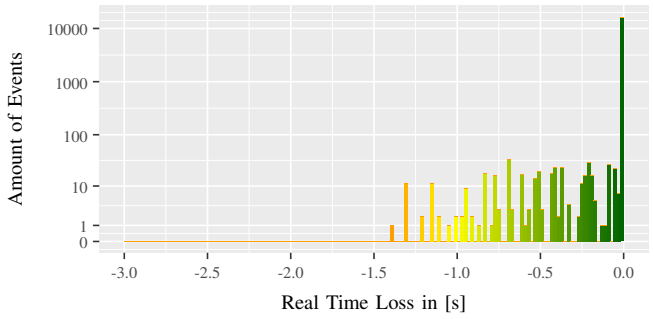
C. OMNeT++ Time Flow

Figure 5 depicts a generic example how the timeline in *OMNeT++* behaves compared to the real-time flow.

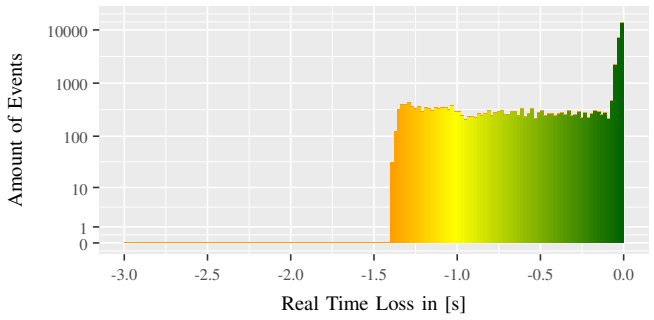
As it can be seen, the absolute time needed to execute one simulation millisecond is sometimes higher than one real millisecond like indicated at millisecond two. If the simulation is behind the wall-clock time, it has to catch up to avoid higher real-time gaps. Hence, the simulation is trying to execute



(a) Average event times

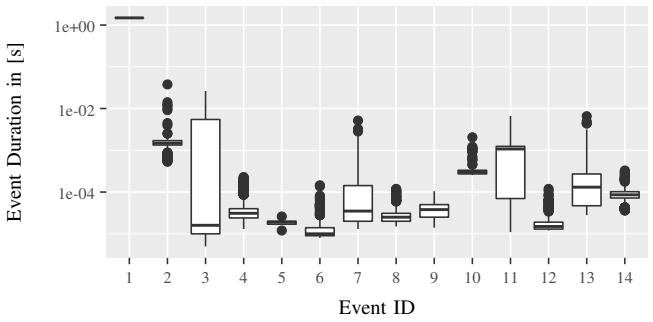


(b) Amount of critical real-time losses in base scenario

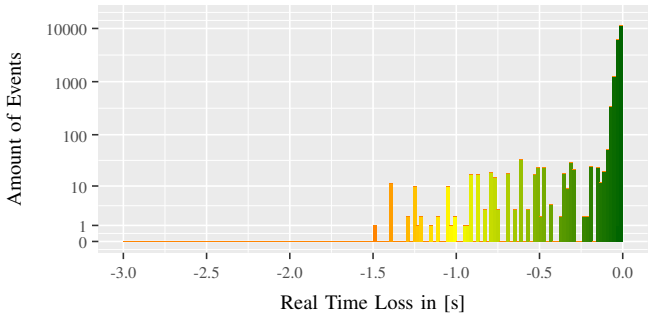


(c) Amount of critical real-time losses in five cars scenario

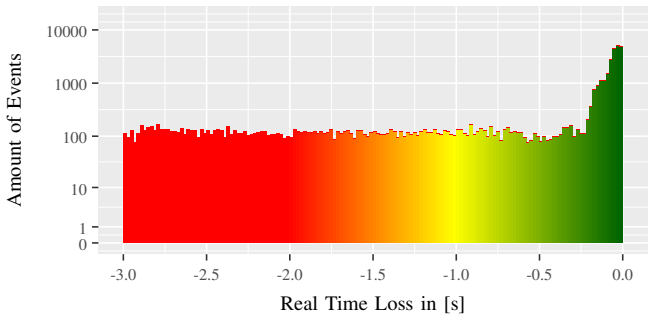
Figure 2. Scenarios executed on the simulation cluster



(a) Average event times



(b) Amount of critical real-time losses in base scenario



(c) Amount of critical real-time losses in five cars scenario

Figure 3. Scenario executed on the laptop computer

events faster than in real-time till simulation time and wall-clock time are matching again. But, even if the *OMNeT++* clock sometimes progresses faster than real time, the fixed time stamp is always behind or equal to the wall-clock time which is ensured by the real-time scheduler. If the next event is located in the future, the scheduler waits till the timestamp of the event matches the wall-clock time. Thus, it is ensured that the DUT must only deal with timestamps in the past and not in the future, which would be much more problematic.

V. Simulation Result Interpretation

Most scenarios in IVC communication do not depend on very accurate timestamps. For example, the Long Position Vector (LPV) [18, Section 8.8] contains the time at which the last geographic position update was received by the vehicle writing the LPV. As in a real-world environment GPS update rates are fluctuating, the timestamp in the LPV is not strongly related to the current wall-clock time.

The LPV timestamp is used to perform duplicate packet detection at the network layer [18, p. 63]. The duplicate packet detection algorithms depend on the sequence number as well as the timestamp. A packet is identified to be a duplicate if the timestamp is lower or equal to an already received packet. As the introduced *RealTimeScheduler* ensures that the simulation time is always behind or equal to the wall-clock time, this algorithm works like expected.

CAMs contain a generation timestamp [16] used, for example, to recognise and avoid replay attacks [2, Section 7.6.1]. However, a CAM is not detected to be a duplicate if there is only a slight difference between wall clock time and the generation timestamp.

A DENM contains various timestamps as well. There is, for example, an expiry time after which a DENM event is terminated. Mostly, this period lasts several seconds but in cases like the dangerous situation [20] trigger, a DENM event lasts only two seconds. Thus, in some situations a real-time

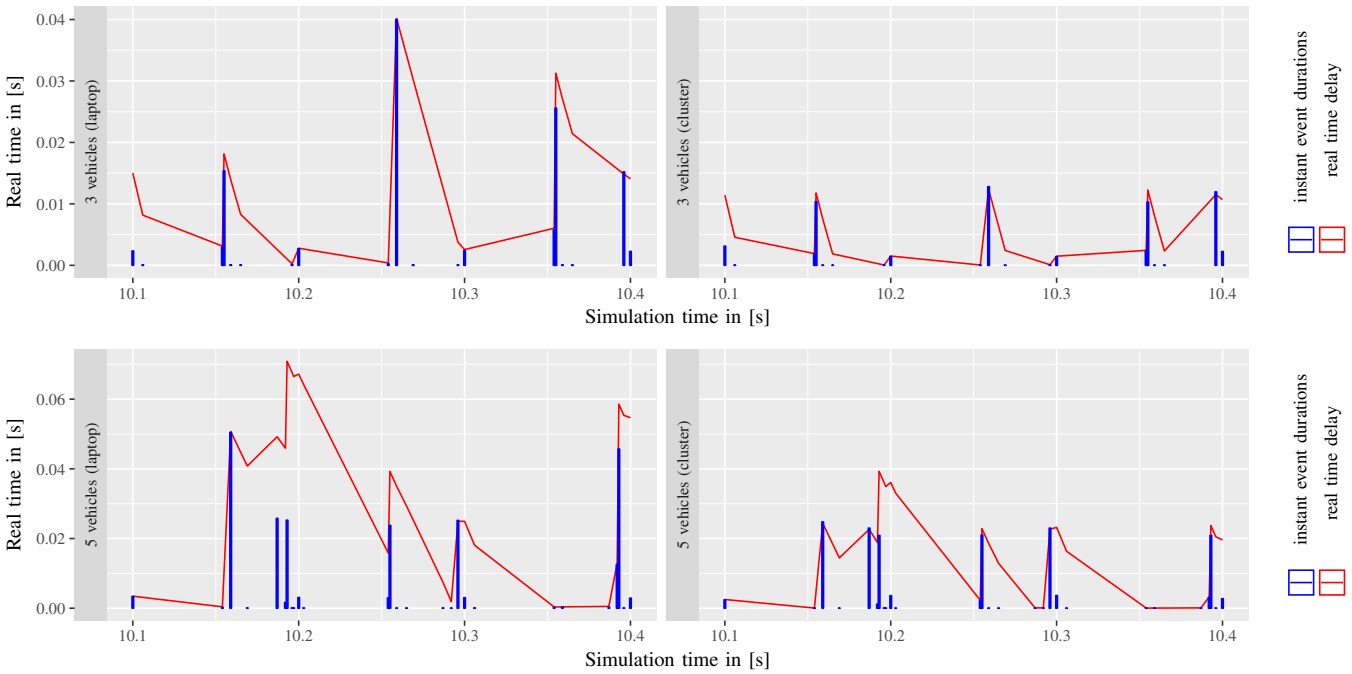


Figure 4. Real-time delays and event durations

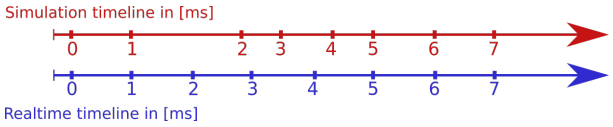


Figure 5. OMNeT++ and real-time timeline

loss higher than one ore two seconds may invalidate a DENM wrongly.

Another issue is the long-range communication using multiple vehicles as forwarders [18]. A MAC layer unicast is followed by an Acknowledgement Frame (ACK), sent by the receiver, to confirm a successful packet transmission. The time span in which the ACK is expected by the initial sender is the time of a Short Interframe Space (SIFS) [1, Section 9.3.2.8]. Determined by the channel bandwith of 10 MHz [21] the SIFS is 32 μs [1, Table 18-17]. This very short time span leads to a potential problem: If a MAC layer unicast is sent to the DUT, it has to respond with an ACK within this SIFS. As depicted in Figure 5 between millisecond two and three, *OMNeT++* could run faster than real time. Thus, the sending vehicle inside *OMNeT++* may not receive this message in the claimed timespan. Hence, this problem must be solved to perform multi-hop tests.

Summarizing, in most cases, a real-time loss in a range of a few milliseconds is not that problematic. Important algorithms like the duplicate packet detection do only rely on linear time flow and do not depend on hard real time. Only a few scenarios like multi-hop testing relay on a very accurate time synchronisation. Thus, *Artery* is basically able to provide online simulation data for a real-time testbed.

VI. Conclusion

This paper presented an approach for extending *Artery* to provide hardware tests. *Artery* is used to facilitate SIL tests in the area of VANETs. As a HIL simulation always depends on real-time data, the online simulated scenarios have to be executed in this manner. It was investigated that this criteria can only be fulfilled if only a few cars are simulated. Also, even if the simulation is overall real-time capable, there are always real-time losses. How significant these losses are depends on the used computation hardware and the scenario complexity. As *OMNeT++* only uses one core when simulating VANETs, a higher single core performance causes a higher execution speed. Also, *Artery* and other wireless communication models tends to produce events to be executed nearly at the same time, causing more significant real-time losses. This is related the fact that one sending event triggers various receiving events.

Also, it was audited in which situations nearly hard real time is required and when real-time drops are bearable: Multi-hop communication strongly depends on real time data, so its not possible to test this feature properly in the current state of the simulation. Most other algorithms do only depend on a steady time flow and are not influenced by real-time losses in the range of a few milliseconds. This is the case for DENM message expiries or the recognition of replay attacks. Also the duplicate packet detection is not affected harmfully.

In conclusion, *Artery* could basically be used to provide data for VANET HIL tests. In case of multi-hop-test scenarios, real-time losses may influence the test results heavily. Other scenarios can be used to provide functional testing of VANET hardware. This scenarios are currently limited to three or four

vehicles simulated vehicles depending on the used hardware.

As this paper presents a work in progress research project, the HIL testbed will be created with the observed behaviour of *OMNeT++* in mind. Thus, to enable multi-hop testing, which is crucial in European VANETs, one idea is to use a Software Defined Radio (SDR) as 802.11p proxy. This allows for a modified MAC layer of the used proxy device to send ACK frames depending on the current state of the simulation. Hence, the proxy device must know all vehicles which can communicate with the DUT in the current simulation step. This idea could provide a basic implementation of a testbed which can handle most functionality of ETSI ITS G5 networks with the constraint that the simulation must run in nearly real time.

So, further work must be done in the field of enhancing simulation speed to achieve faster running simulation scenarios. Moreover, other ways to provide test data for hardware tests can be investigated. This includes, among others, the capturing of the simulated network traffic and playing them back in real time.

Appendix A
Real Time Scheduler

Result: next cEvent
currentRealTimeMiss = *simTime* - *wallClockTime*;
if *(currentRealTimeMiss * -1) > realTimeMissThreshold* **then**
    `// simulation unacceptable slow`
    stop simulation;
else
    eventDuration = *wallClockTime* - *eventStartTime*;
    log *currentRealTimeMiss* and *eventDuration* and *nextEventIdentifier*;
    while *SimTime > wallClockTime* **do**
        `// simulation faster than real time`
        wait;
    end
    set *nextEventIdentifier*;
    set *eventStartTime*;
    return *nextEvent*;
end

Algorithm 1: *cEvent* RealTimeScheduler::takeNextEvent* pseudocode

References

[1] *IEEE 802.11 Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, Institute of Electrical and Electronics Engineers, Mar. 2012.

[2] *EN 302 665 Intelligent Transport Systems (ITS); Communications Architecture*, European Telecommunications Standards Institute, Sep. 2010.

[3] *IEEE 1609.0-2013 IEEE Guide for Wireless Access in Vehicular Environments (WAVE) - Architecture*, 2014.

[4] T. L. Willke, P. Tientrakool, and N. F. Maxemchuk, "A survey of inter-vehicle communication protocols and their applications", *IEEE Communications Surveys Tutorials*, vol. 11, no. 2, pp. 3–20, Second 2009.

[5] R. Riebl, H. J. Günther, C. Facchi, and L. Wolf, "Artery: Extending Veins for VANET applications", in *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, Jun. 2015, pp. 450–456.

[6] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis", *IEEE Transactions on Mobile Computing*, vol. 10, no. 1, pp. 3–15, Jan. 2011.

[7] C. Obermaier, R. Riebl, and C. Facchi, "Dynamic Scenario Control for VANET Simulations", in *Proceedings of IEEE MT-ITS 2017*, Unpublished, 2017.

[8] K. C. Lee, S. h. Lee, R. Cheung, U. Lee, and M. Gerla, "First Experience with CarTorrent in a Real Vehicular Ad Hoc Network Testbed", in *2007 Mobile Networking for Vehicular Environments*, May 2007, pp. 109–114.

[9] A. R. Khan, S. M. Bilal, and M. Othman, "A performance comparison of open source network simulators for wireless networks", in *2012 IEEE International Conference on Control System, Computing and Engineering*, Nov. 2012, pp. 34–38.

[10] G. Nirav and K. M. Sivalingam, "Dynamic routing framework for OMNeT++ based Hardware-In-The-Loop (HITL) network simulation", in *2014 Twentieth National Conference on Communications (NCC)*, Feb. 2014, pp. 1–6.

[11] S. Boehm and M. Kirsche, "Looking into Hardware-in-the-Loop Coupling of OMNeT++ and RoSeNet", in *Proceedings of the "OMNeT++" Community Summit 2015*, 2015.

[12] A. A. Scussel, G. Panholzer, C. Brandauer, and F. von Tüllenburg, "Improvements in OMNeT++/INET Real-Time Scheduler for Emulation Mode", in *Proceedings of the OMNeT++ Community Summit 2015*, 2015.

[13] R. Riebl, C. Obermaier, S. Neumeier, and C. Facchi, "Vanetza: Boosting Research on Inter-Vehicle Communication", in *Proceedings of the 5th GI/ITG KuVS Fachgespräch Inter-Vehicle Communication (FG-IVC 2017)*, Apr. 2017, pp. 37–40.

[14] A. Wegener, M. Piórkowski, M. Raya, *et al.*, "TraCI", in *Proceedings of the 11th communications and networking simulation symposium on - CNS '08*, ACM Press, 2008.

[15] R. Riebl. (2017). Artery, [Online]. Available: https://github.com/riebl/artery (visited on 05/13/2017).

[16] *EN 302 637-2 Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*, European Telecommunications Standards Institute, Nov. 2014.

[17] *Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service*, European Telecommunications Standards Institute, Nov. 2014.

[18] *EN 302 636-4-1 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*, European Telecommunications Standards Institute, Jul. 2014.

[19] R. Riebl, S. Neumeier, and C. Facchi, "Inter-Vehicle Communication on the Run - Experiences From Tweaking Veins Runtime Performance", Ulmer Informatik-Berichte, 2015.

[20] T. Biehle and K. Krumbiegel, *Triggering Conditions and Data Quality - Dangerous Situation*, Release 1.1.0, CAR 2 CAR Communication Consortium, 2015.

[21] *EN 302 663 Intelligent Transport Systems (ITS); Access layer specification for Intelligent Transport Systems operating in the 5 GHz frequency band*, European Telecommunications Standards Institute, Jul. 2013.