PySNS3: A Real-Time Communication Interface and Protocol for Vehicular Ad-Hoc Networks

Tong Wang and Azhar Hussain(✉)

College of Information and Communication Engineering,
Harbin Engineering University, Harbin 150001, China
wangtong@hrbeu.edu.cn, engrazr@gmail.com

Abstract. Vehicular Ad-hoc Network (VANET) being a key part of Intelligent Transportation Systems (ITS), is gaining interest among road authorities, automotive industry, network operators, and public for its numerous solutions related to road safety, comfort, and traffic efficiency. The prohibitive cost of deploying large-scale Testbeds for these solutions has attracted VANET's research community towards cooperative ITS simulation platforms. SUMO is a widely adopted microscopic traffic simulator with a Traffic Control Interface (TraCI) in various programming languages. Recently, Python has become the first priority programming language (used by many companies such as Google, Yahoo!, CERN, NASA, etc.) for data science, web development, embedded applications, artificial intelligence, information security and computation-driven scientific research. Moreover, only Python-based SUMO's TraCI has full support for the up-to-date set of commands. The lack of a Python-based cooperative ITS simulation platform, capable to communicate SUMO's TraCI with a widely adopted network simulator NS3, has led us to develop PySNS3 (a Python-based communication model for SUMO and NS3). We have tested the robustness and reliability of PySNS3 for VANET's experimentation, and compared its mobility as well as communication related simulation results with state-of-the-art NS2-mobility-model. The results have proved the reliability and robustness of the proposed PySNS3.

Keywords: Vehicular Ad-Hoc Networks
Intelligent transport system · Automobile industry · SUMO · NS3
Traffic Control Interface

1 Introduction

The traditional transportation system is becoming increasingly unable to adapt to the challenges of road safety, traffic congestion, fuel efficiency, infotainment and eco-system offered by worldwide society development and unprecedented demands for transportation. Recently, the advancement in wireless communications and automobile industry has brought many vehicular applications that can meet these challenges by deploying VANET strategies for cooperative ITS. However, the cost of deploying large-scale VANET's test beds (from Physical layer to the Application layer) in real traffic environment is prohibitively huge. Therefore VANET's research community mostly rely on cooperative ITS simulation platforms for the analysis of various protocols and interfaces before their actual deployment in road traffic. In ITS simulation study, vehicular mobility and communication modeling for VANET has some distinctive aspects. Traditionally, car manufacturers have deployed safety features such as, safety belts for seats, airbags, bumpers, and anti-lock systems for vehicles. Moreover, in recent years, automobile industry has introduced massive vehicular active safety systems based on sensors (such as radars, lasers and cameras, etc.). VANET has the potential to improve the usage of these new active safety systems [1], which will aid safety factor. It can also help the traffic department to directly manage traffic lights through the knowledge of real-time traffic situations [2]. Bidirectional interaction between network simulator and traffic simulator plays a major role in VANETs simulations [3]. For both safety and non-safety related cases, there is a demand for strong interaction between the network protocol and vehicular mobility. For example, the road traffic information can support efficient routing [4] of vehicles. Hence, there is a unique relationship between mobility and communications in VANET.

Unfortunately, for many years, the traffic simulators have never been created to communicate with network simulators and are designed to be controlled separately, with almost no interaction. However, VANET community has worked in the past few years to define efficient interfaces [5–16] between the two simulators. A communication interface of network and traffic simulator is usually categorized as Isolated, Federated, and Embedded. SUMO is a microscopic traffic simulator [17]. It is one of the most widely adopted traffic simulator for cooperative ITS simulation platforms. SUMO's TraCI support is available in various programming languages such as, Python, Java, C++ and .Net etc. During the past decade, Python has become a de-facto standard for exploratory, interactive, and computation-driven scientific research. The lack of a Python based platform, capable to communicate SUMO's TraCI with a widely adopted network simulator NS3 [18] for VANET simulations has led us to develop PySNS3.

The main contributions of this paper are as follows:

(1) The proposed PySNS3 can test VANET routing protocols (programmed in NS3 either through C++ or Python) by coupling SUMO with NS3 for cooperative ITS solutions.
(2) The proposed model supports up-do-date versions of SUMO and NS3 platforms. Which has made possible the evaluation and usage of various new features in these platforms. On the other hand, recent couplings [11,12,14,15] were not based on Python language. However, only the Python-based TraCI supports all traffic mobility related commands and maintained on daily basis [19].
(3) NS3 has a set of communication protocols for vehicular mobility as well as propagation loss models. A MAC layer model in WAVE [20] (Wireless Access

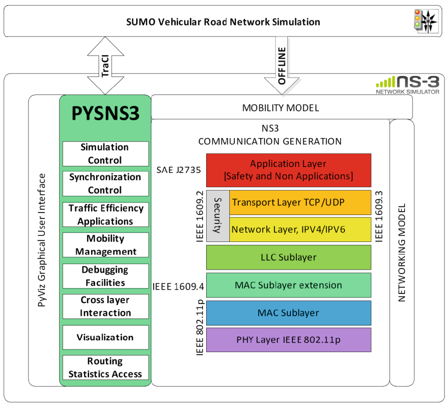
in Vehicular Environment) adapts MAC changes according to mobility of vehicles. However, as per best of our knowledge, online dynamic mobility for nodes in WAVE project was not studied and tested before. Moreover, PySNS3, supports transformation between network-coordinates and geo-coordinates, and vice versa especially in the case of Geographic Routing Protocols as described in [21] for VANET. We have compared VANET's simulation results of PySNS3 with the state-of-the-art NS2-mobility-model [22] in NS3. For this purpose, firstly, we generated an offline-trace-file of urban traffic environment of Harbin City in China. Secondly, we used this offline-trace-file by NS2-mobility-model for performance analysis of WAVE architecture along with GPSR routing protocol to get important network statistics, such as WAVE Packet Delivery Ratio (PDR), MAC-PHY OverHead, and average routing GoodPut. Thirdly, we deployed PySNS3 to manage the dynamic communication between SUMO and NS3 for the same experiment to get one to one comparison of stated statistics in offline and online approaches. Finally, we have compared the effect of varying maximum speed limitation on the vehicular statistics such as WAVE PDR, CO_2 emissions and Fuel consumption as well. The simulation results proved that the proposed PySNS3 communication model can offer online coupling between SUMO and NS3 for cooperative ITS simulations in a much reliable and robust way. Figure 1(a) shows architecture block diagram of PySNS3. The rest of the paper is organized as follows. Section 2 briefly describes the architecture of PySNS3 and simulation approach, Sect. 3 presents the procedure for performance evaluation of PySNS3. Section 4 illustrates the simulation results of the performance evaluation procedure discussed in Sect. 3. Finally, Sect. 5 draws the main conclusions derived from this work.

2 Description of Proposed PySNS3

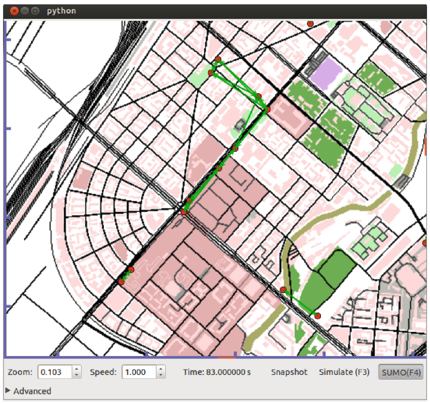
PySNS3 works cooperatively with PyViz to handle the communication between NS3 and SUMO as shown in Fig. 1(a). In NS3, PyViz is a live simulation visualizer, which means that it uses no trace files. It is the most useful tool for debugging purposes, i.e. to figure out if the mobility models are same as expected, where packets are being dropped, etc. PySNS3 exploits the power of Python language to offer robustness as well ease in developing VANET simulations. PySNS3 communication model has made SUMO an optional feature in PyViz GUI as shown in Fig. 1(b). The main idea behind proposed model is the integration of PySNS3 with NS3 to get access to its simulation state and offer dynamic vehicular mobility coupling with SUMO. It provides simulation as well as synchronization control. The performance evaluation of different routing protocols as well as traffic efficiency applications can also be incorporated in PySNS3. It also provides cross layer interaction support, since it has complete access to various layers of network nodes of NS3. The simulation scenario scene from SUMO can be brought inside the PyViz GUI for better understanding and development of different ITS applications. The routing statistics access can offer efficient solutions to the vehicular rerouting. In other words, it can dynamically give the analysis of the relation between Road traffic routing and Network protocol routing.

In order to start SUMO TraCI from PyViz GUI, an optional PySNS3's SUMO launch button (SUMO (F4)) is shown in Fig. 1(b). Moreover, the vehicles are represented by traditional nodes of NS3 and green arrows portraying the in-simulation communications. The process flow of PySNS3 starts by providing the visualizer implementation option to the main NS3 simulation program. At start, the visualizer-thread acquires main simulation lock and the main NS3 topology is scanned for each node and the canvas is created. All nodes are placed on the canvas. The mobility of nodes is provided by an initial-trace-file without movements (for one to one comparison with NS2-mobility-model). Another thread called Simulation-thread waits for the Simulation_GO signal, so that it can acquire the simulation lock. At this point if the SUMO launch support is enabled, then a separate thread is created, we call it SUMO-thread. A flag called SUMO_enable is set to True from its initial state (False). Python based TraCI server then starts listening at a specified port in SUMO. And then the loading of .net.xml, .rou.xml and any other additional file, starts as described in SUMO configuration file. After this, the connection to TraCI server is established. The next step sets the SUMO simulation step to 0 s. At this point the PySNS3 communication control plane waits for another global flag called SUMO_FLAG_GO to be set to True (initial is set to False). The SUMO_FLAG_GO can only be set to True by the simulation-thread. Once this flag is set to True, the SUMO-thread acquires the simulation lock and performs the simulation step. Then the retrieval or assignments of vehicular values, such as position, speed, angle, emission, fuel consumption and traffic lights control, etc. is carried out. At the same time assignment of these values to the NS3 nodes is also performed using python based NS3 getters and setters. The simulation step is incremented and the SUMO_FLAG_GO is set to False and the simulation lock is also released by SUMO-thread. If there is any car inside the SUMO platform, this thread also waits for SUMO_FLAG_GO to be set to True to perform another TraCI simulation step. Otherwise, the TraCI connection is closed and flushed.

The visualizer-thread waits for the simulation start button to initiate the cooperative simulation between NS3 and SUMO. At this instant SIMULATION_GO flag is enabled. This flag sets the Simulation-thread to acquire the simulation lock. The NS3 simulation events are performed for the specified simulation period. After that the simulation-thread releases simulation lock. Meanwhile the visualizer-thread updates the GTK events. The release of simulation lock from Simulation-thread enables the SUMO_FLAG_GO to True only if the SUMO_ENABLE is already in True state. Simulation-thread then waits for the Simulation_GO signal from visualizer to again perform the operations. As SUMO-thread gets the SUMO_FLAG_GO flag to True, it acquires the simulation lock and performs the respective tasks for the same time period. Later on, SUMO-thread releases the simulation lock and sets the SUMO_FLAG_GO to False. And then the visualizer update model lock is acquired and if the SUMO_FLAG_GO is already set to False, then it performs the periodic updates,



(a) The basic architecture of PySNS3



(b) Embedding SUMO's mobility option in PyViz

Fig. 1. The basic architecture of PySNS3 and its SUMO-NS3 mobility coupling (Color figure online)

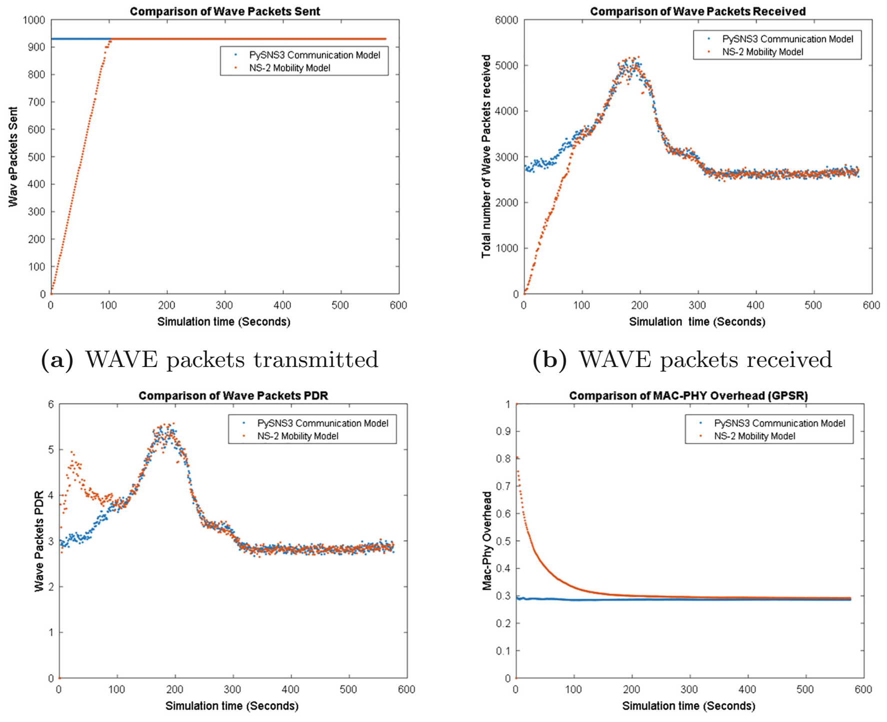
otherwise the update model lock is released and the visualizer lock is acquired to update the nodes on the canvas for their respective positions, taken from NS3. Traditional PyViz's arrows drawing and flow bitrates are also performed. The visualizer plugin is then called upon and the lock is released and then again the Simulation-thread acquires simulation lock. The whole process in this way gets synchronized, which not only enables the robustness of the coupling process between SUMO and NS3, but it also makes the debugging of the mobility related handling much convenient for VANET simulations.

3 Performance Evaluation of PySNS3

The reliability and validity of the proposed scheme is tested by comparing the results obtained by online-mobility of PySNS3 and offline-mobility of state-of-the-art NS2-mobility-model of NS3. The first step is related to the comparison of simulation results of WAVE PDR, MAC-PHY overhead, and average routing GoodPut due to the mobility inputs to NS3 simulator from SUMO through; (i) PySNS3 and (ii) NS2-mobility-model. The purpose of this step is to validate the performance of WAVE architecture in NS3 along with GPSR routing protocol under a City mobility scenario ($5369 \times 4092\,\text{m}^2$ area of Harbin, China). The scenario consists of 93 vehicles traversing random trips on Harbin road network (as shown in Fig. 1(b)) having state-of-the-art Krauss car following model implemented in SUMO. The simulation time is set to 576 s. In performance evaluation scenario, we have considered a Dedicated Short Range Communications (DSRC) broadcast network where each DSRC node has a continuous access to a 10 MHz control channel (CH) for all traffic. All vehicles transmit a 200-byte safety message at 10 Hz with data rate of 6 Mbps. All vehicles also attempt to route additional 64-byte packets at an application rate of 2.048 Kbps to one of 10 other vehicles, selected as sink vehicles. The routing protocol is GPSR. The antenna height of each vehicle is 1.5 m. Transmit power is set to 10 dBm. Regarding the radio propagation models, a combination of the Nakagami probabilistic model (in order to model Multi-Path Fading), and the Two Ray Ground Reflection Model (representing the exponential decay of signal power over distance) is used. The calculation of parameters like WAVE PDR, MAC-PHY OverHead and average routing GoodPut is performed as described in [20]. And finally, in the second step the vehicular statistics like CO_2 emissions and Fuel consumption during each simulation time along with a combination of Friss fading model and Two Ray Ground Reflection loss model are also acquired from SUMO during the NS3 simulation. Moreover, in this step the effect on WAVE PDR by varying maximum speed limits in SUMO through PySNS3 is also evaluated.

4 Simulation Results

The mobility input can be either given offline through various mobility models in NS3, or by direct coupling NS3 with SUMO. In both cases the output should cor-

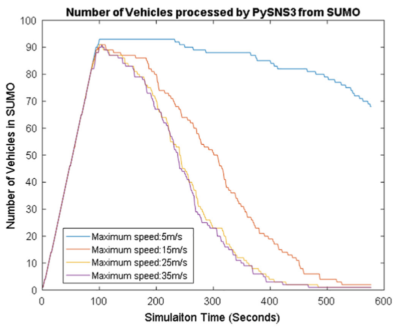


(a) WAVE packets transmitted **(b)** WAVE packets received

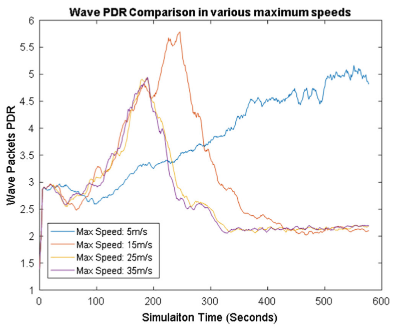
(c) WAVE packets PDR **(d)** MAC-PHY Overhead

Fig. 2. Comparison of PySNS3 with NS2-mobility-model for WAVE simulation results

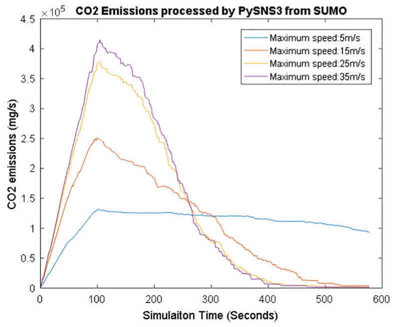
respond to the input mobility. In order to evaluate and validate the performance of PySNS3 with that of NS2-mobility-model, we have performed experiment with specification described in Sect. 3. Basic Safety Message (BSM) packets are sent at regular intervals. BSMs are transmitted assuming the WAVE Short Message Protocol (WSMP), whereas non-BSM data packets are relayed by using GPSR (IP-based routing) protocol. Additionally, to validate the comparison, we have setup the number of WAVE packets for the first 103 s differently for both models as shown in Fig. 2(a). It is also clear that in the case of PySNS3 Communication model each vehicle starts sending WAVE packets with an interval of 0.1 s, right at the start of simulation (i-e. from 0th seconds). Which means each vehicle will send 10 WAVE packets every second and since the number of vehicles is 93, so a total of 930 WAVE packets are sent each second throughout the simulation time (576 s). Comparatively, in NS2-mobility-model the WAVE packets are generated until 103 s as shown in Fig. 2(a). The Fig. 2(b) shows corresponding number of WAVE packets successfully received. The WAVE PDR is shown in Fig. 2(c). It is clear from Fig. 2(c) that the WAVE PDR in both models follows the same pattern after 103 s. Since the vehicles are moving in both models so the respective



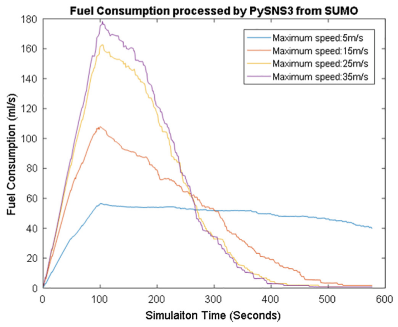
(a) Number of vehicles that have not reached destination



(b) Wave packets PDR for various speeds



(c) CO_2 emissions in mg/s



(d) Fuel Consumption in ml/s

Fig. 3. Performance evaluation results of PySNS3 for SUMO's vehicular and NS3's WAVE PDR statistics

PDR at each simulation second depends on the position and velocity values of vehicles. The respective MAC-PHY OverHead with GPSR protocol is shown in Fig. 2(d). The average routing GoodPut for NS2-mobility-model and PySNS3 is 0.4791 Kbps and 0.4898 Kbps respectively. Figure 3(a) shows the effect on number of vehicles that have not reached their destinations by setting the maximum speed of each vehicle to 5 m/s, 15 m/s, 25 m/s and 35 m/s through PySNS3.

It is clear that for slow speed (5 m/s) the number of vehicles in simulation that have not reached their destinations, at the end of simulation time (576 s) is more than that of 15 m/s to 35 m/s. The Fig. 3(b) shows corresponding effect on WAVE PDR with various speeds under a combination of Friss fading model and Two Ray Ground Reflection loss model. In the 5 m/s scenario, most of the vehicles (about 75%) are still traveling at time 576 s, which roughly corresponds to the situation at 220 s for 15 m/s speed case. The maximum speed limitations corresponding to 25 m/s and 35 m/s have not shown much difference in terms of WAVE packets PDR. Finally, Fig. 3(c) shows the effect of varying max speed on the average vehicular CO_2 emissions. Average fuel consumption (with every vehicle having emission class HBEFA3/PC-G-EU4) during each simulation time is displayed in Fig. 3(d). It is revealed that the fuel consumption increases as the maximum allowable speed for each vehicle is increased.

5 Conclusion

The proposed communication model PySNS3 is a python based platform and it can provide dynamic coupling between SUMO and NS3 in a much reliable and robust way. The proposed scheme has opened a new way towards better understanding and simulation of cooperative ITS applications and it can also be seen as a capability enhancement in the live visualization support of NS3. The performance of proposed model is tested and analyzed for both mobility and communication scenarios. In future, we will use PySNS3 for vehicular real-time rerouting under different VANET routing protocols and traffic congestion scenarios to analyze the impact of traffic congestion on the performance of routing protocols.

Acknowledgment. This paper was supported by the National Natural Science Foundation (61102105), the Engineering and Physical Sciences Research Council of UK (EPSRC) (Funding number EP/N01300X/1), the Harbin Science Fund for Young Reserve Talents (No. 2015RAQXJ008), Key Program for International S&T Cooperation Projects of China No. 2015DFG12150.

References

1. Tong, W., Jiyi, W., He, X., Jinghua, Z., Charles, M.: A cross unequal clustering routing algorithm for sensor network. Measur. Sci. **13**(4), 200–205 (2013)
2. Younes, M.B., Boukerche, A.: An efficient dynamic traffic light scheduling algorithm considering emergency vehicles for intelligent transportation systems. Wirel. Netw. J. Mobile Commun. Comput. Inf. (2017). Springer, New York. https://doi.org/10.1007/s11276-017-1482-5

3. Rondinone, M., Maneros, J., Krajzewicz, D., Bauza, R., Cataldi, P., Hrizi, F., Gozalvez, J., Kumar, V., Rockl, M., Lin, L., Lazaro, O., Leguay, J., Harri, J., Vaz, S., Lopez, Y., Sepulcre, M., Wetterwald, M., Blokpoel, R., Cartolano, F.: iTETRIS: a modular simulation platform for the large scale evaluation of cooperative ITS applications. Simul. Model. Pract. Theory **34**, 99–125 (2013)
4. Tian, R., Zhang, B., Zheng, J., et al.: A new distributed routing protocol using partial traffic information for vehicular ad hoc networks. Wireless Netw. **20**, 1627–1637 (2014). https://doi.org/10.1007/s11276-014-0699-9
5. Bononi, L., Di Felice, M., D'Angelo, G., Bracuto, M., Donatiello, L.: MoVES: a framework for parallel and distributed simulation of wireless vehicular ad hoc networks. Elsevier Comput. Netw. **52**(1), 155–179 (2008)
6. Gorgorin, C., Gradinescu, V., Diaconescu, R., Cristea, V., Iftode, L.: An integrated vehicular and network simulator for vehicular ad-hoc networks. In: Proceedings of the European Simulation and Modelling Conference (ESM), pp. 1–8 (2006)
7. Wang, S.Y., Chou, C.L.: NCTUns tool for wireless vehicular communication network researches. Elsevier Simul. Model. Pract. Theory **17**(7), 1211–1226 (2009)
8. Killat, M., Hartenstein, H.: An empirical model for probability of packet reception in vehicular ad hoc networks. EURASIP J. Wireless Commun. Netw. **2009**, 721301 (2009)
9. Multiple Simulator Interlinking Environment (MSIE) for C2CC in VANETs. http://www.cn.uni-duesseldorf.de/projects/MSIE
10. Mangharam, R., et al.: GrooveSim: a topography-accurate simulator for geographic routing in vehicular networks. In: Proceedings of the 2nd ACM International Workshop on Vehicular Ad hoc Networks (VANET 2005), September 2005
11. Piorkowski, M., Raya, M., Lugo, A.L., Papadimitratos, P., Grossglauser, M., Hubaux, J.-P.: TraNS: realistic joint traffic and network simulator for VANETs. In: Proceedings of the ACM SIGMOBILE Mobile Computing and Communications Review, January 2008
12. Pigne, Y., Danoy, G., Bouvry, P.: A platform for realistic online vehicular network management. In: Proceedings of IEEE GLOBECOM Workshops, pp. 595–599 (2010)
13. Sommer, C., German, R., Dressler, F.: Bidirectionally coupled network and road traffic simulation for improved IVC analysis. IEEE Trans. Mobile Comput. **10**(1), 3–15 (2011)
14. The iTETRIS project. http://www.ict-itetris.eu/10-10-10-community/
15. VSimRTI. https://www.dcaiti.tu-berlin.de/research/simulation/
16. Wu, H., Lee, J., Hunter, M., Fujimoto, R., Guensler, R.L., Ko, J.: Efficiency of simulated vehicle-to-vehicle message propagation on Atlantas I-75 corridor, transportation research record. J. Transp. Res. Board **1910**, 82–89 (2005)
17. Behrisch, M., Bieker, L., Erdmann, J., Krajzewicz, D.: SUMO: simulation of urban mobility: an overview. In: Proceedings of the Third International Conference on Advances in System Simulation (SIMUL 2011), pp. 63–68 (2011)
18. NS3, The network simulator NS-3. http://www.nsnam.org/
19. Interfaces by Programming Languages. http://sumo.dlr.de/wiki/TraCI
20. NS3 WAVE Model. https://www.nsnam.org/docs/models/html/wave.html
21. Wang, T., Cao, Y., Zhou, Y., Li, P.: A survey on geographic routing protocols in delay/disruption tolerant networks. Int. J. Distrib. Sensor Netw. **2016**, 1–12 (2016). Article ID 3174670
22. NS2 mobility Helper. https://www.nsnam.org/docs/models/html/mobility.html