Can Edge Computing fulfill the requirements of automated vehicular services using 5G network ?

Wendlasida Ouedraogo, Andrea Araldo, Badii Jouaber, Hind Castel, and Remy Grunblatt

Telecom SudParis, Institut Polytechnique de Paris; France; {firstname.lastname}@telecom-sudparis.eu

Abstract—Communication and computation services supporting Connected and Automated Vehicles (CAVs) are characterized by stringent requirements, in terms of response time and reliability. Fulfilling these requirements is crucial for ensuring road safety and traffic optimization. The conceptually simple solution of hosting these services in the vehicles increases their cost (mainly due to the installation and maintenance of computation infrastructure) and may drain their battery excessively. Such disadvantages can be tackled via Multi-Access Edge Computing (MEC), consisting in deploying computation capability in network nodes deployed close to the devices (vehicles in this case), such as to satisfy the stringent CAV requirements. However, it is not yet clear under which conditions MEC can support CAV requirements and for which services. To shed light on this question, we conduct a simulation campaign using well-known open-source simulation tools, namely OMNeT++, Simu5G, Veins, INET, and SUMO. We are thus able to provide a reality check on MEC for CAV, pinpointing what are the computation capacities that must be installed in the MEC, to support the different services, and the amount of vehicles that a single MEC node can support. We find that such parameters must vary a lot, depending on the service considered. This study can serve as a preliminary basis for network operators to plan future deployment of MEC to support CAV.

Index Terms—5G Simulation; MEC; Connected and Automated Vehicles

I. INTRODUCTION

In order to safely and efficiently perceive their surroundings, make real-time decisions, and navigate complex environments, CAVs need some computational resources, e.g., CPU, GPU, and RAM, to perform some algorithmic tasks. Computational requirements may fluctuate depending on road traffic conditions. Installing a large amount of computational resources to be ready to perform peak computation at any moment is very costly and would make vehicles too expensive, thus severely restricting the market for CAV. Moreover, a large amount of computation consumes power, thus reducing vehicle autonomy or requiring big, heavy, and expensive batteries. Cloud Computing partially addresses this problem by offering offloading capabilities. However, CAVs have stringent requirements regarding latency and bandwidth that can hardly be met by Cloud Computing [1]. Hence, MEC emerges as an alternative capable of meeting those requirements, thanks to deploying computational resources very close to vehicles, i.e., in the base stations or in the Road Side Units (RSUs). Also, the reliability, the high bandwidth, and the extensive coverage offered by 5G networks, together with the integration of MEC,

offer promising conditions for a large penetration of CAVs. In this context, a question remains open:

With which computational capacity should MEC nodes be equipped in order to support the different CAV services?

To answer this question, we first present a basic queueing theory model to find the lower bounds on the required computational resources (§IV). We then perform a simulation campaign (§V-VI). We simulate the realistic mobility of vehicles using SUMO. Packet exchanges between vehicles and the MEC, as well as CAV service computation, are simulated using essentially OMNeT++ [2], Simu5G [3], and Veins [4]. Our code, released in open source[1] can be used by other researchers as a base to study the feasibility of MEC-based CAV services.

We find that for some CAV services, such as remote driving and cooperative sensing, the amount of required MEC resources is high, and only a few vehicles can be supported at the same time by a MEC node, making MEC a difficult avenue to follow. For other CAV services such as cooperative maneuver and awareness, instead, MEC is more promising, being able to support a larger number of vehicles.

II. BACKGROUND AND MOTIVATION

CAVs are categorized into different levels of automation, as defined by the Society of Automotive Engineers (SAE) [5]. These levels range from level 0 (no automation) to level 5 (full automation). For this work, we consider scenarios with a higher level of automation (from Level 3 to 5), which means that vehicles can perform the majority of driving tasks, such as platooning, collision avoidance, or cooperative sensing.

In these scenarios, vehicles communicate using vehicle-to-everything (V2X) communications, which can be with other vehicles (vehicle-to-vehicle or V2V), the infrastructure (vehicle-to-infrastructure or V2I), pedestrians (vehicle-to-pedestrians or V2P), or the network (vehicle-to-network or V2N). Performing in the edge node computation pertaining to the aforementioned driving tasks relies on V2N communications, enabling communication with the MEC. In this case, we talk about vehicle-to-edge and edge-to-vehicle communications.

Moving a large part of driving-related computation from vehicles to edge nodes provides several advantages. Such advantages have been shown for platooning [6], but they

---

[1]https://github.com/zazim13/Simu5G-MecBasedAV

TABLE I: Use cases requirements

Use cases	End-to-End Latency Threshold	Reliability
Remote driving	20ms	99%
Cooperative sensing	10ms	95%
Cooperative maneuver	100ms	99%
Cooperative awareness	100ms	95%

Fig. 1: Communication behavior of type "Dissemination".

➤ 1: Sending data to MEC
➤ 2: Answering with processed data
➤ 3: Disseminating processed data

extend, more generally, to various CAV applications. Notably, the MEC facilitates interoperability among diverse CAV hardware, enabling a central controller to simplify collaboration between different vehicle types. In addition, edge nodes can be made more resilient by appropriate redundancy practices. If appropriately positioned, edge nodes can be less susceptible to shadowing when communicating with vehicles. Finally, an edge node can aggregate information from multiple vehicles, fostering enhanced cooperation and facilitating global decision-making, as showcased in [7] and [8].

However, while the MEC provides improved latency compared to the Cloud, it does introduce some latency in contrast to on-board computations. Studies such as [9] suggest balancing the computational load between the edge node and the Cloud, based on latency considerations. In our approach, we only focus on leveraging the MEC, as our goal is to comprehend its limitations in supporting CAV services.

III. VEHICULAR SERVICES AND ARCHITECTURE

A. Vehicular applications

We consider the following CAV services, related to the high degree of automation, characterized by stringent requirements, as described in [10].

- *Remote driving* allows vehicles to be driven by a human operator or an application outside the vehicle.
- *Cooperative sensing* involves exchanging sensed data to enhance a vehicle's environmental perception.
- *Cooperative maneuver* consists of exchanging messages to synchronize vehicles' maneuvers, such as lane changing or platooning.
- *Cooperative awareness*: information exchange to inform vehicles about relevant events, such as Emergency Vehicle Alerts, electronic emergency brake signals, etc.

For these use cases, the acceptable requirements are provided in [10] and [11]. Table I summarizes these requirements. The requirements pertain to **end-to-end latency**, signifying the delay from the transmission of a message to its reception at the application level, and **reliability**, indicating the probability of successfully transmitting data within the end-to-end delay threshold, as in [11].

B. Communication behavior

Our objective is to assess the feasibility of deploying applications in the MEC to serve the use cases described in §III-A. Since our paper is not a study on the internals of the aforementioned applications, we made a deliberate decision not to implement each specific application in detail, which would be time-consuming. Instead, we simulate a generic parametric application. By adjusting the parameters to match the application characteristics (as in Table III), we can mimic the behavior of the different applications at a high level, only focusing on network-related characteristics. By mimicking, we mean generating communications that such an application would perform, consuming the same bandwidth, with the same data rate, and demanding the same processing load. We consider two communication behaviors that represent the most distinct services.

Dissemination. For cooperative applications, data such as cooperative sensing, warning alerts, or traffic flow information needs to be shared with nearby vehicles or RSUs. Our approach emphasizes vehicle-to-edge and edge-to-vehicle communication, following the methodology in [6] for platooning services. In this model, the edge node processes data, runs algorithms, and disseminates results to relevant vehicles. We introduce the 'dissemination radius' parameter, creating a circle around the data-producing vehicle for information sharing. Each vehicle wishing to cooperate sends its data to the edge node, which processes it and shares the information in a unicast manner with all vehicles within the dissemination circle. While broadcast is an option, it complicates limiting dissemination to the specified circle. This behavior is depicted in Fig. 1.

Client-Server. For remote driving service, the edge node is generally used to offload certain tasks, such as object recognition and parking assistance (as in [12]). In this paper, we assume that the edge node is completely in charge of remotely operating cars. This means that remotely driven vehicles send data (videos, LiDAR, or sensed data) to the edge node, which handles the execution of the necessary tasks and sends the right commands, e.g., steering, braking, and accelerating.

We choose a proactive behaviour for all applications, meaning that data are periodically sent to the edge node then shared when needed, and deploying the service is trigering the sending of data, no particular event is waited.

IV. PERFORMANCE CHARACTERIZATION

The total delay is composed of several components, as depicted in Fig. 2. Each application running in the MEC (MecApp), related to one service and one vehicle, is allocated a limited amount of processing resources. For the sake of simplicity, we here only focus on the CPU deployed in an edge

node. We assume that the CPU of the edge node is allocated among several MecApps using a fair sharing discipline [13]. A MecApp operates as a queueing system for incoming packets. Queued packets are processed sequentially, with the processing time determined by the allocated CPU and task complexity. The CPU allocation directly influences both processing time and end-to-end delay, a critical factor for meeting reliability requirements in the context of automated vehicles.

If we aim to achieve a reliability of R_{req}, meaning that more than R_{req} percent of the packets should be successfully transmitted within the end-to-end delay requirement D_{req}, we must ensure that for an end-to-end delay D.

$$P(D \leq D_{req}) \geq R_{req} \tag{1}$$

From §V-A2 the packet generation follows a Poisson process, and we also assume that the arrival rate at the MecApp follows a Poisson distribution. In addition, the processing time is assimilated to an exponential distribution. Given that, each MecApp can be assimilated to a queueing system, we can approximate these queues as M/M/1 queues. Consequently, the time taken by each packet within the MecApp (queueing and processing delay) follows an exponential distribution, with a parameter $\mu - \lambda$, where λ represents the packet arrival rate at the MecApp and μ, the processing rate (in packets/s).

To ensure that the requirement can be met, it's crucial that the time taken by each packet within the MecApp D_{mec} respects the condition (1) i.e. a necessary condition to meeting the reliability requirement is :

$$P(D_{mec} \leq D_{req}) \geq R_{req} \tag{2}$$

then we obtain:

$$\mu \geq \lambda - \frac{ln(1 - R_{req})}{D_{req}} \tag{3}$$

where R_{req} is the overall reliability requirement and D_{req} is the overall delay requirement.

Given that:

$$\mu = \frac{CPU}{E(IPR)} \tag{4}$$

Where CPU is the amount of CPU (in instructions/s) allocated to the MecApp and IPR is defined in §V-A2. The minimum CPU that should be allocated to each MecApp in order to respect the necessary condition (2) is:

$$CPU_{min} = \left(\lambda - \frac{\ln(1 - R_{req})}{D_{req}} \right) \cdot E(IPR) \tag{5}$$

We compute these minimum CPU allocations regarding the requirements presented in Table I. They are summarized in Table III. Observe a notable discrepancy among the CPU required for different services.

Fig. 2: Delay components

Fig. 3: Simulation frameworks diagram

V. SIMULATION ENVIRONMENT

We use several open-source software and frameworks to perform our simulations. Fig. 3 illustrates the diagram of interconnectivity within these frameworks. Our work is mainly based on *Simu5G* [3], a popular open-source 5G simulation library, because it provides 3GPP-compliant 5G New Radio Access and integrates MEC features. *Simu5G* is based on the *OMNeT++* framework [2] and integrates *INET* [14], which is an open-source framework providing tools for communication network simulation. In order to enable a vehicular network with realistic mobility, we use *SUMO* [15]. *SUMO* allows us to generate road networks and traffic demand, specifying routes and vehicles with realistic behavior. *SUMO* provides an API [16] for interacting with the vehicles it generates, and *Veins* (especially its subproject *Veins_INET*), another open-source vehicular network simulation framework allows us to access this API within the *OMNeT++* simulation environment and give to vehicles in Simu5G the realistic mobility provided by *SUMO*. With these tools, we can simulate scenarios aligning with our vehicle-to-edge and edge-to-vehicle communication approach.

A. Simulation parameters

1) Radio conditions: Simu5G offers a number of parameters to tune the 5G New Radio network. It also provides a realistic channel model that supports fading, path loss, shadowing, and attenuations. We choose to use macro base stations in an urban scenario. Other parameters are summarized in Table II and are essentially taken from [17].

2) Use cases specifications: Our generic application aims to mimic real-world application behaviors. To achieve this objective, we studied these behaviors in the literature and collected specifications such as distributions, values, and other

TABLE II: Simulation radio parameters

Parameter name	Value
Number of gNBs	1
Carrier frequency	6GHz
Bandwidth	80MHz(100 PRBs)
Numerology	2
Fading (Jakes) + shadowing	enabled
gNB Tx power	46 dBm
gNB antenna gain	8dBi
gNB noise figure	5dB
UE antenna gain	0dBi
UE noise figure	7dB
Path loss model	(3GPP - TR 36.873)
Blershift	5

(a) Remote driving

MIPS	Vehicles 1	2	3	4	5	6	7
2356230	100	100	40	80	40	40	80
749070	100	100	40	80	0	0	0
412090	100	100	0	0	0	0	0
346350	100	100	0	0	0	0	0

(b) Cooperative maneuver

MIPS	Vehicles 1	10	20	30	40	50	60
2356230	100	100	80	20	0	0	0
749070	100	100	20	0	0	0	0
412090	100	100	0	0	0	0	0
346350	100	100	0	0	0	0	0

(c) Cooperative awareness

MIPS	Vehicles 1	20	40	60	80	100	120
2356230	100	100	100	100	100	100	0
749070	100	100	40	0	0	0	0
412090	100	100	0	0	0	0	0
346350	100	100	0	0	0	0	0

(d) Cooperative sensing

MIPS	Vehicles 1	2	3	4	5	6	7
2356230	100	100	40	80	80	0	0
749070	100	100	60	40	20	0	0
412090	60	60	20	0	0	0	0
346350	100	80	0	0	0	0	0

Fig. 4: Success rate of each Edge CPU

metrics for parameters like uplink or downlink bandwidth usage, data rates, and more. These specifications are summarized in Table III.

It is generally assumed in the literature, as in [18], that the packet generation process follows a Poisson distribution for IoT-based applications. Then, assuming a constant data accumulation by IoT devices, the payload size of each sent packet can be approximated as an exponential random distribution.

The processing time is directly related to the size of the packet to be processed, which follows an exponential distribution, we then also assume that the processing time itself is exponentially distributed. In Simu5G's MEC implementation, it is through instructions required by the packet being processed that the processing time is tuned, as they are proportional. For each use case, we consider that each packet requires certain tasks to be performed. We introduce the parameter IPR for Instructions Per Request i.e. the number of instructions required per each request. We detailed in Table III values of IPR for each use case. We extract them from [12], where we can find task instructions requirements for tasks related to automated vehicles. Each use case is attributed with sufficient IPR to handle essential tasks such as steering control, alongside characteristic tasks corresponding to each use case (e.g., object recognition for remote driving or parking assistance for cooperative maneuvers). We use MIPS (Millions of Instructions Per Second) and MI(Million Instructions) as a measure of computational capability and number of instructions to align with Simu5G.

Once processed, an answer with the processed information is transmitted back to the sending vehicle, and in the case of cooperative applications, to neighboring vehicles as well. These transmitted packets include processed information, such as steering control, alerts, acknowledgments, and more. In [19], a downlink bandwidth of 0.25 MBit/s is assumed for self-driven cars, with downlink packets containing steering control information. With a sending rate of 100 messages per second in our remote driving scenarios, this implies an answering rate of 100 messages per second. Consequently, we assume a payload size of 313 bytes for each packet, considering it to be adequate to accommodate all potential types of responses across various use cases.

Each use case is defined by a communication range, repre-

senting the area over which V2X messages can be transmitted. We adopt a circle of dissemination approach, where the radius of this circle determines the communication perimeter. We assume that this radius follows a uniform distribution within the specified range detailed in [10], as indicated in Table III.

3) Simulation scenario: In our simulations, we essentially tune two parameters: the CPU capacity available at the edge, shared among the MecApps of the cars, and the number of injected vehicles. The processor used and their computational capacities are displayed in Tables IV. Each experiment corresponds to a simulation involving a specific edge node processor and a certain number of injected vehicles. Every experiment is repeated five times with different seeds (i.e., five repetitions in OMNeT++), and each simulation lasts 180 seconds. The road map corresponds to a district of Paris of 1 square kilometer.

VI. RESULTS

Figure 4 illustrates the success rates across various experiments, where the success rate is defined as the percentage of repetitions meeting the reliability requirement out of all experiment repetitions. The red limitation denotes a threshold where CPU allocation meets the minimum requirements for each MECApp as outlined in Table III. Simulations below this threshold fail to meet the reliability criteria. However, simulations above this threshold show improved performance but still encounter challenges despite sufficient CPU allocation for each MECApp. This issue stems not only from delays inherent to the MEC node performance, addressed by the condition in (2) which specifies the minimum CPU allocation required for each MEC application, but also from additional uplink and downlink delays. An increase in the number of vehicles leads to reduced bandwidth per vehicle, consequently increasing the end-to-end delay and highlighting the need for scaling edge nodes in proportion to network resources.

With their speed, processors id3 and id4 can effectively handle scenarios such as remote driving or cooperative sensing

TABLE III: Use cases specifications

Use cases	Uplink bandwidth (Mb/s)	Uplink rate (msg/s)	Uplink payload (Bytes)	Downlink payload (Bytes)	IPR [12]	Dissemination radius	Min CPU(MIPS)
Remote driving	32 [20]	Pois(100)	Exp(40000)	313 [19]	Exp(500)	-	165130
Cooperative sensing	10 [10]	Pois(100) [21]	Exp(12500)	313 [19]	Exp(200)	Uniform(0-200m) [10]	79915
Cooperative maneuver	1.3 [10]	Pois(10) [6]	Exp(16250)	313 [19]	Exp(500)	Uniform(0-500m) [10]	28026
Cooperative awareness	0.12	Pois(10) [21]	Exp(1500) [10]	313 [19]	Exp(200)	Uniform(0-500m) [10]	7992

TABLE IV: Processor Processing Speed

Id	Processor	Processing Speed(MIPS) [22]
1	AMD Ryzen Threadripper	2356230
2	AMD Ryzen 9	749070
3	Intel Core i9-9900K	412090
4	Intel Core i5-11600K	346350

for up to 2 vehicles, awareness tasks for 20 vehicles, and maneuvering tasks for 10 vehicles. However, even doubling the processing speed with processor id2 does not significantly increase the number of vehicles that can be managed, with only a modest improvement observed in remote driving and cooperative sensing services. This underscores the challenge of scaling up the number of vehicles through improving the MEC node performance.

Moreover, our observations indicate that services with more stringent requirements support fewer vehicles given the same edge node capacity. Specifically, services like remote driving and cooperative sensing, which demand lower end-to-end delays, can support fewer vehicles than cooperative maneuver and awareness service. The latter services are therefore more promising.

VII. Conclusion

We conducted a simulation campaign using diverse tools to assess the limits of integrating 5G networks and edge computing via the MEC paradigm to handle various CAVs services requirements. Our focus includes evaluating the feasibility of edge-based control, where MEC manages all required computations. We show a basic queuing model to provision the minimum computational capability to satisfy different MEC application requirements. Our results highlight a discrepancy in required resources for specific CAV services. This indicates that even if controlling CAVs from the edge is feasible, the efficiency greatly varies depending on the service. Additionally, we reveal challenges in improving such efficiency by scaling up the edge, as the communication delay can hinder the correct operation of vehicular services from the edge. A limitation of our approach is that we did not conduct an in-depth study of the communication delay nor did we consider cloud or vehicle computational resources. We plan to incorporate these factors in our future works to provide a more comprehensive analysis.

VIII. Acknowledgments

References

[1] B. P. Rimal, D. Pham Van, and M. Maier, "Mobile-edge computing versus centralized cloud computing over a converged fiwi access network," *IEEE Transactions on Network and Service Management*, vol. 14, no. 3, pp. 498–513, 2017.

[2] "Omnet++." [Online]. Available: https://github.com/omnetpp/omnetpp

[3] "Simu5g." [Online]. Available: https://github.com/Unipisa/Simu5G

[4] "Veins." [Online]. Available: https://github.com/sommer/veins

[5] "The 6 levels of vehicle autonomy explained." [Online]. Available: https://www.synopsys.com/automotive/autonomous-driving-levels.html

[6] C. Quadri, V. Mancuso, M. A. Marsan, and G. P. Rossi, "Edge-based platoon control," *Computer Communications*, vol. 181, pp. 17–31, 2022.

[7] F. Malandrino, C. F. Chiasserini, and G. M. Dell'Aera, "Edge-powered assisted driving for connected cars," *IEEE Transactions on Mobile Computing*, vol. 22, no. 2, pp. 874–889, 2023.

[8] S. Tang, B. Chen, H. Iwen, J. Hirsch, S. Fu, Q. Yang, P. Palacharla, N. Wang, X. Wang, and W. Shi, "Vecframe: A vehicular edge computing framework for connected autonomous vehicles," in *2021 IEEE International Conference on Edge Computing (EDGE)*, 2021, pp. 68–77.

[9] K. Sasaki, N. Suzuki, S. Makido, and A. Nakao, "Vehicle control system coordinated between cloud and mobile edge computing," in *2016 55th Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE)*, 2016, pp. 1122–1127.

[10] M. Boban, A. Kousaridas, K. Manolakis, J. Eichinger, and W. Xu, "Connected roads of the future: Use cases, requirements, and design considerations for vehicle-to-everything communications," *IEEE Vehicular Technology Magazine*, vol. 13, no. 3, pp. 110–123, 2018.

[11] *Service requirements for enhanced V2X scenarios*, European Telecommunications Standards Institute, 2022, 3GPP TS 22.186 version 17.0.0 Release 17.

[12] K. Fizza, N. Auluck, and A. Azim, "Improving the schedulability of real-time tasks using fog computing," *IEEE Transactions on Services Computing*, vol. 15, no. 1, pp. 372–385, 2022.

[13] A. Noferi, G. Nardini, G. Stea, and A. Virdis, "Rapid prototyping and performance evaluation of etsi mec-based applications," *Simulation Modelling Practice and Theory*, vol. 123, p. 102700, 2023.

[14] "Inet." [Online]. Available: https://github.com/inet-framework/inet

[15] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[16] "Traci." [Online]. Available: https://sumo.dlr.de/docs/TraCI.html

[17] A. Virdis, G. Nardini, G. Stea, and D. Sabella, "End-to-end performance evaluation of mec deployments in 5g scenarios," *Journal of Sensor and Actuator Networks*, vol. 9, no. 4, p. 57, Dec 2020.

[18] F. Metzger, T. Hoßfeld, A. Bauer, S. Kounev, and P. E. Heegaard, "Modeling of aggregated iot traffic and its application to an iot cloud," *Proceedings of the IEEE*, vol. 107, no. 4, pp. 679–694, 2019.

[19] S. Neumeier, E. A. Walelgne, V. Bajpai, J. Ott, and C. Facchi, "Measuring the feasibility of teleoperated driving in mobile networks," in *2019 Network Traffic Measurement and Analysis Conference (TMA)*, 2019, pp. 113–120.

[20] V. Cislaghi, C. Quadri, V. Mancuso, and M. A. Marsan, "Simulation of tele-operated driving over 5g using carla and omnet++," in *2023 IEEE Vehicular Networking Conference (VNC)*, 2023, pp. 81–88.

[21] *Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*, Intelligent Transport System, 2013, eTSI EN Std. 302 637-2 V.1.3.0.

[22] "Instructions per second." [Online]. Available: https://en.wikipedia.org/wiki/Instructions_per_second