This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Are Turn-by-Turn Navigation Systems of Regular Vehicles Ready for Edge-Assisted Autonomous Vehicles?

Syeda Tanjila Atik, Marco Brocanelli, *Member, IEEE*, and Daniel Grosu, *Senior Member, IEEE*

***Abstract*— Private and public transportation will be dominated by Autonomous Vehicles (AV), which are safer than regular vehicles. However, ensuring good performance for the autonomous features requires fast processing of heavy tasks. Providing each AV with powerful computing resources may result in increased AV cost and decreased driving range. An alternative solution is to install low-power computing hardware on each AV and offload the heavy tasks to powerful nearby edge servers. In this case, the AV's reaction time depends on how quickly the navigation tasks are completed in the edge server. To reduce task completion latency, the edge servers must be equipped with enough network and computing resources to handle the vehicle demands, which show large spatio-temporal variations. Thus, deploying the same resources in different locations may lead to unnecessary resource over-provisioning. In this paper, we leverage simulations using real traffic data to discuss the implications of deploying heterogeneous resources in different city areas to sustain peak versus average demand of edge-assisted AVs. Our analysis indicates that a reduction in network bandwidth and computing cores of up to 60% and 50%, respectively, is achieved by deploying edge resources for the average demand rather than peak demand. We also investigate how the peak-hour demand affects the safe travel time of AVs and find that it can be reduced by approximately 20% if they would be rerouted to areas with a lower edge-resource load. Thus, future research must consider that traditional turn-by-turn navigation systems may not provide the fastest routes for edge-assisted AVs.**

***Index Terms*— Autonomous vehicles, navigation systems, edge computing.**

I. Introduction

A RECENT report [1] estimates that by 2045 as much as half of new vehicle sales could be Autonomous Vehicles (AVs), which are important components of cutting-edge technologies such as internet-of-vehicles (IoV) [2], [3]. The main advantages of AVs are their ability to provide increased productivity, reduced driver stress, reduced energy consumption, and increased safety [4]. In particular, regular (human driven) vehicle safety is generally measured in terms of reaction time and breaking time. While the breaking time is mainly dependent on the mechanical aspects of vehicles (which might be similar to that of AVs), the average reaction time of humans is about 3/4 of a second [5]. To improve reaction time (i.e., safety) over humans, AVs heavily rely on a variety of sensor-generated data [6] (e.g., radar, camera, lidar, ultrasonic sensor) to identify the environment and carry out safe driving operations automatically. For example, each camera frame must be processed on the most appropriate unit (e.g., CPU, GPU, AI accelerator) to identify features such as lanes and other vehicles directions. The navigation system then uses this information to react by updating the AV's speed and direction [7]. In order to ensure improved safety over regular vehicles, it is thus crucial for the end-to-end response time (i.e., from camera frame acquisition to AV reaction) to be minimized.

Given that AV technology is not yet deployed at large scale and it is still being studied in both academia and industry, there are two deployment strategies to help mitigate the challenges. The first one is to equip the AVs with a very powerful computing system [2] to run heavy tasks locally with low response time. However, this may lead to an increased production cost, which may slow down their sale and widespread adoption. The second strategy is to assist the AVs with edge technology, where they can offload heavy tasks to powerful nearby edge servers [8], [9], and achieve faster data processing with lower response time [10], [11]. Compared to the first strategy, the capital cost necessary to install/maintain such servers (e.g., through government incentives) can be amortized by the fact that they are shared across multiple AVs, can help reduce AVs' cost, and may help speed up the transition to a safer and more efficient traffic circulation. However, this strategy also leads to additional challenges that must be further explored.

When data is being offloaded to the nearby edge servers, the analysis or processing of the data vastly depends on the amount of network and computational resources deployed in that edge server. Since the AV keeps moving after offloading its data, it may travel some distance before the computing result from the edge server is received. We call this distance, the *blind distance* since the AV remains *blind* to new features found in the last data offloaded until the computation result is received. With AVs, the blind distance can be bounded *by design* to a certain value for guaranteed safety, at least from the reactiveness point of view. For example, the system designer may want to guarantee a maximum blind distance for edge-assisted AVs of 5 meters. Thus, *the end-to-end response time must be lower or equal to the ratio of blind distance over speed, which can vary over space and time*. As a result, the amount of computing resources deployed at the edge

Manuscript received 23 September 2022; revised 27 March 2023 and 28 April 2023; accepted 8 May 2023. This work was supported in part by the U.S. National Science Foundation under Grant CCF-2118202 and Grant CNS-1948365. The Associate Editor for this article was M. Shojafar. *(Corresponding author: Marco Brocanelli.)*

The authors are with the Department of Computer Science, Wayne State University, Detroit, MI 48202 USA (e-mail: tanjilaatik@wayne.edu; brok@wayne.edu; dgrosu@wayne.edu).

Digital Object Identifier 10.1109/TITS.2023.3275367

1558-0016 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Wayne State University. Downloaded on May 29,2023 at 19:59:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

responsible for processing the offloaded task can limit the AV's *safe speed* that allows the vehicle to meet the chosen blind distance requirement.

In order to reduce the response time and maximize the safe speed, the edge servers must be equipped with enough network and computing resources to handle the vehicle demands. However, because of the variable number of vehicles on the road, this demand is characterized by large *spatio-temporal variations*. As a result, deploying the same amount of resources to all edge servers and/or deploying the resources necessary to handle the peak-hour demand in a specific area may lead to costly and unnecessary over-provisioning of edge resources. On the other hand, deploying a lower amount of resources to limit costs may lead to lowering the safe speed of AVs on a specific path in order to keep a desired bounded blind distance. This may lead to the problem that modern turn-by-turn navigation systems, which provide the fastest route to destination for regular (human-operated) vehicles, may not be able to provide the fastest route for edge-assisted AVs.

In this paper, we explore the design trade offs among the amount of deployed edge resources, desired blind distance, and resulting safe speed for edge-assisted AVs. We develop a simulation environment that leverages a real vehicle transit dataset [12] from the city of Cologne, Germany, to explore heterogeneous edge server configurations that satisfy peak and average AV demand. We find that, when high safety is required (i.e., short blind distance), the peak configurations lead to high over-provisioning due to the high variability of traffic during the day. Thus, cost savings and better utilization can be achieved by deploying the average configuration. Finally, we study the effect of deploying average configurations on the safe speed of autonomous vehicles for various safety requirements and study its effect on the travel time for random routes in the city. *We find that, due to limitations on available edge resources, modern turn-by-turn navigation systems that provide the fastest route to regular vehicles do not necessarily provide the fastest one for edge-assisted AVs.* We hope the discussion and findings of this paper will inspire and motivate future research on AV navigation systems and algorithms, and help determine how to plan resource deployment for AVs.

Specifically, this paper makes the following contributions:

- To the best of our knowledge, this is the first paper to highlight the challenges deriving from heterogeneous edge computing for edge-assisted AVs. Specifically, we provide the first extensive evaluation of the design trade offs among edge resource deployment, safety, and travel time at different city locations and day times.
- We leverage real traffic data from the city of Cologne (Germany) to study the traffic characteristics, e.g., number of vehicles and average speed, and find the likely configuration of edge resources needed in a particular area to ensure a certain safety bound.
- We study several scenarios and compare how the travel time through the same route changes for regular vehicles and edge-assisted AVs. We show that in some cases the current turn-by-turn navigation system fails to provide the fastest route for the AVs because they do not consider the amount of resources deployed at the edge servers and their load at different day times along the navigation path.
- We have developed a simulation environment in Python that uses the real traffic data to evaluate the results. Our analysis has revealed that a reduction in network bandwidth and computing cores of up to 60% and 50%, respectively, is achieved by deploying edge resources for the average demand rather than peak demand. In addition it has also been found that the safe travel time of AVs can be reduced by approximately 20% if they would be rerouted to areas with a lower edge-resource load.

The rest of the paper is organized as follows. Section II reviews the related work. Section III provides an overview of edge-assisted navigation for AVs and describes the approach used for finding the configuration of edge resources. Section IV describes the dataset along with the results from our experiments. Section V discusses the experimental results and Section VI concludes the paper.

II. Related Work

Given that AVs are still in their infancy and have not been deployed at large scale yet, recent research studies have envisioned they will likely be designed in one of two ways. First, all the on-board sensor data is exclusively analyzed locally by deploying powerful computing resources on each AV. Second, to reduce the amount of computing resources deployed on each AV, part of the heavy computation is offloaded from AVs to powerful edge resources shared among nearby AVs.

Mero et al. [13], consider a single computing module to be installed on board of each AV and capable of handling the complex autonomous driving tasks by processing sensor data through deep learning models trained by leveraging Imitation Learning. Souri [3] analyzes the use of AI in connectivity management systems for the IoV and Vehicular Ad Hoc Network (VANET) environments. The study in [6] introduces a virtual machine (VM) placement algorithm, which utilizes the maximum flow and minimum cut theory in order to achieve excellent QoS while reducing the energy usage of IoT devices in the vehicles. The research presented in [14], [15], [16], [17], and [18] consider each vehicle having a full sensor configuration that can navigate on its own without the need for any cooperation with the other vehicles. Researches in [19], [20], and [21] use LiDAR point clouds to implement Simultaneous Localization and Mapping (SLAM) in AVs. The LiDAR sensors are generally expensive and the resulting computation usually needs power-hungry onboard components such as graphics processing units (GPU) [22]. However, none of the above solutions consider that, according to a recent study [23], adding too many resources on board can considerably reduce the driving range of the vehicle.

In order to deal with these challenges, several studies suggest computational offloading. Several researchers [8], [9], [24], [25], propose to offload part of the heavy computation to the edge. Some studies [26], [27], [28], [29] even proposed to move the LiDARs to a suitable place out of the vehicle and move the GPUs at the edge by connecting them with edge servers to be shared among other AVs. Yang et al. [30]

Authorized licensed use limited to: Wayne State University. Downloaded on May 29,2023 at 19:59:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

mitigate the challenge of low on-board object detection accuracy by intelligently offloading blocks of pixels to the edge. Abdallaoui et al. [31] discuss the most popular path planning techniques proposed in literature for navigation problems involving autonomous vehicles. However, none of the above studies provide insights on how heterogeneous edge servers may affect AV's path planning algorithms. *To the best of our knowledge, this is the first work studying the design trade offs of edge resources for future AVs and their consequences in terms of safety and travel time compared to regular vehicles.* We hope the discussion and findings of this paper will inspire new research directions on edge-assisted AVs.

III. System Model and Methodology

In order to conduct our qualitative analysis, we consider a system model where an edge server is placed at the center of an area. The network and computing resources of that server are shared among all the AVs in that area. There are mainly two ways considered for establishing communication between the vehicles and the edge server. One is Device-to-Device (D2D) direct communication and another is multi-hop communication. In this paper we consider direct communication between AVs and local edge server to reduce latency. Figure 1 shows our example system model. An AV offloads its job (computation) to the nearby edge server while being in location s_0 at time t_0 by using the underlying communication network. After a certain transfer time t, the job is received at the edge server where it is scheduled concurrently with other AVs' jobs, based on a certain policy. After it finishes its execution, the result is returned to the AV at time t_1 in location s_1 and the AV reacts according to the received result, e.g., brake or steer. The distance that the AV travels between job offloading and result reception is defined as the *blind distance*, L. The time taken for a job to finish processing at the edge server from the time it is offloaded is defined as the *response time*, r of that job. When the AV receives the result of the previously offloaded job it acquires a new sensor data and offloads another job. Note that in this paper we want to determine the conditions to satisfy a certain blind distance requirement, so considering the AVs to offload jobs periodically rather than sporadically would not change the results of our study.

A. Assumptions

In order to simplify the problem for the ease of understanding, we have made the following assumptions:

- **AV Computation:** All the AVs have the same autonomous navigation software and the related computation relies on the nearby edge servers. Thus, we assume that, by design, all AVs offload the same amount of per-job computation. Given the complexity of developing a reliable AV navigation software, car manufacturers such as General Motors (GM) are likely going to use the same autonomous navigation software across different vehicles to improve reliability and lower the development complexity. We also extend this assumption across car manufacturers. On the other hand, having heterogeneous jobs offloaded to the edge server is unlikely to change the trend of the results presented in this paper since (1) the approach considered to find peak and average configurations would adapt network and computing resources accordingly, and (2) the main reason for the high variations in edge utilization for peak configuration is due to large hourly traffic variations during each day, which is likely to be consistent across regular and autonomous vehicles.
- **Edge-Server Cores:** Each edge server is equipped with a number of generic logical cores and each of them are able to carryout the requested computation within a bounded worst case execution time. We make this assumption to abstract the complexities of dealing with specific CPU types, frequency, and other hardware characteristics. Each logical core can be translated to a specific CPU type with a certain frequency by the system designer.
- **Edge-Server Network:** The AVs are connected through a wireless network interface with the edge server located in the area they are residing in. To keep our study general and avoid tying the results to a specific network technology (e.g., 5G, DSRC), we determine the required total bandwidth, which is the fastest possible data transfer speed [32]. This will be equally shared by the number of AVs connected to the edge server at a particular time in that particular area while jobs are being offloaded. The system designer can use this information to determine the necessary network technology to install and the channel capacity according to parameters such as transmission power, noise, and distance to vehicles.
- **Edge Job Scheduling:** We assume that each edge server schedules jobs according to the non-preemptive Earliest Deadline First (EDF) policy [33]. Specifically, our system tasks are modeled as a constrained-deadline sporadic task model. We choose this policy because it is commonly used to study the behavior of real-time systems. Other schedulers could easily be employed for testing.
- **Downloading Results:** Similar to related work [34], [35], we assume that the time to send back the computation results from edge servers to the AV is negligible, due its generally small data size.

Based on the above assumptions and setup, first we analyze the hourly average traffic speed and vehicle count using a comprehensive real dataset. Then, we vary the required blind distance to obtain a specific deadline for each city area and hour. In Algorithm 1, which is described in next section, we use this deadline to compute the total network bandwidth and number of logical cores required to avoid deadline misses. We call this the *required configuration* of resources and calculate it for every hour of the day and area.

B. Methodology

In this section, we describe the detailed approach we have used to determine the minimum amount of edge network and computing resources necessary to satisfy the total vehicle demand for a fixed maximum blind distance in a certain

Authorized licensed use limited to: Wayne State University. Downloaded on May 29,2023 at 19:59:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

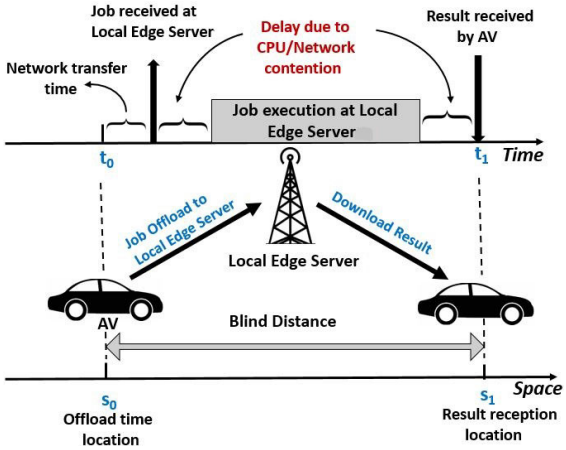


Fig. 1. Overview of the considered system scenario.

Algorithm 1 Configuration-Search

Input: L, blind distance; V, average number of vehicles; S, average speed of the vehicles; D, data size; E, processing time at the edge; W, analysis period; η, multiplicative factor to account for queuing delay at edge server; Δ^b, increment in the bandwidth; Δ^c, increment in the number of logical cores; ϵ^r, response time variation threshold; ϵ^b, maximum bandwidth; M, N, sufficiently large numbers.

1: $d^m \leftarrow M$ ▷ Number of deadline misses
2: $d \leftarrow L/S$ ▷ Relative deadline
3: $t \leftarrow d - \eta E$ ▷ Maximum allowable transfer time
4: $b \leftarrow (D \cdot V)/t$ ▷ Total bandwidth
5: **while** $d^m > 0$ **and** $b < \epsilon^b$ **do**
6: $t \leftarrow (D \cdot V)/b$ ▷ Current transfer time
7: $c \leftarrow 1$ ▷ Number of logical cores
8: $r^{max} \leftarrow 0$ ▷ Maximum response time
9: $\Delta^r \leftarrow N$ ▷ Response time variation
10: **while** $d^m > 0$ **and** $\Delta^r > \epsilon^r$ **do**
11: $d^m, r^{temp} \leftarrow$ SCHED(c, t, E, V, d, W)
12: $\Delta^r \leftarrow \mid (r^{max} - r^{temp})/r^{temp} \mid$
13: $r^{max} \leftarrow r^{temp}$
14: **if** $d^m > 0$ **and** $\Delta^r > \epsilon^r$ **then**
15: $c \leftarrow c + \Delta^c$
16: **if** $d^m > 0$ **then**
17: $b \leftarrow b + \Delta^b$

Output: b, c

geographical area at a specific time of the day. We provide a complete overview of our method in the next sections.

1) Finding Configuration of Resources: Algorithm 1, Configuration-Search describes the procedure of finding the required amount of edge resources for a particular area, time of the day, and blind distance.

The algorithm first initializes the number of deadline misses to a sufficiently large number M, which lets the *while loop* in Line 5 to execute at least the first iteration. In Line 2, the deadline for every job offloaded by the AVs is calculated based on the input blind distance L and input average speed of the vehicles S. In Line 3, we calculate the maximum allowable time t for a job to be transferred to the edge server after it is offloaded. Here, the parameter $\eta \geq 1$ accounts for any queuing delay that may occur at the edge server before the job starts executing, e.g., due to contention with jobs of other AVs. In Line 4, we calculate the initial total bandwidth b required to satisfy the maximum transfer time t for all vehicles served by the same local edge servers. The *while loop* in Line 5 then searches for the minimum number of logical cores necessary to have zero deadline misses (starting from one core in Line 7) and for a progressively increasing network bandwidth, which reduces the transfer time t in Line 6 in each iteration.

In Line 8, the maximum response time r^{max} is initialized to 0. In Line 9, the response time variation Δ^r is initialized to a sufficiently large number N, which lets the *while loop* in Line 10 to execute the first iteration. The algorithm then uses the SCHED algorithm (which is discussed in the next subsection) to schedule vehicle jobs on the edge server and find the maximum response time r^{max} as well as number of deadline misses d^m. This analysis is performed based on the current transfer time t, the number of logical cores c, the average number of vehicles V, the deadline calculated in Line 2, and the time period considered for the analysis W. The *while loop* in Lines 10-15 keeps increasing the number of logical cores by a factor of Δ^c (in Line 15) in each iteration until it leads to zero deadline misses or it finds that increasing the number of logical cores c only leads to a negligible reduction in response time. To check for the latter condition, the response time variation Δ^r is calculated in Line 12 to find how the response time changes compared to the previous iteration. If the response time variation Δ^r is less than a threshold ϵ^r, the algorithm determines that the current bandwidth b is a bottleneck to the system performance, so it needs to be increased. Hence, it is incremented by a factor of Δ^b in Line 17 and the first *while loop* in Line 5 starts executing again with the new total bandwidth b. The number of logical cores is reset to one (Line 7) and the search continues as described above. This process (Lines 6-17) is repeated to increase the network and computing resources in each iteration until it obtains zero deadline misses or the bandwidth reaches a predefined maximum value ϵ^b. When it terminates, the algorithm returns the amount of bandwidth b and number of logical cores c needed to ensure that all the vehicle's offloaded jobs are processed within the deadline.

2) Processing of Jobs at the Local Edge Servers: Algorithm 2, SCHED, describes the process of how offloaded jobs from each AV are processed at the local edge servers. It selects the vehicle job that arrives at the edge server according to the EDF scheduling policy and returns the number of deadline misses as well as the maximum response time of the vehicle jobs at the end of the analysis period W. It first initializes the current time unit of the schedule k, number of deadline misses d^m, and maximum response time of a job r^{max} to 0 (Lines 1-3). In Line 4, based on the input average number of vehicles V and the required number of logical cores c, the number of vehicles V^c served by each logical core is calculated. For simplicity, we consider the job processing and job response time at the heavily loaded logical core. That is the logical core that serves the highest number of vehicles assuming a balanced vehicles-per-core allocation, which is

Authorized licensed use limited to: Wayne State University. Downloaded on May 29,2023 at 19:59:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Algorithm 2 SCHED

Input: c, number of logical cores; t, transfer time; E, processing time at the edge; V, average number of vehicles; d, relative deadline of the jobs; W, analysis period.

1: $k \leftarrow 0$ ▷ Current time unit of the schedule
2: $d^m \leftarrow 0$ ▷ Number of Deadline misses
3: $r^{max} \leftarrow 0$ ▷ Maximum response time
4: $V^c \leftarrow \lceil V/c \rceil$ ▷ Number of vehicles per logical core
5: Let v_i indicate a specific vehicle i.
6: **for** $i \in [1, V^c]$ **do**
7: $o_i \leftarrow 0$ ▷ Time of first offload of vehicle i
8: **while** $k < W$ **do**
9: **for** $i \in [1, V^c]$ **do**
10: $a_i \leftarrow t + o_i$ ▷ Job arrival time at the edge
11: $\delta_i \leftarrow o_i + d$ ▷ Absolute deadline of the job
12: $S \leftarrow \{v_i | a_i \leq k, i \in [1, V^c]\}$
13: **if** $S = \emptyset$ **then**
14: $k \leftarrow k + 1$
15: **else**
16: $j \leftarrow \text{argmin}_{v_i \in S}\{d_i\}$ ▷ Earliest deadline Job
17: $s_j \leftarrow k$ ▷ Job processing start time of vehicle j
18: **if** $s_j > W - E$ **then**
19: **break**
20: $f_j \leftarrow s_j + E$ ▷ Job completion time of vehicle j
21: $k \leftarrow f_j$
22: **if** $f_j > \delta_i$ **then**
23: $d^m \leftarrow d^m + 1$
24: $r \leftarrow f_j - o_j$
25: $r^{max} \leftarrow \max\{r, r^{max}\}$
26: $o_j \leftarrow f_j$

Output: d^m, r^{max}

why we use the ceiling function in Line 4. In Line 7, the algorithm initializes the time o_i to zero when the jobs are offloaded by the vehicles V^c. In Line 10, the arrival time of the vehicle jobs at the edge server queue, a_i, is calculated based on the job offload time, o_i, and job transfer time to the edge, t. In Line 11, the absolute deadline of the jobs δ_i is calculated based on the job offload time o_i and the relative deadline d. Here the subscript i denotes a particular vehicle $v_i \in V^c$.

In Line 12, the vehicle set S is initialized with the vehicles whose job's arrival time a_i is within the current time slot k. Among the vehicles in the set S, the vehicle j having a job with the earliest deadline is chosen to be processed and the current time slot k is updated to when the job processing is finished. As long as the vehicle set S is not empty, the jobs of the vehicles in this set are processed in this manner. This is done in Line 16 to Line 21. If the job completion time f_i is greater than its absolute deadline δ_i, the count for deadline missed jobs d^m is incremented by 1 in Line 23. Every time a job finishes processing, its response time r is calculated in Line 24. Among all the response times, we consider the maximum response time r^{max} to guarantee safety for the AVs in the worst-case scenario (Line 25). The next job offload time o_j of vehicle j is calculated when its previous job finishes its processing at the edge server (Line 26). When the vehicle set S becomes empty, the current time slot is advanced by 1 unit (Lines 13-14). This process is repeated until the end of analysis period (i.e., *while loop* in Line 8). Finally, the algorithm returns the maximum job response time r^{max} and the total number of jobs that missed deadlines, d^m.

Complexity Analysis. The computational complexity of Algorithm 1 can be analyzed by considering ϵ^b, the maximum bandwidth of the network and Δ^b, the increment in the bandwidth, since the *while loop* in Line 5 executes at most $\lceil \epsilon^b/\Delta^b \rceil$ times. Furthermore, Algorithm 2 is executed in each iteration of the second *while loop* in Line 10. The computational complexity of Algorithm 2 can be analyzed considering W as the length of the total analysis period and V as the maximum number of vehicles per logical core c. Therefore the computational complexity of Algorithm 2 is $O(WV)$. In each iteration of the *while loop* in Line 10 of Algorithm 1, the input number of logical cores c is incremented up to V/η, which leads to computational complexity of the loop to be $O(WV \sum_{i=1}^{V/\eta} 1/i) = O(WV \log V)$. Therefore, the computational complexity of Algorithm 1 is $O(\lceil \epsilon^b/\Delta^b \rceil WV \log V)$.

3) Edge-Assisted AVs Safe Speed: From the required edge resource configuration of each hour of the day (found using Algorithm 1), we determine two types of configurations. The *peak configuration* is calculated as the maximum required network bandwidth and number of logical cores over all the hours. That means the peak configuration handles the peak demand without causing any deadline misses of the vehicle jobs. The *average configuration* is calculated as the average of the required network bandwidth and the number of logical cores over all the hours. The average configuration handles the average demand. Afterwards, in order to determine the edge assisted AV's performance with the average configuration and real traffic data from a dataset (see Section IV-A), we leverage simulations that use Algorithm 2 to schedule jobs at the edge server. Specifically, we have used the transfer time calculated from the network bandwidth and number of logical cores found in the average configuration to obtain the maximum response time r^{max} of the jobs at each hour of the day for a certain blind distance, L. Then, we have calculated the AV's safe speed s^s at a particular time and area as follows:

$$s^s = \min\left\{\frac{L}{r^{max}}, s^t\right\}$$

Because the dataset does not provide speed limits, we assume that the AVs cannot exceed the average traffic speed s^t of regular vehicles recorded in the dataset at each specific hour and area. In fact, regular vehicles usually travel either at the maximum speed allowed or at a reduced speed in case of congestion. Note that, the average configuration handles the average demand, but it may lead to deadline misses if AVs travel at regular vehicle speed, especially at rush hours. Thus, for every deadline miss we use the maximum response time to calculate what the maximum safe speed of the AVs should have been to meet the required blind distance, i.e., L/r^{max}. In the next sections we test the effect of various blind distances on the requirement of network and computing resources, and on the AV's safe speed in terms of those two configurations.

Authorized licensed use limited to: Wayne State University. Downloaded on May 29,2023 at 19:59:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Finally, we use this safe speed to further investigate the travel time of the AVs in several routing scenarios.

IV. Experimental Results

In this section, we leverage the approach described in Section III to conduct a qualitative study on the implications of assisting AVs from the edge. Specifically, we first describe our experimental setup to discuss the baseline and the real-world dataset used to conduct our study. Second, we use the data from this dataset to execute Algorithm 1 and compare the peak and average resource configurations in terms of network bandwidth and number of logical cores to deploy in different city areas. Third, we study how using the average configuration affects the AVs' response time and evaluate the safe speed to satisfy a desired blind distance. Finally, we examine the effect of the safe speed on the travel time of edge-assisted AVs compared to that of regular vehicles using the described baseline.

A. Experimental Setup

We have coded the approach described in Section III in Python. The simulations have been executed on an Intel Core i7-8750H CPU at 2.20GHz and 24GB DRAM.

1) Baseline: To the best of our knowledge, there is no existing work focusing on the design trade offs of heterogeneous edge resources providing navigation assistance to AVs. Therefore, we leverage as baseline one of the most popular turn-by-turn navigation systems, i.e., Google Maps. After selecting a random start and end points, we use it to determine 1) the route for regular vehicles with the minimum travel time, and 2) the travel time on an alternative route. We then compare the regular vehicles' travel time on these routes with the AVs' travel time, estimated based on traffic speed and edge resources in each area along the selected route. As a result, this baseline allows us to determine whether modern turn-by-turn navigation systems are ready for edge-assisted autonomous vehicles.

2) Dataset Description: In order to ensure that our experiments are based on realistic data, we have used the vehicular mobility dataset of Cologne, Germany [12]. The dataset includes 24 hours of car traffic traces comprising an area of 400 square kilometers of a typical working day. Although the dataset is from 2013, it is quite complete because it contains timestamps, anonymized vehicle IDs, vehicle speed, longitude and latitude coordinates of vehicles with a one second time granularity. Unlike other newer datasets, it is not restricted to a certain kind of vehicle such as bus or taxi, and includes data of vehicles traveling through both minor and major city roads. Due to the vast amount of data in the dataset, we extracted data for nine sample areas including the heavily congested ones. We call those areas A1 through A9 as shown in Figure 2. Based on the range of typical vehicular wireless interfaces (e.g., DSRC), we choose each area to be of 2 km by 2 km in size. We also assume that the edge servers are located in the center of each area.

For each area, we have calculated the average number of unique vehicles and their average speed in each hour. Figure 3 shows the heatmaps of the average number of vehicles and

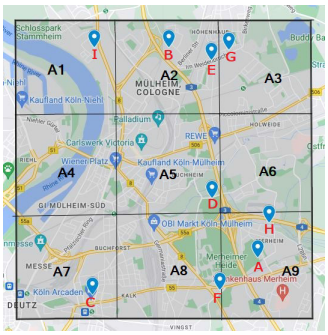
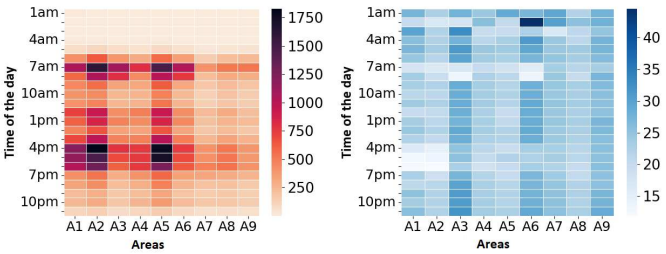


Fig. 2. Selected area locations.

TABLE I
Experimental Parameter Values

Parameter	Value
Blind distance (L)	[2-20] meters
Data size (D^{size})	1.8 Mb
Edge processing time (E)	16 ms
Bandwidth increment (Δ^b)	2 Mbps
Logical core increment (Δ^c)	5
Working period duration (W)	60 seconds
Initial value of d^m and Δ^{res}	100
Max bandwidth (ϵ^b)	118 Gbps
Min response time variation (ϵ^r)	0.005 seconds
Edge queuing delay factor (η)	2



(a) (b)

Fig. 3. Heatmap of (a) the average number of vehicles and (b) the average speed (mph) of the vehicles in the nine areas during the entire day.

average speed for each hour in the nine areas extracted. According to the results in Figure 3(a), similar to most major cities, 7 am and 4 pm are the rush hours of the day with the highest vehicle count in each area. For example, at 4 pm areas A2 and A5 have an average of 1800 vehicles, areas A3 and A6 have 750 vehicles, and areas A7, A8, and A9 have nearly 400 vehicles. Because of the variable traffic at different times of the day and areas, the average vehicle speed is also variable. According to Figure 3(b), during the rush hours the average speed in the area A2 and A5 is less than 16 mph, while areas A3, A6, and A9 have higher speed of 24 mph because of their lower number of vehicles. In order to analyze the effect of having edge-assisted autonomous vehicles in variable traffic and speed, we have used the data of Figures 3(a) and 3(b) as inputs for Algorithm 1. Specifically, while we examine the general results for all nine areas, in the next sections we are going to provide in-depth results by focusing on areas A3, A5, and A7 since they are representative of moderate, heavy, and

Authorized licensed use limited to: Wayne State University. Downloaded on May 29,2023 at 19:59:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

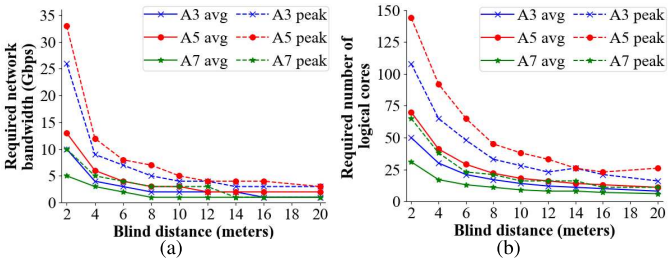


Fig. 4. Comparison of (a) the required network bandwidth and (b) the required number of logical cores between peak and average configurations for areas A3, A5, and A7.

low traffic areas, respectively. The values of the parameters used in our experiments are given in Table I.

3) Robustness to Dataset: We have used the number of regular vehicle data from the above-described dataset in our experiments as it reports the traffic density of different areas at different times. In the near future, when AVs will be deployed at large scale, we can expect to have similar traffic density with similar traffic patterns, e.g., rush hours in early morning and afternoon. Also, the AVs would have to go as fast as the traffic speed of the regular vehicles when there will be traffic to avoid safety concerns. Our approach can be used for any number of vehicles and different cities. In addition, we expect that the trend of the results would not change much as there would always be peak hours and traffic congestion on the road.

B. Blind Distance vs. Edge Resources

In this section, we use the results from Algorithm 1 to study how the resources deployed at the edge change for different input requirements and city areas. The typical human delay to start reacting to a perceived danger is 3/4 of a second [5]. This means that at a speed of 35 mph (typical on city roads) to 60 mph (typical on highways), regular vehicles travel 12 to 20 meters before the human starts reacting, respectively. Since the target of AVs is to improve safety over regular vehicles, in the following study we only consider a range of blind distances of up to 20 meters.

1) Blind Distance vs. Network Bandwidth: Figure 4 shows the network bandwidth and the number of logical cores required for different blind distances while considering the average and peak configurations for the three sample areas A5, A3, and A7, which are representative of higher to lower traffic density, respectively. As discussed earlier, the peak configuration handles the peak demand without deadline misses. As shown in Figure 4(a), in the case of peak configuration, the network bandwidth requirement is 33 Gbps (Giga Bit per Second), 26 Gbps, and 10 Gbps for areas A5, A3, and A7, respectively, considering a blind distance of 2 meters. Thus, area A5, which is the most heavily congested, requires 27% and 230% more network bandwidth than areas A3 and A7, respectively. *This demonstrates that deploying the same amount of network resources in all city areas would not be a reasonable choice.* The peak network bandwidth could be considerably reduced by allowing a higher blind distance, which would come at the cost of a lower AV safety. For example, increasing the blind distance from 2 meters to 10 meters in A5 would reduce the needed peak bandwidth by 85%. Another way to lower resource deployment is to consider the average configuration, which allows to reduce area A5's bandwidth requirement by 60% and 33% compared to the peak configuration for lowest (2 meters) and highest blind distance (20 meters), respectively. However, as we will discuss in next section, the average configuration may slowdown the AVs compared to the regular vehicles during rush hours to meet the required blind distance.

2) Blind Distance vs. Computing Cores: Figure 4(b) shows the number of logical cores required for different blind distances considering the peak and average configurations. Because of the variable traffic density in areas A3, A5, and A7, using the peak configuration would require 108, 144, and 65 logical cores, respectively, which are enough to sustain the traffic demand at all times of the day. Similar to the network requirements, we observe a large spatial variability in the number of logical cores to be deployed in peak configurations. For example, area A7 requires about 55% fewer logical cores compared to area A5 to sustain peak demand. This further demonstrates the unnecessarily high capital expenditures to deploy a uniform amount of resources city-wide. Furthermore, considering the peak configuration, 144 logical cores are required for blind distance of 2 meters in area A5. For the same previously described reason, the number of required logical cores decreases with the increase of the blind distance. However, using the average configuration, the required number of logical cores is much lower, approximately 50% less than what is required for the peak configuration at 2 meters blind distance. Similarly, a blind distance of 12 meters in area A5 would require 33 and 16 logical cores considering peak and average configuration, respectively.

3) Key Takeaway 1: In summary, using the peak configuration would be beneficial during rush hours (e.g., 7 am) because all AVs would be able to always meet the blind distance requirement. However, rush hours only represent a fraction of the entire day time, which means that the edge resources would be severely over provisioned on average. *A better strategy to improve resource utilization and lower capital cost might be to deploy the average configuration and cap the AV's speed during rush hours to meet a pre-established blind distance. In such case, depending on the specific area, the average configuration can help reduce by 60% and 50% the deployed network bandwidth and the number of logical cores, respectively.*

C. Blind Distance vs. Safe Speed

Given the potentially high capital expenditure savings of deploying the average configurations in different city areas, here we study what would be its effect on the AV's responsiveness and safe speed. Figure 5(a) compares the AV's safe speed at blind distances 8, 12, and 16 meters (i.e., circle, star, and cross markers, respectively) with the average speed of regular vehicles from the dataset (i.e., red triangle markers) for areas A3, A5, and A7 (i.e., solid, dashed, and dotted lines,

Authorized licensed use limited to: Wayne State University. Downloaded on May 29,2023 at 19:59:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

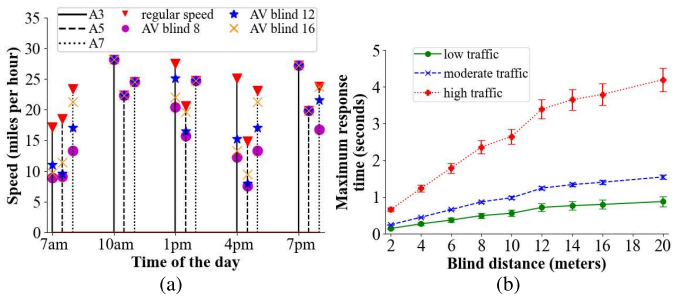


Fig. 5. (a) Comparison of the regular vehicles' speed with the AVs' *safe speed* in the three areas during several hours of the day; (b) Maximum response time for different blind distances with average configuration in area A5.

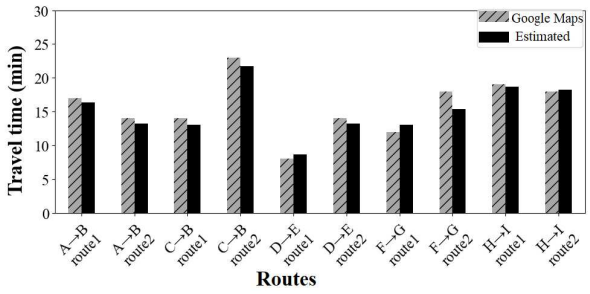


Fig. 6. Comparison of the travel time from Google Maps with the estimated time using the data from the dataset for nine different routes.

respectively). For reasons of clarity, we only show the results at 7 am, 10 am, 1 pm, 4 pm and 7 pm, but other hours show similar trends.

1) Blind Distance vs. AV's Slowdown: To summarize the results across day times, we leverage the traffic density data of the dataset to classify each hour in a low, medium, or high traffic cluster using the K-Means algorithm [36]. On average, the AV's slow down is 0%, 2%, and 68% for low, medium, and high traffic densities, respectively. The worst-case slowdown scenario we found is at 4 pm area A3 and blind distance 8 meters, where the AV's safe speed reduces from 25 mph of the regular vehicles to 21 mph (i.e., 51% slowdown). In general, we also observe a large spatial variation in AV slowdown. For example, at 7pm areas A3 and A5 do not show any slowdown while area A7 experiences an average slow down of 29% and 9% for 8 and 12 meters blind distances, respectively.

2) A Counter-Intuitive Trend: In general, increasing the blind distance increases the deadline of each AV's job, which means that larger response times can still meet deadlines. Figure 5(b) shows how the maximum response time increases with increased blind distance for low, medium, and high traffic. We have shown this result only for area A5 as the other areas exhibit similar trends. We observe that larger blind distances may lead to an overly reduced amount of edge resources deployed for the average configuration, which may lead to the counter-intuitive effect of experiencing a larger slowdown during rush hours. For example, as Figure 5(a) shows, at 4 pm in area A3, the safe speed of AVs decreases from 15 mph to 13 mph when the blind distance is increased from 12 meters to 16 meters. This effect must be taken into consideration when designing an under-provisioned edge infrastructure.

3) Key Takeaway 2: In summary, using the average configuration of resources at the edge server may effectively handle the traffic demands during the medium and low traffic hours, which constitute more than two thirds of the day. *The slowdown that AVs would experience during rush hours could be 1) acceptable, given the potentially high cost savings, and 2) handled by integrating the edge load information into the turn-by-turn navigation system, which could help reduce the AVs travel time by re-routing them through areas with higher safe-speed.*

D. Travel Time: Regular Vehicles vs. Edge-assisted AVs

Given the spatio-temporal variability of the safe speed based on time of the day and location for average configurations, here we leverage the baseline described in Section IV-A to investigate whether the faster travel path for regular vehicles provided by modern turn-by-turn navigation algorithms also maps into the faster path for AVs.

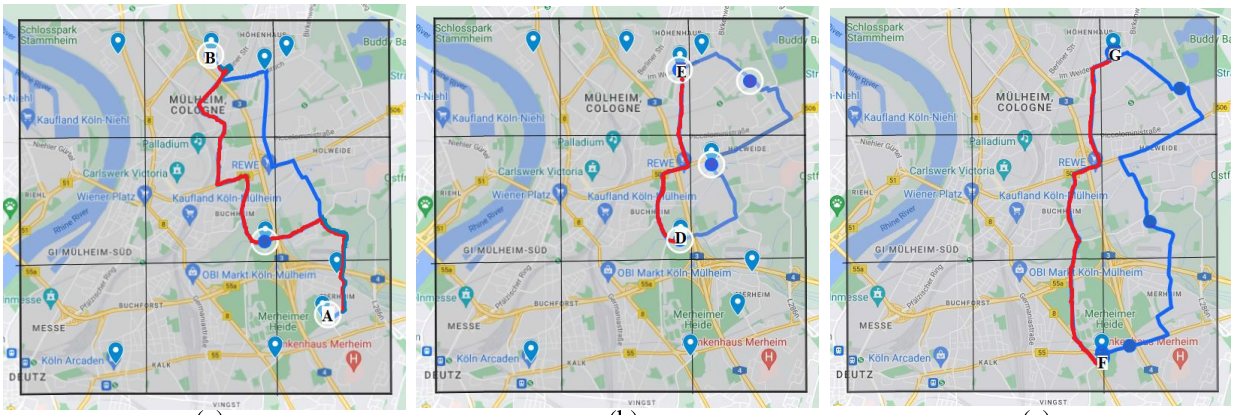
In order to conduct this study, we first need to be able to accurately estimate the travel time of edge-assisted AVs on a certain route. To do so, we have created several scenarios where the AVs need to travel from a certain source to a certain destination during rush hour (see Figure 2 to locate each source/destination). Each scenario has the source and destination located in different areas. Figure 6 compares the travel time for each scenario as given by the Google Maps baseline at 4 pm (one of the rush hours) with the one we calculate using the average speed from the dataset. The results show a good estimation accuracy on average, which validates the results presented in the next paragraphs. Among all the scenarios, we show the detailed results for three of them in Figure 7 and 8.

1) Scenario I: In Figure 7(a), we consider a scenario where the AVs have to go from location A, in area A9, to location B, in area A2, at 4 pm on a working day. We consider two different routes to reach the destination. The first route (red) (provided by Google Maps), requires to go through areas ($A9 \rightarrow A6 \rightarrow A5 \rightarrow A2$) while the second route (blue), takes through the same areas except area A5 comprises a larger portion than area A6. The travel distance of route 1 and route 2 are 6.6 km (kilometers) and 7.6 km, respectively. For regular vehicles, the travel time for route 1 is 13 minutes and route 2 is 16 minutes, respectively. Similarly, route 1 is the fastest route also for the AVs. As shown in Figure 8, the AV can reach the destination 7 minutes, 6 minutes, and 5 minutes earlier considering blind distance of 8 meters, 12 meters, and 16 meters, respectively, via route 1. The reduction in the travel time for the baseline route is due to the fact that the increased travel distance of route 2 is not justified by the AV's safe speed. Thus, in this scenario Google Maps provides the fastest route to both regular vehicles and AVs.

2) Scenario II: There can be cases where the fastest route for regular vehicles may not always be the fastest route for edge-assisted AVs. Figure 7(b) shows such a case. In this scenario, the source (D) is chosen to be in area A5 and destination (E) in area A2. Route 1 (red) $A5 \rightarrow A2$, is the

Authorized licensed use limited to: Wayne State University. Downloaded on May 29,2023 at 19:59:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.



(a) (b) (c)

Fig. 7. Showing two routes; route 1 (red) given by Google Maps and route 2 (blue), an alternative route for (a) Scenario I: Going from $A \rightarrow B$, (b) Scenario II: Going from $D \rightarrow E$, and (c) Scenario III: Going from $F \rightarrow G$.

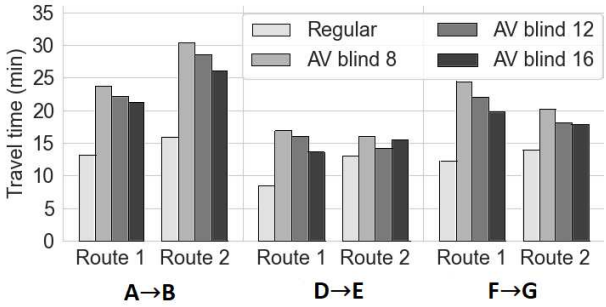


Fig. 8. Comparison of travel time between the two routes for different blind distances for Scenario I (A→ B), Scenario II (D→ E) and Scenario III (F→ G).

shortest one and it is suggested by Google Maps as the fastest route for regular vehicles, as shown in Figure 8. However, this route takes the AV to travel through the two heavy traffic areas while route 2 (blue) $A5 \rightarrow A6 \rightarrow A3 \rightarrow A2$, takes the AVs through the areas having lower traffic at that time. Although the AV needs to travel a longer distance in route 2, the increased safe speed allows to lower their travel time compared to that of route 1. It can be seen from the results in Figure 8 that the AV can reach the destination 1 minute and 2 minutes earlier considering blind distance of 8 meters and 12 meters, if it takes the second route. On the other hand, as discussed in the previous section (Figure 5(a)), sometimes the safe speed is reduced with an increased blind distance due to the lower resources necessary on average. As a result, in Figure 8 for scenario I, the travel time increases by 2 minutes while considering blind distance of 16 meters as compared to 12 meters for route 2. This example scenario clearly shows that the travel time for the AVs is dependent not only on the distance and the traffic density in the selected route but also on the amount of edge resources deployed in different areas throughout the route. Thus, the fastest route shown by the traditional navigation system may not always be the fastest one in case of AVs.

E. Scenario III

Figure 7(c) shows a scenario where the AV has to travel from location F to location G. Route 1 (red) going through the areas ($A8 \rightarrow A9 \rightarrow A6 \rightarrow A5 \rightarrow A2 \rightarrow A3$) is shorter and faster for regular vehicles, as suggested by Google Maps where the travel time for the regular vehicles is 12.41 minutes. However, this route includes the heavy traffic areas A5 and A2. In the case of AVs, with blind distance 8 meters and 12 meters the travel time is approximately 25 minutes and 22 minutes, respectively. If the AV can be rerouted to route 2 (blue) shown in Figure 7(c) avoiding the heavy traffic areas, it can be seen from Figure 8 that the travel time can be reduced to 20 minutes and 18 minutes with blind distance 8 meters and 12 meters, respectively. As the length of the second route is greater than the first one, the travel time for regular vehicle becomes longer but in the case of AVs this route takes approximately 20% less time to reach the destination.

1) Key Takeaway 3: In summary, *we can conclude that while the traditional navigation systems are able to suggest the fastest route for regular vehicles, they are not always efficient in suggesting the fastest routes for the edge-assisted AVs as observed from the results of Scenario III.*

V. Future Research Directions and Challenges

By analyzing all the results shown in Section IV, future research directions in smart transportation systems need to further investigate *how to consider the large spatio-temporal variability of edge-assisted AV demands for an optimal deployment of edge resources in smart cities.* While in this paper we discuss the average configuration that sustains the average demand, future research should focus on optimizing network and computing requirements considering multiple objectives such as total cost, AV slowdown, and safety (in terms of blind distance). In addition, researchers should consider *how to integrate into the turn-by-turn routing algorithms the current edge resource load in different city areas to find the fastest route to destination.* Several challenges can arise from tackling these new issues, including (i) how to measure, exchange, and predict edge load demand across city areas, and (ii) how to coordinate real-time AV routing decisions to collectively optimize travel time considering the predicted edge load and slowdown.

Authorized licensed use limited to: Wayne State University. Downloaded on May 29,2023 at 19:59:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

VI. Conclusion

Motivated by the advantage of edge-assisted AVs in providing greater computing power with reduced capital cost, in this paper, we discussed the implications of deploying different amount of edge resources in different city areas to handle peak versus average traffic demand by leveraging real traffic data. Considering an example system scenario, we analyzed the resource requirements for peak and average configurations. We developed a Python simulation environment that uses real traffic data to evaluate the results. Our analysis revealed that a reduction in network bandwidth and computing cores of up to 60% and 50%, respectively, is achieved by deploying edge resources for the average demand rather than peak demand. We also investigated how the peak-hour demand affects the safe travel time of AVs and find that it can be reduced by approximately 20% if they would be rerouted to areas with a lower edge-resource load. Thus, future research must consider that traditional turn-by-turn navigation systems may not provide the fastest routes for edge-assisted AVs because they do not take into consideration the amount of edge server resources deployed and the computational delay that may occur in the data processing.

References

[1] T. Litman, "Autonomous vehicle implementation predictions: Implications for transport planning," Victoria Transp. Policy Inst., Victoria, BC, Canada, Tech. Rep. 01727624, 2020. [Online]. Available: https://trid.trb.org/View/1678741

[2] A. Hammoud, H. Sami, A. Mourad, H. Otrok, R. Mizouni, and J. Bentahar, "AI, blockchain, and vehicular edge computing for smart and secure IoV: Challenges and directions," *IEEE Internet Things Mag.*, vol. 3, no. 2, pp. 68–73, Jun. 2020.

[3] A. Souri, "Artificial intelligence mechanisms for management of QoS-aware connectivity in Internet of Vehicles," *J. High Speed Netw.*, vol. 2022, pp. 1–10, Mar. 2022.

[4] P. J. Schildkraut. (2021). *AI Regulation: What You Need to Know to Stay Ahead of the Curve*. [Online]. Available: https://www.arnoldporter.com/en/perspectives/publications/2021/06/ai-regulation-what-you-need-to-know

[5] DOT National Highway Traffic Safety Administration. (2015). *Why Your Reaction Time Matters at Speed*. [Online]. Available: https://one.nhtsa.gov/nhtsa/Safety1nNum3ers/august2015/S1N-Aug15-Speeding-1.html

[6] Z. Zhou, M. Shojafar, R. Li, and R. Tafazolli, "EVCT: An efficient VM deployment algorithm for a software-defined data center in a connected and autonomous vehicle environment," *IEEE Trans. Green Commun. Netw.*, vol. 6, no. 3, pp. 1532–1542, Sep. 2022.

[7] H. Wang, B. Kim, J. Xie, and Z. Han, "E-auto: A communication scheme for connected vehicles with edge-assisted autonomous driving," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[8] A. J. Ben Ali, Z. S. Hashemifar, and K. Dantu, "Edge-SLAM: Edge-assisted visual simultaneous localization and mapping," in *Proc. 18th Int. Conf. Mobile Syst., Appl., Services*, Jun. 2020, pp. 325–337.

[9] M. Cui, S. Zhong, B. Li, X. Chen, and K. Huang, "Offloading autonomous driving services via edge computing," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 10535–10547, Oct. 2020.

[10] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A survey on end-edge-cloud orchestrated network computing paradigms: Transparent computing, mobile edge computing, fog computing, and cloudlet," *ACM Comput. Surv.*, vol. 52, no. 6, pp. 1–36, Nov. 2020.

[11] M. Li, N. Cheng, J. Gao, Y. Wang, L. Zhao, and X. Shen, "Energy-efficient UAV-assisted mobile edge computing: Resource allocation and trajectory optimization," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3424–3438, Mar. 2020.

[12] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas, "Generation and analysis of a large-scale urban vehicular mobility dataset," *IEEE Trans. Mobile Comput.*, vol. 13, no. 5, pp. 1061–1075, May 2014.

[13] L. Le Mero, D. Yi, M. Dianati, and A. Mouzakitis, "A survey on imitation learning techniques for end-to-end autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 9, pp. 14128–14147, Sep. 2022.

[14] J. Knuth and P. Barooah, "Distributed collaborative localization of multiple vehicles from relative pose measurements," in *Proc. 47th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2009, pp. 314–321.

[15] I. M. Rekleitis, G. Dudek, and E. E. Milios, "Multi-robot cooperative localization: A study of trade-offs between efficiency and accuracy," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Mar. 2002, pp. 2690–2695.

[16] A. Asvadi, C. Premebida, P. Peixoto, and U. Nunes, "3D LiDAR-based static and moving obstacle detection in driving environments: An approach based on voxels and multi-region ground planes," *Robot. Auto. Syst.*, vol. 83, pp. 299–311, Sep. 2016.

[17] N. Bernini, M. Bertozzi, L. Castangia, M. Patander, and M. Sabbatelli, "Real-time obstacle detection using stereo vision for autonomous ground vehicles: A survey," in *Proc. 17th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2014, pp. 873–878.

[18] A. Broggi, S. Cattani, M. Patander, M. Sabbatelli, and P. Zani, "A full-3D voxel-based dynamic obstacle detection for urban scenario using stereo vision," in *Proc. 16th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2013, pp. 71–76.

[19] M. Masmoudi, H. Ghazzai, M. Frikha, and Y. Massoud, "Object detection learning techniques for autonomous vehicle applications," in *Proc. IEEE Int. Conf. Veh. Electron. Saf. (ICVES)*, Sep. 2019, pp. 1–5.

[20] C. Cadena et al., "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Trans. Robot.*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016.

[21] M. Masmoudi, H. Ghazzai, M. Frikha, and Y. Massoud, "Autonomous car-following approach based on real-time video frames processing," in *Proc. IEEE Int. Conf. Veh. Electron. Saf. (ICVES)*, Sep. 2019, pp. 1–6.

[22] V. Venugopal and S. Kannan, "Accelerating real-time LiDAR data processing using GPUs," in *Proc. IEEE 56th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2013, pp. 1168–1171.

[23] S.-C. Lin et al., "The architectural implications of autonomous driving: Constraints and acceleration," in *Proc. 23rd Int. Conf. Architectural Support Program. Lang. Operating Syst.*, Mar. 2018, pp. 751–766.

[24] K.-L. Wright, A. Sivakumar, P. Steenkiste, B. Yu, and F. Bai, "Cloud-SLAM: Edge offloading of stateful vehicular applications," in *Proc. IEEE/ACM Symp. Edge Comput. (SEC)*, Nov. 2020, pp. 139–151.

[25] A. Ashok, P. Steenkiste, and F. Bai, "Adaptive cloud offloading for vehicular applications," in *Proc. IEEE Veh. Netw. Conf. (VNC)*, Dec. 2016, pp. 1–8.

[26] M. C. Lucic, H. Ghazzai, A. Alsharoa, and Y. Massoud, "A latency-aware task offloading in mobile edge computing network for distributed elevated LiDAR," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Oct. 2020, pp. 1–5.

[27] N. Jayaweera, N. Rajatheva, and M. Latva-aho, "Autonomous driving without a burden: View from outside with elevated LiDAR," in *Proc. IEEE 89th Veh. Technol. Conf. (VTC-Spring)*, Apr. 2019, pp. 1–7.

[28] M. C. Lucic, H. Ghazzai, and Y. Massoud, "A generalized and dynamic framework for solar-powered roadside transmitter unit planning," in *Proc. IEEE Int. Syst. Conf. (SysCon)*, Apr. 2019, pp. 1–7.

[29] M. C. Lucic, H. Ghazzai, and Y. Massoud, "A low complexity space-time algorithm for green ITS-roadside unit planning," in *Proc. IEEE 62nd Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2019, pp. 570–573.

[30] Z. Yang et al., "EdgeDuet: Tiling small object detection for edge assisted autonomous mobile vision," *IEEE/ACM Trans. Netw.*, early access, Dec. 2, 2022, doi: 10.1109/TNET.2022.3223412.

[31] S. Abdallaoui, E.-H. Aglzim, A. Chaibet, and A. Kribèche, "Thorough review analysis of safe control of autonomous vehicles: Path planning and navigation techniques," *Energies*, vol. 15, no. 4, p. 1358, Feb. 2022.

[32] D. Comer, *Computer Networks and Internets*. Upper Saddle River, NJ, USA: Prentice-Hall, 2009. [Online]. Available: https://books.google.com/books?id=tm-evHmOs3oC

[33] A. K. Mok, "Multiprocessor scheduling in a hard real-time environment," in *Proc. 7th Texas Conf. Compt. Syst.*, 1978, pp. 1–10.

[34] J. Wang, D. Feng, S. Zhang, J. Tang, and T. Q. S. Quek, "Computation offloading for mobile edge computing enabled vehicular networks," *IEEE Access*, vol. 7, pp. 62624–62632, 2019.

[35] W. Chen, Y. Zhu, J. Liu, and Y. Chen, "Enhancing mobile edge computing with efficient load balancing using load estimation in ultra-dense network," *Sensors*, vol. 21, no. 9, p. 3135, Apr. 2021.

[36] S. Lloyd, "Least squares quantization in PCM," *IEEE Trans. Inf. Theory*, vol. 28, no. 2, pp. 129–137, 1982, doi: 10.1109/TIT.1982.1056489.

Authorized licensed use limited to: Wayne State University. Downloaded on May 29,2023 at 19:59:25 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Syeda Tanjila Atik received the B.Sc. and M.Sc. degrees in information technology from Jahangirnagar University, Bangladesh, in 2015 and 2017, respectively. She is currently pursuing the Ph.D. degree with the Department of Computer Science, Wayne State University. She is also a Student Member of the Energy-aware Autonomous Systems Laboratory (EAS-Lab). Her research interests include edge computing, the Internet of Things, autonomous mobile robots, and machine learning.

Marco Brocanelli (Member, IEEE) received the B.E. and M.E. degrees in control systems from the University of Rome Tor Vergata, Italy, and the Ph.D. degree in electrical and computer engineering program from The Ohio State University in August 2018. He is currently an Assistant Professor with the Department of Computer Science, Wayne State University, and the Director of the Energy-aware Autonomous Systems Laboratory (EAS-Lab). His research interests include cyber-physical systems, energy-aware systems, the Internet of Things (IoT), edge computing, embedded, and real-time systems.

Daniel Grosu (Senior Member, IEEE) received the Diploma degree in engineering (automatic control and industrial informatics) from the Technical University of Iaşi, Romania, in 1994, and the M.Sc. and Ph.D. degrees in computer science from The University of Texas at San Antonio, in 2002 and 2003, respectively. He is currently an Associate Professor with the Department of Computer Science, Wayne State University. His research interests include parallel and distributed computing, approximation algorithms, and topics at the border of computer science, game theory and economics. He is a Senior Member of the ACM and the IEEE Computer Society. He is an IEEE Computer Society Distinguished Contributor. He serves an Associate Editor and a member for the editorial boards of *ACM Computing Surveys*, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, and IEEE TRANSACTIONS ON CLOUD COMPUTING.

Authorized licensed use limited to: Wayne State University. Downloaded on May 29,2023 at 19:59:25 UTC from IEEE Xplore. Restrictions apply.