On Time Constraints for Internet-Connected Multi-User Real-Time Traffic Simulation

Marie-Christin H. Oczko*, Lukas Stratmann*, Florian Klingler†, and Falko Dressler*
* School of Electrical Engineering and Computer Science, TU Berlin, Germany
† Department of Computer Science, TU Ilmenau, Germany
{oczko, stratmann, dressler}@ccs-labs.org, florian.klingler@tu-ilmenau.de

***Abstract*—In recent years, the inclusion of vulnerable road users (VRUs) such as bicyclists in road traffic systems has become a topic of general interest. However, the development and testing with humans in the loop are complicated. Co-simulation of cyclists on a training stand together with road traffic and vehicular networking simulation helps to get more insights into traffic interactions. This approach is currently limited as that only one real-time cycling stand can be supported. Therefore, we extend a centralized intelligent transportation systems (ITS) simulation to allow multiple real-time users simultaneously. We developed an architecture building upon new extrapolation and convergence algorithms to deal with communication lags. Our proof-of-concept for internet-connected multi-user real-time simulation confirms the general feasibility and allows us to gain insights into its technical limitations.**

I. Introduction

Looking back at the last decades, one observes enormous progress in the domain of vehicular networking and cooperative driving. Many car makers have already equipped their recent models with cellular and WiFi modems; some have also added vehicle-to-everything (V2X) communication technologies. So far, much of this work focuses almost exclusively on cars but leaves out communication and coordination with vulnerable road users (VRUs) such as pedestrians and bicyclists.

A possible solution is the co-simulation of human behavior together with intelligent transportation systems (ITS) simulation (road traffic and V2X communication). Such coupling of real-time systems (e.g., hardware-in-the-loop (HiL) simulation) and event-driven simulation (e.g., OMNeT++, SUMO, Veins) was first introduced by Buse et al. [1]. In previous work, we developed the virtual cycling environment (VCE) [2], which integrates a bicyclist on a training stand into a virtual cooperative driving scenario. Similar activities have been reported in [3], [4]. This allows the development and, most importantly, testing of safety solutions in a safe environment.

Now, integrating one real-time system with an event-driven simulation framework has been challenging already. In this paper, we go one step further and investigate the challenges of a multi-user concept, where multiple real-time systems simulators or instances of our VCE connect for live interaction as well as to the simulation framework for road traffic and V2X communication. We identified communication lag as the most critical issue. This may simply be caused by latency but also by packet loss. Extrapolation and convergence algorithms help to overcome such communication lag if designed appropriately. We developed a proof-of-concept to show the feasibility and to study the limits of the system. Our results show that using a state-of-the-art computer system, about 20 users can be supported for experimentation on safety solutions based on V2X communication among the VRUs and nearby cars.

Our contributions can be summarized as follows:

- We explore the challenges of internet-connected multi-user integration into real-time traffic simulation;
- we introduce an architecture to couple distributed VCE instances with a central Veins simulation;
- we present extrapolation and convergence algorithms to handle communication lag; and
- we developed a proof-of-concept implementation to obtain first insights into the technical limits of the system.

II. Related Work

A vast number of traffic simulators with different complexity levels and purposes have been designed. Various single-user setups have been developed to research human behavior in traffic scenarios [3]–[5]. With the increasing need to study complex scenarios focusing on human interaction, the focus has begun to shift to multi-user simulation and the resulting requirements [6]. Abdelgawad et al. [7] designed a networked driving simulator for testing, training, and development purposes in cooperative vehicle systems and autonomous driving systems. In their setup, two driving simulators are connected via a central station controlling the simulation session. However, all of these systems are not able to also co-simulate complex scenarios with additional simulated traffic, support for multiple participants at the same time, and, most importantly, V2X communication at the same time.

To combine a real-time HiL setup with discrete-time simulators for traffic and V2X simulation, Buse et al. [1] developed the ego vehicle interface (EVI). Besides the ability to study advanced driver assistance systems (ADAS) in a HiL setup with integrated V2X communication, the potential application range of the EVI is huge. We are particularly interested in the use case of studying bicyclists' behavior in traffic scenarios. For this, in earlier work, we developed the VCE [8], which enables a single person on a stationary bicycle trainer to interact with potential ADAS applications and simulated traffic. In our work, we use and further extend the VCE to study internet-connected distributed multi-user simulation.

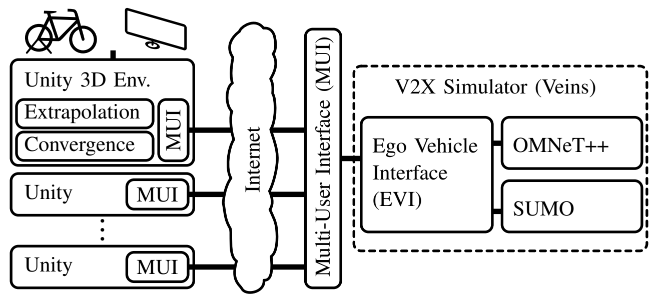


Figure 1. Architecture extending VCE with the MUI to connect multiple Unity instances with the EVI. The MUI is split into two parts, one integrated within Unity and one connected to the EVI.

With coupling simulators and executing distributed simulation, multiple challenges have arisen. Tranninger et al. [9] discussed strategies for fault-tolerant coupling of real-time components with a focus on data loss, faults, and time-varying delays. Stettinger et al. [10] highlighted the general difficulties of coupling real-time simulators caused by communication delays. On the positive side, Schreiber et al. [11] showed the feasibility of X-in-the-loop tests of automotive systems with distributed laboratories, located in Germany, the Netherlands, South Africa, and the USA. Furthermore, Aramrattana et al. [12] tested a distributed simulator for cooperative ITS.

To be able to deal with network delays, extrapolation, as also often used in multiplayer-computer games, needs to be introduced to the simulation environment. One approach to realize extrapolation is dead reckoning, which, for example, has been used for distributed cars by Chen and Liu [13]. The authors included additional knowledge about routes in the simulation to increase their prediction accuracy, introducing path-assisted dead reckoning. To overcome latencies between geographically-distributed experiment setups, we apply a similar solution.

III. Multi-Player Support for VCE

A. Architecture

The current architecture for the co-simulation of an ego user (i.e., a bicyclist in the case of the VCE) and the Veins simulator (consisting of the network simulator OMNeT++ handling V2X communication and SUMO handling road traffic) is based on the ego vehicle interface [1]. We extend this architecture by adding the possibility to integrate multiple ego users. These can be HiL setups, car driving simulators, or bicycles on a training stand as used in the VCE. In the following, we refer to these ego users by their Unity3D[1] visualization. The extended architecture is depicted in Figure 1. As it can be seen, an additional interface, called the MUI, between the EVI and the Unity-based ego users, is introduced. The MUI organizes the data exchange between multiple Unity instances and coordinates the message exchange with the EVI. Technically, the MUI consists of two different components: a *Unity Connector* responsible for coordinating the communication with various Unity instances, and an *EVI Connector*, handling the communication with the EVI. During the initialization phase, the Unity Connector waits until each Unity instance has connected to the MUI, more precisely until it has received at least one position update from each entity. This approach is applied to ease the connection from multiple experiment sites, which could come with varying delays, and it guarantees that the expected number of Unity instances is connected at least once. After initialization, the Unity Connector only waits until a certain timeout to guarantee the necessary 100 ms update intervals of the EVI. Additionally, the Unity Connector will collect position updates from the different Unity instances and join them together in one single EVI message. As a result, the EVI treats all updates as updates from a single Unity instance, with multiple human-controlled vehicles connected to it. This technique offers the advantage of full transparency for the EVI. In the other direction, the Unity Connector needs to distribute traffic updates from the EVI to the different connected Unity instances.

[1]https://unity.com/

B. Challenges and Feasibility

A known challenge in internet-connected distributed simulations that also commonly affects multi-player computer games is communication latency and its variations (e.g., intermittent jitter). To some extent, it is possible to prevent simulated vehicles from visually jumping by introducing interpolation, extrapolation, and convergence techniques.

Interpolation can be applied to smooth the transition of an object between two received position updates since the frame rate of the 3D environment is typically higher than the frequency of network updates. This has already been implemented in the single-user VCE. *Extrapolation* can be used to predict the motion of a vehicle even when the next position update is still pending. One possible approach is dead reckoning, in which predictions are based on the last known position, velocity, and, optionally, acceleration. As a consequence, there might be deviations in the predicted and the actual position, which can be detected later based on received updates. *Convergence* helps correcting these mistakes, ideally in a visually smooth and inconspicuous manner. Typically, a vehicle recovering from a significant position deviation would start a *convergence period*, during which it smoothly transitions to the convergence point and catches up with the correct position. In a vehicular context, it can be useful to limit changes in linear and angular velocity of a vehicle to prevent unrealistic motion during the convergence period [13].

To assess whether implementing a multi-user VCE would be feasible at all and to gain first insights into the requirements

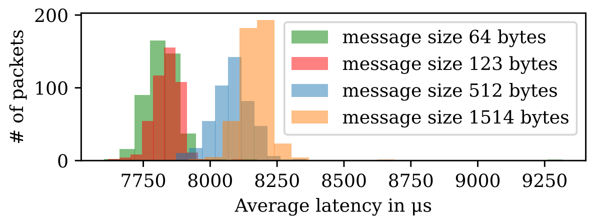


Figure 2. Histogram of the observed communication latencies between Paderborn and Berlin for message sizes of 64 B, 123 B, 512 B, and 1514 B.

for an extrapolation algorithm, we performed the following preliminary experiments. By default, the VCE operates with update intervals of 100 ms. Communication delays should stay below this threshold to allow for smooth operation even in large simulation scenarios requiring longer computation time.

To check the feasibility of our concepts, we first used *tcp-dump*[2] to determine the maximum size of messages transmitted between the 3D environment of the VCE and the EVI in our reference scenario. In the direction from the 3D environment to the EVI this turned out to be 123 B and 1514 B in the reverse direction. We used *SockPerf*[3] to transmit TCP packets of size 64 B, 123 B, 512 B, and 1514 B in intervals of 15 min between Paderborn and Berlin over the course of six days. As shown in Figure 2, the observed communication latencies stayed well below 10 ms, with an average of 7.82 ms for 64 B messages, 7.83 ms for 123 B, 8.1 ms for 512 B, and 8.2 ms for 1514 B.

C. Extrapolation and Convergence

Since the different simulator instances are connected via the internet, delays could adversely affect the visualization and update rate of the various simulation components. Furthermore, to provide a smoothly running environment, delays and packet loss effects must be counteracted. Thus, we introduced an extrapolation algorithm to the internet-based multi-user VCE. It is applied to predict vehicle behavior when traffic updates are delayed to support a smooth and realistic visualization. When developing an extrapolation and convergence concept for vehicles, certain assumptions can be made. Usually, traffic updates in the multi-user VCE should arrive regularly (e.g., every 100 ms), which means that extrapolation should only be applied for small time intervals. Furthermore, most situations being examined with the VCE involve bicycles and are restricted to city traffic. Thus, we can assume that the convergence algorithm only has to handle small deviations of predicted and actual position received via an update.

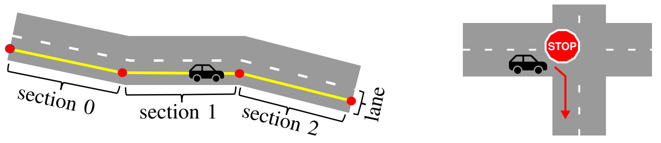
We started with the concepts introduced by Chen and Liu [13] and included path knowledge in our predictions. However, this information was not available in the original Unity system. For that purpose, we reused the SUMO file representing the street network information as a graph structure.

We realized two approaches for extrapolation, distinguishing human-controlled vehicles and other road users. Human-controlled vehicles behave less predictably than simulated ones, and, in the case of bicycles, usually travel quite slowly. Thus, we simply extrapolate the position of a human-controlled vehicle based on its current speed, position, and direction. We used the equation from [13]: $s = s_0 + v\Delta t$, where s refers to the next position, s_0 to the last received position, v to the velocity, and t to the time that has passed since the last update. For simulated vehicles, the behavior is more predictable. As long as the vehicle is not close to an intersection, it is most likely to keep its speed and keep following the lane.

In the SUMO configuration file, each lane has its shape, more precisely its center line, defined by a sequence of coordinates,

[2]https://man7.org/linux/man-pages/man8/tc-netem.8.html

[3]https://manpages.debian.org/bullseye/sockperf/sockperf.1.en.html



(a) Case A (b) Case B

Figure 3. Case A: The vehicle continues following the lane during extrapolation. The lane consists of at least one section. The center of the lane, here yellow, is defined by shape-defining points (red). Case B: If the vehicle is close to an intersection, it stops shortly in front of it.

as shown in Figure 3a. We compute the two points defining the center of the vehicle's current street segment and the direction it is driving:

- To get the closest shape-defining point p_1, our algorithm iterates over all shape-defining points of the lane, computes the distance to each, and returns the closest one.
- Next, we compute the index of this point in the lane's list of shape-defining points.
- We check whether our vehicle is moving to or away from point p_1. Based on that, we can compute the index of the second point p_2 defining our current street section, and by that, get its coordinates from the SUMO file.
- As a result, we can calculate a vector $streetSection$ based on p_1, p_2, representing a direction parallel to the center of the lane.
- Our next step is to compute the angle of the vector $streetSection$, which gives us the degree we have to adjust the direction of the vehicle.
- Based on the angle, the new vehicle's yaw rate is computed, as well as its new speed vector.
- Using this speed vector, the old vehicle position, and the passed time since the last update, we can compute the new position of the vehicle.

Otherwise, as shown in Figure 3b, if a vehicle is close to an intersection, we stop the vehicle by keeping its position and setting its speed to zero. Even though the turn signals of the vehicle are available in Unity, it is not always possible to unambiguously determine the lane the vehicle will continue on after the intersection as there might be multiple lanes on the left and right of the intersection. Thus, we avoid jumps due to wrong predictions but we have to compromise on the next update, which will forward the vehicle faster than in reality.

IV. Results

We evaluated the practicality of our multi-user VCE in terms of performance, stability, and scalability. Our main goal is to connect virtual cycling environments from different sites via the internet to conduct joint experiments. As a proof-of-concept, we connected a cyclist at TU Berlin with a cyclist at Paderborn University in a coupled simulation. For about ten minutes, both participants rode their bicycles and interacted with both the simulated traffic and each other. Comparing screen recordings of the experiment sites, we confirmed that both cyclists experienced the same situation at the same time. To gain more insights into scalability and stability, we investigated the

Table I
SCALABILITY OF THE CO-SIMULATION OF MULTIPLE EGO VEHICLES

# of ego vehicles	Visualization	MUI	EVI
2 to 10	smooth	normal	normal
20	environment/car visualization (slightly) lagging	normal	normal
30	environment/car visualization lagging, cars missing	normal	partly delayed

Table II
NETWORK TRAFFIC MEASUREMENTS FROM THE MUI TO A UNITY INSTANCE FOR AN INCREASING NUMBER OF UNITY INSTANCES AND DIFFERENT EVI UPDATE INTERVALS

	# of Unities				
EVI	1	3	5	10	20
100 ms	27.4 pps	29.3 pps	32.2 pps	31.5 pps	31.3 pps
95 ms	29.1 pps	30.6 pps	33.1 pps	31.0 pps	33.3 pps
80 ms	31.8 pps	32.1 pps	35.0 pps	36.2 pps	34.6 pps
60 ms	29.0 pps	37.0 pps	35.7 pps	31.3 pps	17.3 pps
50 ms	27.5 pps	37.6 pps	36.4 pps	30.9 pps	3.0 pps

impact of the number of ego vehicles on the simulation quality. For that, we connected a varying number of Unity instances (2 to 30) to our newly developed MUI. For reproducibility, we used a well-defined traffic scenario. For each configuration, we measured the data rate, the packet size distribution, and the number of messages between the different simulators. Additionally, we monitored the changes in the visualization in Unity, as well as in the MUI and the EVI behavior. As a platform, we used a computer with an AMD Ryzen$^{\text{TM}}$ 7 5800X and an MSI RTX 3070 graphics card. The results are listed in Table I. Starting at twenty connections, the observed visualization quality quickly decreased; some cars behaved jerky. Additionally, the general visualization quality decreased slightly. These effects were even more pronounced for thirty connections, marking the limit where the Unity simulation executed on the laptop became unusable. Yet, for realistic experiments, this seems a suitable performance to conduct experiments on vulnerable road users.

Based on these results, we looked further into the update interval. We measured packet rate from the MUI to a Unity instance for an increasing number of Unity instances and different EVI update intervals as shown in Table II. Comparing update intervals of 80–100 ms, the packet rate slightly increases for a decreasing update interval for an equal number of Unity connections. However, they started decreasing after further decreasing the update interval to 60 ms. We conclude that changes in the EVI update interval can help to improve the quality of the simulation.

V. CONCLUSION

We investigated opportunities and challenges related to coupling multiple internet-connected real-time simulators in a joint environment with event-based road traffic and a V2X simulation toolkit. This was already challenging for a single ego vehicle case with the ego vehicle interface. Our application domain is the development and testing of safety solutions for vulnerable road users, specifically bicyclists. We developed a novel architecture and implemented the multi-user interface (MUI) as a proof of concept. The core functionality of the system is based on extrapolation and convergence algorithms to cope with communication lags. Starting from available solutions in the scientific literature, we extended these for our use case. In a set of experiments, we confirmed the general functionality and also showed the technical limits of the system.

ACKNOWLEDGMENT

This work was supported in part by the Federal Ministry of Education and Research (BMBF, Germany) within the 6G Platform under Grant 16KISK050.

REFERENCES

[1] D. S. Buse, M. Schettler, N. Kothe, P. Reinold, C. Sommer, and F. Dressler, "Bridging Worlds: Integrating Hardware-in-the-Loop Testing with Large-Scale VANET Simulation," in *14th IEEE/IFIP Conference on Wireless On demand Network Systems and Services (WONS 2018)*, Isola 2000, France: IEEE, Feb. 2018, pp. 33–36.

[2] M.-C. H. Oczko, L. Stratmann, M. Franke, J. Heinovski, D. S. Buse, F. Klingler, and F. Dressler, "Integrating Haptic Signals with V2X-based Safety Systems for Vulnerable Road Users," in *IEEE International Conference on Computing, Networking and Communications (ICNC 2020)*, Big Island, HI: IEEE, Feb. 2020, pp. 692–697.

[3] M. Aramrattana, T. Larsson, J. Jansson, and A. Nåbo, "A simulation framework for cooperative intelligent transport systems testing and evaluation," *Elsevier Transportation Research Part F: Traffic Psychology and Behaviour*, 2017.

[4] A. Hussein, A. Díaz-Álvarez, J. M. Armingol, and C. Olaverri-Monreal, "3DCoAutoSim: Simulator for Cooperative ADAS and Automated Vehicles," in *21st IEEE International Conference on Intelligent Transportation Systems (ITSC 2018)*, Maui, HI: IEEE, Nov. 2018.

[5] F. Schramka, S. Arisona, M. Joos, and A. Erath, "Development of virtual reality cycling simulator," *Journal of Computers*, vol. 13, no. 6, 2018.

[6] K. Abdelgawad, J. Gausemeier, R. Dumitrescu, M. Grafe, J. Stöcklein, and J. Berssenbrügge, "Networked Driving Simulation: Applications, State of the Art, and Design Considerations," *Designs*, vol. 1, no. 1, Jun. 2017.

[7] K. Abdelgawad, S. Henning, P. Biemelt, S. Gausemeier, and A. Traechtler, "Networked Driving Simulation for Future Autonomous and Cooperative Vehicle Systems," in *8. VDI/VDE Fachtagung Automatisiertes Fahren und vernetzte Mobilität*, Berlin, Germany: VDI, Jul. 2017.

[8] J. Heinovski, L. Stratmann, D. S. Buse, F. Klingler, M. Franke, M.-C. H. Oczko, C. Sommer, I. Scharlau, and F. Dressler, "Modeling Cycling Behavior to Improve Bicyclists' Safety at Intersections – A Networking Perspective," in *20th IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM 2019)*, Washington, D.C.: IEEE, Jun. 2019.

[9] M. Tranninger, T. Haid, G. Stettinger, M. Benedikt, and M. Horn, "Fault-tolerant Coupling of Real-Time Systems: A Case Study," in *3rd Conference on Control and Fault-Tolerant Systems (SysTol 2016)*, Barcelona, Spain: IEEE, Sep. 2016.

[10] G. Stettinger, M. Benedikt, N. Thek, and J. Zehetner, "On the difficulities of real-time co-simulation," in *International Conference on Computational Methods for Coupled Problems in Science and Engineering (COUPLED 2013)*, Santa Eulàlia, Spain: CIMNE, Jun. 2013.

[11] V. Schreiber, V. Ivanov, K. Augsburg, M. Noack, B. Shyrokau, C. Sandu, and P. S. Els, "Shared and Distributed X-in-the-Loop Tests for Automotive Systems: Feasibility Study," *IEEE Access*, vol. 6, pp. 4017–4026, May 2018.

[12] M. Aramrattana, A. Andersson, F. Reichenberg, N. Mellegård, and H. Burden, "Testing cooperative intelligent transport systems in distributed simulators," *Elsevier Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 65, pp. 206–216, Aug. 2019.

[13] Y. Chen and E. S. Liu, "Comparing Dead Reckoning Algorithms for Distributed Car Simulations," in *ACM SIGSIM Conference on Principles of Advanced Discrete Simulation (PADS 2018)*, Rome, Italy: ACM, May 2018.