Wireless Networks
https://doi.org/10.1007/s11276-023-03432-3

ORIGINAL PAPER

Edge computing in the loop simulation framework for automotive use cases evaluation

Levente Márk Maller[1,2] · Péter Suskovics[2] · László Bokor[1,3]

Accepted: 5 June 2023

Abstract
Edge architectures provide local, decentralized services, enabling balancing network traffic and distributing hardware resources. Later, many new use cases can be implemented by combining the advantages of the edge computing concept with the services of 5G systems. One of the biggest beneficiaries of this could be the Vehicle-to-Cloud (V2C) technology, where it is necessary to efficiently process large amounts of data resulting from Vehicle-to-Everything communication (V2X) services. In specific use cases, this makes it possible to process sensor data collectively, enhanced by fusion, which promotes a more effective virtual representation of the real world. The effective implementation of these technologies is a complex task. One of the most important steps before tests on actual infrastructures with real vehicles is evaluating and validating edge cloud systems. We present a solution for this problem, the Cloud-in-the-Loop (CiL) simulation framework. It can orchestrate a real-size, telco-grade level, Kubernetes-based edge cloud infrastructure based on information gathered from a traffic simulator and performing fine-grained benchmarking and data collection. In addition to the performance analysis of the edge system, it also enables an in-depth examination of cloud-native applications serving complex automotive use cases. In this paper, we focus on presenting the developed framework and its capabilities by utilizing the system with implemented test applications, and give an example of testing QoS and QoE aspects of the edge cloud-based V2C concept.

Keywords Edge cloud · Automotive use cases · Cloud-in-the-loop simulation · Kubernetes · Cloud-native applications · 5G Telco cloud

1 Introduction

In recent years, cloud-native-based services have gained more and more recognition. The technology is already used in many areas, from information communication through web services to banking systems. Cloud-native applications are easily scalable, deployable, and flexible software packages and, due to their design, can be run in any cloud-based environment. Cloud-based technologies have also expanded into a new area called the edge computing paradigm, the most crucial feature of which is that hardware resources are distributed. As a result, the services are closer to the consumers, thereby speeding up information processing and ensuring high-performance computing for the applications that provide the services. By applying these innovative technologies, new solutions have also emerged, such as the 5G Service-Based Architecture (SBA) [1] approach, which implements the network

✉ Levente Márk Maller
lmaller@hit.bme.hu; levente.maller@ericsson.com

Péter Suskovics
peter.suskovics@ericsson.com

László Bokor
bokorl@hit.bme.hu

[1] Department of Networked Systems and Services, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics, Műegyetem Rkp. 3., Budapest 1111, Hungary

[2] Technology and Innovation, Digital Services, Ericsson, Magyar Tudósok Körútja 11., Budapest 1117, Hungary

[3] ELKH-BME Cloud Applications Research Group, Magyar Tudósok Krt. 2, Budapest 1117, Hungary

Published online: 04 July 2023

functions that build up the core network of mobile networks with cloud-native applications. With this approach, modern 5G systems can be integrated with edge cloud deployments, thereby taking benefit of the most significant advantages of the two new technologies. One of the largest European standardization organizations, European Telecommunications Standards Institute (ETSI), is also actively integrating the two systems [2]. Furthermore, the organization has already achieved several results in unifying edge cloud systems. This model is called Multi-Access Edge Computing (MEC) [3]. Many service areas will be able to utilize the opportunities created by the combination of 5G and MEC systems. One such area is Vehicle-to-Cloud (V2C) communication. The application of these technologies opens up many new use cases, such as sensor fusion applications based on collective perception. These rely on deep learning-based object detection technologies, for which the distributed resources of the MEC systems provide an excellent execution environment. An example of this is the High Definition (Local) Maps use case, which will benefit from the potential of MEC systems effectively. By processing the environmental data transmitted by the vehicles with the help of HD Maps, it is possible to implement continuously updated local maps running on edge servers. These multi-layered maps can contain various information, such as processed environmental information transmitted by vehicles and virtual representations of other traffic participants [4]. Also, use cases heavily relying on environment detection could also benefit from technologies, such as Object Pose Estimation [5], that enable the 3D modeling of objects based on camera data. Furthermore, with the help of 5G systems, a low-latency, high-data-speed communication can be provided, which ensures a stable connection between user equipments (UEs) and edge servers.

The effective implementation of these technologies is a complex problem, and many requirements must be met. The realization of such infrastructures poses multiple challenges. As V2C services rely on cloud applications, it is essential to investigate resource availability and scalability. Moreover, properly orchestrating the service-providing entities in the edge cloud environment is crucial to efficiently address the UEs' mobility. Also, the dimensioning of these networks proves to be a complex problem. Furthermore, building a real system is a costly task. Before this, it is essential to carry out tests related to the operation and performance of these systems. Test systems that can replace expensive real-life tests provide the apparent basis for this. Among others, these aspects motivated the design and implementation of the presented Cloud-in-the-Loop (CiL) simulation framework [6], which can behave accordingly and process information generated in a simulated environment. This information allows it to manage a real, integrated cloud environment using implemented algorithms and logic. Our CiL approach currently integrates a traffic simulator called SUMO [7][7] with a Kubernetes (k8s) based distributed cloud environment. With the help of the framework, the operation of edge cloud systems and cloud-native applications modeling V2C use cases can be investigated and evaluated. Using the CiL-Simulator, we have already presented preliminary research results in [6]. Since then, we have made numerous improvements to the system. We have successfully integrated the framework into a telco-grade edge cloud environment and extended the central component – implementing the whole framework's orchestration – with new features and functions executing more detailed and accurate measurements. Furthermore, we also developed new supporting application components compatible with the framework that can demonstrate the framework's functions. In the paper, we present the following major contributions:

- A Cloud-in-the-Loop simulation framework concept integrating simulation environments with real cloud deployments;
- A proposed implementation of the Cloud-in-the-Loop method on telco-grade hardware/software environment for investigating V2C use cases;
- A methodology for testing edge cloud-based object detection applications using basic QoE metrics.
- A methodology for examining QoS parameters affecting the operation of edge cloud-based applications in the framework

The following section provides an overview of related works and literature. Section 3 of the paper will give a comprehensive presentation of the CiL framework and the improvements we have made since our initial version. After that, in Sect. 4, we present the example use cases implemented in the framework. Then, in Sect. 5, we propose test measurements with a methodology that can be performed with the help of the framework. Finally, we conclude the article in Sect. 6.

2 Related works

Prior to the publication of this article, we came across several studies that discuss the possibilities and research of cloud-based vehicle communication solutions. Furthermore, we have read about several environments similar to the framework we have implemented. Some tools aim to investigate edge cloud environments and V2C technologies (or both). However, these are not primarily focused on studying these technologies in the way our solution does. The Cloud-in-the-Loop simulation framework's main

advantage is that it makes it possible to integrate various real cloud environments and applications while it enables simulating user equipments in multiple scenarios. Generally, cloud simulators aim to examine infrastructures based on predefined models and the model-based implementation of actual cloud components and functions in a simulated environment. Opposed to these methods of investigating cloud systems, our solution utilizes an actual cloud deployment and provides an opportunity to collect metrics of real-time operation. Regardless, there are numerous cloud simulators available in the literature and open-source repositories. For further details, we recommend [9], which is an excellent survey on the topic, and [10] which presents a new tool focusing on cloud energy consumption but gives a comprehensive review of existing cloud simulators. Also, for cloud-based V2X-related frameworks, we suggest [11], which presents a detailed overview of the subject. However, this section presents the tools and solutions that we found more closely related to our CiL framework. Below we provide a summary of our findings on the related tools and frameworks.dSPACE, a company that specializes in software and hardware simulation tools, also has a system [12] that implements a V2Cloud Hardware-in-the-Loop simulator. The purpose of this system is to examine and validate the entire V2N communication chain, however, based on the available information, the system focuses mainly on the examination of LTE and 5G systems. To accomplish this, an Anritsu[1] radio communication test station is used. Moreover, unlike the open-source components and APIs we use, it operates in a closed software environment. Like dSPACE, Dell, a major IT company, is researching the subject as well [13]. The company's Hardware-in-the-Loop Autonomous driving simulation system aims to test ADAS / AD solutions. It uses Amazon's AWS cloud system for cloud application testing, which works with real-world vehicle sensor data replayed with microsecond accuracy. These systems have already been theoretically studied in the 2017 publication X-in-the-Loop Test Methods for Cloud-based Vehicle Functions [14]. The article discusses in detail the challenges that the automotive environment poses to cloud systems. The article introduces the Hardware / Model-in-the-Loop approaches that build on cloud-based wireless systems. One of the main messages of the article is that the systems implementing these concepts can greatly help the examination and validation of cloud-based vehicle communication systems in the future. Test systems integrating cloud infrastructure also help in other domains, as detailed in the 2016 Wide-area control of power systems using cloud-in-the-loop feedback article [15], in which the authors examined the control of a large-scale, simulated electrical network using a Cloud-in-the-Loop test system. Recently, also a new Hardware-in-the-Loop framework was presented, the CarTest V2X Simulation Framework [16], which enables testing real V2X devices (for e.g. vehicle-to-vehicle communications) while it completely simulates the actual vehicles.

Also, several software programs are available for complete simulation testing of edge cloud systems that can be used to implement various measurements (Table 1). Mostly, these tools use only simulation for investigating cloud environments and do not necessarily implement V2X-based models and communication. Despite these limitations, they play a prominent role in cloud system modeling and simulation. Free software called OPNET [17] can be used to model computer network environments in which different measurements can be executed. For modeling more complex wired and wireless networks, the OMNeT + + framework [18] can provide a solution that can use a wide range of additional software components. Furthermore, the system can be used to model 4G LTE mobile networks, which is implemented by the SimuLTE [19] software package. However, in recent years, simulation tools specifically designed to model cloud computing and distributed systems have also become available, such as CLOUDS Lab's CloudSim software [20] or the iFogSim [21] software package, which can be used to model IoT and Fog Computing architectures. In automotive applications, however, the use of traffic simulators is essential, the best solution is the SUMO traffic simulator, and there are ready-made, implemented systems, such as the Veins [22] or the Artery [23] software package, which integrates SUMO and OMNeT + + software. It is important to highlight the previously mentioned SimuLTE's successor, the Simu5G software package [24]. The simulator allows one to study systems based on 5G New Radio (NR) radio access technology. The possibilities of Simu5G were introduced in a publication published in 2020 [24]. In the test environment presented in the article, the processes of containerized applications can be tested in an edge cloud system cooperating with 5G architecture. With the help of this emulated network, the processes of containerized applications, their network communication can be examined with conditions that also characterize 5G networks. In addition to examining the modeled architecture, simulation environments of this kind can also be of great help in the development of applications based on similar systems. Also, some tools focus better on testing V2X-related cloud operations. Such as the Telco Cloud Simulator [25, 26], which enables the investigation of various types of operations of telco clouds, but it can also support cloud performance evaluation with mobile users and V2X communication. There are also simulation frameworks for more specific tests, such as the iCanCloud that can be used to evaluate vehicular ad-hoc

[1] Anritsu official website: https://www.anritsu.com/en-gb/.

Table 1 Comparison of related simulators

Tool name and relevant publications	Edge cloud implementation	Edge cloud technology	Cloud resource management	V2X support	System observability	Real-time, responsivity	Supported applications	Vehicle simulation	Purpose
Cloud-in-the-Loop simulation framework [6]	Real	Kubernetes (k8s, k3s, etc.)	Operations and resource types provided by Kubernetes: pods, services, scalability, replica sets (redundancy management)	Yes (optional V2C)	Benchmarking hardware resources (e.g., CPU, RAM), High precision benchmarking, multi-layer metrics (e.g., Prometheus [28])	Yes	Real, containerized, edge-cloud applications (e.g., AI-based object detection)	SUMO	Integrated system (design, dimensioning, performance analysis)
CloudSim [29]	Virtual (EdgeCloudSim)	Based on Java models	Based on Java models	No	No	No	Based on Java models	No	Modeling and simulation
iFogSim [30]	Virtual	Based on Java models	Based on Java models	No	No	No	Based on Java models	No	Modeling and simulation
EdgeCloudSim [31, 32]	Virtual	Based on Java models	Based on Java models	No	No	No	Based on Java models	No	Modeling and simulation
SimuLTE/ Simu5G [24, 33, 34]	Virtual	Based on C + + models	Based on C + + models (very simple actual implementation)	Yes (cellular-based, V2C)	No	Yes (for end-to-end simulation [33])	Based on C + + models, Real MEC apps	SUMO (requires Veins/ Artery integration)	Modeling and simulation, application testing
Veins/Artery [35, 36]	Virtual (requires Simu5G integration [37, 38])	Based on C + + models (Requires Simu5G integration e.g.)	Based on C + + models (very simple actual implementation)	Yes (multiple type)	No	No	Based on C + + models, Real MEC apps (requires Simu5G integratio)	SUMO	Modeling and simulation
CarTest [16]	No	No	No	Yes (V2V, V2I)	V2X device system level observability	Yes	Real V2X applications	Virtual Test Drive (VTD) as a simulator	Application testing and performance analysis
iCanCloud [27, 39]	Virtual	Based on C + + models	Based on C + + models	Yes (optional)	No	No	Based on C + + models	Traffic flow prediction	Modeling and simulation
Telco Cloud Simulator [25, 26]	Virtual	Based on Java models	Based on Java models	Yes (traffic profile-based)	No	No	Based on Java models (application profile based)	Multiple patterns of mass movements	Modeling and simulation

network cloud architectures [27]. Comparing these tools to our solution evidently shows that the CiL framework requires capable hardware to be used, and also, any tested environment has to be adequately integrated. Furthermore, it currently doesn't support 5G functionalities. Despite its limitations, the CiL allows testing and evaluating real cloud deployments and applications, supporting detailed benchmarking of relevant, cloud-focused hardware-software systems. These features later help design, dimensioning, and performance analysis of the tested environments (e.g., products, deployment/placement configurations, etc.), enabling them to be optimized and appropriately used in real-life scenarios.

3 The cloud-in-the-loop simulation framework

The Cloud-in-the-Loop (CiL) simulation framework [6] results from several years of research and development and provides the basis for the achievements presented in this work. Based on information extracted from a simulator that models the behavior of user devices, the framework can orchestrate a closely integrated, real, distributed cloud-based environment and run and test real cloud-native applications in it. In the current construction of the framework, it is configured to examine V2C use cases; accordingly, the simulation environment is provided by a very versatile, multi-modal traffic simulation software called SUMO [7]. The orchestration of the distributed cloud-based environment is realized by Kubernetes (k8s) [40], a widely used, open-source platform, which already plays a prominent role in operating SBA-based 5G Core networks nowadays [41] and can also be used excellently for managing edge cloud systems. The CiL framework consists of three main system components (Fig. 1):

- Automotive traffic simulator: The simulation environment is realized by the microscopic and continuous multi-modal traffic simulator called SUMO. The software can model real, large-scale road networks and simulate detailed, high-precision traffic models. Furthermore, detailed information on the behavior of each simulated object and vehicle can be extracted, and their run-time configuration is also provided.
- CiL Orchestrator (CiL-O): The orchestration component is a software developed in Java, which is one of the essential elements of the framework. This is where the control of the distributed cloud-based environment is realized based on the information extracted and processed from the simulator. The Orchestrator uses the Traffic Control Interface (TraCI) [8] to establish a connection to SUMO. It enables the CiL Orchestrator to access information on the simulated objects (e.g., vehicle position, vehicle speed) in every simulation step (which is controlled by the CiL-O). Then, it processes the acquired data of the individual vehicles to orchestrate the service-providing applications in the distributed cloud environment and manages the client applications representing the vehicle-side functionalities. These functionalities enable the testing and investigation of various V2C-based automotive use cases by implementing different algorithms and logics in the Orchestrator. Furthermore, this component ensures detailed data collection of Docker-level events from every node during the measurements.
- Distributed cloud environment: The distributed cloud environment is realized by a Kubernetes (k8s) platform-based real-scale hardware cluster and interconnected with the CiL Orchestrator using the official k8s Java library [42]. The nodes of k8s that form a cluster represent the edge servers of a MEC infrastructure (e.g., n1, n2 in Fig. 1). The k8s enables cloud-native applications that implement automotive use cases to be deployed on these distributed resources. These applications provide services to the investigated vehicles (modeled by client applications) in the V2C model. According to vehicle mobility (and the implemented use case), these edge applications can also be relocated between nodes, the management of which is provided by CiL-O.

An important aspect of MEC systems is examining the effect of switching between edge resources resulting from the mobility of user devices, which affects the system's operation in many ways. In such cases, relocating applications that provide services to user devices may also be necessary, and service continuity must also be ensured. The system must also manage the redirection of the network connection giving access to the servers. In addition to relocation, the performance and operation of the system are also affected by the network and resource load, which largely depends on the number of devices served, the type of applications providing the service, and the data traffic generated. With the help of the CiL framework, based on the information extracted from the simulator, the operation of real applications can be examined, and fine-grained benchmarking and data collection can also be performed. Accordingly, many improvements have been made to the system and its supporting applications since we published our initial results in [6]:

- Telco-grade level, real distributed edge infrastructure was integrated into the framework.
- Specialized client and server application components were implemented to test UDP traffic and generate/evaluate QoS metrics.

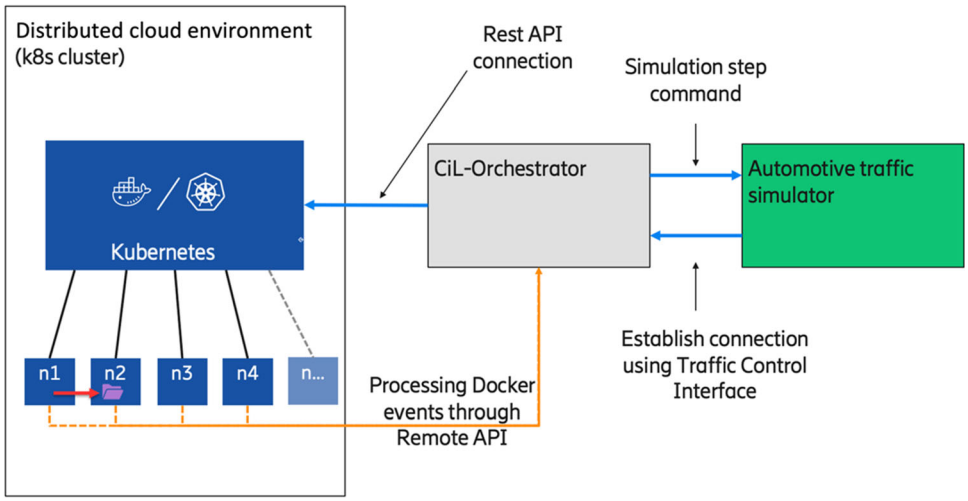

Fig. 1 The architecture of the Cloud-in-the-Loop simulation framework

- Application components were designed and deployed that model IP mobility to enable the UDP test traffic tools to directly forward packets to edge applications running on the corresponding node. Kubernetes networking no longer performs the entire traffic management as before.
- A deep learning-based application component was designed and implemented, modeling an automotive use case featuring AI-supported object detection for MEC-level sensor fusion and related applications.

3.1 Scenario definition process

SUMO provides an opportunity to implement and investigate any given traffic situation. The simulation traffic map used for presenting the framework's capabilities—which provides an excellent basis for testing edge systems—has already been implemented during previous tests [6]. The map is based on Hungary, Budapest XI. district's urban environment around Infopark. The resources (servers) of the integrated edge cloud environment are located virtually on this map. To achieve this, we defined so-called latency zones (Fig. 2) for the two edge servers of the cluster, which determine which resource serves the vehicle moving in the given position. These zones represent the limit within which their associated edge server can still fulfill the given latency requirements with acceptable reliability. In reality, the shape of these zones is affected by countless factors, such as the location of the base stations or the network structure. However, the zones created in the current simulation environment implement only one possible layout among many, but it is ideal from the point of view of testing edge cloud systems. Because the measurements carried out in the framework currently focus on the tests of the application components that model the operation of V2C use cases and the operation of the distributed system.

In our model, every vehicle represented by a client application served by a dedicated (backend) edge application running on the edge servers. According to the two zones, the two worker nodes of Kubernetes act as Edge servers, so if vehicles move in certain zones, the worker node belonging to that zone must run the backend applications that serve the vehicles. The simulation data generated by SUMO is processed by CiL-Orchestrator, which then controls the pods [43] and services [44] running on the Kubernetes cluster and the client applications running on the application server. Thus, according to the vehicles' relative position to the zones in the simulation, the CiL-O manages the backend applications' relocation, and sets the server and client applications' network configuration to ensure proper service access. During the simulations, cars follow predefined routes designed to model traffic in an urban environment.

3.2 CiL-Orchestrator

The most important task of Orchestrator is to establish and maintain a connection between the framework system's components. This is where the algorithms and logic that model the operation of edge cloud-based vehicle communication solutions are implemented. The processing of the simulation data and the related calculations are performed within each simulation step. We can examine vehicles with a specific ID in the program code, and the CiL-Orchestrator performs the operations based on the data describing the simulated objects. The position of a vehicle is stored in a variable during each simulation step. We can also examine and model migration events in edge cloud systems in the implemented proof-of-concept use cases. The algorithms and functions that implement the use case's functions are located in the CiL-Orchestrator's *zonemigration* class. The function of the class performing the corresponding tasks is

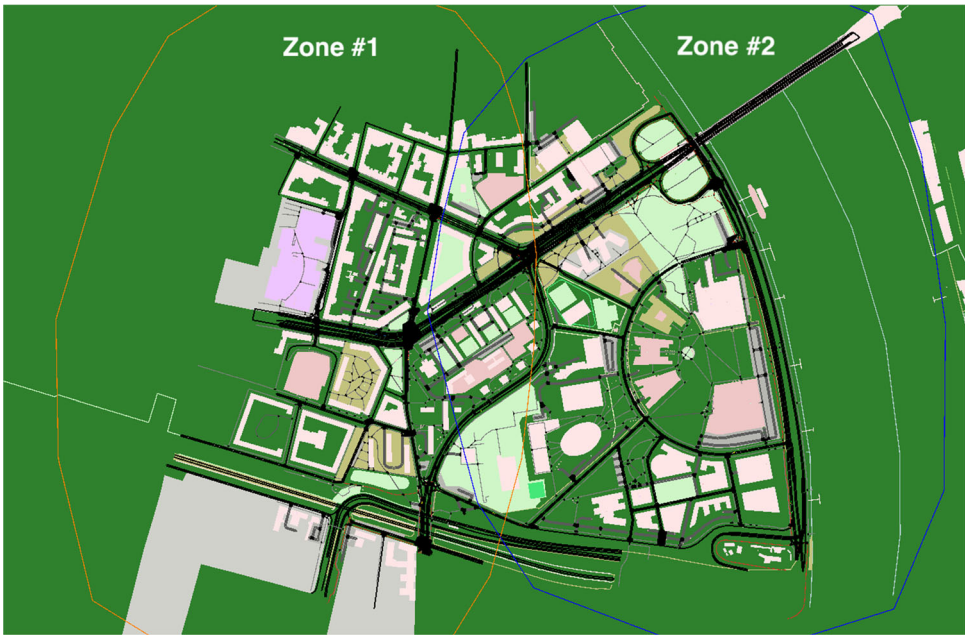

Fig. 2 Simulation map for the implemented test cases

hence called by the program during each simulation step, passing the current vehicle positions, the Kubernetes connection configuration, and other information necessary for any arbitrarily implemented function. Therefore, the coordinates of the vehicle can be compared with the access zones stored in the *managedZones* list, and its location in the network topology can be determined. To calculate this, the program uses the *contains(int x, int y)* function of Java's *Polygon* class, which can calculate from a coordinate passed as a parameter whether the given polygon includes it. As a result, it is possible to determine in which zones a particular vehicle is located, i.e., based on its position, which edge server serves it. Thus, migration operations also become feasible on the framework's distributed network. The software can also process many other variables that describe vehicles, allowing for various implementable use cases. Based on the processed simulation data, the orchestrator component can also manage applications implementing automotive use cases, including client and server applications realizing application-level network traffic. In our proof-of-concept scenarios, we implemented a live migration logic (Fig. 3) using the CiL-O. In this strategy, the orchestrator examines the vehicles and the zones' relative positions (in every simulation step) and, using this information, orchestrates the cloud environment and the application components. Suppose a vehicle enters a new zone (through the overlapped zone area). In that case, the implemented algorithm relocates the service-providing edge application by triggering the deployment of a new instance in the destination zone's edge server. It cleans up the previously used instance as soon as it is ready. Also, it re-configures the client applications (representing the vehicles) to connect the services provided by newly deployed edge applications.

Accessing Kubernetes from the CiL-Orchestrator is based on API calls. To establish communication with the distributed network, we used the official Java client library of Kubernetes [42] and implemented it in the CiL-Orchestrator. According to the use cases implemented in the CiL-Orchestrator, the software ensures the creation of the correct control signals for Kubernetes.

The Orchestrator component is capable of collecting fine-grained, Kubernetes-level event information. Monitoring event information that can be retrieved from the Kubernetes API server has only timestamps accurate to seconds. In order to get more precise time information on the containerized applications' lifecycle events, the Docker API can be used for fetching nanosecond-accurate timestamps. While in Kubernetes-level, the event's time data is only available through the master node via the k8s master-api, the respective docker-level events' time information is accessible directly on each node via the docker.sock UNIX socket. In our implementation, we chose this latter approach, and in addition, we exposed the socket instances to TCP/IP ports. Thus, the framework is capable of seamlessly access on all nodes the nanosecond-grade timestamps of container events.

3.3 Distributed cloud environment

The first important task of the cluster design was planning the network, hardware, and software elements that implement an actual cloud hardware environment based on the edge cloud paradigm. To evaluate various V2C scenarios

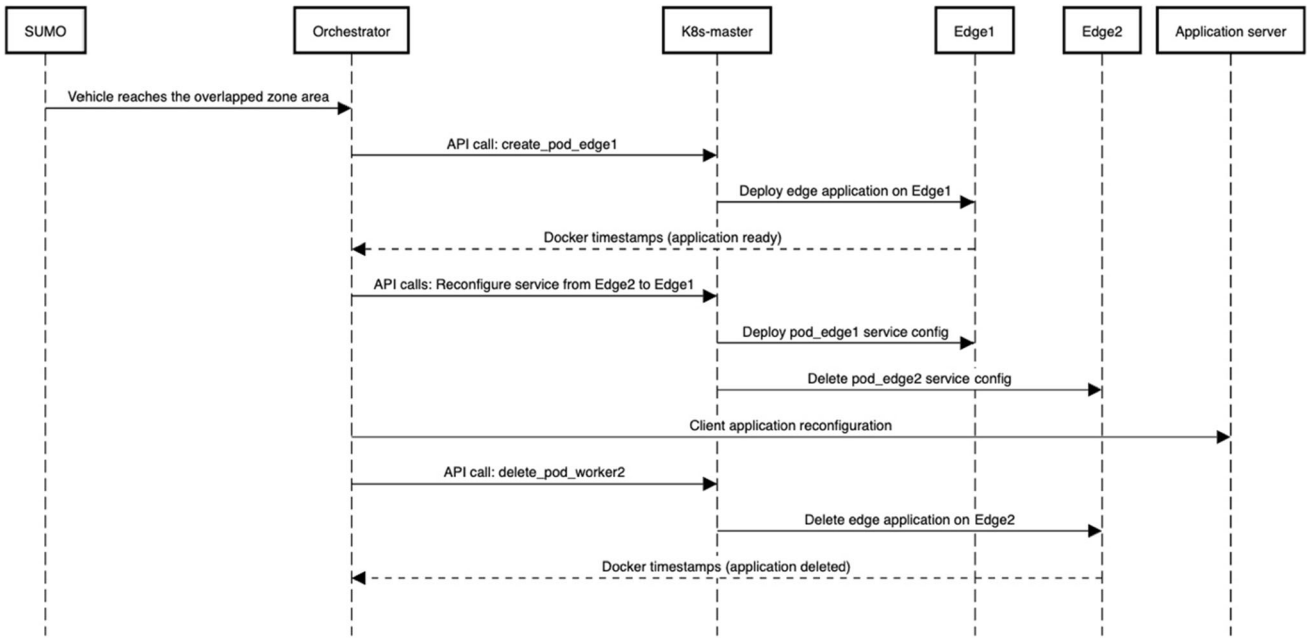

Fig. 3 The implemented live migration logic

and use cases, the k8s cluster was deployed with three worker nodes and a master node, which also runs the main software components of the Cloud-in-the-Loop framework. The details of the hardware components are shown in Table 2.

3.3.1 The layout of the devices

The deployment also includes a server entirely dedicated to running applications that realize and model the V2C application layer. Each server is interconnected through a high-performance switch with two 10 Gbit ports, an Operation and Maintenance, and a traffic port. These links can also be aggregated to create a 20 Gbit/s bonded connection between the servers. The selection and arrangement of the devices aimed to develop an actual distributed cloud deployment that could be integrated into a real network. According to this intention, the hardware can solve demanding computational tasks and serve many clients. Thus, this hardware platform enables the investigation and evaluation of use cases and validation of the concept on a telco-grade level. The cluster design also allows the integration of 5G functionalities using e.g., a Local Packet Gateway [45] in the future to which a high-performance server is prepared. The hardware environment consists of 6 devices (Fig. 4).

3.3.2 Software environment

To run the Cloud-in-the-Loop frameworks and the Kubernetes platforms' components, selecting an operating system that also properly utilizes the hardware's capabilities was necessary. All of the devices were installed with Ubuntu 18.04., a very stable and reliable version with good compatibility with various software used in the framework. With a functioning OS, the next step was to install Kubernetes, which required multiple preparatory steps. These include disabling the SWAP function (to make the software work properly), configuring the iptables and firewalls, and, most importantly, installing the Docker engine (version 20.10.7) on the nodes. In the following step, we installed Kubernetes (version 1.23.3). In the Kubernetes environment, Pod-to-Pod networking [46] is critical to cluster networking. It enables communication between the pods that encapsulate the containerized applications. It is a necessary function for the platform to operate. This type of communication can be realized by different technologies that are implemented mostly by third-party software components. In this implementation, the system is installed with the Container Network Interface (CNI) plugin called Calico [47]. Unlike other CNI plugins, Calico realizes the communication in the network layer (layer 3), using BGP routing protocol instead of relying on network virtualization. This way, additional

Table 2 The specification of the cloud environment

Device model	Role	Hardware description	Software	Networking
Dell R630	Kubernetes master node, Running the CiL Orchestrator, and the traffic simulator component	CPU: 28 cores (56 threads), 2.4 GHz, 3.3 GHz w/ Turbo RAM: 128 GB, 2.4 GHz Storage: 4 × 372 GB SSD	OS: Ubuntu 18.04 Kubernetes: version 1.23.3 (CNI: Calico) Docker: version 20.10.7	2 × 10 Gbit/s
Dell R630	Edge server 1 (k8s: worker1 node)	CPU: 24 cores (48 threads), 2.5 GHz, 3.3 GHz w/ Turbo RAM: 128 GB, 2133 MHz Storage: 4 × 372 GB SSD	OS: Ubuntu 18.04 Kubernetes: version 1.23.3 (CNI: Calico) Docker: version 20.10.7	2 × 10 Gbit/s
Dell R630	Edge server 2 (k8s: worker2 node)	CPU: 24 cores (48 threads), 2.5 GHz, 3.3 GHz w/ Turbo RAM: 128 GB, 2133 MHz Storage: 4 × 372 GB SSD	OS: Ubuntu 18.04 Kubernetes: version 1.23.3 (CNI: Calico) Docker: version 20.10.7	2 × 10 Gbit/s
Lenovo × 3650	Edge server 3 (k8s: worker3 node)	CPU: 20 cores (40 Threads), 2.3 GHz RAM: 144 GB, 2133 MHz Storage: 5 TB SSD/HDD	OS: Ubuntu 18.04 Kubernetes: version 1.23.3 (CNI: Calico) Docker: version 20.10.7	2 × 10 Gbit/s
Lenovo × 3650	Application server: running client and central server application components	CPU: 20 cores (40 Threads), 2.3 GHz RAM: 144 GB, 2133 MHz Storage: 5 TB SSD/HDD	OS: Ubuntu 18.04	2 × 10 Gbit/s
Dell R640	Not in use currently. Its future task: Running Cloud-Native 5G network functions [45]	CPU: 40 cores (80 Threads), 2 GHz, 3.7 GHz w/ Turbo RAM: 384 GB, 2666 MHz Storage: 8 × 480 GB SSD	-	2 × 10 Gbit/s

Fig. 4 The actual hardware environment (on the left)

encapsulation of the packets can be avoided, resulting in better performance [48].

4 Investigated case studies for testing edge cloud environments

4.1 Implementation of a UDP traffic benchmarking tool for testing quality of service (QoS)

One of the most critical aspects of benchmarking edge cloud systems is evaluating the effects of network and hardware resource load resulting from serving user devices. Furthermore, it is crucial to consider that since it is a distributed system, this load is distributed among the individual resources according to the users and the implemented load-balancing algorithms. During the planning of MEC services and systems, it is a vital aspect to be able to ensure the QoS required by the requirements in all cases. This is particularly important in areas with strict requirements, such as vehicular use cases. Errors and outages in application-level network traffic cause degradation of service quality. To investigate this in the CiL framework, it became necessary to integrate an application that can be appropriately scaled and is capable of generating network traffic.

On the other hand, it can also create QoS metrics based on the generated data. In this way, the load caused by various automotive use cases can be modeled on the actual edge cloud system integrated with the framework, and the system's performance can also be examined in terms of service quality. The performance and operation of the systems can be well examined from the point of view of the packet loss metrics of the traffic generated by the applications, a UDP-based network test traffic is ideal for such measurements. Later, we also plan to implement TCP-based performance tests to get a more comprehensive picture of the system characteristics. The basis of the self-developed UDP traffic benchmarking tool was an open-source application [49] developed in Python, which primarily focuses on measuring network latency. We made several modifications to the application, preparing it for proper operation in a distributed environment. During the presented tests, the application was running in three instances:

- Client-side application: This component compiles and forwards the packets with the appropriate information, which is used to evaluate the generated data traffic and detect network errors.
- Edge-side application: This component receives the network traffic generated by the client and ensures the processing of the information extracted from the packets and sending the relevant metrics data to the server-side application.
- Server-side application: This component receives the metrics data extracted by the edge application, then generates QoS metrics based on them.

4.2 Deep learning-based automotive use case implementation for testing quality of experience (QoE)

One of the most promising functionalities of edge cloud systems is the possibility of outsourcing computing tasks. This can be especially beneficial in areas such as Vehicle-to-Cloud communication. Modern vehicles are equipped with many sensors, including high-resolution cameras and LIDARs. Using these devices, vehicles can collect large amounts of raw data about their environment, which are processed to support various automotive use cases. The processing of environmental information requires high-performance hardware resources, and the vehicles must also share the information extracted from the processed data. With the help of edge cloud systems supported by 5G and beyond cellular networks, it is possible for the processing of sensor data to be implemented by the resources of the distributed environment. This makes it possible to process data collected from individual sources jointly, increasing the accuracy and reliability of environment detection. One of the most efficient ways to process sensor data for object detection is to use Deep Learning-based networks. Using the CiL framework, our goal in this work was to investigate the operation of an edge cloud-compatible, cloud-native V2C application based on this technology. In the first step, it was necessary to define and implement a use case that could be used to test the functionality of these technologies.

According to the implemented use case scenario, a given V2C-capable vehicle collects environmental data using its camera sensors and then transmits it to the edge server currently serving it. On the edge server, an AI-based application processes the video stream and sends relevant feedback information to the vehicle based on the data collected from its environment. In a later phase, the scheme can also support MEC-aided sensor fusion [50] and misbehavior detection [51] purposes. To model the use case and to be able to generate QoE metrics using the framework, we designed and developed a two-component application (Fig. 5). The operation of the application is detailed in Sect. 5.

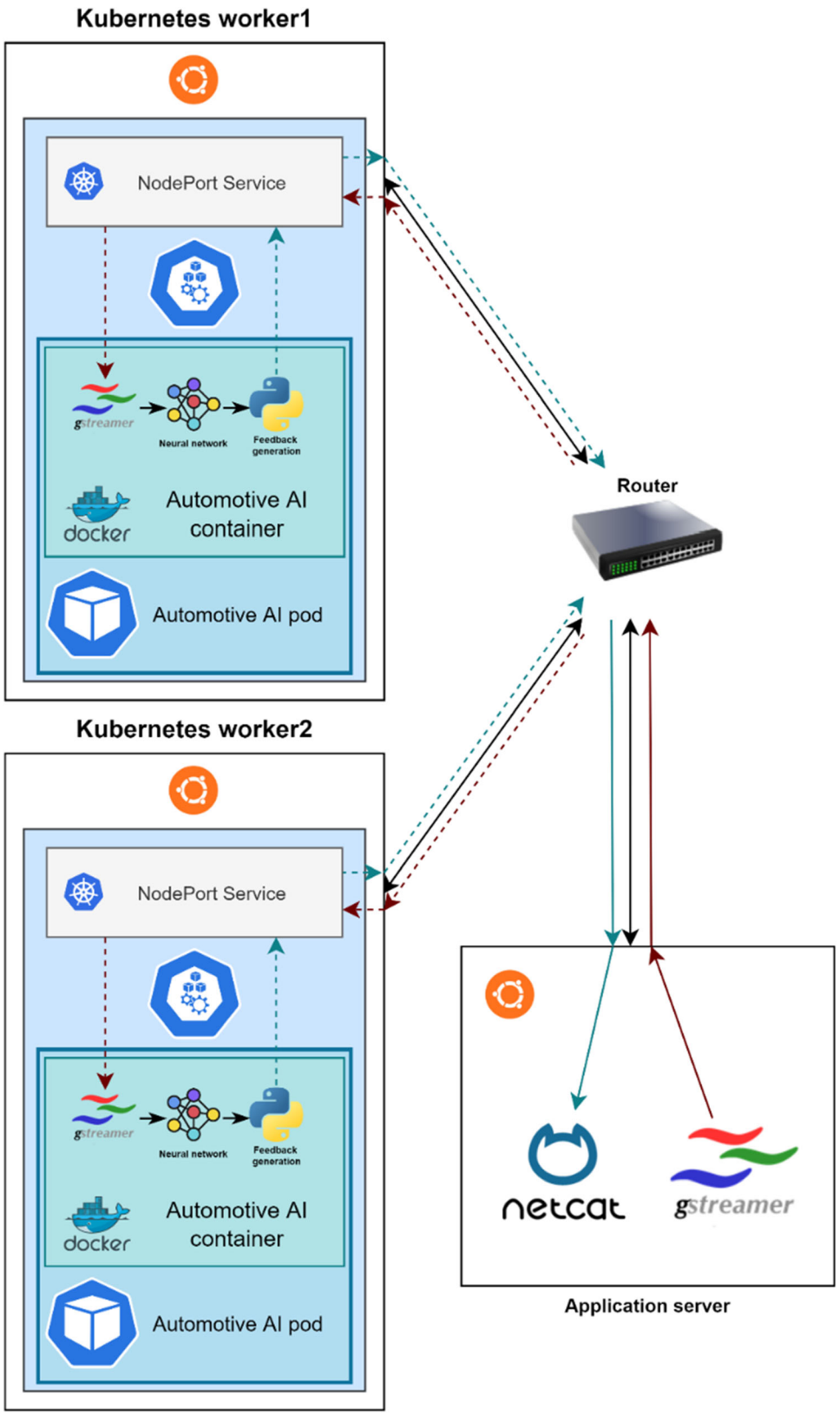

Fig. 5 The integration of the AI-based automotive use case

5 Test measurements

The most significant advantage of the proposed framework is that it can simulate the operation and characteristics of mobile UEs (vehicles). Based on that simulated information it can manage HW/SW implementations of telco cloud environments and applications. Opposed to other solutions, it can test the operation of actual services on telco-grade hardware and investigate the effects of the mobility of client devices without utilizing real, moving UEs. This paper presents how the CiL framework can be used to test and evaluate distributed cloud environments for V2C use cases. We present the framework's capabilities by examining applications implementing automotive use cases and the performance of the integrated k8s-based distributed environment. For this, it was first necessary to create a suitable test case.

The most critical aspect of offloading computing tasks offered by edge cloud systems is that the systems must ensure high Quality of Service (QoS) and the proper Quality of Experience (QoE) at the application level. In the case of the AI-based object detection application, we examined the QoE based on a predefined KPI, which characterizes the application's performance and the distributed cloud-based environment during the measurements. During the tests, with the help of the client-side applications representing V2C vehicles (Sect. 4.2), we transmitted a 30-s reference video material implementing raw data from a camera sensor to the edge-side applications. We also deployed two copies of the application components implementing the AI engine to both k8s worker nodes implementing edge servers integrated into the simulation environment. We calculated the performance of the object detection application and the QoE provided by it from detecting a car-type object on the reference video (Fig. 6).

The application detects and classifies objects frame by frame. It determines the confidence value for each recognized object, which shows how accurately it identifies an object type based on the trained model. At every frame evaluation, the application sends feedback to the client about the recognition confidence of car-type objects. During the measurements, the average of the (frame-by-frame) recognition confidences of the 30-s reference video provides the KPI based on which the system's operation can be examined under different traffic and network load scenarios:

$$ARC = \frac{\sum_{i=1}^{N} \frac{\sum_{j=1}^{M_i} RC_{i,j}}{M_i}}{N} \quad (1)$$

where ARC stands for the average recognition confidence, N is the number of measurements, M_i is the number of object detections performed in the i-th measurement, and $RC_{i,j}$ is the j-th recognition confidence of the i-th measurement. An AI engine ran on both edges of the simulation environment during the tests. Accordingly, we ran two clients in the two latency zones (Sect. 3.1), which forwarded the video stream to the edge servers corresponding to the zones. We performed the measurements according to several scenarios. To test the system's and the application's performance, we generated background load UDP packet traffic using our self-developed UDP benchmarking tool (Sect. 4.1) and the simulator, according to different vehicle numbers and data speeds. In order to achieve this, we placed vehicles performing movements causing application relocation operations (zone switching) into the simulation environment and generated a background load corresponding to the number of cars.

Fig. 6 Visualization of the object detection on the reference video (You Tube link: https://www.youtube.com/watch?v=u-CTsTZxRBI&t=218s)

5.1 Evaluating the QoS of the integrated distributed environment

We also performed Quality of Service performance tests on the integrated telco-grade edge cloud system to present the framework's capabilities. For this, we used the UDP benchmarking tool presented in Sect. 4.1. We used UDP relay applications running on the edges to conduct the tests. In the simulator, we created a vehicle for each client application, which causes the (live migration type) relocation of the relay applications on the edge servers due to the zone changes resulting from their mobility. Each vehicle is served by an edge application running on the edge server belonging to its zone. Edge applications forward the relevant metric data extracted from UDP packets received from clients to the server-side applications running on the application server. In this way, detailed QoS metrics describing the system's performance can be generated based on the evaluations carried out on the server side (Fig. 7). The results obtained in this way describe the effect of background load (vehicle number, generated data traffic) and the effects of application relocation events.

We performed measurements with 150, 200, 250, 300, and 350 simulated vehicles within the 1.5 km^2 map area and data traffic initiated from the client side with data rates of 1 Mbit/s, 2 Mbit/s, 3 Mbit/s, 4 Mbit/s, and 5 Mbit/s. The average of 500 pcs 30 s sessions in each scenario gives the results. We divided the results into those measurement results where application relocation occurred during the 30 s sessions and those measurement results where no migration occurred for the cars under test. The results are based on the packet loss rates from the QoS metrics (Figs. 8, 9, and 10). Based on these tests, the performance of the system under a given load and the impacts of relocation events can be indicated.

As expected, the measurement results indicate that a lower load (fewer vehicles and lower data speeds) results in low packet loss ratios (PLR). It can be observed in Fig. 8, where the aggregated results are presented, that the average PLRs are in the 0–5% range, except in the cases of 300 vehicles/5 Mbps (7.114% PLR), 350 vehicles/4 Mbps (9.54% PLR) and 350 vehicles/5 Mbps (23.60% PLR). It indicates that the resulting load in these cases can significantly deteriorate the edge infrastructure performance. Comparing the results where no relocations occurred (Fig. 8) with the results influenced by relocations (Fig. 9 shows that the effect of application relocations significantly increases the PLRs. In most vehicle number and data speed

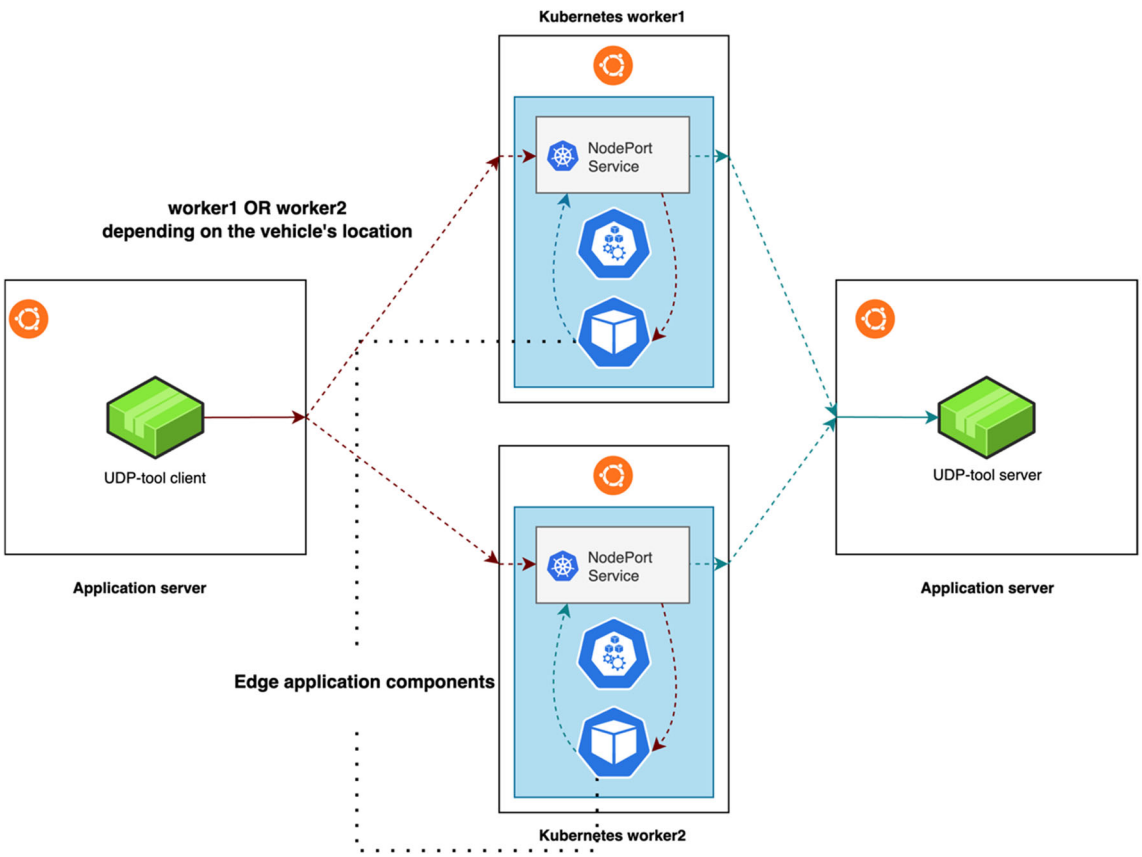

Fig. 7 Application components of the QoS tests

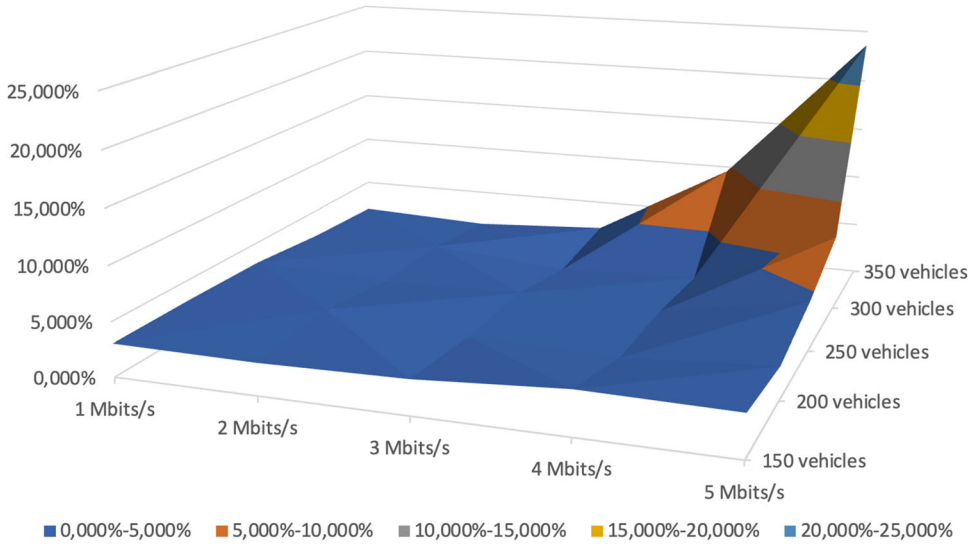

Fig. 8 Visualizing aggregated packet loss results

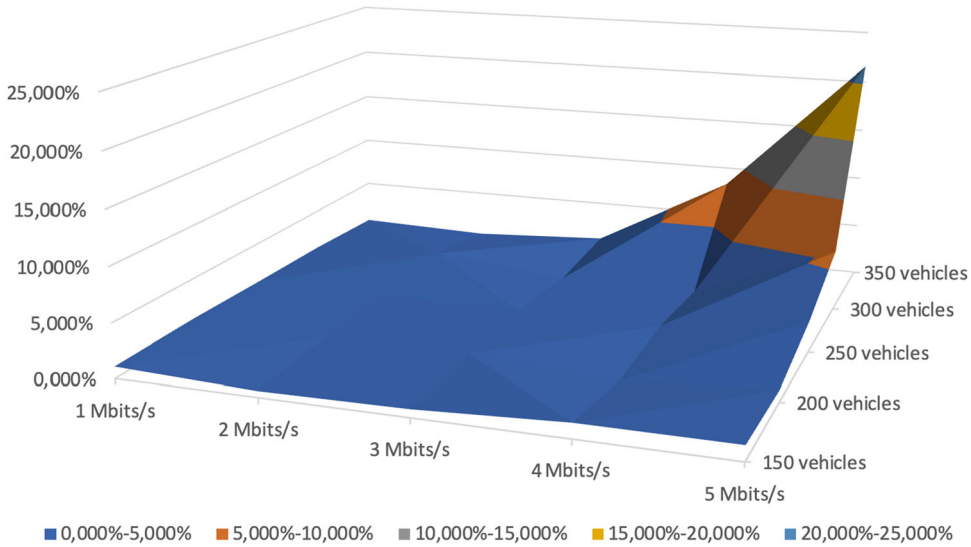

Fig. 9 Visualizing packet loss results without relocation events

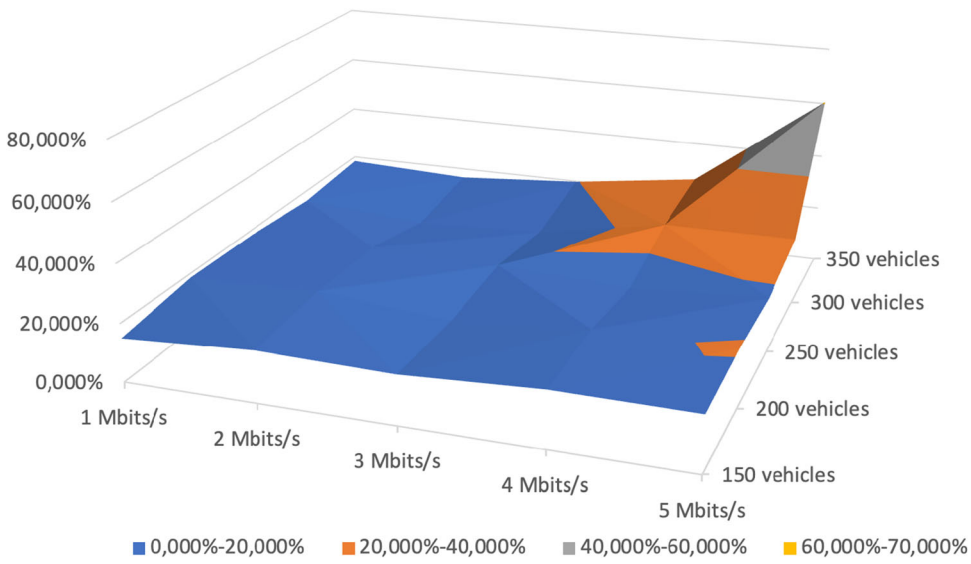

Fig. 10 Visualizing packet loss results due to relocation events

configurations, the average PLRs are in the 0–2% range without the relocation events, and during relocations, these values increase to 15–20%. Furthermore, in the case of the 350 vehicle/5 Mbps case, the average PLR increases from 21.56% (with no relocation) to 60.56%. Thus, in use cases where relocation is a part of the operation, addressing these effects on the QoS will be essential, as service continuity is one of the most critical aspects of V2C.

5.2 Evaluation of the Deep Learning-based automotive use case implementation

As discussed in the introduction of Sect. 5, we present the results generated from the average recognition confidences during the measurements with the application components implementing the Deep Learning-based automotive use case. We tested the application using methods that implement the background load, which is the basis of the measurement results presented in Sect. 5.1. For this, we created various measurement scenarios with 100, 200, and 300 simulated vehicles and data traffic initiated from the client side with data rates of 1, 2, 3, 4, and 5 Mbits/s. In these, we calculated how the average recognition confidence of car-type objects in the reference videos changes due to a given background load (Table 3 and Fig. 11). The results are provided by the total average of the confidences generated by the application instances running on the two worker nodes, with around 100 measurements for each scenario (1 measurement is given by the confidence values calculated for each frame of the 30-s reference video) per node. In order to be able to compare the results, we first measured the KPI without load, which shows what QoE the AI engine can provide if the distributed system does not serve any other clients. The average recognition confidence, in this case, was 52.80% based on 718 measurements (per node).

With the below-introduced test measurements, we present that the framework can run applications modeling complex, edge-computing supported automotive use cases that can be examined in detail. In this case, we can conclude that the impact of the background load on the system occurring in these measurement scenarios essentially does not affect the efficiency of the application. This clearly shows that changes in the background load do not necessarily cause these differences in the measurement results. In addition to the averages, when examining the individual measurement points, it can also be seen that the distribution of the recognition confidence measurements for the scenario without background load and the scenario with maximum generated load (300 vehicles, 5 Mbps) is almost identical (Figs. 12, 13). To properly investigate the load effects on the operation of object detection applications, increasing the vehicle number and data speeds will be required by defining new test configurations. Also, to design more comprehensive QoE metrics, it's necessary to gather more information on the operation of the application. For this, we also plan to store the forwarded camera sensor videos, use post-process evaluation on objective QoE, and extract specific information that describes the effects of operating in the edge infrastructure under load. This way, it will be possible to check the effects leading to lost frames that can be critical from the object recognition and sensor fusion/collective perception point of view.

Here, with the introduced initial measurements, we only focused on presenting the capabilities of our proposed framework. Therefore, we do not reach general conclusions on the integrated cloud infrastructure or an actual V2C use case with these results. The presented case studies and applications were developed to showcase our CiL framework, to model the operation of real V2C applications, and, using these, to indicate how edge systems are affected in certain operating modes. However, by improving the test cases, tools, metrics, and vehicle traffic models, we aim to utilize the framework to dimension edge networks, validate use cases and develop methods to optimize 5G and beyond edge-computing system operations. With improving the QoS and the proposed AI-based QoE measurement methodology, we plan to design a model that can produce QoE prediction based on the QoS metrics.

Table 3 The averages, minimums, and maximums of the object detection confidences in different scenarios

	1 Mbits/s (%)	2 Mbits/s (%)	3 Mbits/s (%)	4 Mbits/s (%)	5 Mbits/s (%)
100 vehicles	Avg: 52.92	Avg: 52.91	Avg: 52.71	Avg: 52.86	Avg: 52.88
	Min: 47.78	Min: 47.41	Min: 46.43	Min: 46.59	Min: 48.42
	Max: 54.48	Max: 54.39	Max: 54.23	Max: 54.60	Max: 55.05
200 vehicles	Avg: 52.87	Avg: 52.86	Avg: 52.73	Avg: 52.71	Avg: 52.78
	Min: 46.04	Min: 47.27	Min: 46.46	Min: 46.33	Min: 48.18
	Max: 54.63	Max: 54.71	Max: 54.36	Max: 55.05	Max: 54.89
300 vehicles	Avg: 52.84	Avg: 52.77	Avg: 52.84	Avg: 52.64	Avg: 52.80
	Min: 45.81	Min: 46.91	Min: 46.97	Min: 46.95	Min: 46.52
	Max: 55.15	Max: 55.39	Max: 54.75	Max: 54.77	Max: 55.68

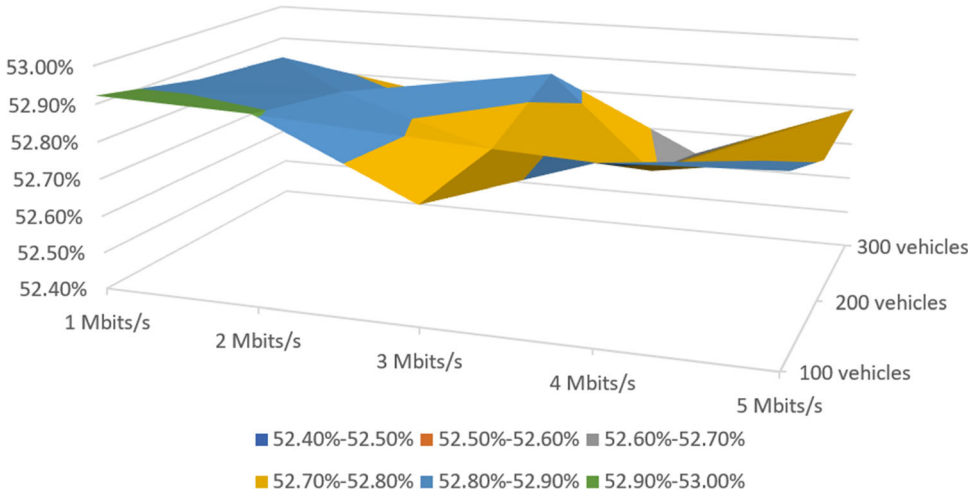

Fig. 11 Visualizing the averages of the object detection confidences in different scenarios

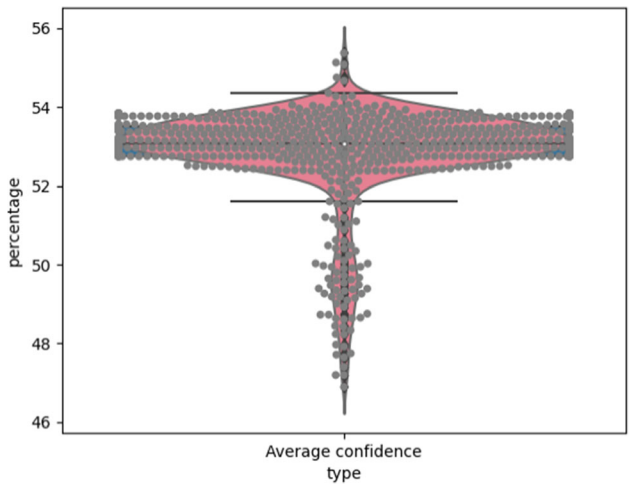

Fig. 12 Distribution of individual recognition confidence measurement results in a scenario without background load (on worker1 node)

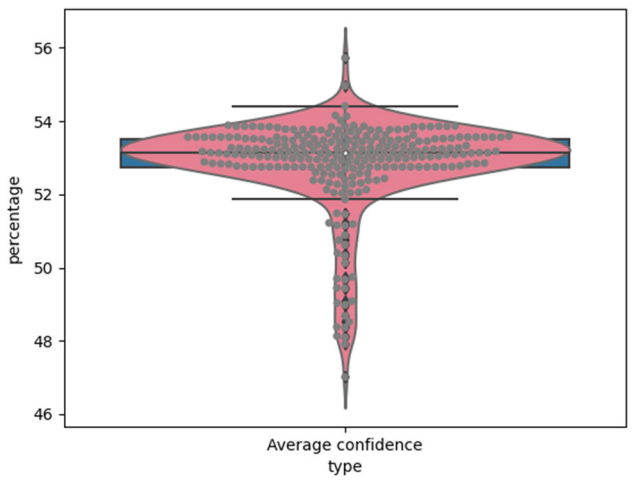

Fig. 13 The distribution of individual recognition confidence measurement results with 300 vehicles and 5 Mbps (on worker1 node)

6 Conclusion

In the paper, we briefly presented the technological foundations of edge cloud-based VC2 communication and its importance in the future. After that, we detailed the focused literature on tools designed for simulating V2C or edge cloud systems, like the framework we developed. Then, we presented the concept and the structure of our proposed Cloud-in-the-Loop simulation framework and its components. The simulation environment, orchestration component, and the real distributed cloud-based network integrated into the framework are covered in detail. After the presentation of the system's operating principle, we introduced proposed test cases of the framework, with the help of which the integrated edge environment and cloud-native applications implementing vehicle communication solutions can be investigated. Finally, we presented how the framework can be used to test edge cloud-based V2C use cases from QoS and QoE perspectives. Overall, it can be declared that based on the results so far, the Cloud-in-the-Loop simulation framework can be excellently utilized for testing V2C use cases on real edge systems.

However, our solution currently focuses on testing the edge cloud environments, and we do not investigate the mobile network aspects of the services, which is crucial for the operation of these use cases. In order to carry out more detailed and in-depth examinations, further improvement of the system is necessary. Future goals include further development of the framework and the test methodologies; one of the priority targets is integrating 5G core network components and such improving our QoS and QoE test models and metrics to produce more precise and more relevant performance measurements. We can use the framework with higher precision to properly model V2C use cases inside the 5G ecosystem, supporting network dimensioning, edge infrastructure planning, edge node

placement strategies evaluation, and use case validation. Furthermore, it will make it possible to develop QoS and QoE correlation models that can help with various optimization tasks of 5G and beyond architectures.

Funding Open access funding provided by Budapest University of Technology and Economics. Project No. TKP2021-NVA-02 has been implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme.

Data availability Data sharing is not applicable to this article as no datasets were generated or analyzed during the current study.

References

1. Ericsson. *5G core (5GC)*. Ericsson. Retrieved April 1, 2023, from https://www.ericsson.com/en/core-network/5g-core
2. Sami Kekki et al. *MEC in 5G networks (ETSI White Paper No. 2)*. ETSI. Retrieved from April 1, 2023, https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf
3. ETSI. *Multi-access edge computing (MEC); framework and reference architecture (ETSI GS MEC 003)*. ETSI. Retrieved April 1, 2023, from https://www.etsi.org/deliver/etsi_gs/MEC/001_099/003/03.01.01_60/gs_MEC003v030101p.pdf
4. AECC. *Operational behavior of a high definition map application white paper*. AECC. Retrieved April 1, 2023, from https://aecc.org/wp-content/uploads/2020/07/Operational_Behavior_of_a_High_Definition_Map_Application.pdf
5. Nejatishahidin, N., Fayyazsanavi, P., & Košecka, J. (2022). Object pose estimation using mid-level visual representations. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2022*, 13105–13111. https://doi.org/10.1109/IROS47612.2022.9981452
6. L. Maller, P. Suskovics, and L. Bokor, 'Cloud-in-the-Loop simulation of C-V2X application relocation distortions in Kubernetes based Edge Cloud environment', in *2022 26th international conference on information technology (IT)*, 2022, pp. 1–4. doi: https://doi.org/10.1109/IT54280.2022.9743520.
7. Lopez P.A. et al. Microscopic Traffic Simulation using SUMO, in *The 21st IEEE international conference on intelligent transportation systems*, IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/
8. SUMO. (2023, March 30). *Traffic control interface (TraCI)*. SUMO. Retrieved April 1, 2023, from https://sumo.dlr.de/docs/TraCI.html
9. Mansouri, N., Ghafari, R., & Zade, B. M. H. (2020). Cloud computing simulators: A comprehensive review. *Simulation Modelling Practice and Theory, 104*, 102144. https://doi.org/10.1016/j.simpat.2020.102144
10. Lago, D. G., da Silva, R. A. C., Madeira, E. R. M., da Fonseca, N. L. S., & Medhi, D. (2021). 'SinergyCloud: A simulator for evaluation of energy consumption in data centers and hybrid clouds. *Simulation Modelling Practice and Theory, 110*, 102329. https://doi.org/10.1016/j.simpat.2021.102329
11. Ahmed, B., Malik, A. W., Hafeez, T., & Ahmed, N. (2019). Services and simulation frameworks for vehicular cloud computing: A contemporary survey. *EURASIP Journal on Wireless Communications and Networking, 2019*(1), 4. https://doi.org/10.1186/s13638-018-1315-y
12. Anritsu and dSPACE. *Anritsu and dSPACE to accelerate simulation and testing of 5G automotive applications–joint showcase at MWC 2020*. dSPACE. Retrieved April 1, 2023, from https://www.dspace.com/en/pub/home/news/dspace_pressroom/press/20200101.cfm#175_51153_1
13. Dell. (2020). *Hardware-in-the-loop autonomous driving simulation*. Dell. Retrieved April 1, 2023, from https://www.delltechnologies.com/asset/en-ae/products/storage/briefs-summaries/dell-emc-aws-natl-instruments-hil-solution-overview.pdf
14. Milani F., Blaschke V., Johannaber M., and Beidl C., 'X-in-the-loop test methods for cloud-based vehicle functions 2017
15. Weiss, M., Zhang, J., & Chakrabortty, A. (2016). Wide-area control of power systems using cloud-in-the-loop feedback. *IEEE Global Conference on Signal and Information Processing (GlobalSIP), 2016*, 831–835. https://doi.org/10.1109/GlobalSIP.2016.7905959
16. Wang, J., & Zhu, Y. (2022). A hardware-in-the-loop V2X simulation framework: CarTest. *Sensors, 22*(13), 102. https://doi.org/10.3390/s22135019
17. OPNET. (2023). *OPNET Network simulator*. OPNET. Retrieved April 1, 2023, from https://opnetprojects.com/opnet-network-simulator/
18. OMNeT++. (2023). *OMNeT++ Documentation.* OMNeT++. Retrieved April 1, 2023, from https://omnetpp.org/documentation/
19. Virdis A., Stea G., and Nardini G., Simulating LTE/LTE-advanced networks with simuLTE. in *Simulation and modeling methodologies, technologies and applications*, M. S. Obaidat, T. Ören, J. Kacprzyk, and J. Filipe, Eds., Cham: Springer International Publishing, 2015, pp. 83–105. https://doi.org/10.1007/978-3-319-26470-7_5
20. CloudSim. *A framework for modeling and simulation of cloud computing infrastructures and services*. The cloud computing and distributed systems (CLOUDS) laboratory, University of Melbourne. Retrieved April 1, 2023, from http://www.cloudbus.org/cloudsim/
21. Gupta, H., Dastjerdi, A. V., Ghosh, S. K., & Buyya, R. (2016). 'iFogSim: A toolkit for modeling and simulation of resource management techniques in internet of things. *Edge and Fog Computing Environments'. arXiv*. https://doi.org/10.48550/ARXIV.1606.02007
22. Christoph Sommer. *Veins, the Open Source vehicular network simulation framework - Documentation*. Vechiles in Network Simulation (Veins). Retrieved April 1, 2023, from https://veins.car2x.org/documentation/
23. Riebl R., Obermaier C., and Günther H.-J. Artery: Large scale simulation environment for ITS applications', in *recent advances in network simulation: The OMNeT++ environment and its ecosystem*, A. Virdis and M. Kirsche, Eds., Cham: Springer International Publishing, 2019, pp. 365–406. doi: https://doi.org/10.1007/978-3-030-12842-5_12.

24. G. Nardini, G. Stea, A. Virdis, D. Sabella, and P. 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*, 2020, pp. 1–7. doi: https://doi.org/10.1109/PIMRC48278.2020.9217177
25. Hegyi P., Varga N., and Bokor L. An advanced telco cloud simulator and its usage on modelling multi-cloud and 5G multi-access environments', in *2018 21st conference on innovation in clouds, internet and networks and workshops (ICIN)*, 2018, pp. 1–3. doi: https://doi.org/10.1109/ICIN.2018.8401637.
26. Hegyi P., and Varga J. Telco Cloud Simulator, in *2019 IEEE 24th international workshop on computer aided modeling and design of communication links and networks (CAMAD)*, 2019, pp. 1–7. doi: https://doi.org/10.1109/CAMAD.2019.8858483.
27. Abdelatif S., Makhlouf D., and Roose P., Extended iCanCloud simulation framework for VANET-Cloud architectures in *3rd international conference on networking and advanced systems*, Annaba, Algeria, 2017. [Online]. Available: https://hal-univ-pau.archives-ouvertes.fr/hal-02464156
28. Prometheus. *What is prometheus?*. Prometheus. Retrieved April 1, 2023, from https://prometheus.io/docs/introduction/overview/
29. Beloglazov A., and Buyya R., Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers, *Concurrency and Computation: Practice and Experience*, pp. 1–24, Jan. 2011.
30. Mahmud, M., Pallewatta, S., Goudarzi, M., & Buyya, R. (2022). iFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments. *Journal of Systems and Software., 190*, 111351. https://doi.org/10.1016/j.jss.2022.111351
31. Sonmez C., Ozgovde A., and Ersoy C., EdgeCloudSim: An environment for performance evaluation of Edge Computing systems', in *2017 Second International Conference on Fog and Mobile Edge Computing (FMEC)*, 2017, pp. 39–44. doi: https://doi.org/10.1109/FMEC.2017.7946405
32. Sonmez, C., Tunca, C., Ozgovde, A., & Ersoy, C. (2021). Machine learning-based workload orchestrator for vehicular edge computing. *IEEE Transactions on Intelligent Transportation Systems, 22*(4), 2239–2251. https://doi.org/10.1109/TITS.2020.3024233
33. Nardini, G., Stea, G., & Virdis, A. (2021). Scalable real-time emulation of 5G networks with simu5G. *IEEE Access, 9*, 148504–148520. https://doi.org/10.1109/ACCESS.2021.3123873
34. Noferi, A., Nardini, G., Stea, G., & Virdis, A. (2023). Rapid prototyping and performance evaluation of ETSI MEC-based applications. *Simulation Modelling Practice and Theory, 123*, 102700. https://doi.org/10.1016/j.simpat.2022.102700
35. Sommer, C., German, R., & Dressler, F. (2011). Bidirectionally coupled network and road traffic simulation for improved IVC Analysis. *IEEE Transactions on Mobile Computing, 10*(1), 3–15. https://doi.org/10.1109/TMC.2010.133
36. Hegde A., Festag A., Artery-C: An OMNeT++ Based Discrete Event Simulation Framework for Cellular V2X, in *proceedings of the 23rd international acm conference on modeling, analysis and simulation of wireless and mobile systems*, in MSWiM '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 47–51. doi: https://doi.org/10.1145/3416010.3423240.
37. Kovács G.A., Bokor L., Integrating artery and simu5G: A mobile edge computing use case for collective perception-based V2X safety applications, in *2022 45th international conference on telecommunications and signal processing (TSP)*, 2022, pp. 360–366. doi: https://doi.org/10.1109/TSP55681.2022.9851276.
38. G. Kovács and L. Bokor, 'Towards realistic simulation of MEC-based Collective Perception: an initial edge service design for the Artery/Simu5G framework', Jan. 2023, pp. 53–58. doi: https://doi.org/10.3311/WINS2023-010.
39. G. G. Castañé, A. Núñez, and J. Carretero, 'iCanCloud: A brief architecture overview', in *2012 IEEE 10th international symposium on parallel and distributed processing with applications*, 2012, pp. 853–854. doi: https://doi.org/10.1109/ISPA.2012.131.
40. Kubernetes. *Kubernetes-Overview*. Kubernetes. Retrieved April 1, 2023, from https://kubernetes.io/docs/concepts/overview/
41. Henrik B. and Rakesh B. Why Kubernetes over bare metal infrastructure is optimal for cloud native applications, May 03, 2022. https://www.ericsson.com/en/blog/2022/5/kubernetes-over-bare-metal-cloud-infrastructure-why-its-important-and-what-you-need-to-know
42. kubernetes-client. Kubernetes Java Client - Home. github. Retrieved April 1, 2023, from github. https://github.com/kubernetes-client/java/wiki
43. Kubernetes. *Kubernetes - Pods*. Retrieved April 1, 2023, from https://kubernetes.io/docs/concepts/workloads/pods/
44. Kubernetes. *Kubernetes - Service*. Retrieved April 1, 2023, from https://kubernetes.io/docs/concepts/services-networking/service/
45. Ericsson. *Embrace the 5G edge opportunity*. Ericsson. Retrieved April 1, 2023, from https://www.ericsson.com/494ce3/assets/local/core-network/doc/5g-core-local-packet-gateway-datasheet.pdf
46. Kubernetes. Kubernetes - cluster networking. Kubernetes. Retrieved April 1, 2023, from https://kubernetes.io/docs/concepts/cluster-administration/networking/
47. Calico. *About Calico*. Tigera. Retrieved April 1, 2023, from https://docs.tigera.io/calico/latest/about
48. Rancher. Comparing Kubernetes CNI Providers: Flannel, Calico, Canal, and Weave. Rancher by SUSE. Retrieved April 1, 2023, from https://www.suse.com/c/rancher_blog/comparing-kubernetes-cni-providers-flannel-calico-canal-and-weave/
49. Chuanyu X. Udp-latency - README. github. Retrieved April (2022) 1, 2023, from https://github.com/ChuanyuXue/udp-latency/blob/main/README.md
50. Munir, A., Blasch, E., Kwon, J., Kong, J., & Aved, A. (2021). Artificial intelligence and data fusion at the edge. *IEEE Aerospace and Electronic Systems Magazine, 36*(7), 62–78. https://doi.org/10.1109/MAES.2020.3043072
51. van der Heijden, R. W., Dietzel, S., Leinmüller, T., & Kargl, F. (2019). Survey on misbehavior detection in cooperative intelligent transportation systems. *IEEE Commun. Surv. Tutor., 21*(1), 779–811. https://doi.org/10.1109/COMST.2018.2873088

Levente Márk Maller received his M.Sc degree in electrical engineering from the Department of Networked Systems and Services (HIT), Budapest University of Technology and Economics (BME) in 2023. Since 2023, he is a Ph.D. student at the BME Doctoral School of Informatics, Department of Networked Systems and Services, member of the Multimedia Networks and Services Laboratory (MEDIANETS). He is also a developer at Ericsson where he works on edge user plane solutions for 5G networks. His research interest areas are edge cloud networks, edge computing, and Vehicle-to-Cloud communication.

Péter Suskovics received his M.Sc. degree in computer engineering from the Department of Telecommunications, Budapest University of Technology and Economics (BME) in 2008 and completed his Ph.D. studies at the Doctoral School of Informatics of BME in 2011. He joined Ericsson in 2007 as a software developer, later he drove several product development and innovation projects related to telecommunication services. Currently, he is a senior system architect of 5G cloud software and services, specializing in characteristics analysis and performance management. His main interest areas cover data analytics, AI/ML, and the adaptation of 5G/6G technologies in industrial use cases. As a delegate of the Automotive Edge Computing Consortium, he focuses on requirement specification, architecture definition, and use case development.

László Bokor received his Ph.D. degree in computer engineering from the Budapest University of Technology and Economics (BME) in 2014. He is currently an associate professor at the Department of Networked Systems and Services, BME, Budapest, 1117, Hungary. His research interest focuses on V2X communications in Intelligent Transportation Systems. He is a member of several professional organizations, such as the IEEE ITS Society, the Hungarian Standards Institution's Technical Committee for ITS, and the ITS Hungary Association.