Application Synchronization among Multiple MEC Servers in Connected Vehicle Scenarios

Tiia Ojanperä*, Hans van den Berg†‡, Wieger IJntema†, Ramon de Souza Schwartz† and Miodrag Djurica†
*VTT Technical Research Centre of Finland, Kaitoväylä 1, Oulu, Finland, Email: {firstname.lastname}@vtt.fi
†TNO, the Netherlands Organisation for Applied Scientific Research, Anna van Buerenplein 1, The Hague, the Netherlands,
Email: {j.l.vandenberg, wieger.ijntema, ramon.desouzaschwartz, miodrag.djurica}@tno.nl
‡ University of Twente, Drienerlolaan 5, 7522 NB Enschede, the Netherlands

***Abstract*—Multi-access Edge Computing (MEC) is an emerging technology aimed to improve the communication latency and scalability e.g. of cloud-based connected vehicle applications and services. This is accomplished by bringing services closer to the end users, that is, to the network edge. In a high-mobility scenario, such as one involving vehicles, there will be a need for handovers between different MEC servers in order to maintain the required communication latency. Part of the application data may also be relevant to multiple MEC servers, covering overlapping geographical areas or being hosted by different network operators. This paper focuses on these situations and studies the problem of maintaining service continuity and synchronization of relevant data among multiple MEC servers to support vehicular applications. The analysis is conducted in the context of two example applications with different requirements, namely platoon management and shared world model, with promising results regarding their suitability for future implementation.**

Keywords*—Automotive services, edge computing, orchestration, system architecture.

I. Introduction

The advance towards fully automated driving is an evolutionary process where dependable, low latency communication among vehicles plays a crucial role as well as seamless connectivity to the internet enabling supportive applications and services that cannot be realized efficiently in a fully centralized way. Enhancing the operation of connected and automated vehicles with these applications will require bringing the cloud infrastructure and services physically closer to the vehicles. Multi-access Edge Computing (MEC) [1] aims at realizing this by incorporating servers into the networked system architecture. Depending on the deployment scenario and service requirements, MEC servers may be placed by the cloud and telecom operators such that they are even collocated with the wireless access points (AP). Thus, the MEC servers can be used for achieving low-enough latencies even for safety critical services for connected vehicles.

In connected vehicle scenarios [2], the use cases of MEC include *computation task offloading* and *vehicular edge cloud services*. In the former, a computation task is transferred partially or wholly from a vehicle for execution on a MEC server, and only the result is fed back to the vehicle. In the latter, MEC servers are used for hosting cloud applications and services at the network edge. The edge cloud services may collect various and even vast amounts of real-time vehicular sensor and IoT data, including video and LiDAR, for localized processing and analysis. IoT data from other sources, such as the roadside infrastructure or traffic management service, can be incorporated as well for enhanced situational awareness. For such services, MEC can provide better scalability by reducing the traffic load on the core network in addition to lower communication latencies than traditional cloud services.

To fully benefit from MEC, vehicles should connect automatically to the optimal MEC server for the edge services [3]. Vehicles are highly mobile, thus the optimal MEC server changes whenever a vehicle moves too far from the serving MEC server. The notion of *far* in this case depends on the requirements of the application. The limiting factor may be the tolerated communication delay, or it may refer to the boundaries of the geographical area served by an edge application instance. Another example is roaming, in which the MEC server may change due to the changing of the network operator when crossing a country border. The handover process should include the selection of the optimal MEC server [4] and ensure service continuity despite the server change. Furthermore, if each vehicle is connected to a single operator network/MEC infrastructure at a time, some services may benefit from vehicular application data exchange across multiple operators, servicing the same geographical area.

This paper studies two connected and automated vehicle applications that rely on IoT data and MEC and are currently being developed in the EU AUTOPILOT project [5], viz. Platoon Management (PM) and Shared World Model (SWM). For the applications, MEC holds near time critical data, such as dynamic local maps (HD maps) or information for vehicles that want to join a platoon. In this paper, we describe an overall system architecture and specific designs for supporting the applications' deployment in MEC. Particular emphasis is put on the issue of synchronizing application data among multiple MEC servers, e.g. when a platoon moves from one MEC service area to another one or to enhance the situational awareness across the service areas. For each application, detailed requirements for data synchronization between vehicles and MEC server as well as among neighboring MEC servers are provided. Based on the specifications and requirements, we provide an initial assessment of the applications' delay performance and the scalability of their designs in realistic settings. Given the promising results of the feasibility study of the proposed overall solution, we will move to the implementation phase and experimental evaluation in our future work.

The rest of the paper is organized as follows. Section II discusses the related work. Section III introduces the use cases of MEC. Section IV describes the overall system architecture

and application designs. Section V provides the performance assessment. Finally, Section VI concludes the paper.

II. Related Work

In the MEC reference architecture [1], defined by ETSI, the Mobile Edge Orchestrator (MEO) is the entity responsible for managing applications and resources in the whole MEC system, including also connectivity and mobility support. The standard defines three options for maintaining continuity of the service between a MEC application and UE when the MEC server changes: *(i) Application state relocation*, *(ii) Application instance relocation within a MEC system*, and *(iii) Application instance relocation between a MEC system and an external cloud*. The implementation of the approaches is however out of the scope of ETSI's standard. Furthermore, the triggering of the MEC server change depends on the optimization criteria used. For example, the optimization of low-latency LTE/5G services within a MEC system is studied in [4].

The mechanisms needed for supporting session continuity in the case of a handover and application state or instance relocation depend on the application type. That is, if the application is *stateful* or *stateless* [6]. The stateless paradigm is more flexibile as it maintains information and session state only in the client side. It is also a key enabler of the microservice architecture [6], [7], which is a means of developing software applications as a suite of independently deployable, small, modular services, in which each service runs a unique process and communicates through a well-defined, lightweight mechanism such as REST. The approach is different from the traditional monolith application, built as a single autonomous unit, and is thus seen as a potential approach for implementing MEC applications and other future 5G services [7]–[10].

Replicated distributed storage can be utilized together with stateless applications in order to store state, thereby avoiding too large client request messages. However, in the case of a storage-dependent application, the system must support data synchronization, managing the periodical propagation of data storage, in order to maintain data volume coherency and thus ensure session continuity despite migration [6].

Especially lightweight virtualization technologies, such as Docker, are seen as potential platforms for deploying applications in MEC [6]. Some works, proposing solutions for application migration using VMs or containers, already exist in the literature. A management architecture and replica-based approach for ultra-short latency service provisioning in mobile MEC environments is proposed in [6]. The solution is evaluated in the context of stateless applications deployed in Docker containers. Approaches for moving a network service function on the fly from cloud to an edge node are discussed in [11]. The paper presents experimental results for the VM migration case, which is the heaviest one in terms of bandwidth consumption and service down time compared to application or session transfer. A generic layered migration framework, using incremental file synchronization, is proposed for applications that are encapsulated in a container or VM in [12].

This paper advances the literature by conducting an initial feasibility analysis of the operation of vehicular applications in a MEC system. The analysis is based on two actual connected and automated vehicle application designs, presented in this paper. To the best of our knowledge, this is the first analysis of MEC use in platoon management or shared world model.

III. Connected Vehicle Use Cases for MEC

This section introduces the two vehicular use case applications for MEC considered in this paper. Both applications benefit from MEC but have different requirements regarding the data synchronization among multiple MEC servers.

A. Platoon Management

Platooning relates to the function where vehicles automatically follow one another in a relatively close distance. Driving in a platoon requires the vehicles to use V2V communications to anticipate timely on maneuvers of the other vehicles in the platoon. Several benefits for platooning exist, such as improvement of traffic throughput and homogeneity, enhancement of traffic safety, and reduction of fuel consumption and emissions.

Platooning includes three distinct phases: finding a platoon to join and navigating to it (formation), merging with (driving) a platoon and maintaining a platoon. The first one is non-time critical and contains high level (discovery, authentication) information on available platoons, and it typically would run in the service cloud provider. The second, would run on a MEC node, since it needs to have information on the current state of the platoon, which goes beyond the V2V communication range and is used to assist vehicles joining the platoon. Finally, the third one, maintaining the platoon, would typically run on the vehicles and uses V2V communication.

In this paper, we focus on the platoon manager, which maintains information about the status of platoons, or "virtual vehicles", and their environment. The platoon manager is also assisting or controlling the platoons based on enhanced situational awareness. The service must be ubiquitous along the route of the platoon, despite of changing of the serving MEC server. The service may need to continue even, if the platoon roams out of the serving operator's network coverage area, for example, when crossing a country border.

B. Shared World Model

Automated vehicles rely on their embedded environmental perception sensors, such as cameras, LiDAR, radar to build the so-called *world model* that is used as basis for crucial tasks such as path planning. The ego vehicle's world model provides a high-level description of the vehicle's surroundings which can be extended with SWM data, coming from other sources when connected roadside services.

In this paper, we consider vehicles that continuously push their local world model data to a MEC-hosted application that maintains a broader view of its served geographical area. In this way, vehicles traveling in the area can effectively extend their range of awareness and react more timely and efficiently to hazardous situations. Although the relevance of the world model data is mainly local to a single MEC server, neighboring MEC servers may share some overlapping geographical areas or data of interest. Maintaining an extensive world model of the surroundings, therefore, requires information exchange between MEC servers and in some cases even between MEC servers hosted by different operators.

IV. System Architecture

This section introduces the overall system architecture and design decisions of the connected and automated driving MEC applications. It also analyzes the requirements of the applications on data synchronization within the MEC system.

A. Overview

The overall system architecture considered in this paper is illustrated in Fig. 1. The architecture includes two types of vehicles, namely connected and automated vehicles. Vehicles use 4G/5G connectivity (Uu interface) to access the cloud services (V2N), whereas 4G/5G (PC5 interface) or ITS-G5 are used for communicating with the other vehicles (V2V) and roadside infrastructure (V2I). The roadside infrastructure can provide information to the cloud services via the Uu interface. Depending on the prevailing business model, vehicles and roadside infrastructure may connect to different mobile network operator (MNO) networks. Two MNO networks are included in Fig. 1 for the sake of completeness.

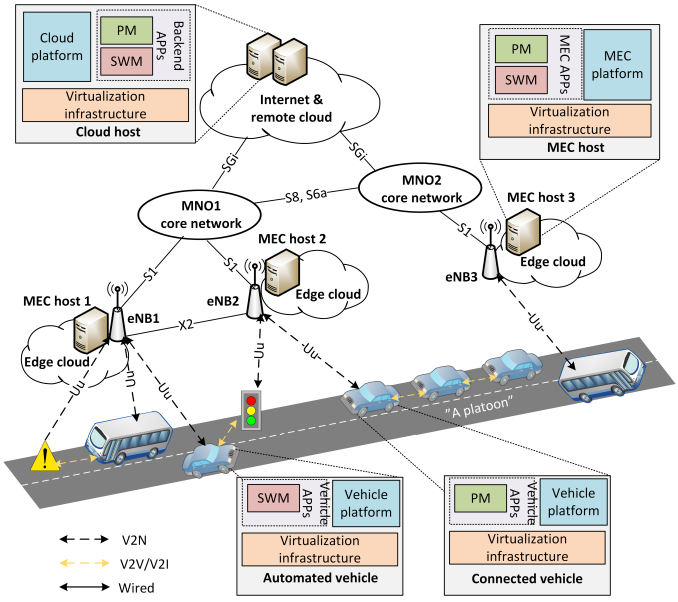

Fig. 1. A network and node architecture for the connected and automated vehicle applications.

The distributed cloud infrastructure in Fig. 1 includes a remote cloud facility and edge clouds, implemented with MEC and located physically closer to the vehicles at the edge of the mobile network. The figure also illustrates the placement of the PM and SWM applications within the architecture. The PM and SWM services are implemented using distributed application functionalities running in the vehicles, edge, and cloud. For example, for PM, this means that non-time critical tasks are handled by the backend PM application (e.g. handling AAA of participants wanting to platoon or rendezvous point planning), near-time critical task are placed in the MEC hosts (e.g. managing process of vehicles joining or leaving platoon or SWM services), and finally the platooning function, residing in vehicles, maintain the vehicle in a platoon (keeping distance, speeding up or slowing down, changing lanes, etc.). Fig. 1 also suggests that virtualization technologies may be utilized for flexible application function management in the architecture.

For communicating application data or instances between the MEC hosts for synchronization or relocation purposes, four main routing options are depicted in Fig. 1. In the MNO1 network, the communications can take place via the direct X2 interface between eNB1 and eNB2 as long as there is enough capacity. Alternatively, the data goes through the core network (S1). When exchanging data between MEC applications hosted by different MNOs, the communications takes a longer route via both the MNO1 and MNO2 core networks. Finally, data that is neither time-critical nor bandwidth intensive may be distributed via the backend applications.

B. The Platoon Management Application

The PM application design, focusing on the functionalities located in the vehicle and MEC, is depicted in Fig. 2. In the MEC host, the Platoon Manager component manages the local platoons in the area. The state of a platoon is stored in a local platoon status database. It also facilitates platoon formation and platoon disengaging, as well as vehicles joining/leaving the platoon. Other vehicles that are not yet in V2V communication range can use the PM application to start communicating with nearby platoons during the formation process. Furthermore, it provides platoon-specific tactical-level advice, concerning traffic lights, lanes, speed, engaging, and disengaging. Other functionalities, residing in the cloud (not depicted), take care of matching vehicles with existing or new platoons and delegates tactical decisions to the correct MEC server.

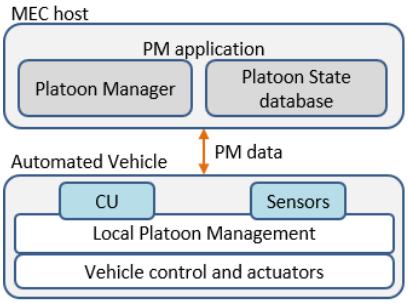

Fig. 2. The platoon management application architecture.

In a dynamic situation, as moving vehicles represent, the changing of a MEC host becomes inevitable. To operate seamlessly, the PM application has specific requirements for data synchronization among multiple MEC servers:

- **Type:** Handover event-based synchronization. Platoon states need to be transferred only for platoons that are leaving the local PM (control) area in order to avoid interruptions in the vehicle merge/leave processes or loss of information on the current platoon state.
- **Session continuity support:** Yes. Platoon formation can happen any moment, which means that the platoon status can possibly change during a handover from a local PM area to another.
- **Type and amount of data transferred:** Platoon status messages are sent by each vehicle to the PM at every second. Advisory data messages for the searching and formation platooning phases are sent by the PM to

vehicle at every second. The three messages together account for approximately between 200 and 300 bytes.

- **Criticality of the data:** The messages concern tactical-level advices that are not time-critical but at the same time affect the platoon behavior from a safety point of view, since vehicles are expected to follow the advices. When vehicles are in V2V range, V2V communication will take over the operational control.

C. The Shared World Model Application

The SWM application architecture is shown in Fig. 3. In the vehicle, a *vehicle world model* is continuously updated based on the fusion of both the ego and SWM data. The ego world model is based on internal sensor data and the SWM is collected from the MEC host. Control and actuator components use the vehicle world model for tasks, such as path planning, requiring perception knowledge of the environment.

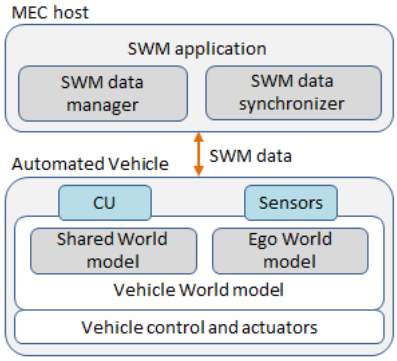

Fig. 3. The shared world model application architecture.

In the MEC host, two main components are in place. The SWM data manager collects SWM data coming from multiple vehicles in the vicinity and fuses it to keep a coherent representation of the local environment. It also filters and sends out relevant SWM data back to vehicles to increase their perception awareness. The SWM data synchronizer transfers data to neighboring MEC hosts that might be relevant to vehicles traveling in adjacent geographical regions. Considering that the data transferred to other MEC hosts has a lower level of criticality, the frequency of such synchronization events might be lower compared to local world model updates.

The requirements of the SWM application regarding data synchronization among multiple MEC servers include:

- **Type**: Continuous or periodical synchronization.
- **Session continuity support:** No. World model data concerns periodic data pushed by vehicles that are relevant locally and for a short duration of time.
- **Type and amount of data transferred**: Header, road/lane polygon data, and data about detected objects (relative position, orientation, covariance matrix) accounts for about 3.5 kB per world model message sent when assuming that up to 10 objects are detected simultaneously. The higher the frequency at which these messages are sent, the more up-to-date the extended world model of each vehicle will be. For close distances (e.g., time gap between vehicle and world model object of 0.5 seconds), we assume a minimum of 10Hz. For longer distances, the frequency could be as low as 1Hz for world model data concerning objects pushed to neighboring MEC hosts.
- **Criticality of the data:** Both time and space aspects affect the level of criticality of the data to a particular vehicle. The world model data has its highest relevance when it includes up-to-date (fresh) data about a nearby region that might directly affect the vehicle's path in the coming (milli-)seconds.

D. Management of the MEC Applications

Each MEC system has a central component, called MEO [1], that is aware of the deployed MEC hosts, their capabilities and services, instantiated applications, as well as the network topology. It is also the entity responsible of ensuring that the requirements, indicated by the applications in terms of the resources, services, location, and performance (e.g. maximum allowed latency), are met. MEO selects the target MEC host for an application and triggers the relocation procedure, when needed. In our system architecture, we assume one MEO per MNO, and the PM and SWM applications require specific support from MEO. First of all, the considered applications are most likely installed for a long term in selected MEC hosts, covering the roads where the PM and SWM services are available. The area needs to be split between the individual MEC hosts, with partial overlaps; thus, requiring the knowledge of the overall MEC system topology from MEO. Second, the applications are location-specific. Thus, they will rely on the location service (e.g. geolocation or Cell ID based) of the MEC system. Third, the vehicles or platoons need to connect to the MEC host, serving the geographical area, in which they are located. The MEC host change should be triggered proactively whenever the vehicles or platoons are approaching the border of the geographical area of the serving MEC host or application instance. Also, latency may play a role in the triggering of the MEC host change. Fourth, the MEC host change should be seamless for the PM application. In addition, both the applications require knowledge of the neighboring MEC servers to enable data synchronization. Finally, depending on the business model, information exchange between MEOs and MEC applications hosted by different operators may be needed. This will require collaboration between operators.

V. Initial Performance Assessment

This section provides an initial assessment of the performance of the two selected applications, based on the proposed architecture and designs as well as the specific features and requirements of the applications described in Section IV.

A. Platoon Management

Given the size and frequency of the data messages to be transferred, the traffic load generated by the PM application on the radio access network is very low, i.e. less than 250Kbit/s per platoon in most practical situations (up to 100 vehicles in platoon) both for up- and downlink. Latencies due to data processing and transmission (roundtrip) will be negligible compared to the 1-second heartbeat of the message exchange.

Even when several platoons are simultaneously served by the same base station/MEC host, there will be no performance problems (unless the serving cell is heavily loaded due to traffic from other applications).

In case of a handover of a platoon between base stations the Platoon State database at the current MEC host has to be copied to the MEC host at the other base station. This may be done (proactively!) via the X2 interface between the involved base stations or via the core network that connects the base stations, cf. Fig. 1; if the delay requirements permit, the core network option is preferred. Given a transfer delay requirement of about 1 second, and assuming that the amount of data in the Platoon State database is limited to the number of vehicles in the platoon times the size of the platoon status messages ($<$ 300 bytes) sent by the individual vehicles, we dont expect any capacity and performance problems here. In particular, the core network option for data transfer between MEC hosts seems very well appropriate.

B. Shared World Model

The traffic load generated by the SWM application on the radio access network is considerably higher than for PM, but in most practical situations it won‘t cause any problems. For example, in the case that 50 cars are involved, the traffic rate (for downlink as well as for uplink) will be less than 10 x 50 x 3.5 Kbytes per second, which is about 14 Mbit/s (for downlink, it's important that multicast transmission can be used). Message delays (roundtrip) will in general also be no problem, but considerably higher transmission speeds are required than in the PM case in order to keep roundtrip delay within the system's heartbeat of (max) 10Hz (e.g., in case of 50 cars, 30 Mbit/s transmission speed in downlink and 1 Mbit/s per vehicle in uplink would still suffice).

The SWM data synchronizer in the MEC host transfers data to neighboring MEC hosts at lower frequency than local updates, i.e. one message per second or less. So the traffic load generated for synchronization between MEC hosts will be 1.4 Mbit/s or less per MEC host, if 50 cars per MEC are involved (50 x 3.5 Kbytes). The data transfer between MEC hosts may be done via the X2 interface between the involved base stations or via the core network that connects these base stations, cf. Fig. 1. Given these example values and the system's relatively low heartbeat for this event type we don't expect any performance or capacity problems here. In particular, the core network option for data transfer between MEC hosts seems very well appropriate for this purpose.

C. Next Steps

Overall, this initial assessment does not reveal potential capacity or performance problems for the two applications considered in this paper. Their designs seem well scalable for use in practical circumstances. However, details such as processing times for the involved data analyses and other computations (e.g. for orchestration) have not been taken into account in our considerations. Although we don't expect large effects from these factors, detailed system simulations and experiments in a testbed, or field trials, are needed for further analysis of the applications' performance in various scenarios.

VI. Conclusion

The paper studied the problem of synchronizing application data among multiple MEC servers in order to ensure session continuity and enhanced situational awareness for connected and automated vehicle applications. The analysis was conducted in the context of two applications: Platoon Management and Shared World Models. We proposed an overall system architecture for the vehicular services as well as more detailed application designs, including the placement of the application functionalities in the distributed cloud environment. We also analyzed the requirements of the applications in terms of data synchronization, management, and orchestration. Furthermore, we conducted an initial assessment of the performance of the applications within the proposed system architecture. As a conclusion we can state that the overall concept and the application designs seem well scalable and thus feasible for practical implementation. Therefore, the future work includes the implementation of the proposed applications, which is currently ongoing in AUTOPILOT. We also plan to evaluate the operation of the applications and their data synchronization in a real testbed environment, including multiple MEC servers.

Acknowledgment

The work was conducted in the AUTOPILOT project and in cooperation with TNO Automotive Helmond. AUTOPILOT has received funding from the European Union's H2020 research and innovation programme under grant agreement No 731993. The work was supported also by the Challenge Finland 5G-SAFE project, partially funded by Business Finland. The authors thank for the support and collaboration.

References

[1] ETSI, "Mobile edge computing (MEC): Framework and reference architecture," ETSI GS MEC 003 V1.1.1 (2016-03), March 2016.

[2] 5G PPP, "5G automotive vision," White paper, October 2015.

[3] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks," *IEEE Vehicular Technology Magazine*, June 2017.

[4] J. Heinonen, P. Korja, T. Partti, H. Flinck, and P. Pöyhönen, "Mobility management enhancements for 5G low latency services," in *IEEE ICC'16*, May 2016.

[5] AUTOPILOT, "Project website," www.autopilot-project.eu.

[6] I. Farris, T. Taleb, H. Flinck, and A. Iera, "Providing ultra-short latency to user-centric 5G applications at the mobile network edge," *Wiley Transactions on Emerging Telecommunications Technologies*, 2017.

[7] D. Namiot and M. Sneps-Sneppe, "On micro-services architecture," *International Journal of Open Information Technologies*, vol. 2, no. 9, pp. 24–27, 2014.

[8] 5G PPP Architecture WG, "5G architecture white paper revision 2.0," Online. Available: https://5g-ppp.eu/white-papers/, December 2017.

[9] L. Li, Y. Li, and R. Hou, "A novel mobile edge computing-based architecture for future cellular vehicular networks," in *IEEE Wireless Communications and Networking Conference (WCNC)*, March 2017.

[10] T. Taleb, K. Samdanis, B. Mada, H. Flinck, S. Dutta, and D. Sabella, "On multi-access edge computing: A survey of the emerging 5G network edge cloud architecture and orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, 2017.

[11] K. Haensge, R. Szczepanski, H. Einsiedler, and M. Maruschke, "Moving service functions into edge cloud environments on the fly," in *ITG-Fachtagung Mobilkommunikation*, May 2017.

[12] A. Machen, S. Wangy, K. Leung, B. J. Koy, and T. Salonidis, "Poster: Migrating running applications across mobile edge clouds," in *MobiCom'16*, October 2016.