EdgeDrive: Supporting Advanced Driver Assistance Systems using Mobile Edge Clouds Networks

Sumit Maheshwari[§], Wuyang Zhang[§], Ivan Seskar[§], Yanyong Zhang[§†] and Dipankar Raychaudhuri[§]
[§]WINLAB, Rutgers University, North Brunswick, NJ, USA
{sumitm, wuyang, seskar, yyzhang, ray}@winlab.rutgers.edu
[†] University of Science and Technology of China
yanyongz@ustc.edu.cn

***Abstract*—In this paper, we present EdgeDrive, a networked edge cloud services framework which can support low-latency applications during mobility taking into account needs of the driver, nature of the required service and key network features. We implement head-mounted device (HMD) based Augmented Reality (AR) ADAS applications such as navigation, weather notification and annotation based assistance to drive the evaluation. These services are then coupled with the Mobile Edge Clouds (MECs) wherein the container based service migration is enabled based upon migration cost and required Quality of Experience (QoE) to support mobility. An emulator based evaluation is carried out on the ORBIT testbed using realistic San Francisco taxicab traces running over nine edge cloud nodes and AR HMD being used by drivers. The experiments show that the EdgeDrive can support low-latency ADAS applications with an average system latency less than 100 ms for the applications under consideration.**

***Index Terms*—ADAS, self-driving, Mobile Edge Cloud Computing, Service Migration, Augmented Reality, Cloud computing.**

I. Introduction

Advanced Driver Assistance Systems (ADAS) are expected to become increasingly important to the automotive industry [1]. ADAS focuses on assisting drivers by providing timely critical information such as real-time navigation, safe driving limits, pedestrian crossing, and sensor calibration. Recent advances in ADAS focus on applications such as 3D mapping [2], Internet of Vehicles (IoV) [3], and holographic displays [4] requiring uninterrupted information exchange between connected cars, offloading computation to the cloud computing servers, and handling mobility through novel communication as well as networking architectures [5], [6].

Head-mounted device (HMD) based Augmented Reality (AR) provides a user-friendly method for drivers of vehicles to interact with the surrounding environment by supplementing real world with the contextual information, e.g., traffic status, stop notification [7]. Furthermore, gesture enabled head-mounted AR devices make this interaction hands-free thus improving the user Quality of Experience (QoE). Nevertheless, the core functions of AR enabled driving assistant applications (perception, annotation, visualization and sensing) are usually computation and data intensive [8], and the on-board computing capabilities of currently available AR devices such as Microsoft Hololens [9] and Google glass [10] are insufficient to perform these intensive tasks. For instance, the latest Hololens has only 1.04 GHz CPU clock rate, and 2GB RAM. The constrained capability cannot satisfy the service demands associated with processing analytics over a single frame within 30 ms and delivering a 60 frame per second [11].

Research supported under NSF Future Internet Architecture - Next Phase (FIA-NP) Award CNS-134529

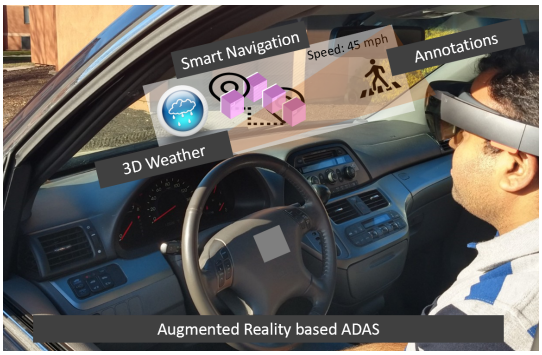

Fig. 1. AR-enable ADAS Applications. The dashboard displays the weather, navigation and surrounding information.

Further, the applications running on these devices rely on shared information obtained from nearby vehicles or roadway infrastructure to interact with the environment. For example, self-driving vehicles receive surrounding information from other vehicles which have knowledge about distant traffic for better informing driving decisions. This inter-vehicle information exchange can be realized either as a peer-to-peer (V2V) application or as a cloud service based on edge cloud infrastructure. As central cloud servers may be at distant geographic location from the source of data generation, conventional offloading is likely to introduce significant network latency. For a client instance in New Jersey which connects to Amazon EC2 cloud servers located in West Virginia, Oregon and California, the round-trip latency *alone* is 17, 104 and 112ms, with achievable bandwidths of 50, 18 and 16 Mbps, respectively.

Mobile Edge clouds (MECs) bring computation, storage and networking close to the user thereby promising to support stringent latency requirements for AR applications [12]–[14]. Latency is a critical factor in user QoE for AR applications. For example, in an HMD, the combined network and processing latency while using an AWS cloud is more than 100 ms at which 50 degrees per second head or object rotation introduces

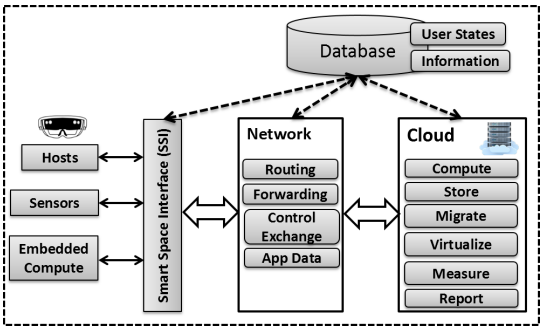

Fig. 2. ADAS System Design

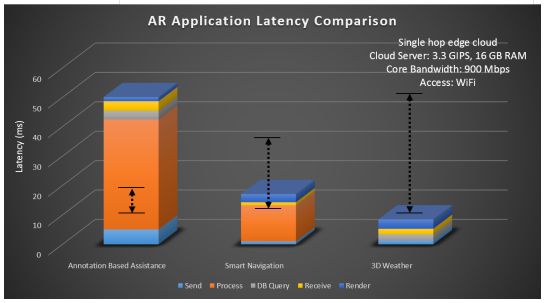

Fig. 3. ADAS Application Latency Comparison

5 degrees of angular error [15]. In case of objects closer to the user, this implies that the user views a virtual coffee cup in the air instead of on the table and for farther objects, the error only accumulates. MECs provide a way to reduce this latency, and hence the overall errors by providing responses faster than the user perceivable latency.

This paper presents, EdgeDrive, an edge cloud based end-to-end system to minimize the latency of AR applications for ADAS. We implement and deploy several example ADAS based AR applications such as smart navigation, weather notification, and annotation based assistance as shown in Figure 1. The applications are augmented with networking and edge cloud components emulated using the ORBIT [16] testbed with demonstrated methods to reduce overall system latency by edge cloud techniques such as caching, application specific routing, dynamic server selection and edge cloud migration [17]. A large-scale emulation using nine edge cloud locations in San Francisco is used along with realistic taxicab traces to showcase dynamic server migration. It is shown that the MEC system can improve AR based ADAS performance when augmented with additional capabilities such as service containerization, and its migration.

The rest of paper is organized as follows. Section II describes the above mentioned ADAS applications while detailing smart navigation application for the techniques used to reduce its delay. Section III presents the EdgeDrive architecture using MECs. Section IV describes the emulation set-up used to carry out the experiments. Results and discussion are presented in Section V and Section VI concludes the paper.

II. ADAS Applications

In order to test the functionalities of ADAS system, we identified three key applications namely, (a) annotation based assistance, (b) smart navigation, and (c) 3D weather. These applications are chosen as they cover different features and requirements such as the latency, caching, throughput and compute as listed in Table 1. In particular, the first application supports the driver by embedding the processed surrounding information onto the AR device using image processing. The second application provides navigation support by embedding 3D objects onto the path where the driver collects the objects similar to a game play. The third application projects the current location's and destination's weather information on to the driver's AR display.

TABLE I
Requirements for Sample ADAS Applications

Feature	Annotation	Smart Navigation	3D Weather
Latency	Low	Medium	High
Database/Caching	No	Yes	Yes
Throughput	High	Medium	Low
Compute	High	Medium	Low

A. ADAS System Design

The system consists of devices (e.g. Hololens), network (e.g. routers) and the service functions placed at each cloud servers. The server runs on an edge cloud which is typically one hop away from the Access Point (AP). The network, in addition to routing and forwarding, also supports control exchange functions among the edge clouds. The edge cloud provides features such as computation, storage, resource virtualization, service migration, and performance monitoring. The applications are developed using Unity and C#. The design framework is as shown in Figure 2. The APIs developed at the smart space interface allows user to seamlessly access the services from the cloud server. The user states are accessible across the network using a logically centralized database.

By deploying this three-layered, device-router-edge cloud system on the ORBIT testbed, the function level latency is measured for all the applications as shown in Figure 3. The arrows represent the required latency thresholds which is low for annotation based application as the response should be in real-time as the user's head movement. The lower bound represents the latency which is sufficient to provide satisfied quality of experience to the user. The smart navigation application spends most of the time in image processing using OpenCV and therefore requires a highly computational, lightly loaded edge cloud for offloading the compute. The 3D weather application is able to serve without delays using its caching function retrieving the statistics from a central weather server periodically. It can be noted that despite edge cloud being one hop away, the annotation based assistance application is unable to achieve latency bounds and there require techniques such as distributed task computing [11] which are not studied in this work.

B. Smart Navigation

Smart Navigation is one of the most crucial features in ADAS. First, we describe the challenges of developing an

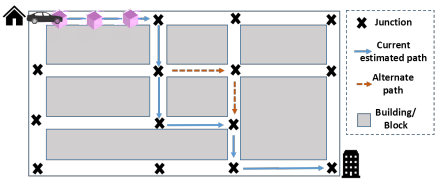

Fig. 4. Smart Navigation in ADAS

uninterrupted navigation service and then detail the suitable design choices.

Localization with Hololens: Navigation needs GPS location information of the device. Yet, currently available Hololens devices do not support the GPS module. Therefore, we rely upon mapping coordinates based upon accelerometer and gyroscope sensor readings for localization. The map is divided into set of junctions and at each junction, the route is recalculated based upon current Hololens coordinate and destination coordinates. Therefore, the complete navigation system relies upon manipulating coordinates as read by local sensors and thus the implementation functions without the GPS as shown in Figure 4.

High Uplink Bandwidth: Hololens requires to send the stream of images to the server continuously. This is expensive as it requires high uplink bandwidth. In order to optimize this, we send only when the image changes more than 5% (can be varied) as compared to the last sent image and thereby also saving server compute. A lightweight bitmap based hash function is locally employed on the HMD for the image comparison.

Path Rerouting: Sometime the vehicle may take paths other than that suggested by navigation. This is handled by continuously comparing the coordinates of line connecting two suggested junctions. If the user's current coordinates do not lie on the line, the path is rerouted to the next optimal path. Finally, the path is always selected based upon the minimum distance between the source and the destination considering all the possible combinations.

Receiving Real-time Responses: The image processing and path calculations are offloaded to the neighboring edge cloud server and is detailed in the next section.

In a general ADAS system, a tagged metadata image stream is used for all the services rendered to an HMD. Therefore, in this work, we have employed image streaming irrespective of the application.

III. EdgeDrive System

In this section, the EdgeDrive system components and its capabilities are described.

A. System Components

The system has the following components as shown in Figure 5. (1) **End Devices**: These are hosts, sensors and embedded compute devices such as FGPA which can push/pull

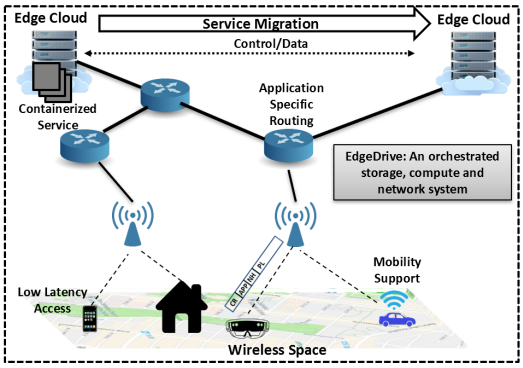

Fig. 5. EdgeDrive System

data from the system; (2) **Smart Space Interface**: It allows to connect heterogeneous devices to the edge cloud by mapping their API requirement to the deployed system; (3) **Network**: Its functions are routing, forwarding and control. The deployed network has additional feature which allows application state to be pushed to the cloud and vice-versa. This simple action allows application specific routing to best available edge cloud while keeping the control distributed in the network; (4) **Edge Cloud**: Each instance of edge cloud can compute, store, migrate resources by virtualization (such as containers), measure and report statistics to the network and neighbors; (5) **Database**: It contains user states and space specific information which can be cached at the edge cloud proactively or on-demand.

B. AR using HoloLens

HoloLens provides a state of the art technique to achieve AR functionalities by its unique features such as frame of references (stationary and attached), spatial anchoring and spatial mapping. This implies that a virtual object (hologram) can be placed and oriented at a fixed location in the real-world (mixed reality) and can be retrieved at the same location later anytime. The specially designed spatial coordinate system can be used to derive other coordinate systems or can be used as-is to determine device's own position in the three-dimension space.

C. EdgeDrive Capabilities

The key features of EdgeDrive are described as follows. **Service Containerization:** All the services such as navigation, annotation and weather reporting are containerized at the edge cloud servers. Containerization provides benefits such as service and user level isolation, faster start and stop, and easy migration.

Mobility Support using Container Migration: EdgeDrive provides ADAS application mobility support by migrating containers across edge cloud nodes. Each edge cloud node has a performance monitor, migration manager and controller which with the help of neighboring edge cloud resource information decides the target as shown in Figure 6. The migration[1] cost is assessed using the equation in [18], for the

[1]The authors would like to thank Shalini Choudhury for her help in container migration system formulation.

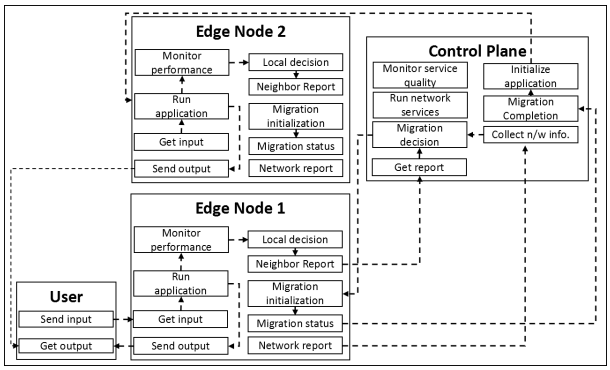

Fig. 6. Container Migration in EdgeDrive

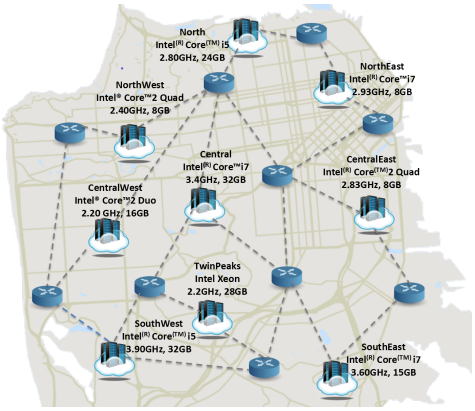

Fig. 7. San Francisco Edge Cloud Network

parameters shown in Table II, as follows:

$$C_m = \sum_{j=1}^{N} \left[\frac{(k_{in,j} + k_{out,j}) * r_{pd,j} * s_{page,j}}{s_{p,j} * (1 - load_j)} \right] + \sum_{j=1}^{N} \sum_{\substack{l=1 \\ (l \neq j)}}^{N} \left[\frac{k_{in,j,l} * r_{pd,j} * s_{page,j}}{b_{i,j,l}} \right] \quad (1)$$

In the above equation, the former part provides time for computing while the latter estimates the migration time due to inter-edge bandwidth. From [18], the decision migration algorithm ShareOn is used in EdgeDrive for estimating the target location based upon network bandwidth, compute capability and current utilization of the destination with the following objective function:

$$\min . \left[\max . \left\{ \frac{(k_{in,j} + k_{out,j}) * s_{page,j} * r_{pd,j}}{s_{p,j} * (1 - load_j)} + \sum_{\substack{l=1 \\ (l \neq j)}}^{N} \frac{k_{in,j,l} * s_{page,j} * r_{pd,j}}{b_{i,j,l}}, \forall j \in [1, N] \right] \right. \quad (2)$$

TABLE II
CONTAINER MIGRATION PARAMETERS

Param	Description	Param	Description
r_{pd}	page dirty rate (KBps)	s_{page}	page size (KB)
s_p	processor speed (GIPS)	m	RAM (GB)
b_i	inter-edge bw (Gbps)	t_n	network latency (ms)
$load$	CPU load at MEC (0–1)	k	#running containers
k_{in}	#containers received	k_{out}	#containers sent

In Equation 2, the migration time in a continuous migration process (wherein multiple containers overlap) is optimized by minimizing the time of maximum cost migration. The control plane exchange is similar to that of [19] where the neighbors exchange their load and compute information with each other.

IV. EXPERIMENTAL DETAILS

The experiment is set up at the sandbox-9 (SB9) on the ORBIT [16] testbed. Nine nodes are set-up based on the topology of a San Francisco edge cloud network with variable inter-edge bandwidth (randomly chosen between 100-900 Mbps) connectivity as shown in Figure 7. The nodes are placed based on the population density in SFO and have heterogeneous memory, processing speed (Ref: Fig.7) and cpu load (varied from 0 to 1). Real SFO taxicab traces are used from the heavy traffic routes across the city. Each of these users are assumed to be using the smart navigation application as described earlier. The source and destination are chosen randomly. The low-latency requirement of the navigation application is aimed to be fulfilled by EdgeDrive by: (a) pre-caching and reevaluating navigation information at each junction and the line connecting the neighboring junctions, (b) associating user to the best available edge cloud, (c) providing service transfer support using container migration, and (d) enabling network embedded application specific routing.

A. Emulating ADAS

The ADAS applications are emulated on the ORBIT testbed by setting in-lab source and destination for navigation. The junctions are pre-defined at every intersection and realistic dimensions are obtained using a laser rangefinder. A navigation algorithm finds the shortest distance between the source and the destination. The coordinates are initialized at (x,y,z) = (0,0,0) using OpenCV based recognizable chilitags placed at the source. For the entire experiment, z (height) is set to 0. At source, the user looks at the marker using Hololens and the navigation begins for the set destination. The user then collects the pink cubes as shown in Figure 8(b) to navigate towards the destination. The image stream rate from HoloLens is varied from 10fps to 60fps for the walking user to emulate vehicular mobility. For a large scale experiment, 536 mobile users are injected into the system using a custom script emulating from the SFO taxicabs traces and thereby loading servers. Similarly, the annotation application is emulated by obtaining real-time printer information such as ink status as shown in Figure 8(c). MySQL database is used for storage and Apache for server code.

B. Container Migration

As the services run on containers at the nine edge clouds, the migration is self-triggered locally at an edge considering

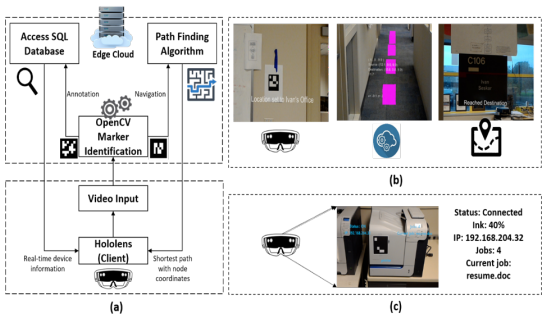

Fig. 8. Emulating ADAS Applications at WINLAB

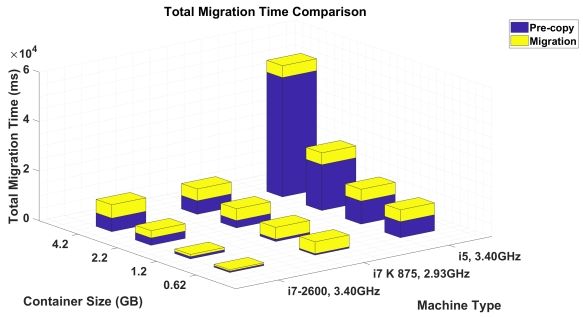

Fig. 9. Impact of Machine type and Container Size on Total Migration Time (Bandwidth=912 Mbps)

migration cost and the parameters as described in ShareOn. The migration has three stages: (a) decision, (b) initiation and (c) completion. During the decision phase, the right edge cloud is chosen for migrating a running containerized service. During the initiation phase, the pre-copy of container begins thus copying the pages from source to the destination. Finally, during the completion phase, source edge cloud node informs the destination edge cloud node to run the newly received container and itself discards the old container upon receiving confirmation from the destination. The complete process is automated using Python and shell scripts.

V. Results and Discussion

This section presents the results obtained using the experiments as detailed earlier.

A. Parameters Impacting Container Migration

Container migration consists of pre-copy, migration and post-copy. For stateless applications such as annotation, weather and navigation, post-copy is not required and therefore merely migrating the dependencies enables the destination node to restart the services. Figure 9 shows the impact of machine type and container size on the total migration time. It can be seen that the migration time for a 4.2 GB container running on an Intel i5 machine can be as high as 60 seconds which is less useful for real-time applications described in this paper. Therefore, while deciding service migration, along with the bandwidth, machine type plays an important role. The other parameters of interest are system utilization (load), processing capabilities of the edge cloud and the available RAM.

B. Factors Affecting Application Performance

The input from HoloLens in case of weather and smart navigation applications is primarily the coordinates (a few KBs) of user whereas in case of annotation application, continuous stream of images (avg. 30fps) is sent from the device to the server. Therefore, the annotation application is the most affected by the system load as shown in Figure 10. As the weather application relies upon the pre-fetched data in the edge cloud, the current system load does not affect its performance. The navigation application intermittently requests server to

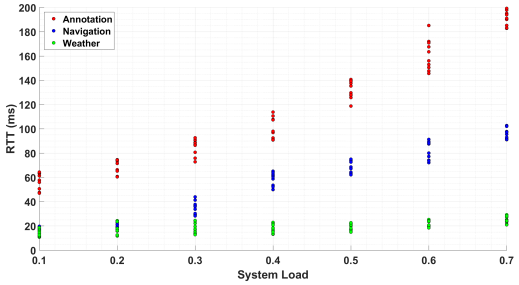

Fig. 10. Impact of System Load on Application Performance

calculate the path and therefore its round trip response time grows when the system load is high.

Figure 11(a) depicts that when the frame per second sent from the HoloLens to the edge cloud server is low, the server is unable to determine the object and therefore the response time is high. When the fps is increased, OpenCV is able to detect the marker or object and therefore the annotation appears in less than 70 ms. As the fps is increased, the server is loaded with the heavy processing and therefore the RTT increases again.

Figure 11(b) shows the impact of mobility on the navigation performance of HoloLens. The in-lab mobility is normalized from 0 to 1 and error is defined as junction missed due to mobility which otherwise would have provided the shortest path. The percentage error is calculated by running the experiment multiple times and then averaging the miss rate.

C. Effect of Container Migration

Figure 12(a) shows the latency for the navigation application for a single random user when the system is loaded at 0.5 (50% CPU utilization). As the user is mobile, the latency depends upon whether the user is moving closer to the assigned edge cloud or not. Employing ShareOn based EdgeDrive, the application latency for the user drops after the time tick 20 at which point the migration is complete and user's service is migrated to another edge cloud.

Figure 12(b) shows the average latency for the whole system for all the applications when the CPU load is 0.5. For the higher compute demanding applications, EdgeDrive is able to

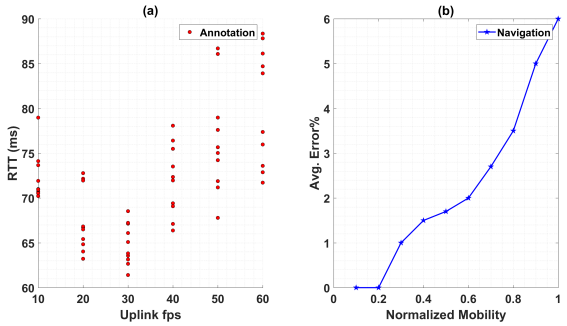

Fig. 11. Impact of Various Parameters on the Application Performance. (a) Uplink fps affects the RTT for Annotation Application and (b) Mobility affects the Accuracy of Navigation

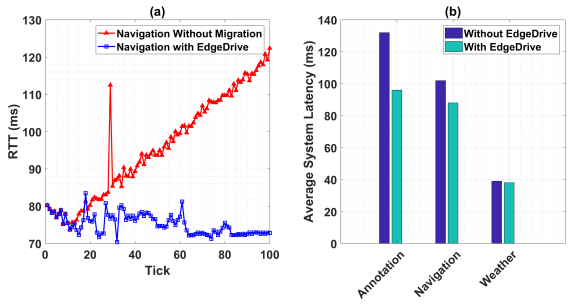

Fig. 12. Latency Performance using EdgeDrive. (a) Single User Latency with and without migration and (b) Average System Performance with and without Migration

provide better QoE as the latency is significantly lower than the system without migration.

VI. Conclusion

In this paper, we proposed EdgeDrive, a mobile edge cloud based compute, network and storage orchestrated architecture to support advanced driver assistance systems (ADAS). Using key Augmented Reality (AR) applications developed for the head mounted display (HMD), it is demonstrated that EdgeDrive is able to provide low-latency service to the driver during mobility. The experiment is set-up on the ORBIT testbed for real time emulation by deploying set of edge cloud nodes, container based service migration system, SFO taxicab traces for mobility and the network support.

The key observations from this study are: (1) machine type plays a crucial role in deciding migration, (2) applications requiring higher compute for instance annotation based assistance should be offloaded to the closest available edge cloud, (3) increasing frames per second sent to the edge cloud server does not necessarily improve the application performance, (4) the latency of applications requiring prefetched data cannot be further optimized, and (5) service migration should consider network bandwidth, system load, and compute capability of the source and the destination. Our future work includes evaluating EdgeDrive on a large-scale testbed such as COSMOS [20] where realistic mobility data can be evaluated.

Acknowledgement

The work is supported under NSF Future Internet Architecture - Next Phase (FIA-NP) Award CNS-134529. We would like to thank Kishore Kumar Anand, Anirudh Balaji and Krishnamohan P. for their help in the application development.

References

[1] D. Geronimo, A. M. Lopez, A. D. Sappa, and T. Graf, "Survey of pedestrian detection for advanced driver assistance systems," *IEEE transactions on pattern analysis and machine intelligence*, vol. 32, no. 7, pp. 1239–1258, 2010.

[2] R. W. Wolcott and R. M. Eustice, "Visual localization within lidar maps for automated urban driving," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 176–183, IEEE, 2014.

[3] M. Gerla, E.-K. Lee, G. Pau, and U. Lee, "Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds," in *Internet of Things (WF-IoT), 2014 IEEE World Forum on*, pp. 241–246, IEEE, 2014.

[4] S. Tay, P.-A. Blanche, R. Voorakaranam, A. Tunç, W. Lin, S. Rokutanda, T. Gu, D. Flores, P. Wang, G. Li, *et al.*, "An updatable holographic three-dimensional display," *Nature*, vol. 451, no. 7179, p. 694, 2008.

[5] F. Bronzino, S. Maheshwari, I. Seskar, and D. Raychaudhuri, "Novn: named-object based virtual network architecture," in *Proceedings of the 20th International Conference on Distributed Computing and Networking*, pp. 90–99, ACM, 2019.

[6] D. Raychaudhuri, K. Nagaraja, and A. Venkataramani, "Mobilityfirst: a robust and trustworthy mobility-centric architecture for the future internet," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 16, no. 3, pp. 2–13, 2012.

[7] W. Zhang, B. Han, and P. Hui, "Jaguar: Low latency mobile augmented reality with flexible tracking," in *2018 ACM Multimedia Conference on Multimedia Conference*, pp. 355–363, ACM, 2018.

[8] Z. Huang, W. Li, P. Hui, and C. Peylo, "Cloudridar: A cloud-based architecture for mobile augmented reality," in *Proceedings of the 2014 workshop on Mobile augmented reality and robotic technology-based systems*, pp. 29–34, ACM, 2014.

[9] "Microsoft Hololens." https://www.microsoft.com/en-us/hololens .

[10] "Google Glass." https://developers.google.com/glass/ .

[11] W. Zhang, S. Li, L. Liu, Z. Jia, Z. Yanyong, and D. Raychaudhuri, "Heteroedge: Orchestration of real-time vision applications on heterogeneous edge clouds," in *INFOCOM, 2019 Proceedings IEEE*, IEEE, 2019.

[12] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, 2017.

[13] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the internet of things," in *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pp. 13–16, ACM, 2012.

[14] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[15] R. Azuma, "Tracking requirements for augmented reality," *Communications of the ACM*, vol. 36, no. 7, pp. 50–51, 1993.

[16] D. Raychaudhuri, I. Seskar, M. Ott, S. Ganu, K. Ramachandran, H. Kremo, R. Siracusa, H. Liu, and M. Singh, "Overview of the orbit radio grid testbed for evaluation of next-generation wireless network protocols," in *Wireless Communications and Networking Conference, 2005 IEEE*, vol. 3, pp. 1664–1669, IEEE, 2005.

[17] W. Zhang, Y. Hu, Y. Zhang, and D. Raychaudhuri, "Segue: Quality of service aware edge cloud service migration," in *Cloud Computing Technology and Science (CloudCom), 2016 IEEE International Conference on*, pp. 344–351, IEEE, 2016.

[18] S. Maheshwari, S. Choudhury, I. Seskar, and D. Raychaudhuri, "Traffic-aware dynamic container migration for real-time support in mobile edge clouds," in *2018 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*, pp. 1–6, IEEE, 2018.

[19] S. Maheshwari, D. Raychaudhuri, I. Seskar, and F. Bronzino, "Scalability and performance evaluation of edge cloud systems for latency constrained applications," in *2018 IEEE/ACM Symposium on Edge Computing (SEC)*, pp. 286–299, IEEE, 2018.

[20] "Cloud Enhanced Open Software Defined Mobile Wireless Testbed for City-Scale Deploymen." http://cosmos-lab.org/.