Proceedings of the 11th International Conference on Applied Informatics
Eger, Hungary, January 29–31, 2020, published at http://ceur-ws.org

Extensions and Usage of Veins/Plexe to Evaluate QoS Requirements of Cooperative Platooning*

András Wippelhauser, László Bokor

Budapest University of Technology and Economics, Department of Networked Systems and Services, Budapest, Hungary
andras.wippelhauser@gmail.com, bokorl@hit.bme.hu

Abstract

Vehicle-to-Anything (V2X) communication is expected to make traffic more efficient and safe by creating an essential infrastructure for Cooperative Intelligent Transport Systems (C-ITS). Cooperative platooning is a C-ITS application controlling a group of vehicles and maintaining small intervehicle distances even at high speeds. The small distance between the vehicles requires an extended vision for the adaptive cruise control (ACC) algorithms, which can be provided through advanced V2X communication, such as creating an extension of ACC called cooperative ACC (CACC). The Quality of Service (QoS) parameters of the V2X communication influences the reliability of the CACC algorithms. In this paper, we modeled and examined the effects of QoS parameters on the platooning control algorithms using Veins/Plexe as the base simulation framework.

Keywords: V2X, cooperative platooning, ACC/CACC algorithms, QoS

1. Introduction

1.1. V2X principles and context

Vehicle-to-Anything communication is direct and fast (very low-latency), but not expected to provide high throughput. V2X communication protocols are designed

*The research reported in this paper was supported by the Higher Education Excellence Program in the frame of Artificial Intelligence research area of Budapest University of Technology and Economics (BME FIKP-MI/FM) and by the EFOP-3.6.2-16-2017-00013 project.

for fully distributed networks that are able to operate without any centralized or cellular-like control and management systems [12]. This eliminates the risk of service outages either due to lack of coverage or system failure. The two major existing physical layers for V2X services are the IEEE 802.11p (IEEE-802.11-2016) [1] and the LTE-V2X protcols [2]. Above the complex PHY and MAC layers, simple network and transport protocols are implemented to support the so-called Facility Layer signaling messages which can share key properties about the vehicles or the surrounding environment. The applications built on top of these messages can be categorized into different days or phases of deployment (in 1-5 levels) [3, 13]. The platooning application is a Day 3 V2X application because it also relies on intention data, and the full control is taken from the driver among specific circumstances.

1.2. Platooning application

The platooning term refers to a set of vehicles, where the corresponding vehicles drive in one line, following each other. We assume that the control of the first car in the group (the platoon leader) is solved either with a professional driver or with a fully autonomous car. The steering of every other, so-called following vehicle, is considered to be handled in terms of this paper. The cooperative term refers to platooning applications, where the algorithm of the lateral acceleration can also take motion state information of the non-adjacent vehicles into account. The platoon uses V2X communication features to gain information about non-adjacent vehicles. Having information about non-adjacent vehicles is a must because it is a mathematically proven fact that the linear response algorithms which try to maintain a fixed intervehicle gap require such information to ensure the string stability property [4, 14]. This strong requirement indicates that disturbances in communication can have a massive effect on the safety of the platoon [15].

1.3. Scope of the examination

The main topic of this article is the inspection of CACC algorithms for platooning in circumstances where the communication link has varying QoS parameters. To reach this goal, we extended the existing framework of Veins/Plexe [5] to be able to define the QoS parameters for every inter-vehicle V2X link explicitly. We executed simulations using homogenous platoons controlled by models of existing control algorithms (ACC and CACC types) with various latency, jitter, and packet loss parameters. We analyzed the circumstances if any collision or dangerous behavior applies. The rest of the paper is organized as follows. First, we give a short overview of the background focusing on the main communication techniques and control algorithms applied in platoons. Then we introduce the simulation environment we used and extended, followed by the measurement details, the simulation scenarios, results, and their analysis. Finally, we conclude the paper and draw our future research plans.

2. Background

2.1. Communication between platoon members

The PHY and MAC layers of existing, available V2X communication techniques are based on the IEEE 802.11p [1] or the LTE-V2X [2] standards. The cooperative platooning application requires regular updates about the motion states of the platoon member vehicles and some additional messages to maintain the platoon members, roles, and maneuvers [16, 17].

EU and US standardization provide two similar message types to implement the periodic update ("heartbeat") feature. These two message types are Cooperative Awareness Messages (CAM) [8] and Basic Safety Messages (BSM) [9]. The CAM and BSM all contain the required motion state information, and they both send the information with a maximum frequency of 10 Hz. These two message types are both modular and have many optional fields, including additional status information about the vehicle state.

The other type of messages handle management tasks for platoons. These messages are responsible for maintaining the platoon member vehicles, the leader of the platoon, the maneuvers made by the platoon, for example, merging two platoons, changing the lines, or splitting. Currently, these types of messages and solutions based on them are under standardization and evaluation [6, 18, 19, 20].

2.2. Control algorithms

The platooning application was described first in the 1960s [7]. Since then, multiple predecessor algorithms were present. Maybe the most important invention in this topic was the work mathematically proved that the elimination of the string stability problem in case of using linear response algorithms that try to maintain a fixed intervehicle gap is only possible if the information is not only available from the adjacent vehicles [4].

A control algorithm for vehicle platooning is expected to produce the desired acceleration or deceleration value as output to determine the lateral dynamics of the controlled vehicle. The steering of the controlled vehicle is considered to be solved. The control of the platoon leader is handled by a fully autonomous vehicle or a human driver.

Existing control algorithms can be categorized by the inputs of they are relying on. If the control algorithm has only information about the adjacent vehicles, then the solution is ACC. At the same time, the algorithm also uses information about non-adjacent vehicles, and then it is a CACC. CACC algorithms can have the string stability property – unlike the ACC –, making the technique able to be a secure basis for platooning control.

Simulational examination of the platooning application is a reasonable method because it can provide realistic data much cost-efficiently than using field tests. Several articles are examining the quality of service (QoS) requirements of the platooning application from various approaches via simulation. The Plexe article [5]

defines the packet length, implements the MAC and the physical layer protocol, and simulates various platooning maneuvers, so it examines the safety of the platooning application with communicational disturbances. Another approach [11] is to examine the platooning application if short term disturbances occur, for example, the device reboots. This article claimed that – depending on the speed change and distance between the vehicles – if the loss of communication occurs for 2 seconds, the vehicles will collide. Our approach is different from the ones mentioned earlier. We defined QoS parameters, which are explicitly defined. Our sensor model also includes a radar in front of the vehicles which can be able to handle communicational outages.

3. Simulation Environment

3.1. Plexe

As part of our work, we selected a simulation framework that is able to simulate the platooning traffic with support for parametrized lateral control algorithms. The framework had to support V2X communication, as well. The Plexe framework was chosen, which is an extension of the Veins simulation framework [5, 21]. It is designed to support the realistic simulation of various platooning scenarios. The Plexe – or the Veins – framework consists of the following three elements.

- The SUMO microscopic simulation framework [22]. SUMO implements traffic simulation with different driving models. It makes it possible to import maps from OpenStreetMap [10] or implement any driving model, in our case, any platooning control algorithm. The SUMO simulation is accessible through a so-called TraCI interface, which is a network-based interface with extensible instruction sets. The Plexe implements four cruise control algorithms to demonstrate the limitations of the platooning application.

- The OMNeT++ discrete-time event-driven network simulation framework [23]. This framework simulates network traffic through various terrestrial situations with various parameters given. Plexe uses the communication implemented in the Veins framework and extends it with additional functionalities. Plexe implements a higher layer unicast protocol and basic beaconing protocols with corresponding message definitions to support signaling messages of the cooperative platooning applications. Plexe also defines a superclass for platooning applications, which can pass wirelessly received data to the controllers and also logs the motion data.

- The Veins open source vehicular network simulation framework [21]. The Veins module connects OMNeT++ and SUMO through the SUMO's TraCI interface. Simulation parameters are stored in the configuration file of OMNeT++, the driving scenario, and the platooning control algorithm is implemented in the SUMO framework. The Veins module itself compiles in a

shared object file, which is loaded by OMNeT++. Veins manages the SUMO simulation, the network simulation, and the TraCI connection as well. The Plexe framework extends Veins with a custom unicast message type and the implementation of the messages which are responsible for exchanging the information about the vehicle's motion state.

3.2. The examined communication parameters

To examine QoS requirements of cooperative platooning control algorithms, we selected the Average Packet Loss, Latency, and Jitter as the most important communication QoS parameters.

The average packet loss is interpreted here as a certain percent of the signaling packets of the cooperative algorithm that does not arrive at the receiver. The loss of each packet is independent and follows a Bernoulli distribution, where the probability – and the mean – value of the packet loss is the configurable packet loss parameter. The L represents the event of packet loss.

$$L \sim B(p)$$

Latency represents the amount of time, which corresponds to the time difference between sending a signaling packet and receiving it. The latency is caused by processing delays, communicational delays, and many other reasons.

$$t_{\text{latency}} = t_{\text{receive}} - t_{\text{send}}$$

The jitter parameter in our analysis follows a normal distribution, with an adjustable dispersion parameter and 0 as the mean parameter. This definition of the latency would allow the simulator to send a packet before the generation of the packet, which would not be realistic, nor implementable, so this effect is filtered. The latency and the jitter parameters together form a normal distribution where the mean – μ – is the configured latency parameter and the dispersion – σ – is the configured jitter parameter.

$$t_{\text{Latency with Jitter}} = \begin{cases} N(\mu, \sigma) & \text{if} > 0 \\ 0 & \text{otherwise} \end{cases}$$

3.3. Implementation of explicit QoS parameter declaration

To perform the desired analysis, we had to extend the Plexe framework with the support of an explicit declaration of communication QoS parameters. For batch execution, the values of the QoS parameters must be given in the configuration file of OMNeT++. Therefore we introduced proper TraCI messages at the Veins side to enable the configuration as an adjustable parameter. The above mentioned three explicit QoS parameters were implemented as receiver functions. The latency and the jitter were implemented as OMNeT++ self messages parameterized with a certain latency. A new application class was introduced at the Veins side called

QoSApp containing two instances from helper objects: one of them implements the draw of the probabilistic packet loss using the selected Bernoulli distribution, the other implements the draw of the latency values using the normal distribution. The QoSApp class is responsible for the implementation of the packet loss as well. It stores the message frame of the delayed message and sends a self message to schedule the delivery of the packets. As the self message arrives at the QoSApp, it makes the corresponding message available for the cooperative platooning application.

3.4. The examined algorithms

There are multiple approaches to ACC/CACC algorithms with different design visions and objectives. We examined four different algorithms that were already implemented in Plexe. We did not change the built-in parameters of the algorithms to stay consistent with the original framework.

- Plexe ACC. This algorithm only considers the motion state of the preceding vehicle. Therefore, this algorithm can not ensure string stability with a low headway time gap. The safety of the platoon can only be ensured if quite long distances – time gaps – are maintained between the vehicles. The ACC algorithm relies on the distance between the preceding and the ego vehicle and the speed of the ego and the preceding vehicle.

- Plexe CACC. The CACC algorithm uses V2X communication to gather information from the preceding and the leader vehicle. This algorithm counts with the information obtained from the leader and preceding vehicle. It uses the acceleration, speed, and distance values as well.

- Plexe PLOEG. The PLOEG algorithm uses information only from the preceding vehicle transmitted via V2X communication in the Plexe implementation. This approach can be realistic because the acceleration is the second derivate of the distance. If the acceleration sensing is based on distance sensing, the accumulated errors can lead to unreliable results. This problem can be solved by using communication to share the measurements of the sensors in the preceding vehicle.

- Plexe CONSENSUS. The Consensus algorithm can count on every member of the platoon. This algorithm has an adjacency matrix filled with properly chosen coefficients. The Plexe implementation of the Consensus algorithm only uses the leader and the preceding vehicle data obtained from V2X communication.

4. Simulation Results and Analysis

4.1. Simulation scenarios

Plexe implements the two most important safety-relevant scenarios in terms of the platooning application. In both of them, the first car is driven by the simulator on a flat and straight highway with different characteristics of speed.

The Braking scenario is responsible for testing emergencies where the platoon has to stop in a very short distance. The leader vehicle brakes hard, and the follower vehicles have to stop without colliding the other cars.

In the Sinusoidal scenario, the speed of the leader vehicle follows a sinus curve. This scenario is responsible for testing whether the algorithms can ensure safe operation in situations where they face an excitation with a constant frequency. The desired answer of the algorithm is a decaying behavior where the algorithm smoothens the speed oscillation of the leader car. It is easy to consider that an ascending answer would lead to collisions. This ability is called string stability.

4.2. Simulation execution

The introduced QoS parameters, the two crucial scenarios, and the applied algorithms were defined as adjustable variables in the OMNeT++ configuration file. The QoS implementation introduced significant random factors in the simulation runs – without the QoS implementation, and the simulation runs did not show an indeterministic behavior.

Figure 1: The platoon contained 8 vehicles.

In order to get statistically relevant results, it was necessary to execute each simulations multiple times. The OMNeT++ framework executed each combination of the available variables, where the available free factors were the following:

- the examined algorithm. The simulation ran on four different ACC/CACC algorithms, where the first algorithm was represented twice because it ran with two different configuration parameters;
- the packet loss rate, parametrized from 0 percent until 70 percent with 10 percent steps;
- the standard deviation of the jitter could be 0 or 500 milliseconds;
- the delay parameter could be 0 and 1 second;
- the two key scenarios;

- finally, an additional helper parameter was introduced to execute the test cases multiple times.

Each scenario covered a 60 seconds long situation on a straight highway. The platoon contained eight vehicles, a leader and seven follower vehicles, as it is visible in Fig 1. The platoon used a dedicated lane; the surrounding traffic was not a disturbing factor. Each simulation run took approximately 10 seconds for the framework to simulate. During our analysis, we executed thousands of different simulations and produced more gigabytes of data to be processed.

4.3. The developed analysis tool

A novel tool was developed with the capability to handle the multiple simulation runs. The analysis tool was developed to process and visualize the most critical measured Key Performance Indicators (KPIs) of the examined algorithms. The logging of the measured data in the simulation framework was performed on the Veins side, where each platoon participants sent their movement data to play the communication parts of the overall simulation. In the platooning application the following KPIs were logged and analyzed:

- the distance between the platoon members, where the logged parameter is the distance between the adjacent vehicles;
- speed of the platoon member vehicles;
- the acceleration of the platoon member vehicles.

The logging is implemented with standard OMNeT++ methods on the communication simulation side, with a defined frequency, along with the beaconing messages.

The distance between the platoon member vehicles is the most important evaluation criterion because it shows if the vehicles collide, which is not acceptable due to the strict safety requirements of the platooning application. The speed and the acceleration data shows whether the algorithms can perform according to the design principles also in suboptimal communicational situations or not. The tool, in its current form, calculates the minimum, maximum, deviation, and mean values of the measured data.

4.4. Evaluation method and simulation results

The main goal of the analysis was to point on the most sensible QoS parameter of the examined algorithm and also to specify the algorithm's numerical limits. It is important to note that the algorithms were used with the coefficients/parameters defined in the article where they were used. This means that not all of the algorithms try to keep the same time gap or distances, so a direct comparison is not applicable. During the evaluation, we used the developed analysis tool to evaluate the simulation results.

The experience of the analysis is that most of the algorithms are sensitive for the latency, especially when having a braking scenario. Most of the algorithms could handle the Sinusoidal scenario, but some of them could only handle it because they had a relatively wide safety gap. If having a Sinusoidal gain, the correct answer from the algorithms is a decaying behavior, which we could see from some of the algorithms. In the Braking scenario, however, some of the algorithms did not respond hard enough to stop without colliding. This behavior is potentially caused by choosing too small coefficients, which creates a smooth response in most of the time, but when having a hard brake situation, this approach does not perform well.

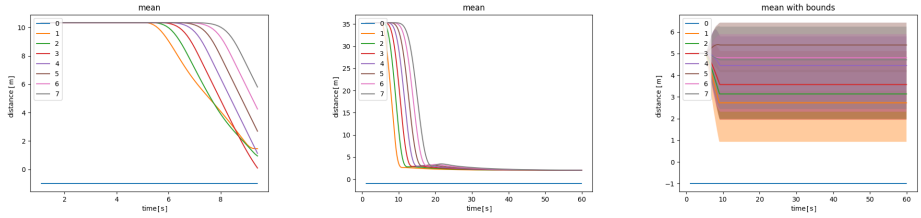

Figure 2: Braking scenario: ACC algorithm with 0.3 and 1.2 seconds time headway and CACC algorithm with 30% packet loss

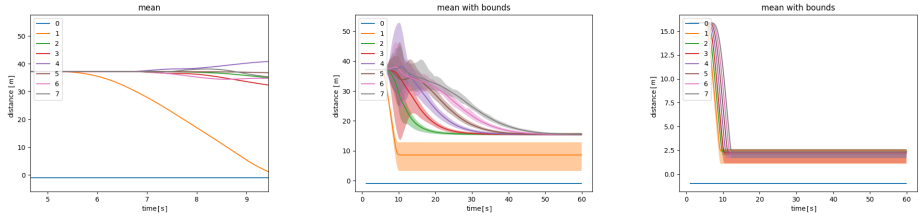

Figure 3: Braking scenario: Consensus algorithm with 0.5 seconds delay and 70% packet loss, PLOEG algorithm with 30% packet loss

The algorithms have different identified weaknesses and strengths:

- ACC: The performance of the ACC depended strongly on the chosen time gap between the vehicles. If the time gap was great enough, the algorithm could avoid the collision in any case, as one can see in Fig 2, which means that this algorithm can be a fallback solution. However, the lack of string stability property does not make it able to be a full solution, as one can see in Fig 4.
- CACC: The CACC handled the Sinusoidal scenario relatively well – which is visible on Fig 6 –, but it is visible that with higher coefficients the algorithm could result in a better performance when braking hard as one can see on Fig 2. The algorithm collides if having too high latency or jitter values.

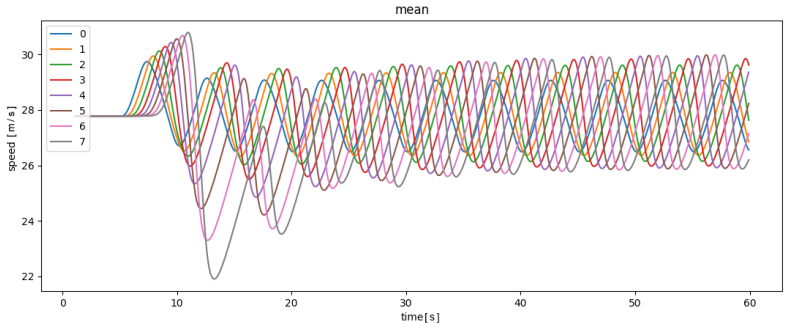

Figure 4: Sinusoidal scenario: ACC algorithm 0.3 seconds time gap

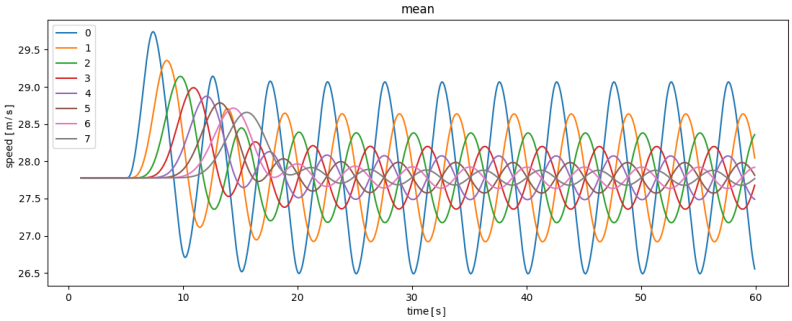

Figure 5: Sinusoidal scenario: ACC algorithm 1.2 seconds time gap

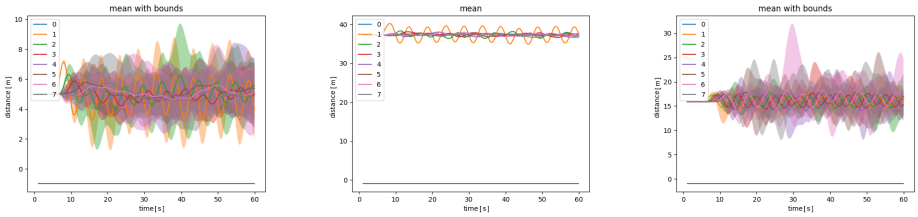

Figure 6: Sinusoidal scenario: CACC algorithm with 60%packet loss , CONSENSUS algorithm with 70% packet loss, PLOEG algorithm with 70% packet loss

- PLOEG: The PLOEG algorithm was susceptible to the latency property, which is proven by Fig 3 and Fig 6.
- CONSENSUS: The consensus algorithm was sensitive for the latency property but relatively resistive for the jitter. This is visible on Fig 3 and Fig 6.

5. Conclusions

The experience of the performed analysis is that most of the examined CACC algorithms are sensitive for the latency, especially when having a braking scenario. Most of the algorithms could handle effects in the Sinusoidal scenario, but some of them could only manage it because they had a relatively wide safety gap. If having a Sinusoidal gain, the correct answer from the algorithms is a decaying behavior, which we could see from some of the algorithms. In the Braking scenario, however, some of the algorithms did not respond hard enough to stop without colliding. This behavior is potentially caused by choosing too small coefficients, which creates a smooth response most of the time, but when having a hard brake situation, this approach does not perform well. As an experience, future algorithms could recognize some critical situations like hard braking, and they could handle these situations with different control sequences.

As a part of our future work, we plan to extend the set of examined CACC algorithms, and also to explore the circumstances where the obtained minimal QoS parameters are present due to the dense traffic. We want to implement control algorithms and new maneuvers as well based on the Artery framework [24].

Acknowledgements. The authors would like to thank András Váradi and several other experts from Commsignia Ltd. for their comments and guiding discussions.

References

[1] IEEE – Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN MAC and PHY Specifications Amendment 6: Wireless Access in Vehicular Environments, vol., no., pp. 1–51, 15 July 2010.

[2] ISO 17515-3:2019(en) Intelligent transport systems – Evolved-universal terrestrial radio access network – Part 3: LTE-V2X (2019-08).

[3] https://www.car-2-car.org/fileadmin/documents/General_Documents/C2CCC_WP_2072_RoadmapDay2AndBeyond.pdf, https://www.car-2-car.org/fileadmin/documents/General_Documents/C2C-CC_MoU_on_Deployment_Oct_2012.pdf

[4] Seiler, P., Pant, A., Hedrick, K., 2004, Disturbance propagation in vehicle strings, IEEE Transactions on Automatic Control, vol. 49, no. 10, pp. 1835–1841.

[5] M. Segata, S. Joerer, B. Bloessl, C. Sommer, F. Dressler, R. L. Cigno, 2014 IEEE Vehicular Networking Conference (VNC) – Plexe: A platooning extension for Veins, 2014, p. 53-60, DOI: 10.1109/VNC.2014.7013309.

[6] http://www.platooningensemble.eu/

[7] W. Levine, M. Athans, On the optimal error regulation of a string of moving vehicles. IEEE Transactions on Automatic Control, 11(3):355 361, July 1966.

[8] ETSI EN 302 637-2 V1.4.1 (2019-04) Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service.

[9] Dedicated Short Range Communications (DSRC) Message Set Dictionary J2735_201603.

[10] OpenStreetMap contributors, https://www.openstreetmap.org/

[11] N. T. Tangirala et al., Analysis of Packet drops and Channel Crowding in Vehicle Platooning using V2X communication, 2018 IEEE Symp. Series on Comp. Int. (SSCI), p. 281–286, 2018.

[12] H. Zhou, W. Xu, J. Chen, W. Wang, Evolutionary V2X Technologies Toward the Internet of Vehicles: Challenges and Opportunities, in Proceedings of the IEEE, vol. 108, no. 2, pp. 308–323, Feb. 2020, DOI: 10.1109/JPROC.2019.2961937.

[13] G. Naik, B. Choudhury, J. Park, IEEE 802.11bd & 5G NR V2X: Evolution of Radio Access Technologies for V2X Communications, in IEEE Access, vol. 7, pp. 70169–70184, 2019, DOI: 10.1109/ACCESS.2019.2919489.

[14] B. Tian, X. Deng, Z. Xu, Y. Zhang, X. Zhao, Modeling and Numerical Analysis on Communication Delay Boundary for CACC String Stability, in IEEE Access, vol. 7, pp. 168870–168884, 2019, DOI: 10.1109/ACCESS.2019.2954978.

[15] H. Xing, J. Ploeg, H. Nijmeijer, Compensation of Communication Delays in a Cooperative ACC System, in IEEE Transactions on Vehicular Technology, vol. 69, no. 2, pp. 1177–1189, Feb. 2020, DOI: 10.1109/TVT.2019.2960114.

[16] International Organization for Standardization (ISO, 2019).ISO 20035:2019 Intelligent transport systems – Cooperative adaptive cruise control systems (CACC) – Performance requirements and test procedures

[17] K. C. Dey et al., A Review of Communication, Driver Characteristics, and Controls Aspects of Cooperative Adaptive Cruise Control (CACC), in IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 2, pp. 491–509, Feb. 2016, DOI: 10.1109/TITS.2015.2483063.

[18] S. Zhu, D. Goswami, H. Li, Evaluation Platform of Platoon Control Algorithms in Complex Communication Scenarios, 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring), Kuala Lumpur, Malaysia, 2019, pp. 1–5, DOI: 10.1109/VTCSpring.2019.8746477.

[19] W. Chen, Y. Li, K. Zhang, T. Zheng, H. Feng, Platoon control for connected vehicles based on the V2X communications: Design and implementation, 2018 Chinese Control And Decision Conference (CCDC), Shenyang, 2018, pp. 6552–6557, DOI: 10.1109/CCDC.2018.8408282.

[20] P. Wang, B. Di, H. Zhang, K. Bian, L. Song, Platoon Cooperation in Cellular V2X Networks for 5G and Beyond, in IEEE Transactions on Wireless Communications, vol. 18, no. 8, pp. 3919–3932, Aug. 2019, DOI: 10.1109/TWC.2019.2919602.

[21] Sommer, C., Eckhoff, D., Brummer, A., Buse, D. S., Hagenauer, F., Joerer, S., Segata, M., (2019). Veins: The Open Source Vehicular Network Simulation Framework. In Recent Advances in Network Simulation (pp. 215—252). Springer International Publishing. DOI: 10.1007/978-3-030-12842-5_6

[22] P. A. Lopez et al., Microscopic Traffic Simulation using SUMO, 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Maui, HI, 2018, pp. 2575–2582, DOI: 10.1109/ITSC.2018.8569938.

[23] András Varga, Rudolf Hornig, 2008. An overview of the OMNeT++ simulation environment. In Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops (Simutools '08). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), Brussels, BEL, Article 60, 1–10.

[24] Riebl, R., Obermaier, C., Günther, H.-J., (2019). Artery: Large Scale Simulation Environment for ITS Applications. In Recent Advances in Network Simulation (pp. 365–406). Springer International Publishing. DOI: 10.1007/978-3-030-12842-5_12.