Coupling Microscopic Mobility and Mobile Network Emulation for Pedestrian Communication Applications *

Matthias Rupp, Stefan Schuhbäck, and Lars Wischhof

Dept. of Computer Science and Mathematics
Munich University of Applied Sciences
Munich, Germany
mrupp@hm.edu, stefan.schuhbaeck@hm.edu, lars.wischhof@hm.edu

Abstract

Network emulation is a well-established method for demonstrating and testing real devices and mobile apps in a controlled scenario. This paper reports preliminary results for an open-source extension of the CrowNet pedestrian communication framework. It enables the interaction between simulated and real devices using the emulation feature of OMNeT++. The interaction is handled by several OMNeT++ modules that can be combined to match different use-cases. Initial timing measurements have been conducted for an example application which creates decentralized pedestrian density maps based on pedestrian communication. The results indicate that the approach is feasible for scenarios with a limited number of pedestrians. This limitation is mainly due to the real-time simulation requirements in coupled emulation.

1 Introduction

In Intelligent Transportation Systems (ITS) and related areas, wireless communication of vehicles and infrastructure is of fundamental importance and has been a major focus of research for more than two decades. Vehicular communication protocols and suitable communication models and simulation tools have been developed and are widely used. Communication of and with pedestrians, despite often being mentioned as a possible "X" in Vehicle-to-X (V2X) communication, did not receive that much attention. This changed in recent years, when the potential of pedestrian communication for safety and protection of human lives received more attention, e.g. suitable message formats such as the Vulnerable Road User (VRU) Awareness Messages (VAM) [1] were standardized and are considered in industry forums such as the 5G Automotive Association (5GAA). In parallel, also the research in this area gained momentum, e.g. to evaluate the impact of VAM generation rate adaptation on the awareness of VRUs [6].

When these techniques are deployed in future, it is very likely that the smartphone of a pedestrian will send these and other messages, i.e. it will be the main communication device of a pedestrian – comparable to the role that an OnBoard Unit (OBU) has for vehicles. Thus, the research community needs to develop mobile apps for pedestrian communication, test them in realistic settings and must be able to demonstrate the benefits of the applications. As a first step to enable the two latter ones, this paper presents a straight-forward extension of the OMNeT++ based open-source framework CrowNet, which allows to run a pedestrian communication app on an Android smartphone within an emulated pedestrian communication scenario. CrowNet itself

---

combines several public OMNeT++ models (see Sec. 3.1) to simulate the interaction between mobile communication and pedestrian mobility.

Our main motivations for developing the extension are threefold: a) testing mobile applications in a reproducible pedestrian communication scenario, e.g. within a Continuous Integration (CI) pipeline, b) demonstrating the usefulness of a mobile app based on pedestrian communication, c) enabling user-studies. While we assume that the framework is suitable for a wide range of applications, we illustrate its usage on the example of a mobile app generating pedestrian density maps.

Contributions: The main contributions of this paper are the following: 1. a novel open-source extension of the CrowNet OMNeT++ framework which allows to test and demonstrate pedestrian communication based apps on real smartphones, 2. illustrating the concept using an example-application which generates local pedestrian density maps, 3. preliminary results proving the feasibility of the proposed approach.

2 Related Work

While we are not aware of any other public open-source framework for network emulation specifically for mobile apps for pedestrians, related work exists regarding several aspects.

2.1 Network Emulation

Emulation test beds for mobile apps have been investigated by several research projects: Hetu et.al. developed a tool called Similitude [4], which couples the road traffic simulator SUMO with several Android mobile device emulators and (optionally) the open-source network simulator ns-3 to evaluate intelligent transportation systems. Although their concept of coupling Android devices and network simulation is somewhat similar to our approach, the CrowNet emulation presented here is designed for pedestrian communication and therefore includes specific pedestrian mobility models (see Sec. 2.2 and 3.1). A different approach for testing Android applications was presented in [14], where a hybrid emulation test bed specifically for military applications was presented. However, it does neither include a detailed model for 4G/5G cellular communication nor a detailed pedestrian mobility model.

The general aspect of testing real devices and applications within test beds applying network emulation has also been considered multiple authors: E.g., in [11], Saki et. al. use ns-3 with the LENA LTE model to perform system-level network emulation of a LTE network and demonstrate that it can be used to predict the performance of VoIP applications in real LTE networks. More recently, Nardini et. al. published results on using the system-level network simulator Simu5G in an emulation test bed for multi-access edge computing [8].

2.2 Pedestrian Mobility Simulation

A realistic modelling of the pedestrian mobility has a significant influence on the observed properties of a mobile network [3]. While simplified mobility models for pedestrians are available in most traffic simulation systems, an accurate microscopic modelling requires specific models such as the Social Force Model (SFM) [2] or the more advanced Optimal Steps Model (OSM) [5, 16] Therefore, several pedestrian and crowd simulation frameworks have been implemented. The open-source crowd simulator Vadere [5] is widely used and has been validated by several experimental studies. It is used for pedestrian mobility simulation in this paper.
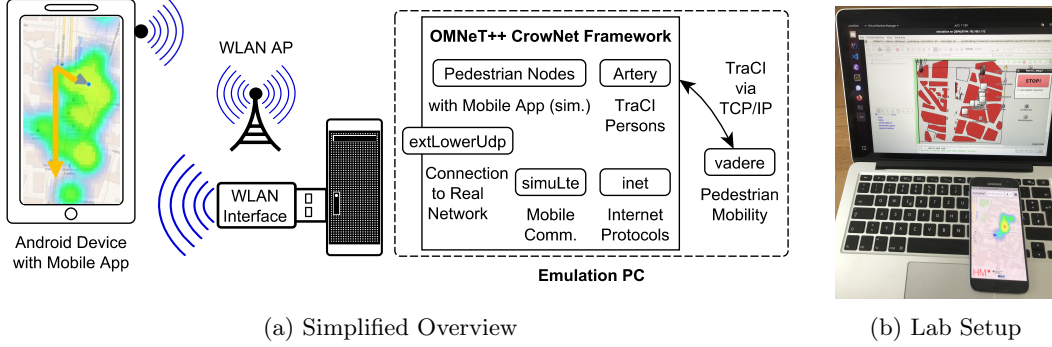
(a) Simplified Overview   (b) Lab Setup

Figure 1: System architecture: Emulation environment for pedestrian communication.

3   System Architecture

The basic idea is illustrated in Fig. 1a: One or more mobile devices are connected via local WLAN to a PC which executes the emulation environment.[1] Packets sent by the real mobile device are captured by the network interface of the emulation PC and injected into the emulation environment (via the *extLowerUdp* module, as explained in more detail in Sec. 4). Packet injection is performed by a place-holder of the real device within the simulation. Thus, packets are received and processed by other (simulated) pedestrians within communication range. Vice versa, packets received by the real-device's place-holder are sent via WLAN to the real mobile device and processed there. Since a realistic mobility simulation is of high importance, we use Vadere (see Sec. 2) as a mobility provider in the coupled emulation.

The next subsections first present the basic approach for coupling pedestrian mobility simulation with mobile network simulation. Afterwards, an example application is outlined.

3.1   Coupling Pedestrian Mobility and Mobile Network Simulation

CrowNet[2] is an open-source simulation framework that combines multiple simulators from the communication and mobility domain. It allows the simulation of scenarios where the interaction between these domains is an integral part of the simulation study. For the communication domain, the discrete event simulator OMNeT++ (version 6.0 Preview 11) is used in conjunction with the widely-used models INET (version 4.3.2), Artery [10], SimuLTE [9]/Simu5G [8]. For the mobility domain, SUMO [7] as well as the crowd dynamics framework Vadere [5] (see Sec. 2) are supported.

To connect the simulators, CrowNet uses the Traffic Control Interface (TraCI) as the inter-process communication to interlock the simulation loops of the selected simulators. This synchronization allows the exchange and modification of simulation state between the mobility and communication domain. The initial integration ([12]) was based on Veins [15]. However, the current TraCI connection management uses Artery to harmonize the implementation between Vadere and SUMO nodes.

The example scenario presented in this paper uses broadcast LTE-A side link communication

---

[1]We assume that the data rate available on the local WLAN link is high and that the additional delay is low compared to the delay caused by pedestrian communication and that it can therefore be neglected.

[2]Sources are publicly available at `https://github.com/roVer-HM/crownet/tree/emulation_omnetsummit21`, general information on the CrowNet framework can be found at `https://crownet.org`

(PC5 interface) to disseminate pedestrian density maps. For modelling pedestrian mobility, the Optimal Steps Model (OSM), provided by Vadere, is applied. The OSM model uses navigation fields which encode the shortest geodesic distance from the current position to a pre-selected target while modelling each step [5].

3.2   Use-Case Example: Decentralized Pedestrian Density Maps

Emulation based test beds are helpful to test real application in network situation which are not easy to reproduce in real life. As an example, we use a decentralized application that provides pedestrian density information for the local area in the form of a Decentralized Pedestrian Density (DPD) map.
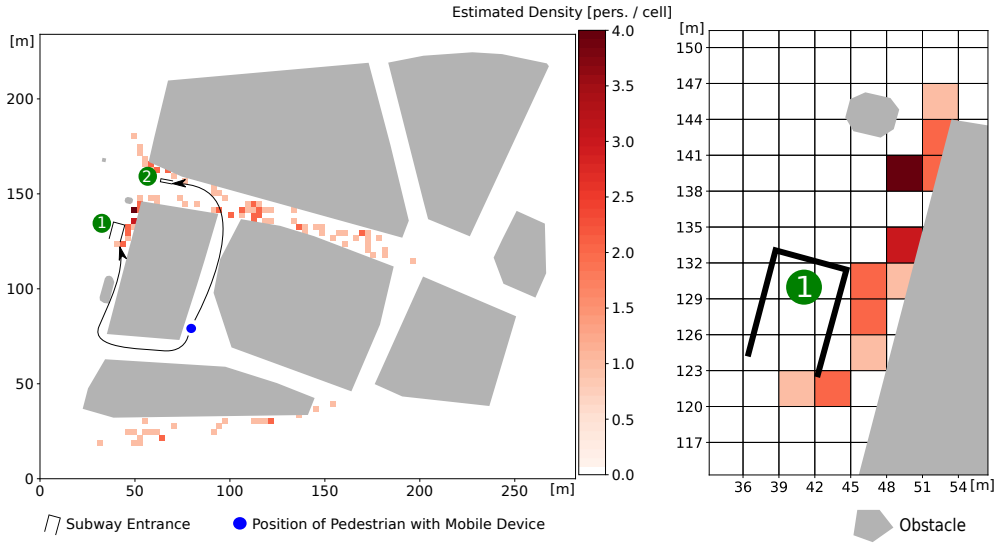

Figure 2: Left: DPD map (based on [13]). The highlighted pedestrian (blue dot) can select an alternative route. Right: Enlarged view of target area. Color indicates estimated density.

DPD maps allow the creation of decentralized awareness of the surrounding area which is an important context parameter needed for multiple urban centric mobility applications such as high crowed density detection and avoidance (e.g. to reduce Covid-19 infections), route and capacity planing for public transportation or individual real time route planing.

DPD maps are created by aggregating periodically broadcasted position beacons from neighboring nodes into a two dimensional density map of the surrounding area (Fig. 2). These maps are re-broadcasted to share the local density data with neighboring nodes, allowing them to merge this view with there own perception and create a larger area of awareness [13].

The decentralized creation and dissemination approach makes it hard to test the application on real hardware. To get meaningful scenarios, multiple devices have to be used by real pedestrians in a real urban setup. These tests are time and resource expensive. On the other hand, only testing the scenario in a simulated environment does not take into account the effects of real hardware. Thus, using real devices within the OMNeT++ emulation testbed allows to have the advantages of real devices but still achieves reproducibility and scalability though simulation.
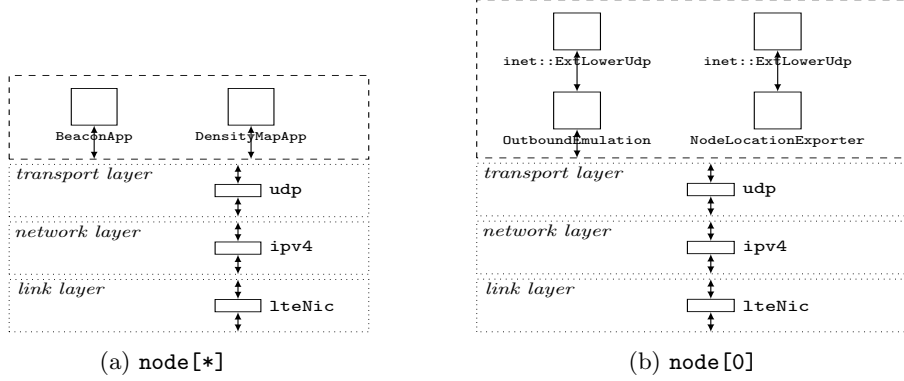
(a) `node[*]`   (b) `node[0]`

Figure 3: Structure of a generic node and `node[0]`, serving as emulation bridge. The position of `node[0]` is controlled by Vadere, therefore, the InboundEmulation module is not needed.

3.2.1   DPD Android App

While the simulation model for DPD maps example was already publicly available [13], for testing the proposed emulation framework, an identical application had to be implemented on a real Android smartphone. It was implemented in Kotlin using the usual Android development tools. This Android app calculates and visualizes the pedestrian density in the surrounding area. This is achieved by exchanging location messages, serialized as Protocol Buffers, between participating devices.

For data dissemination in a local area, the Android application currently cannot use LTE-A sidelink communication via PC5 interface, since the available commercial smartphones do not implement this mode of cellular communication, yet. Therefore, the Android app uses broadcast messages sent via WLAN for local communication as a workaround.

Density maps consist of rectangular cells, e.g. 3m × 3m. The real device obtains its local position using the Android Location Services in form of WGS84 coordinates. In order to simplify the mapping of the current position to a DPD cell, a Universal Transverse Mercator (UTM) projection is applied. Each cell then is identified by the UTM coordinates of its top-left corner.

When two real devices communicate, the WGS84-based location is converted to UTM, encoded as DPD location data and sent in form of a LocationMessage via UDP/IP/WLAN broadcast to the second device.

4   Network Emulation for Mobile Apps

To meet our goal of connecting Android applications running in a real network to the simulation, we developed a bridge that forwards and converts the position beacons between the real and emulated network. The offset between real- and simulated world coordinates can be specified in the modules' configurations. Each bridge-module connects one real device to the simulation. The bridge-module is located in the application layer of a specific node, which now represents the real device inside the simulation. In scenarios where one device is connected, the bridging is handled by `node[0]` (Fig. 3). The node does not need any applications for the dissemination of beacons and density maps because this task is handled by the coupled device. The bridge-module consists of three modules:

OutboundEmulation receives the position beacons from `node[1..n]` and forwards them to the mobile application, applying necessary coordinate projections. The module consists of 2 sub-modules: The `DensityMessageHandler` receives position beacons and converts them to the required protocol buffers format for the mobile application. `DensityMessageHandler` is connected to an `inet::ExtLowerUdp` module which sends the data over the real network to the connected devices via an UDP socket.

NodeLocationExporter exports the simulated position of `node[0]` to the mobile application. This allows to mock the device's location using mobility data from Vadere. Again, the `inet::ExtLowerUdp` module is used to send the data to the real-device(s).

InboundEmulation receives protocol buffers location beacons generated by the mobile application and sets the position of `node[0]` to the position of the real device. The beacons are received by a `inet::ExtLowerUdp` socket which is connected to the `InboundEmulation` module. The data is serialized to a simple string-based format.

The framework currently supports two modes for setting the location of the real mobile devices:

1. If the NodeLocationExporter module is used (see Fig. 4a), the app's location is controlled by Vadere. This allows to test location-aware applications without the need of actually moving the device which makes these tests reproducible and independent of error sources such as the GPS reception.If the NodeLocationExporter module is used (see Fig. 4a), the app's location is controlled by Vadere. This allows to test location-aware applications without the need of actually moving the device which makes these tests reproducible and independent of error sources such as the GPS reception.

2. If the InboundEmulation module receives mobility information from the real mobile application, it moves its place-holder (e.g. `node[0]`) accordingly (Fig. 4b). This variant is mainly attractive for interactive demonstrations where the real device is moved within a real urban scenario, which is replicated in the coupled simulation.
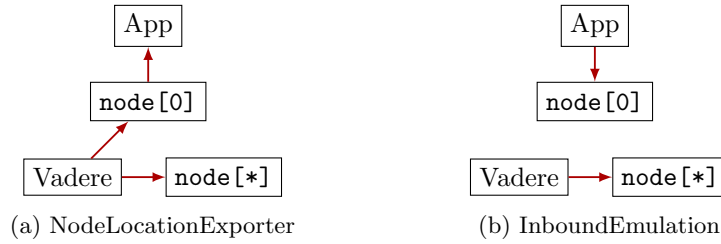

(a) NodeLocationExporter   (b) InboundEmulation

Figure 4: Flow of mobility data between Vadere, the simulated nodes, and the coupled app.

4.1 Scheduling

To satisfy the timing requirements of both the real and simulated nodes, it is important that the simulation time is synchronized with the wall-clock time. The `inet::RealTimeScheduler`, which is required by the `inet::ExtLowerUdp` module, performs this synchronization. However, we observed that due to the fact that the scheduler starts before the connection to
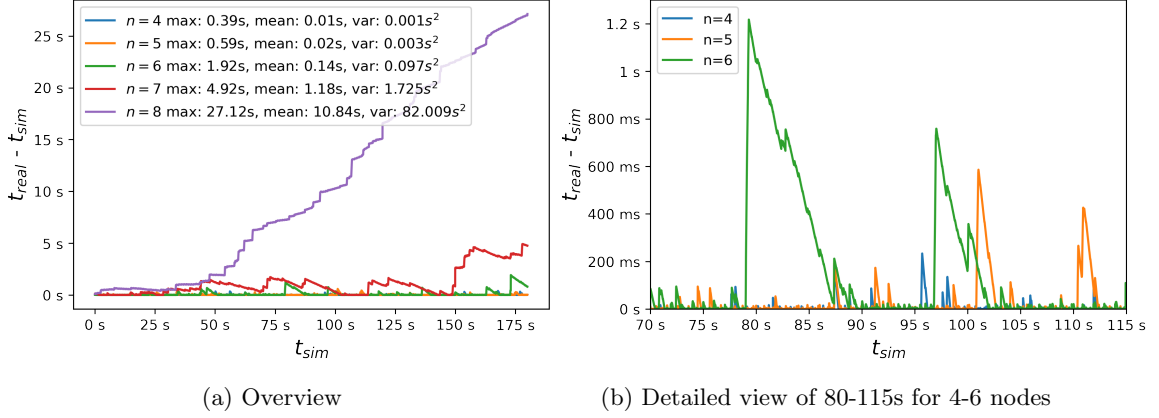
(a) Overview

(b) Detailed view of 80-115s for 4-6 nodes

Figure 5: Observed offset of real-time and simulation time ($t_{real}-t_{sim}$) depending on the number of simulated nodes n. On a low-end machine, a value of $n > 6$ leads to offsets exceeding 5s.

Vadere is established, there is an initial delay between t_{sim} and t_{real}. Therefore we created a custom scheduler class called `crownet::EmulationScheduler`, which inherits from `inet::RealTimeScheduler` and includes a method to synchronize the simulation time back to the wall-clock time as well as some methods for timing analysis. During testing, it showed that the most important factor whether or not the simulation time can keep up with the wall-clock time is the number of simulated nodes (see Sec. 5).

5   Measurement Results

As a proof-of-concept, an urban simulation scenario with one eNodeB and a varying number of UEs/nodes was configured. Tab. 1a [3] lists the most relevant parameters, the complete sources are available in the CrowNet repository, see Sec. 3.1. To measure the timing behavior depending on the numbers of simulated nodes, both the real-time and simulation time were recorded at a 10 ms interval. Then, the offset between real-time and simulation time $t_{real} - t_{sim}$ was calculated. All measurements were performed on a low-end virtual machine (2 CPU-cores, 12GB RAM, Ubuntu Linux).

Fig. 5a shows the a plot for scenarios with 4 to 8 nodes. Synchronized emulation is possible for up to 6 nodes on this machine. Fig. 5b reveals that although scenarios with $n < 7$ in general have a low offset $t_{real} - t_{sim}$ near zero, there are periods where the simulation time lags behind the real-time for several seconds. The lag increases with increasing number of nodes. However, the observed lags of $t_{sim} - t_{real} < 5s$ are negligible for our use-case.

6   Conclusions

The presented extension of the CrowNet framework allows network emulation for mobile apps based on pedestrian communication. However, emulation currently is only feasible for scenarios with a very limited number of simulated nodes. For a low-end machine, the offset between real

---

[3]For full simulation setup see https://github.com/roVer-HM/crownet/tree/emulation_omnetsummit21/crownet/simulations/emulation_omnetsummit21

time and simulation time can only be kept low for scenarios with up to six nodes. Using our emulation strategy with larger scenarios would require a performance optimization of the simulation, for example a lower level of detail for the system-level mobile communication simulation or parallel execution. Despite the limited scalability, the presented framework is a useful tool for small-scale tests or demonstrations with real hardware devices.

In future, we will upgrade the communication model to 5G (using Simu5G) and perform in-depth measurements to analyze the system in detail. We will especially try to find the main cause of the performance problems using a profiler tool. The goal is also to integrate more data formats in our emulation framework to support more types of mobile applications.

Table 1: Most relevant parameters used for the emulation Config: pedmap_emulation.

(a) Pedestrian Mobility

Symbol	Value	Description
T_{inter}	60.0 s	pedestrians inter arrival time
N_{ped}	1..10	number of pedestrians
B	415x394 m	simulation bound
	$1.34\ \text{ms}^{-1}$	mean of ped. speed-distribution
	$0.26\ \text{ms}^{-1}$	std. dev. of ped. speed-distribution
	0.195 m	pedestrian radius
	0.45 m	ped. potential intimate space width
	1.2 m	ped. potential personal space width
	50.0	ped. potential height
	0.8 m	obstacle potential width
	6.0	obstacle potential height
	1.2	intimate space factor
	1	personal space power
	1	intimate space power

(b) Communication

Symbol	Value	Description
T_{beacon}	1.0 s	inter-transmission time beacon packets
S_{beacon}	224 B	payload of beacon packets
T_{map}	2.0 s	inter-transmission time map packets
S_{map}	1000 B	payload of map packets
B	2.6 GHz	carrier frequency
N_{RB}	20	number of resource blocks
M_{PL}	URBAN-MICROCELL	pathloss scenario (ITU-R M2135-1)
h_{UE}	1.5 m	hight of user equipment
h_{eNB}	25.0 m	hight of eNodeB
$P_{\text{TX, eNB}}$	20 dBm	transmission power of eNB (reduced)
$P_{\text{TX, UE}}$	20 dBm	transmission power of UE
D_{RLC}	UM	RLC type (UM: unacknowledged mode)
Q_{mac}	10000 B	mac queue size
$Q_{\text{rlc\_um}}$	5×10^6 B	rlc queue size

References

[1] ETSI. Ts 103 300-3 v2.1.1 (2020-11), intelligent transport system (its);vulnerable road users (vru) awareness;part 3: Specification of vru awareness basic service;release 2. Technical report, ETSI.

[2] Francesco Farina, Daniele Fontanelli, Andrea Garulli, Antonio Giannitrapani, and Domenico Prattichizzo. Walking ahead: The headed social force model. *PLOS ONE*, 12(1):1–23, 01 2017.

[3] Soren Hahn, Dennis M. Rose, Jaroslaw Sulak, and Thomas Kurner. Impact of realistic pedestrian mobility modelling in the context of mobile network simulation scenarios. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*. IEEE, 5 2015.

[4] Seth N. Hetu, Vahid Saber Hamishagi, and Li-Shiuan Peh. Similitude: Interfacing a traffic simulator and network simulator with emulated android clients. In *2014 IEEE 80th Vehicular Technology Conference (VTC2014-Fall)*. IEEE.

[5] Benedikt Kleinmeier, Benedikt Zönnchen, Marion Gödel, and Gerta Köster. Vadere: An open-source simulation framework to promote interdisciplinary understanding. *Collective Dynamics*, 4, 2019.

[6] Tomás Lara, Alexis Yáñez, Sandra Céspedes, and Abdelhakim Senhaji Hafid. Impact of safety message generation rules on the awareness of vulnerable road users. 21(10):3375.

[7] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flotterod, Robert Hilbrich, Leonhard Lucken, Johannes Rummel, Peter Wagner, and Evamarie WieBner. Microscopic traffic simulation using SUMO. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 11 2018.

[8] Giovanni Nardini, Giovanni Stea, Antonio Virdis, Dario Sabella, and Purvi Thakkar. Using simu5g as a realtime network emulator to test MEC apps in an end-to-end 5g testbed. In *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*. IEEE.

[9] Giovanni Nardini, Antonio Virdis, and Giovanni Stea. Modeling network-controlled device-to-device communications in SimuLTE. *Sensors*, 18(10):3551, 10 2018.

[10] Raphael Riebl, Hendrik-Jorn Gunther, Christian Facchi, and Lars Wolf. Artery: Extending veins for VANET applications. In *2015 International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*. IEEE, 6 2015.

[11] Sergio Sakai, Gilberto Gambugge, Ricardo Takaki, Jorge Seki, Juliano Bazzo, and Joao Paulo Miranda. Performance comparison of a custom emulation-based test environment against a real-world LTE testbed. In *Proceedings of the 2015 Workshop on ns-3 - WNS3 '15*. ACM Press, 2015.

[12] Stefan Schuhbäck, Nico Daßler, Lars Wischhof, and Gerta Köster. Towards a bidirectional coupling of pedestrian dynamics and mobile communication simulation. In *Proceedings of the OMNeT++ Community Summit 2019*.

[13] Stefan Schuhbäck and Lars Wischhof. Decentralized pedestrian density maps based on sidelink communication. In *ICC 2021 - 2021 IEEE International Conference on Communications (ICC)*, pages 1–6, 2021.

[14] Constantin Serban, Alexander Poylisher, Angelo Sapello, Yitzchak Gottlieb, C. Jason Chiang, and Ritu Chadha. Testing android devices for tactical networks: A hybrid emulation testbed approach. In *MILCOM 2015 - 2015 IEEE Military Communications Conference*. IEEE.

[15] Christoph Sommer, Reinhard German, and Falco Dressler. Bidirectionally coupled network and road traffic simulation for improved IVC analysis. *IEEE Transactions on Mobile Computing*, 10(1):3–15, 1 2011.

[16] Isabella von Sivers and Gerta Köster. Realistic stride length adaptation in the optimal steps model. In M. Chraibi, M. Boltes, A. Schadschneider, and A. Seyfried, editors, *Traffic and Granular Flow '13*, pages 171–178. Springer, 2015.