R-CAV: On-Demand Edge Computing Platform for Connected Autonomous Vehicles

Mohammad Aminul Hoque
Dept. of Computer Science
University of Alabama at Birmingham
Birmingham, AL 35294, USA
mahoque@uab.edu

Raiful Hasan
Dept. of Computer Science
University of Alabama at Birmingham
Birmingham, AL 35294, USA
raiful@uab.edu

Ragib Hasan
Dept. of Computer Science
University of Alabama at Birmingham
Birmingham, AL 35294, USA
ragib@uab.edu

2021 IEEE 7th World Forum on Internet of Things (WF-IoT) | 978-1-6654-4431-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/WF-IoT51360.2021.9595160

***Abstract*—Connected Autonomous Vehicles (CAVs) have achieved significant improvements in recent years. The CAVs can share sensor data to improve autonomous driving performance and enhance road safety. CAV architecture depends on roadside edge servers for latency-sensitive applications. The roadside edge servers are equipped with high-performance embedded edge computing devices that perform calculations with low power requirements. As the number of vehicles varies over different times of the day and vehicles can request for different CAV applications, the computation requirements for roadside edge computing platform can also vary. Hence, a framework for dynamic deployment of edge computing platforms can ensure CAV applications' performance and proper usage of the devices. In this paper, we propose R-CAV – a framework for drone-based roadside edge server deployment that provides roadside units (RSUs) based on the computation requirement. Our proof of concept implementation for object detection algorithm using Nvidia Jetson nano demonstrates the proposed framework's feasibility. We posit that the framework will enhance the intelligent transport system vision by ensuring CAV applications' quality of service.**

***Index Terms*—Connected Autonomous Vehicles, Edge Computing, Perception, Drones, Smart City, Unmanned Ariel Vehicles.**

I. Introduction

Autonomous vehicles (AVs) are expected to play a huge role in the urban traffic system to enhance road safety. Development in sensing technologies, computer vision algorithms, and high-performance edge computing devices has resulted in the evolution of Advanced Driving Assistance Systems (ADAS), and Autonomous Driving (AD) technologies [1]. The AVs can to be connected for sharing sensing data to improve AD and ADAS. This advanced vehicular environment is termed Connected Autonomous Vehicles (CAV). The CAV platform enables different applications such as cooperative perception [2], intelligent traffic control [3], collision warning, creating and updating HD maps [4], etc. CAV architecture in the urban area requires roadside edge computing platforms for the smooth operation of CAV applications. However, requirements for such an edge computing platform may vary depending on different constraints, such as different computation requirements for different CAV applications, a variable number of cars present on the road, etc. Hence, a framework is required to ensure the quality of service (QoS) of the CAV applications and proper usage of the edge computing devices.

The CAV environment is a typical edge computing system with three main components: vehicle, edge, and cloud [5]. Each vehicle is equipped with proper sensors and an edge computing system that integrates different AD functions. The CAVs can communicate with the cloud, roadside units (RSUs), and other CAVs to share information that enables various latency constraint applications. Sending sensor data to the cloud for calculation is not feasible due to the bandwidth requirement and incurred unacceptable latency. The roadside edge servers provide low-powered and high-performance edge computing devices that can efficiently process shared sensor data from different vehicles to extract more critical information, which is usually beyond the area visible to any single vehicle. Such advanced information can be beneficial for different AD functionalities such as path planning and trajectory generation. However, adaptability with computation requirements depending on variable latency constraints, applications, and the number of vehicles is crucial. A fixed roadside infrastructure may not provide the required QoS when the computation requirement is high. Additionally, the devices may remain idle when there are not enough computation requirements. Hence, a framework is essential to provide roadside edge computing devices based on the computation requirements. Considering the importance of roadside edge servers in the CAV applications, a drone-based RSU deployment framework can provide roadside edge servers. Moreover, the dynamic deployment of the RSUs can ensure the appropriate usage of the devices.

In this paper, we propose a drone-based on-demand RSU deployment framework named R-CAV that provides edge computing devices based on computation requirements. Drones are unmanned aerial vehicles (UAV) that are easy-to-deploy, re-programmable from anywhere, and excellent for carrying small payloads [6]. In the R-CAV framework, the drones are equipped with proper edge computing hardware and communication devices. The drones fly to the assigned intersection and land on a specific place close to the intersection. The framework considers the drone's low flight time and optimizes energy consumption and time for a particular service. The battery level is also monitored to ensure the safe return of the drone. The R-CAV framework offers several features that provide efficient computation offloading from the vehicles. It can adopt variable latency constraints and application requirements from different vehicles. Moreover, R-CAV also supports scalability by assigning new devices or removing existing unused devices based on current computation requirements.

Contribution: The contributions of this paper are as follows:

1) We propose R-CAV – a framework that provides on-demand edge computing devices in RSUs using drones to enable different latency constraint CAV applications.

Authorized licensed use limited to: UNIV OF ALABAMA-BIRMINGHAM. Downloaded on March 05,2023 at 04:45:15 UTC from IEEE Xplore. Restrictions apply.

2) We analyze the performance of the framework in terms of scalability and resource requirements to demonstrate the feasibility.

Organization: The rest of the paper is organized as follows: Section II explains the related background and motivation behind the proposed framework. Section III contains the framework details. Section IV provides the experiment and evaluation of the proposed framework. Section V contains the related works and we conclude in section VI.

II. Background and Motivation

This section provides the background regarding autonomous vehicular edge computing architecture where the vehicles can communicate and share sensor data. Moreover, we also explain the motivation behind the framework.

A. Autonomous vehicular edge computing

Connected autonomous vehicles form a typical edge computing system that provides different context aware services in close proximity. The are three components of autonomous vehicular edge computing or CAV architecture: the AV, roadside edge server or roadside units (RSUs), and the cloud. Typically, all the AVs are equipped with specialized hardware such as Nvidia Drive PX2, which is capable of analyzing the sensor data to make real-time driving decisions. The road side edge servers or roadside units (RSUs) also contain edge computing device such as Intel fog node reference based CPU-FPGA cluster, Tensor Processing Unit (TPU) based cluster, and Nvidia GPU [7]. AVs can communicate with each other and roadside edge servers through vehicle-to-everything (V2X) communication using Dedicated Short Range Communication (DSRC) [8]. Federal Communications Commission (FCC) has reserved 5.850 to 5.925 GHz frequency band for Vehicle-to-Everything (V2X) communications [9]. The US Department of Transportation is developing rules for vehicle to vehicle (V2V) and vehicle to infrastructure (V2I) communication. The AVs and the RSUs can also communicate with the cloud using 4G/LTE. Figure 1 gives an overview of the autonomous vehicular edge computing architecture.

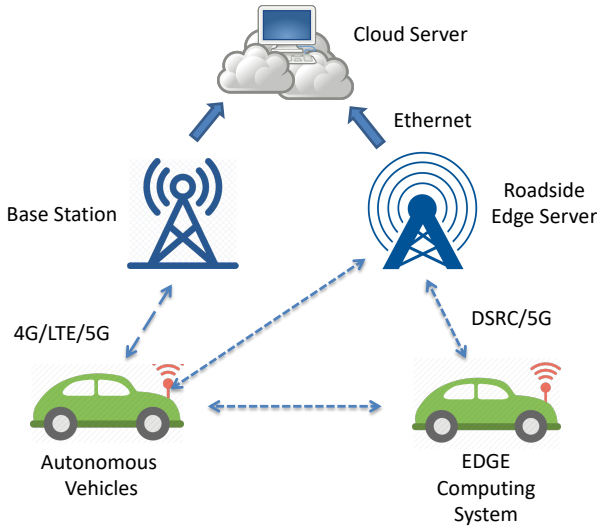

Fig. 1. Autonomous vehicular edge computing architecture

B. Motivation

Autonomous driving technology has achieved significant improvement in recent years. More than 40 companies are currently working on AD, and ADAS technologies [10]. Around 200 million cars are expected to be connected within 2025 to share sensor data and enable intelligent vehicular applications [11]. Edge-assisted RSUs are required for these heterogeneous CAV applications to ensure QoS. However, ensuring proper computation capability in RSUs is essential for the smooth operation of these applications. The CAV applications that can be highly benefited from the proposed framework are:

Cooperative perception: Cooperative perception is one of the killer applications of CAV environment. AV's vision area can be extremely limited due to obstacles and other vehicles on the road. The LiDAR data from multiple vehicles can be merged by high-level feature fusion using the edge computing devices in RSUs [2]. Such high-level feature fusion for cooperative perception enhances the perception capability of the AVs. Camera data from multiple vehicles can also enable cooperative perception application.

Smart traffic controller: In CAV environment, connected vehicles can broadcast basic safety messages (BSMs) that contain location, speed, heading, and acceleration. The intelligent traffic control system takes a snapshot of trajectories from the BSMs received from all the vehicles and calculates estimated time to arrive (ETA) [3]. The roadside edge servers can use the data to run a machine learning model to reduce the overall wait time in the intersection.

Enhancing security: The sensors of an AV can be the victim of different attacks. For example, the GPS spoofing attack can move the vehicle off the road when the spoofed GPS signal dominates the Extended Kalman Filter (EKF) output [12]. Spoofed objects created by fake LiDAR point clouds [13] can reduce object detection algorithms' accuracy. Sensor data from other vehicles can help the AV to detect spoofed objects or other ongoing attacks. Moreover, shared sensor data can reduce object detection failure [14].

HD mapping: AVs require HD maps to determine the precise location on the road. However, the road scenario can change rapidly due to different reasons (e.g., road closure), requiring the HD map to be updated and disseminated to all the AVs. Edge-assisted RSUs can perform necessary updates in HD maps using the sensor data received from the AVs [4].

On-demand provisioning of computation resources can ensure the QoS of the applications discussed above. For this purpose, drones are one of the most feasible mediums for carrying the edge computing devices to the service area considering the small payload carrying capability. They are already being used for various purposes, such as package delivery [15] and power line monitoring [16]. Drones can be controlled remotely to provide updated instructions and interactive services. Nowadays, edge computing devices are becoming smaller and possess excellent computation capabilities with very low power requirement. Such properties of edge computing devices have made the drones one of the most prominent mediums for carrying and deploying them as RSUs.

Authorized licensed use limited to: UNIV OF ALABAMA-BIRMINGHAM. Downloaded on March 05,2023 at 04:45:15 UTC from IEEE Xplore. Restrictions apply.

III. Proposed Framework

In this section, we explain the details of the R-CAV framework. The framework explanation contains the details of hardware setup inside a drone, R-CAV service framework, optimized drone selection, and task assignment mechanism.

A. Hardware setup

A roadside unit consists of different hardware to operate correctly, such as an edge computing device that serves as an edge server, control board, 4G/LTE cellular communication device, battery for power supply, a V2X communication module, and a GPS device. Here we explain all the hardware components of a drone-based RSU.

Edge server (edge computing platform): Edge computing devices are continuously improving with enhanced computation capability. Nvidia is one of the leading manufacturers of embedded edge accelerator devices. Their popular AI platforms include Jetson TX1, Jetson TX2, Jetson Nano, Jetson Xavier, etc. Google has an edge computing device named Coral tensor processing unit (TPU) that uses TensorFlow lite and provides high-speed neural network performance at a low power cost. Besides these, Intel has some other high-performance edge devices such as integrated graphics processing unit (eGPU), vision processing unit (VPU), and field programmable gate array (FPGA). Considering different aspects, such as power requirement, weight, calculation capability, and price, we choose Jetson Nano as the edge computing device to mount in the drone. Jetson nano is a power-efficient embedded AI computing device powered by NVIDIA Maxwell GPU architecture with 128 CUDA cores. The device has a quad-core ARM A57 CPU with 4 GB memory and 16 GB eMMC. The CPU-GPU architecture in Jetson nano is suitable for accelerating deep learning calculations, which is an integral part of the CAV applications. The other edge computing devices discussed above can also be mounted in the drone, which can provide different computation capabilities.

Control board: Raspberry Pi is an excellent component that can work as the control board for controlling the roadside edge servers. The controller device can communicate with the vehicles to send and receive data, update the instructions received from the central cloud, send back the service summary, and control the drone trajectory.

V2X communication kit: V2X communication is required for communication among the vehicles (V2V) and communication between vehicle and roadside infrastructures (V2I) such as roadside edge servers. Such V2X communication kits include UBX-P3, Qualcomm C-V2X, Qualcomm AG15 , NXP SAF5400, etc. The drone is also equipped with a 4G/LTE modem to communicate with the cloud.

Power source: A 5V power supply is required to run the control board and edge computing device. In our framework, a LiPo 2S battery works as the power source for the devices to operate properly.

B. R-CAV framework

R-CAV framework consists of three separate modules: cloud resource management module, task management module, and the drone reporting module. Figure 3 provides the detailed

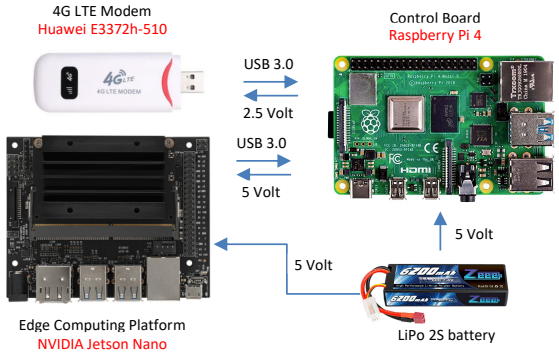

Fig. 2. Hardware setup inside a drone

architecture of the R-CAV framework. The components are explained below:

1) Cloud resource management module: The module is responsible for managing all the drones in different drone centers and assign new drones based on the request received from the task management module. The module runs as a service in the cloud. It has two components:

Resource requirement analyzer: The component is responsible for analyzing the requirements of received resource requests and figuring out the drones with the best suitable devices from the Resource Database. This component also considers the constraints for optimizing service time and energy consumption during drone selection.

Resource pool manager: The resource pool manager maintains the details of all the drones and installed devices in them, such as the control board, communication devices, edge computing devices, etc. in the resource database. It also keeps tract of the drones which are currently providing service.

2) Task management module: The task management module works in a specific intersection on the road and responsible for assigning and monitoring tasks to the drones based on the service requests received from the vehicles. The task management module has the following components:

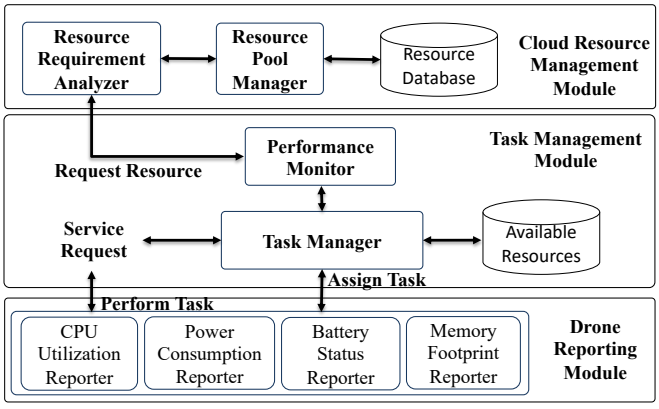

Fig. 3. System Framework of R-CAV

Task Manager: The task manager is a program that runs in a device in a fixed infrastructure at each intersection. The component is responsible for assigning tasks that are received from the vehicles to compatible drone-based RSUs. Moreover, task manager also keeps track of the tasks' status, and updates the available resource list.

Authorized licensed use limited to: UNIV OF ALABAMA-BIRMINGHAM. Downloaded on March 05,2023 at 04:45:15 UTC from IEEE Xplore. Restrictions apply.

Performance monitor: The performance monitor component tracks the resource utilization in terms of power consumption, memory, bandwidth, and CPU utilization of the edge computing devices in RSUs and also ensures satisfactory QoS for all the running applications. It can send new resource request to the cloud resource management module if the number of available drones drops below a threshold. For example, if more than 80% of the drones become busy in providing service, then the component may send new resource request. Moreover, the component can send back the drones to the drone center which remain idle for a specific period of time, do not work properly, or have low battery status.

3) Drone Reporting Module: The drone reporting module is responsible for reporting the current status of the services periodically to the task management module. The control board executes the drone reporting module components.

CPU Utilization Reporter: CPU utilization refers to different utilization metrics of edge computing devices such as memory bandwidth, cache, CPU cycles, etc. CPU utilization reporter periodically checks these metrics and sends the report task management module.

Power Consumption Reporter: This component is responsible for keeping track of consumed energy by the control board and edge computing device for executing CAV applications.

Battery Status Reporter: The battery status reporter component monitors and reports the drone battery status to determine the drone's return time. Drones are expected to maintain at least 20% battery after returning at the end of the service.

Memory Footprint Reporter: In R-CAV architecture, different CAV applications may take different amounts of memory for calculation. The memory footprint reporter tracks and reports the edge computing device's memory usage.

C. Optimizing drone selection

Optimizing the drone selection refers to the optimization of power consumption by the drone and total service time. The cloud resource management module selects the drones based on these optimization constraints and resource requirements to satisfy the QoS. If there are n available drones and m resource requests, we can define an optimization problem as follows:

$$minimize \sum_{j=1}^{n}\sum_{i=1}^{m}\{E_j \times x_i + T_j \times y_i\} \times a_{ij} \quad (1)$$

such that

$$\forall j \epsilon n : E_j \times a_{ij} \leq E_i \quad (2)$$

$$\forall j \epsilon n : T_j \times a_{ij} \leq T_i \quad (3)$$

$$\forall j \epsilon n : \sum_{i=1}^{m} a_{ij} \leq 1 \quad (4)$$

$$a_{ij} \epsilon \{0, 1\} \quad (5)$$

Here, x_i and y_i denotes the optimization weights of energy consumption and time for a specific service i where $0 \leq x_i, y_i \leq 1$ and $x + y = 1$. The weights influence the drone selection and service strategy. More weight on y_i optimizes the service latency by selecting more drones with higher computation and bandwidth capability, such as Nvidia Jetson TX2 or Nvidia Jetson Xavier NX, which incurs more energy consumption. Conversely, a higher value in x_i optimizes total energy consumption by selecting fewer drones that may result in a higher latency in service. For example, cooperative perception requires low latency, while creating or updating an HD map allows more time. However, irrespective of the value of x_i and y_i, each drone j is required to maintain a threshold of energy consumption and time for providing a service i. The thresholds are denoted by E_i and T_i, respectively (explained by constraints 2 and 3). The constraints ensure that the time and energy consumption maintains the quality of service by not crossing the threshold. Constraint 4 ensures that each drone is assigned to provide service at one intersection. Finally, constraint 5 denotes the decision variable a_{ij} that decides whether the drone j is selected to provide service i.

D. Task management in R-CAV

In the R-CAV framework, a vehicle can send a task request, and the task manager assigns an RSU to perform the task. Figure 4 shows the task management process in R-CAV. The task management steps are as follows: **Step 1:** A vehicle sends a request to perform a task to the task manager with application details and required throughput. **Step 2:** The task manager figures out appropriate RSU based on the task's requirement, such as application type, latency requirements, available RSUs, computation capability requirement to run the application, etc. If no drone is found among the drones providing service currently, a new drone boots up and is assigned for that service. Algorithm 1 provides the algorithm for generic task assignment. **Step 3:** The task manager creates a token for the assigned drone and sends it to the vehicle. **Step 4 & 5:** The vehicle connects with the RSU using the token and starts sending the sensor data and receive results. The RSU adds the sensor data inside a queue and performs the calculation by extracting the data from the queue. **Step 6:** The token becomes invalid once the service ends, and the RSU informs the task manager about the end of the service.

Algorithm 1: Generic Task Assignment Algorithm

Input: Vehicle ID $id_{vehicle}$, Task $task$, Throughput requirement req_{thp}, Application $application$
Output: Chosen drone D for the task $task$

$List < Drone > activeDrones =$ $getActiveDrones(application, req_{thp})$
for D **in** $activeDrones$ **do**
  if $D.canOperate(req_{thp}, task)$ **then**
    $D.assign(task)$
    return D
  end if
end for
$List < Drone > availableDrones =$ $getInactiveDrones(application, req_{thp})$
for D **in** $availableDrones$ **do**
  if $D.canOperate(req_{thp}, task)$ **then**
    $bootDrone(D, application)$
    $D.assign(task)$
    return D
  end if
end for
Drop $task$ and return $null$

Authorized licensed use limited to: UNIV OF ALABAMA-BIRMINGHAM. Downloaded on March 05,2023 at 04:45:15 UTC from IEEE Xplore. Restrictions apply.

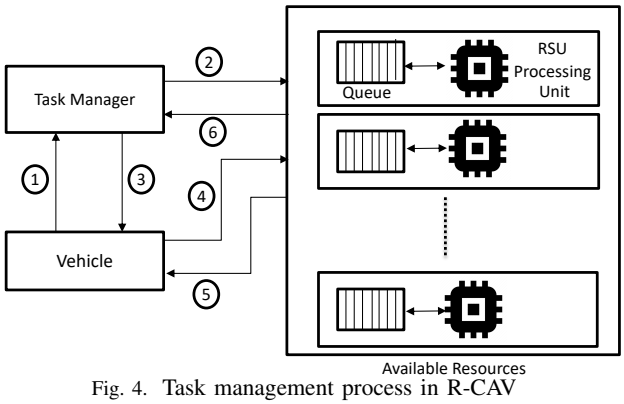

Fig. 4. Task management process in R-CAV

IV. Experiment and Evaluation

In this section, we explain the experiments to evaluate the proposed framework. Initially, we used a simulator to analyze the mobility pattern of vehicles in an urban environment. Later, we implemented an R-CAV framework prototype and evaluated using an object detection algorithm based on the vehicle mobility pattern. Through our experiments, we tried to answer the following questions:

1) What can be the potential resource requirements for a particular intersection in an urban area?
2) Can the framework provide the output of a task within the required timeframe for varying demand of resources, i.e., is the framework scalable?

A. Mobility pattern of vehicles

To predict the arrival of tasks rate, we need to understand vehicles' mobility patterns in urban areas. For this purpose, we used Carla, an unreal engine-based AD simulator. We selected Town01 for the experiment, which was an urban simulation area. We picked an intersection in Town01 and assumed that one or more drone-based RSU are deployed at the intersection. We varied the total number of vehicles to generate different driving scenarios and calculated the number of vehicles present inside the communication range of the RSU at a timestamp. The communication range was set according to the DSRC standards. For each scenario, the simulation was run for around 25 minutes. Figure 5 shows the number of vehicles present inside the communication range of an RSU.

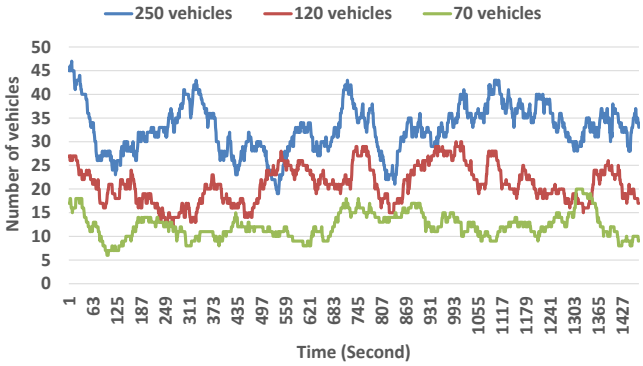

Fig. 5. Number of cars present inside the communication range of one RSU with varying total number of cars present in simulation area

B. Number of drones

We implemented a prototype of the R-CAV framework in python. The framework booted new virtual drones when required that contained computation capability similar to a Jetson Nano device. As different CAV applications required different computation capabilities, we used the TinyYOLO object detection algorithm as our experiment's standard. Different benchmark study shows that jetson nano can operate at around 25 frames per second (FPS) for TinyYOLO [17]. We also observed a similar result in our empirical study where the device provided around 23 to 26 FPS for the same algorithm. Different vehicles could send service requests for object detection tasks with varying throughput constraints in our implemented prototype. Initially, the task manager checked whether it was possible to assign the task to an active drone. Otherwise, the task manager booted a new virtual drone assuming the drone had computation capability similar to the jetson nano device. We calculated the total number of booted drones for a different number of vehicles. We also varied the throughput requirements and observed the total number of drones booted. Figure 6 shows the number of required drones concerning the number of vehicles present inside the communication range of an RSU.

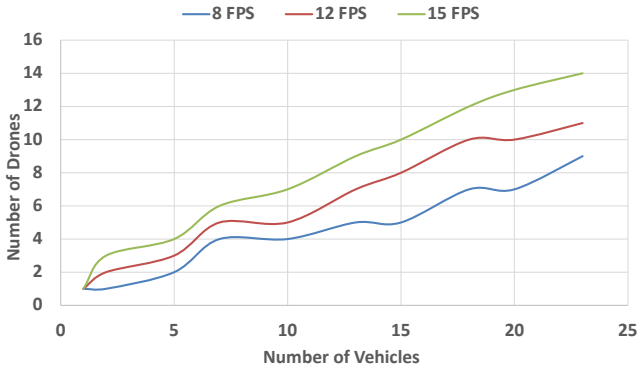

Fig. 6. Number of drones required for different number of vehicles with variable QoS requirements

C. Scalability

To test the framework's scalability, we calculated latency probability distribution for image processing tasks using TinyYOLO with varying resource requirements. In the experiment, we used three Jetson Nano devices where each device had an external memory of 64 GB. In our empirical study, the Jetson nano device took around 0.04 seconds to perform the object detection task for one image. We used KITTI benchmark dataset for our experiment to calculate the object detection latency. The number of vehicles was varied where each vehicle sent a request with a fixed throughput requirement to the R-CAV framework. The throughput requirement was set to 7 FPS. Figure 7 shows the CDF of image processing latency. The R-CAV framework successfully fulfilled the throughput requirements as the CDF went to 1 within acceptable latency. However, the framework dropped tasks after accepting requests from 10 vehicles as no more device was available. The framework also incurred a slight overhead that increased the latency when more vehicles sent service requests. Our experiment result indicated that the framework was successfully able to scale the service with an increased number of vehicles. Hence, the result showed the scalability of the R-CAV framework.

Authorized licensed use limited to: UNIV OF ALABAMA-BIRMINGHAM. Downloaded on March 05,2023 at 04:45:15 UTC from IEEE Xplore. Restrictions apply.

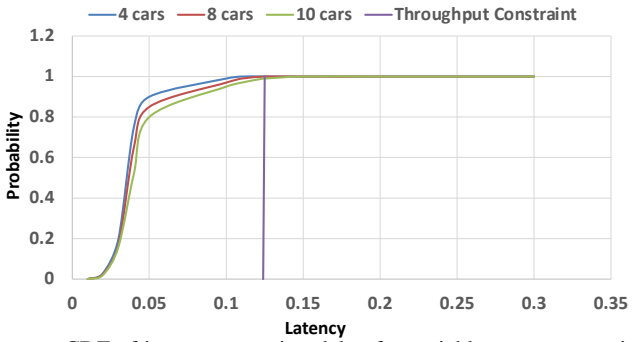

Fig. 7. CDF of image processing delay for variable resource requirements

V. Related Works

We observe the usage of drones for different connected vehicle applications in the literature. VESPER [18] is a real-time data processing framework for connected vehicles capable of adapting different algorithms based on link quality. However, the framework compromises with the accuracy to reduce latency. R-CAV overcomes the shortcoming through on-demand provisioning of RSUs. Besides data-intensive CAV applications, different research works proposed efficient RSU placement methods for VANETs. Seliem et al. [19] proposed a drone-based packet delivery service for highway VANET architecture. Menouar et al. [20] present different UAV-enabled intelligent transportation system applications for smart cities such as flying accident report agent, flying police eye, etc.

Using drones for strategic cloudlet installation has been focused on different research works. Bhatta et al. [21] proposed a cost-aware cloudlet placement in an edge-computing system that reduces the execution time. Moreover, Drones are used for IoT-based applications to collect sensor data, reduce communication time and energy-cost. Such applications include crowd surveillance [22], machine-to-machine based drone management system for different IoT based services [23], etc. Drones can also be used for provisioning or down-scaling the number of IoT gateways to enhance QoS of an IoT network [24]. Most of these research works have focused on specific drone-based applications. However, little attention has been focused on developing an on-demand edge computing platform deployment framework for CAV applications using drones. Our work focuses on this issue to ensure scalability and QoS of CAV applications.

VI. Conclusion and Future Work

Autonomous vehicles are expected to be an integral part of the urban transportation system which can share sensor data to enable different CAV applications. Such applications require deployment of roadside edge servers. However, the computation capability requirement in RSUs may vary. Hence, we propose a framework for on-demand RSU deployment using drones. Our proof of concept implementation and evaluation using Nvidia Jetson nano for object detection algorithm at different throughput constraints shows the feasibility of the proposed framework. In the future, we plan to evaluate the framework with other CAV applications such as cooperative perception using LiDAR data and HD mapping. We also plan to evaluate the performance using other metrics such as memory bandwidth usage, cache behavior, and QoS-resource utilization curve.

Acknowledgement

This research was supported by the National Science Foundation through awards ECCS-1952090, DGE-1723768, ACI-1642078, and CNS-1351038.

References

[1] S. O.-R. A. V. S. Committee *et al.*, "Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems," *SAE Standard J*, vol. 3016, pp. 1–16, 2014.
[2] Q. Chen and X. Ma, "F-cooper: feature based cooperative perception for autonomous vehicle edge computing system using 3d point clouds," in *ACM/IEEE SEC*, 2019, pp. 88–100.
[3] S. E. Huang, W. Wong, Y. Feng, Q. A. Chen, Z. M. Mao, and H. X. Liu, "Impact evaluation of falsified data attacks on connected vehicle based traffic signal control," *ACM AutoSec*, 2021.
[4] J. Zhang and K. B. Letaief, "Mobile edge intelligence and computing for the internet of vehicles," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 246–261, 2019.
[5] S. Liu, L. Liu, J. Tang, B. Yu, Y. Wang, and W. Shi, "Edge computing for autonomous driving: Opportunities and challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1697–1716, 2019.
[6] D. Analyst, "Why drones are the future of the internet of things," 2014. [Online]. Available: https://droneanalyst.com/2014/12/01/drones-are-the-future-of-iot/
[7] L. Liu and Y. Yao, "Equinox: A road-side edge computing experimental platform for cavs," in *IEEE MetroCAD*. IEEE, 2020, pp. 41–42.
[8] USDoT, "Vehicle safetycommunications project - final rep." Apr. 2006.
[9] G. M. N. Ali and E. Chan, "Co-operative data access in multiple road side units (rsus)-based vehicular ad hoc networks (vanets)," in *ATNAC*. IEEE, 2011, pp. 1–6.
[10] "40+ corporations working on autonomous vehicles," 2020. [Online]. Available: https://www.cbinsights.com/research/autonomous-driverless-vehicles-corporations-list/
[11] "200 million connected cars by 2025 says juniper research," 2021. [Online]. Available: https://which-50.com/200-million-connected-cars-by-2025-says-juniper-research/
[12] J. Shen and J. Y. Won, "Drift with devil: Security of multi-sensor fusion based localization in high-level autonomous driving under gps spoofing," *USENIX Security*, 2020.
[13] Y. Cao and C. Xiao, "Adversarial sensor attack on lidar-based perception in autonomous driving," in *ACM CCS*, 2019, pp. 2267–2281.
[14] S. Saxena and Isukapati, "Multiagent sensor fusion for connected & autonomous vehicles to enhance navigation safety," in *IEEE ITSC*. IEEE, 2019, pp. 2490–2495.
[15] Amazon, "Amazon prime air," 2016. [Online]. Available: https://www.amazon.com/b?node=8037720011
[16] A. Varghese, J. Gubbi, H. Sharma, and P. Balamuralidhar, "Power infrastructure monitoring and damage detection using drone captured images," in *IJCNN*. IEEE, 2017, pp. 1681–1687.
[17] "Jetson nano: Deep learning inference benchmarks," 2021. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-dl-inference-benchmarks
[18] K.-L. Wright, "Vesper: A real-time processing framework for vehicle perception augmentation," in *IEEE IECCO*, 2019.
[19] H. Seliem, R. Shahidi, and M. S. Shehata, "Drone-based highway-vanet and das service," *IEEE Access*, vol. 6, pp. 20 125–20 137, 2018.
[20] H. Menouar and I. Guvenc, "Uav-enabled intelligent transportation systems for the smart city: Applications and challenges," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 22–28, 2017.
[21] D. Bhatta and L. Mashayekhy, "Cost-aware cloudlet placement in edge computing systems," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, 2019, pp. 292–294.
[22] N. H. Motlagh, M. Bagaa, and T. Taleb, "Uav-based iot platform: A crowd surveillance use case," *IEEE Communications Magazine*, vol. 55, no. 2, pp. 128–134, 2017.
[23] S.-C. Choi, N.-M. Sung, J.-H. Park, I.-Y. Ahn, and J. Kim, "Enabling drone as a service: Onem2m-based uav/drone management system," in *ICUFN*. IEEE, 2017, pp. 18–20.
[24] M. A. Hoque, M. Hossain, and R. Hasan, "IGaaS: An IoT Gateway-as-a-Service for On-demand Provisioning of IoT Gateways," in *2020 WF-IoT*. IEEE, 2020.

Authorized licensed use limited to: UNIV OF ALABAMA-BIRMINGHAM. Downloaded on March 05,2023 at 04:45:15 UTC from IEEE Xplore. Restrictions apply.