Demonstration of Mobile Edge Cloud for 5G Connected Cars

Oleksandr Zhdanenko¹, Jianhui Liu², Roberto Torre¹, Stanislav Mudriievskiy¹, Hani Salah¹, Giang T. Nguyen¹, and Frank H. P. Fitzek¹

¹Technical University of Dresden, Germany. E-mail: {firstname.lastname}@mailbox.tu-dresden.de
²Aarhus University, Denmark. E-mail: jianhui.liu@eng.au.dk

Abstract—This demonstration aims to show both the utility of Mobile Edge Cloud (MEC) for 5G connected cars, as well as the impact of MEC server selection (i.e. migration) strategy on latency. The demonstration simulates cars moving according to a realistic model inside the city of Munich, and implements different MEC server selection strategies. A player can steer an ambulance using a controller, and can also choose one among four server selection strategies while driving. With each strategy, the player will experience a different latency.

I. INTRODUCTION

As demonstrated in [?], connected cars are one of the most demanding use cases for 5G. They can increase safety on roads by overcoming slow human reactions. Furthermore, they can improve traffic management, thus reduce fuel consumption and CO₂ emissions.

The realization of connected cars faces several technical challenges. One notable challenge is to manage the traffic in a timely manner, based on data collected from the cars. Mobile Edge Cloud (MEC), a networking concept that enables high computing capabilities at the network edge, can address the aforementioned challenge by performing related processing and data collection tasks close to the cars. However, orchestration among MEC servers to select the server that gives the lowest overall latency is still an open problem.

We present a demonstration through which we illustrate both: (i) the role that MEC can play to realize 5G connected cars, and (ii) the impact of MEC server selection on the latency. Audience can interact with the demonstration through a 3D game in which they drive a car inside a virtual city, with the possibility to switch among different MEC server selection strategies. The demonstration setup is designed so that it can also serve as a testbed for future research on MEC and connected cars.

II. DEMONSTRATION

A. Scenario

This demonstration (Fig 1) is based on a game in which a player drives an ambulance through waypoints in a virtual city. The player controls the ambulance using a gamepad, with the help of in-vehicle camera (Fig 2). The traffic system consists of autonomous cars, with a moderate density. The communication between the ambulance and the autonomous cars is implemented over virtual base stations. More precisely, the ambulance sends its position and destination to a server through the nearest base station. Next, the server calculates the optimal route for the ambulance as well as the optimal behavior of the cars nearby the ambulance. The sever then commands the ambulance and the cars to behave accordingly. In such a scenario, both the communication latency and the computational latency should be low to achieve timely, optimal traffic management.

Fig. 1: Demonstration Setup

Fig. 2: In-vehicle View
The demonstration implements the following four modes, each representing a MEC server selection (i.e. cloud migration) strategy:

1) **Static cloud (no migration):** An instance of the service is deployed on a server located in a (likely far away) datacenter. The communication latency here is high.

2) **Distance-based MEC:** The MEC is automatically migrated to the base station geographically closest to the ambulance. This results in a significant reduction of the communication latency. However, the computational latency can vary, depending on the server’s load (in terms of CPU, RAM, and storage).

3) **Load-based MEC:** The MEC is automatically migrated to the base station hosting the least loaded server. The computational latency is low in this case. However, this strategy cannot guarantee a satisfactory communication latency.

4) **Distance and load-based MEC (hybrid):** The MEC placement depends on both the distance to the ambulance and loads of MEC servers. The goal here is to avoid selecting servers that can result in critical total latency values.

The capabilities and limitations of each strategy will be demonstrated. In particular, the player and the spectators will notice the differences between the four strategies, in terms of the total time the ambulance needs to drive through all waypoints as well as the reaction time of the other cars.

**B. Implementation**

The demonstration consists of the following four interconnected entities, which are realized by several technologies:

1) **Cars** are simulated entities implemented in Unity 3D [2]. One car (the ambulance) is manually steered by a participant via a controller, while the rest of the cars select their ways randomly.

2) **Data collectors** aggregate data from the cars, including their GPS coordinates and odometry. The realization of the data collectors highly depends on the scale of the coordination application, ranging from tailor-made pub-sub software to highly capable frameworks, like Kafka [7]. The collection can be performed with multiple aggregation points organized into a hierarchy.

3) **Analytical entities** coordinate activities of the cars traveling in a certain coverage area, making sure to avoid collisions. More precisely, these entities calculate the best location that the steered car should reach next, visualized as a circle on the road. The calculated location is also sent to the nearby cars to avoid collisions.

The migration of the analytical entities can be done either reactively using state-of-the-art virtualization technologies, such as virtual machines or containers, or proactively as described in [7].

4) **MEC orchestrator** is in charge of selecting the server using one of the above-described migration strategies.

The setup and the layout of the demonstration are shown in Fig. 1 and Fig. 3, respectively. The cloud and the MEC infrastructure, the data collectors, and the analytical entities are encapsulated in Docker containers, and orchestrated by Kubernetes [7]. The MEC orchestrator is developed on top of Kubernetes. We use a PC for the MEC orchestrator and another PC for visualization. In addition, we use nine Raspberry Pis, each representing the computing infrastructure of one base station. To make it easy for the audience to identify the active base station, i.e. the one to which the MEC is migrated, a light toy fan is automatically activated on the corresponding Raspberry Pi.

A preliminary version of this demonstration has been shown at the IEEE 5G Summit in September 2018. A teaser of the demonstration is available at [2].

ACKNOWLEDGMENT

This work has been supported by the Federal Ministry of Education and Research of the Federal Republic of Germany (BMBF) in the framework of the 5G NetMobil project with funding number 16KIS0687, ECSEL JU under the project H2020 737469 AutoDrive, and the European Union’s H2020 research and innovation program under grant agreement H2020-MCSA-ITN- 2016-SECRET 722424. The authors alone are responsible for the content of the paper.

REFERENCES