Interactive test tool for interoperable C-ITS development

Alexey Voronov[1], Cristofer Englund[1,2]
Hoai Bengtsson[1], Lei Chen[1]
Viktoria Swedish ICT[1]
Lindholmspiren 3A
SE-417 56 Gothenburg, Sweden
{alexey.voronov, cristofer.englund,
hoai.bengtsson, lei.chen}@viktoria.se

Department of Intelligent Systems[2]
School of Information Technology
Halmstad University, Box 823, 30118, Halmstad, Sweden

Jeroen Ploeg[3], Jan de Jongh[4], Jacco van de Sluis[3]
TNO[3] Integrated Vehicle Safety Department,
5700 AT Helmond, The Netherlands
{jeroen.ploeg, jan.dejongh, jacco.vandesluis}@tno.nl

TNO[4], ICT, Oude Waalsdorpperweg 63,
2697 AK The Hague, The Netherlands

***Abstract*—This paper presents the architecture of an Interactive Test Tool (ITT) for interoperability testing of Cooperative Intelligent Transport Systems (C-ITS). Cooperative systems are developed by different manufacturers at different locations, which makes interoperability testing a tedious task. Up until now, interoperability testing is performed during physical meetings where the C-ITS devices are placed within range of wireless communication, and messages are exchanged. The ITT allows distributed (e.g. over the Internet) interoperability testing starting from the network Transport Layer and all the way up to the Application Layer, e.g. to platooning. ITT clients can be implemented as Hardware-in-the-Loop, thus allowing to combine physical and virtual vehicles. Since the ITT considers each client as a black box, manufacturers can test together without revealing internal implementations to each other.**

The architecture of the ITT allows users to easily switch between physical wireless networking and virtual ITT networking. Therefore, only one implementation of the ITS communication stack is required for both development and testing. This reduces the work overhead and ensures that the stack that is used during the testing is the one deployed in the real world.

I. Introduction

Cooperative systems reinforced by Vehicle-to-Vehicle and Vehicle-to-Infrastructure communication (V2X) are expected to improve safety and throughput and lower the environmental impact in the future traffic system. With the support from proximity sensors, such as cameras and radars, vehicular safety systems have already become more efficient and the future employment of V2X communication will enable the utilization of sensors beyond the line of sight to increase the awareness of the road users. In order to boost the development and realisation of Cooperative Intelligent Transportation Systems (C-ITS), the European Telecommunications Standards Institute (ETSI) and the European Committee for Standardization (CEN) recently released V2X standards [8].

Standardization is one way to harmonize and speed up the development of C-ITS. Plug-tests arranged by the standardization organizations, e.g. ETSI, is another approach to facilitate the development of C-ITS. During these tests, developers gathers to evaluate their communication devices and verify that they are interoperable, i.e. that they can send and receive messages between each other. In this work we highlight another approach that also facilitates the development of C-ITS, namely an Interactive Test Tool (ITT) that enables distributed interoperability testing of C-ITS vehicles, for example, regarding platooning cooperation.

The ITT is developed within the i-GAME project [1], which is financed by the European Commission within the 7th Framework Programme. Within i-GAME, the Interoperable Grand Cooperative Driving Challenge (GCDC) 2016 is arranged. GCDC is a challenge where participants compete in cooperative and automated driving scenarios. The i-GAME project provides interaction protocols and corresponding communication message sets to safely perform the scenarios. The scenarios in GCDC are chosen to reflect the current technical challenges within cooperative and automated driving. The project is accelerating innovation in a multi-vendor environment. Teams from different organizations and countries come together with vehicles of different type and of different brands to collaborate and compete within cooperative and automated driving.

Testing is a large part of the development process and is typically associated with high cost. For the communication equipment used in cooperative systems two main activities characterize the test process. Conformance testing [12] is the first where communication equipment is tested and verified towards the standard. The second is interoperability testing where the equipment is tested towards other implementations [7]. Traditionally, interoperability testing requires physical meetings of several manufacturers [7]. Since the standard can be interpreted in different ways the interoperability testing is performed towards all implementations to assure that all versions are interoperable. This is a costly and time consuming process.

The ITT facilitates the distributed development of cooperative systems by allowing developers to test and evaluate their implementations over the Internet with other development teams. The ITT allows distributed (virtual) interoperability testing of, for example, Network and Application layers, thus making it a valuable resource to facilitate testing and accelerate development of C-ITS. Another unique benefit of the ITT is that it allows testing of the real implementation of the cooperative application where real vehicles, sensors and physical environment can be part of the test.

A. Related architectures

The ITT is a simple domain-specific tool for distributed simulation of C-ITS systems. General-purpose distributed simulation systems and architectures include High-Level Architecture (HLA, [3], [6]) and Distributed Interactive Simulation (DIS, [2]). HLA is a general purpose architecture for distributed computer simulation systems; it is a specification of the interface between client and HLA Run-time Infrastructure. DIS is a network protocol for data exchange for distributed computer simulation systems. DIS Protocol Data Units contain information about, for example, the position and orientation of entities in the world, and describes electronic warfare, logistics, collisions, and simulation management. HLA can be used even for real-time distributed simulations of peer-to-peer communication [10].

Co-simulation is a simulation of a coupled problem, where different subsystems are modelled and simulated in a distributed manner. The coupled simulation is carried out by running the subsystems in a black-box manner. The subsystems exchange data during the simulation. The Functional Mock-up Interface (FMI) for co-simulation is an interface standard for the solution of time-dependent coupled systems consisting of subsystems that are continuous in time or time-discrete. FMI provides interfaces between server (master) and clients (slaves) and addresses both data exchange and algorithmic issues. Similar to FMI, the ITT provides an infrastructure for data exchange, but only within the C-ITS domain. Similar to co-simulation, ITT considers each vehicle as a black-box subsystem. The ITT does not cover the algorithmic issues of simulation but rather the logic connections between different users.

Hardware-in-the-Loop (HiL) simulation is a technique to develop and test real-time systems by allowing the (physical) real-time system (System Under Test, SUT) to interact with the simulation of the so-called *plant*, where the plant is the environment in which the SUT operates. For example, if the hardware SUT is a real-time vehicle controller, then the plant includes the simulated vehicle, the road and other entities that might influence the controller behaviour. A HiL simulation includes electrical emulation of sensors and actuators. These electrical emulations act as the interface between the plant simulation and the physical SUT. The plant simulation emulates sensors that are then read by the SUT. The SUT outputs actuator control signals that change variable values in the plant simulation. The ITT allows HiL simulation of, for example, the communication stack, where networking is running on the dedicated hardware. The ITT also supports HiL simulation of the complete vehicle, for example using a vehicle mounted on a roller bench if the vehicle's sensors are emulated appropriately, where only the wireless communication is simulated. Please note that HiL requires real-time operation of the ITT, see Section III-C for details.

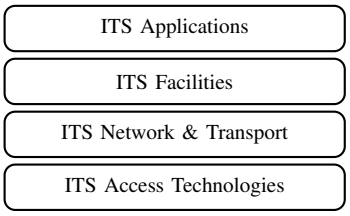


Fig. 1. Reference architecture for C-ITS.

This paper describes the design and architecture of the ITT. The remainder of the paper is organized as follows. Section II describes the general C-ITS architecture, the communication standards and requirements. Section III describes the ITT architecture, and Section IV presents an example of an ITT client. Section V concludes the paper.

II. Cooperative ITS

The communication standard proposed by ETSI [8] summarizes a number of finalized standards namely the C-ITS communication architecture, EN 302 665, the GeoNetworking architecture, TS 102 636-3 and EN 302 636-3, the user and application requirements, TS 102 894-(1-2) series, the C-ITS over public cellular networks, TR 102 962, and the interoperability testing standards, EG 202 798. In [4] an overview of the the C-ITS standard [8] is given along with the concept of EU's cooperative intelligent transport systems and a detailed description of the functional architecture. The fundamental building blocks of C-ITS are the ITS-stations that implement the above standards to enable connectivity between road users. ITS-stations may be stationary, for example integrated within the infrastructure to wirelessly provide information about a traffic light or about the location of a roadwork. They may also be mobile and integrated within vehicles to inform the surroundings about vehicle's current speed, position, direction, etc.

A general reference ITS architecture is illustrated in Fig. 1. At the top layer is the ITS Application layer where the applications are executed. The next layer is the ITS Facilities layer where the Cooperative Awareness Messages (CAM), Decentralized Environment Notification Messages (DENM) and eventually any other custom messages are handled (encoded/decoded) and communicated to/from the ITS Network & Transport layer. This layer contains the BTP (Basic Transport Protocol) and GeoNetworking protocol that forwards the messages to the physical hardware at the ITS Access Technology layer.

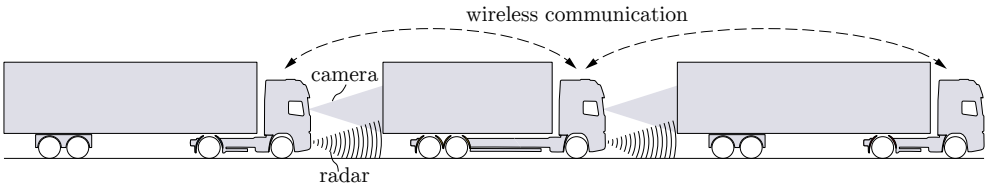


Fig. 2. Example of C-ITS platooning application.

III. Interactive Test Tool architecture

In a C-ITS system, interaction between cooperative vehicles is established in two ways: first, through the wireless exchange of information, and second, through the perception of the environment by means of on-board sensors, see Fig. 2. Hence, the ITT establishes two channels between the vehicles, see Fig. 3. In the ITT the channels connects two or more (physical or simulated) vehicles equipped with ITS-stations for co-simulation:

- one channel for V2X communication and,
- one channel for ground truth data for sensor modelling.

For the V2X channel, the ITT replaces the access and physical layers of the ITS-G5 communication stack [8] with a Virtual Private Network (VPN) tunnel. For the Ground Truth (GT) channel, vehicles and the ITT Server establish TCP connections, through which the self-reported ground truth data is exchanged between all vehicles, so that each vehicle has a complete set of ground truth data to synthesize its sensor readings.

A. Ground Truth channel

The Ground Truth data that the vehicles exchange through the ITT are listed in Table I. These data are encoded in the JSON format [5] and are exchanged between each client vehicle and the ITT Server through a TCP connection. On the first step of the simulation, the ITT Server provides initial positions, speeds and orientations for all vehicles within the co-simulation. In the second step, the ITT gathers the self-reported ground truth data and redistributes a complete set of ground truth data to all vehicles, so that each vehicle has a complete view of the world. Source code of an example ITT Server implementation is available at https://github.com/alexvoronov/itt-gt.

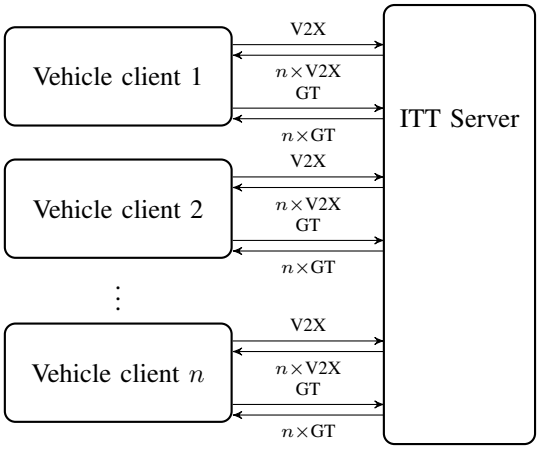


Fig. 3. Interactive Test Tool architecture.

TABLE I
Ground Truth data.

Parameter name	Units
Simulation time stamp	UTC string, ISO 8601
Id	Integer
Latitude	Degrees
Longitude	Degrees
Altitude	Meters above mean sea level
Speed	Meters per second
Heading	Degrees from north
Orientation	Degrees from north
Yaw rate	Degrees per second
Acceleration longitudinal	Meters per second squared
Acceleration latitudinal	Meters per second squared
Vehicle length	Meters
Vehicle width	Meters
Distance from ref. point to rear bumper	Meters

As mentioned above, the ITT provides initial conditions to all participants in the first simulation step. This is a natural operation for virtual vehicle models. But even a physical vehicle can participate in a co-simulation. A vehicle can be physically placed on a huge asphalt field or in a laboratory on a roller bench (dynamometer), while in the simulation the vehicle will behave as if it was where the ITT positioned it.

From the second iteration onwards, vehicles decide themselves their positions using, for example, vehicle dynamics models or readings from the wheel sensors. This self-decided position and orientation is communicated to the ITT. Redistribution of such self-reported ground truth data to all participants is made within the ITT. With a complete set of ground truth data, each vehicle can create appropriate sensor readings using sensor models. In the simplest setup, the sensor model just calculates an angle at which the nearby vehicle would have been detected by an on-board sensor. In a more advanced case, a detailed sensor model describes the physical characteristics of the specific on-board sensor. Another example of an advanced Hardware-in-the-Loop simulation is where a virtual vehicle is injected into a physical world (such injection is sometimes called *augmented reality*); in such a case, the sensor model combines information from physical sensors with information received from the ITT to generate the final sensor readings for the vehicle controller.

In the current setup, the ITT distributes only vehicle-

reported ground truth data. Static data, like topography, road width, road markings, number of lanes etc. have to be handled outside of the ITT. The geographical area where the ITT will position all participating clients is agreed in advance, and each participant have to acquire the needed static data themselves, for example, from a static map or from visiting the site and measuring it with a 3D scanner. For future setups, the ITT may also include a map and other environmental properties (friction, weather, etc.) of the virtual testing area.

B. V2X channel

The V2X channel is established by directing Data Link Layer traffic of the ITS station into a virtual network interface of an OpenVPN client (a so-called `tap` interface), instead of a network interface of a physical network card (e.g. `wlan0` on Linux). The OpenVPN client forwards all Data Link Layer data to an OpenVPN entity of the ITT Server. The server then redistributes all V2X data to appropriate clients. In the simplest setup, all clients are connected to the same virtual Ethernet segment, thus receiving all messages. In a more advanced implementation, the ITT Server may control delivery of V2X data, emulating packet loss, latencies, or wireless range limits, using, for example, network simulators like ns-3 [9].

The data exchanged through the V2X channel of the ITT are raw Ethernet frames, making the V2X channel very flexible. Payloads transferred by V2X channel may include (but are not limited to) CAM, DENM, or any custom messages sent through GeoNetworking, it can even be IPv4/IPv6 traffic if such need should arise.

Since the ITT replaces only the lowest layers of the ITS stack, the client vehicles may use the same ITS stack in physical and in virtual testing, making it Software-in-the-Loop or, if ITS stack is running on dedicated communication hardware, Hardware-in-the-Loop simulation. Ability to reuse the ITS stack is both cost-efficient and ensures that the tested implementation is the one that is later deployed.

C. Time management

ITT can work in one of two modes:

- Real-time mode,
- Non-real-time mode.

In real-time mode, all vehicles have their own real-time clock. In this mode, the time to collect and redistribute ground truth data and V2X data is mutually agreed in advance to be bound by a time constant that is acceptable for real-time controllers of all participating clients, e.g. 10 ms. That is, all clients have to generate data and deliver it to the ITT Server, and ITT Server have to gather all data and redistribute it to all clients, all within 10 ms. This is almost impossible to achieve if ITT is running over the Internet, but might be achievable in low-latency Local Area Networks.

In non-real-time mode, all clients are also time-driven, but the ITT provides the time update by communicating the current time stamp to all clients, which need not be advancing at the same pace as real time. Iterations of the vehicle controller are triggered by reception of ground truth data from the ITT Server. Non-real-time mode allows clients to be geographically distributed, as well as tolerates network latencies. Even imperfect networking is tolerated since both V2X channel and Ground Truth channel are running over TCP, which ensures the delivery of the data even in case of packets loss over the Internet.

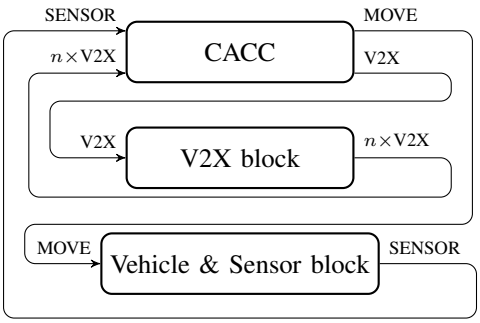


Fig. 4. Vehicle controller.

Non-real-time mode is only suitable for co-simulations where all participants are virtual, and can work using the time provided by the ITT. As soon as any client is a physical vehicle, the whole co-simulation has to run in real-time mode.

D. Isolation of Intellectual Property

ITT can connect any type of clients, both physical and virtual, the only requirements ITT imposes is the ability to run an OpenVPN client for V2X communication and the ability to exchange JSON-encoded ground truth data through a TCP connection, no other data is shared between the vehicles. The clients may run on different operating systems or in different simulation environments. This allows different users (even competing vendors) to perform tests with each other without revealing the details of their implementations.

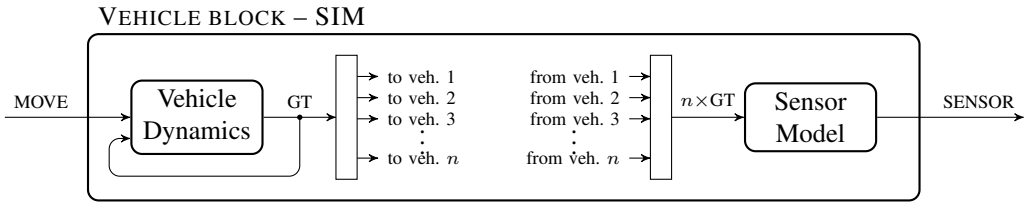
IV. Example structure of a vehicle client

Consider testing a vehicle controller for Cooperative Adaptive Cruise Control [11] (CACC). Such controller, depicted in Fig. 4, would output two groups of signals: signals for actuator, called MOVE in the figure, and signals to be transferred using V2X communication to other vehicles, called V2X in the figure. The controller would have two groups of inputs: sensor inputs, called SENSOR in the figure (for example, readings from an on-board radar generated from GT data from all other, n, vehicles), and signals received from other vehicles through V2X communication. Since vehicle receives V2X signals from n other vehicles, the input is called $n\times$V2X in the figure.

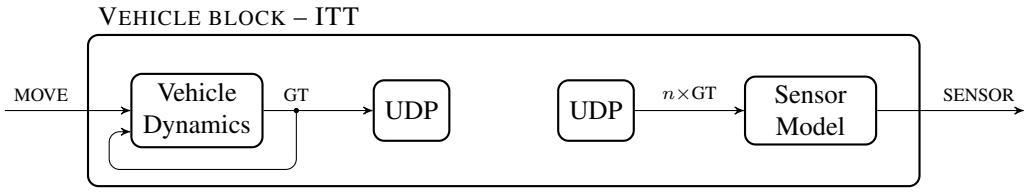
A. MOVE *and* SENSOR *signals*

A physical vehicle would send MOVE signals directly to the controller of an actuator, and receive SENSOR signals directly from on-board sensor systems.

To run the controller in a simulated environment, physical actuators and sensors have to be replaced by their models. One such implementation of the vehicle replacement block is shown in Fig. 5a. It consists of a Vehicle Dynamics model, a



(a) For monolithic multi-vehicle simulation



(b) For Interactive Test Tool

Fig. 5. Vehicle replacement block.

Sensor Model and an infrastructure to exchange Ground Truth data between vehicles for use in their Sensor Models.

The Vehicle Dynamics model takes MOVE signals and previous Ground Truth of the vehicle (denoted GT in the figure; it consists of position, speed and other parameters, see Table I) and calculates the Ground Truth for the next time step.

The Sensor Model takes as input Ground Truth data of all vehicles and produces appropriate SENSOR signals.

Within the vehicle replacement block, the Ground Truth data for each vehicle is forwarded to all other vehicles. One possible implementation, shown in Fig. 5a, multiplexes Vehicle Dynamics outputs and hard-wires them to appropriate inputs in Sensor Models of all other vehicles, thus requiring that the simulation is monolithic and the number of vehicles is fixed.

It is easy to adapt monolithic implementation of the vehicle replacement block to the ITT. GT data will be forwarded to the ITT server, and would be received by all other vehicles from the ITT server. This can be implemented, for example, through an UDP gateway, as shown in Fig. 5b. An UDP gateway is chosen due to availability of UDP in MATLAB/Simulink and other simulation tools. A separate ITT client adapter would accept UDP packets from the controller simulator and would pack and unpack the necessary data and communicate it via TCP to the ITT. Such an ITT GT client adapter is shown in Fig. 6. Example source code of ITT GT client adapter is available at http://github.com/alexvoronov/itt-gt.

If a physical vehicle is connected to the ITT, then MOVE signals are sent to vehicle actuators, and changes in GT data are collected from vehicle sensors (accelerometers, wheel rotation sensors etc). Note that absolute positioning using GPS might be disabled if the vehicle has to behave as if it was driving in a different geographical location. SENSOR signals are still generated by the sensor model using $n\times$GT data from the ITT.

B. V2X signals

A physical vehicle can have an ITS stack implemented as in Fig. 7. Such stack would receive signals from a CACC

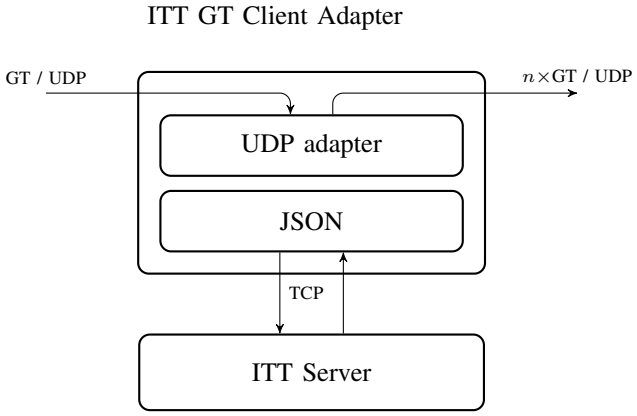


Fig. 6. ITT Ground Truth client adapter.

controller via UDP through a V2X module shown in Fig. 8. Exactly the same setup can be used for the ITT, the only difference being that the ITS stack will use `tap` virtual network interface towards OpenVPN client for ITT instead of `wlan` network interface to a physical network card for a physical vehicle. In the ITT, a virtual `tap` interface is connected to a Virtual Private Network (VPN) running on OpenVPN.

V. CONCLUSION AND FUTURE WORK

This paper presented an Interactive Test Tool (ITT) architecture for interoperability testing of C-ITS systems. The ITT allows for testing of such systems on all layers, up to and including the application layer. The tool may also include actual (physical or simulated) interaction between vehicles, also taking into account on-board sensors. The ITT enables V2X communication and provides ground truth data about all vehicles to all ITT clients, even if the clients are geographically distributed. ITT clients can be either virtual or physical vehicles. The ITT provides two modes of operation: real-time mode and non-real-time mode. When at least one client is a

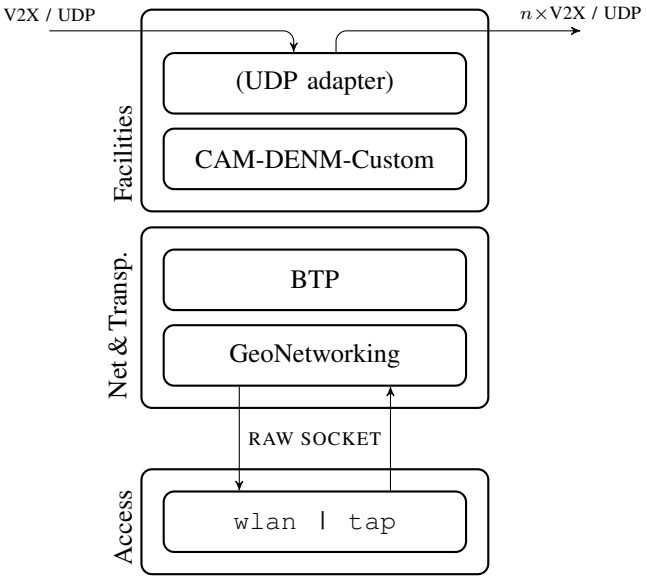


Fig. 7. ITS stack for both physical vehicle and for Interactive Test Tool. When stack is used for physical vehicle, `wlan` interface towards physical network card and antenna is used. When stack is used for Interactive Test Tool, `tap` interface is used towards OpenVPN client.

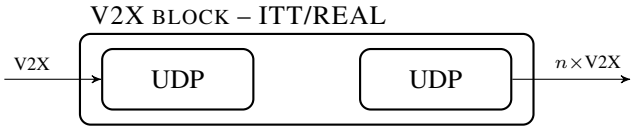


Fig. 8. V2X module for use with ITT and with physical vehicle.

physical vehicle, the ITT requires to be run in real-time mode; when all clients are virtual, the ITT can be run in non-real-time mode. Non-real-time mode allows clients to be connected to the ITT through an ordinary Internet connection. Since no implementation details are shared between vehicles, competing vendors can perform co-simulations and interoperability testing without revealing internal details. An implementation-agnostic interface of the ITT allows co-simulation between heterogeneous simulation environments, as well as enables Hardware-in-the-Loop simulations. The ITT will be further developed and implemented in the scope of the European i-GAME project.

ACKNOWLEDGMENT

This work was performed within the i-GAME project that has received funding from the European Union's Seventh Framework Programme for research, technological development and demonstration under grant agreement no 612035.

REFERENCES

[1] i-GAME, www.gcdc.net.

[2] IEEE Standard for Distributed Interactive Simulation - Application Protocols. *IEEE Std 1278.1a-1998*, 1998.

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

[4] Lei Chen and Cristofer Englund. Cooperative ITS - EU standards to accelerate cooperative mobility. In *The 3rd International Conference on Connected Vehicles & Expo (ICCVE 2014)*.

[5] Douglas Crockford. The JSON Data Interchange Format. Standard ECMA-404. Technical report, CH-1204 Geneva, 2013.

[6] Carlos Bragança de Oliveira and António Carvalho Brito. A Business Language For A Distributed Simulation Framework. *26th European Simulation and Modelling Conference - ESM'2012*, 2012.

[7] Fabian de Ponte Müller, JuanMaría Reveriego Sierra, Bernhard Kloiber, Matthias Röckl, and Thomas Strang. Interoperability Testing Suite for C2X Communication Components. In Thomas Strang, Andreas Festag, Alexey Vinel, Rashid Mehmood, Cristina Rico Garcia, and Matthias Röckl, editors, *Communication Technologies for Vehicles SE - 18*, volume 6596 of *Lecture Notes in Computer Science*, pages 203–215. Springer Berlin Heidelberg, 2011.

[8] ETSI. TR 101 607 Intelligent transport systems (ITS); Cooperative ITS (C-ITS); Release 1, 2013.

[9] Thomas R Henderson, Mathieu Lacage, George F Riley, C Dowell, and J B Kopena. Network simulations with the ns-3 simulator. *SIGCOMM demonstration*, 15:17, 2008.

[10] Thom McLean, Richard Fujimoto, and Brad Fitzgibbons. Middleware for real-time distributed simulations. *Concurrency and Computation: Practice and Experience*, 16(15):1483–1501, 2004.

[11] Jeroen Ploeg, Nathan Van De Wouw, and Henk Nijmeijer. Lp string stability of cascaded systems: Application to vehicle platooning. *IEEE Transactions on Control Systems Technology*, 22(2):786–793, 2014.

[12] J. Baños Polglase, C. Cárdenas Angelat, and A. Plaza Ortega. Test Tools for Road Safety Telematic Systems. In *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*, pages 2560–2564, April 2007.