Adaptive Computation Partitioning and Offloading in Real-Time Sustainable Vehicular Edge Computing

Yu-Jen Ku, *Student Member, IEEE*, Sabur Baidya, *Member, IEEE*, and Sujit Dey, *Fellow, IEEE*

***Abstract*—In this paper, we explore the feasibility of solar-powered road-side unit (SRSU)-assisted vehicular edge computing (VEC) system, where SRSU is equipped with small cell base station (SBS) and VEC server, both of which are powered solely by solar energy. However, the limited capacity of solar energy, VEC server's computing, and SBS's bandwidth resources may prohibit vehicle users (VUs) from offloading their vehicular applications to VEC server for better service quality. We address this challenge by dynamically determining vehicular task partitioning and offloading, VEC server's system configuration, and vehicular application level adjustment decisions. We aim at minimizing the end-to-end delay of vehicular applications while maximizing their application level performance (e.g., accuracy). We also implement an object detection vehicular application on an edge computing platform and measure the corresponding energy consumption, computation delay, and detection accuracy performance to establish empirical models for the SRSU-assisted VEC system. We then propose a dynamic programming-based heuristic algorithm which jointly makes the task partitioning and offloading, as well as system and application-level adaption decisions in real-time. We build a simulation framework with the above empirical models to evaluate the proposed algorithm. The simulation results show that our proposed approach can significantly reduce the end-to-end delay while maximizing the detection accuracy compared to existing techniques.**

***Index Terms*—Vehicular applications, edge computing, task partitioning, task offloading, split computing, renewable energy, solar power, road side unit.**

I. Introduction

THE rapid advancement in vehicular technology in recent years has enabled the modern vehicles to be equipped with a wide range of vehicular applications, many of which are based on compute-intensive machine learning based algorithms. The vehicular local computing (VLC) units often cannot satisfy the computing demands of such applications, due to limited computing resources, or contention with other applications. A promising solution to resolve this problem is using the emerging new generation of Road-side Units (RSUs), consisting of a small cell base station (SBS) and a vehicular edge computing (VEC) [1], [2] server. The VEC servers being one hop wireless distance away from the Vehicle Users (VUs), provide much lower communication delay compared to using conventional cloud computing resources, or even mobile edge computing units in wireless networks. VEC servers, although inferior to cloud or mobile edge computing servers, have more computing capabilities than individual VLC resources in most vehicles. By VUs offloading the compute-intensive vehicular applications like object detection to RSUs, VUs can receive better service quality and improve driving experience.

However, transportation systems need dense deployment of RSUs, especially in urban areas, to support the high density of VUs. The dense deployment will significantly increase the cellular networks' energy consumption, thus worsening the carbon footprint. Recent studies have projected 110 million tons of carbon dioxide equivalent (CO_{2e}) emitted by the operation of base stations in global cellular networks in 2030 [3], [4]. Therefore, the future dense RSUs should be deployed without increasing the cellular network's greenhouse gas emission burden. In our previous work [5], we proposed the use of Solar-powered RSU (SRSU), which consists of SBS, VEC server, and a self-sustained solar energy system. Note that in an SRSU, the generated solar energy is limited and fluctuating. If solar energy cannot meet the SRSU's power demand, the SRSU will need to reduce its computing and communication loads by preventing some of the VUs from offloading their applications to the VEC server.

In this work, we assume that each VU has a VLC node which can be supplemented with VEC resource to execute the VU's application. To efficiently utilize the computation resources of VLC node and VEC server, we consider a dynamic offloading scenario, where different subtasks of an application can be chosen to be either executed locally at the VLC node, or offloaded to a VEC server. The dynamic offloading decisions will depend on current computing, communication, and energy resources of the serving SRSU, as well as the VLC node capacity and channel condition for each VU.

In our previous work [5], we aimed at minimizing the disruption of vehicular applications due to the limited solar energy supply in real-time by optimally partitioning and executing the application tasks to either VLC nodes or VEC servers. The vehicular application is considered as disrupted when its delay requirement cannot be satisfied. However, the proposed method does not consider the potential latency improvement

Manuscript received May 7, 2021; revised August 16, 2021; accepted September 16, 2021. This work was supported in part by the National Science Foundation under Grant CNS-1619184, the UC San Diego Center for Wireless Communications, and the Smart Transportation Innovation Program (STIP). The review of this paper was coordinated by A. L. Grieco. *(Corresponding author: Yu-Jen Ku.)*

Yu-Jen Ku and Sujit Dey are with the Department of Electrical and Computer Engineering, University of California, San Diego, La Jolla, CA 92093 USA (e-mail: yuku@eng.ucsd.edu; dey@eng.ucsd.edu).

Sabur Baidya is with the Department of Computer Science and Engineering, University of Louisville, Louisville, KY 40292 USA (e-mail: sabur.baidya@louisville.edu).

Digital Object Identifier 10.1109/TVT.2021.3119585

under an energy constraint by dynamically changing the system configuration of the VEC server. In one of our recent studies [6], we showed the average computation delay can be further optimized satisfying a given energy constraint by appropriately configuring the CPU and GPU frequencies of the VEC server. Additionally, the proposed offloading method in [5] does not consider the potential delay improvement achievable by adapting application-level performance parameters. For example, machine vision based vehicular applications can lower their processed image quality for reducing the size of the data for offloading to further minimize the transmission delay, however, with possible impact on application-level performance, such as object detection accuracy.

In this paper, given the current channel condition and VLC node capacity of each VU, as well as the current computing, communication and energy resources of the SRSU, we aim at minimizing the end-to-end delay of the vehicular application and maximizing the application's performance. We propose to dynamically determine the task partitioning and offloading, VEC server's system configuration, and VUs' application-level performance adaptions. The decisions are calculated in real-time to accommodate the rapidly changing locations and channel conditions of the VUs. To show in real-world the benefit of our proposed method, we consider a vehicular object detection application, which is an essential building block for various complex vehicular applications such as Advanced driver-assistance systems (ADAS), path planning, and navigation. We implement the object detection application using SSD-MobileNetV2 [7] on an edge computing platform Nvidia Jetson TX2 Board [8]. With extensive experiments, we establish empirical energy consumption, end-to-end delay, and detection accuracy models, which are used in simulation-based evaluations for the proposed method. The simulation results show that our proposed approach can significantly reduce the end-to-end delay while maximizing the detection accuracy compared to existing strategies.

The main contributions of this work are summarized below.

1) To the best of our knowledge, this is the first work to optimize delay and accuracy performance of a vehicular object detection application for the SRSU-assisted VEC system using task partitioning and offloading, as well as joint system and application-level adaptations.
2) Specifically, we develop a technique which determines in real-time the optimal VEC server's hardware configuration, image quality for detection, and task partitioning and offloading decisions for an SRSU-assisted VEC system.
3) Using a real-world edge computing platform, we establish empirical models of the energy consumption, computing capacity, end-to-end delay, and accuracy for an SRSU-assisted VEC system.
4) To demonstrate the effectiveness of the proposed technique, we develop a simulation framework consisting of real-world solar generation, urban traffic traces, and the above empirical models. The simulation results show that the proposed approach significantly improves the end-to-end delay and accuracy compared to existing techniques.

II. Related Work

There have been many studies on computation task offloading for vehicular edge computing [9]–[12]. These studies focus on task offloading strategies which leverage computing resources at the edge to minimize the task completion delay [9], [11], [12], while maximizing the edge computing resource utilization [10] or the number of offloaded tasks [9]. Their approaches address the challenges in highly varying bandwidths under the constraint on computation delay [11] or vehicle's energy consumption [12] for real-time applications. However, these techniques do not study the trade-off between leveraging VLC node and VEC server, and hence can not be used for task partitioning according to the computing capacities of both VLC node and VEC server. Since in our considered scenario, both the capacities of VLC node and VEC server are limited, these resources need to be carefully allocated to computation tasks to achieve real-time computation delay.

To facilitate computation capacity-aware task offloading, [13] proposed a learning-based task partitioning and scheduling algorithm which partitions and assigns subtasks among multiple VEC servers to minimize the completion delay and handover-induced service disruption. The technique requires data exchange between multiple RSUs, which will cause prohibitively large communication delay when applied to our high-data volume vehicular perception applications. [14]–[16], on the other hand, study the optimal computation task partitioning between VEC server and VLC node by proposing joint task partitioning and offloading as well as SBS communication and VEC server computation resource allocation methods. Among these studies, [14] and [15] aim at minimizing the system cost in terms of utilized communication and computation resources, under delay constraints while [16] jointly minimizing the task execution delay and the utilized VEC server's computing resources. However, the partitioning techniques proposed in these studies cannot be applied to subtasks with task dependencies as they assume the computation tasks can be arbitrary partitioned, offloaded, and executed in parallel.

To consider task dependency during partitioning and offloading, [17], [18] divide the sequential convolution layers of a Deep Neural Network (DNN) into several independent subtasks. These subtasks can be executed in parallel on multiple edge nodes to minimize task completion time [17] or the utilized edge server memory [18] as well as the communication overhead for task offloading. However, the proposed parallel partitioning techniques cannot leverage the potential reduction of communication delay with sequential partitioning. On the other hand, [19] and [20] enable local devices to early stop a DNN and offload the result to edge server. The edge server can choose to adopt the result or further execute the rest layers of the DNN. [19] aims at minimizing the execution delay and [20] aims at minimizing the utilized communication and computing resources on local and edge devices while maximizing the object detection results transmitted to the edge server under communication resource constraint. [21] proposes a real-time task partitioning and bandwidth allocation strategy to maximize the throughput (i.e. the number of processed data per second) using limited edge

TABLE I
SUMMARY OF KEY NOTATIONS AND ABBREVIATIONS

Notation	Description	Notation	Description
τ	the duration of current time slot	$\mathcal{I}$	current VU set
b	the SRSU for study	i	the index of VU
η_{bi}	the current SNR of uplink transmission from VU i to SRSU b	I	total number of VUs in the VU set
$r_{b,i}$	uplink transmission rate from VU i to SRSU b	$\mathcal{W}_i$	bandwidth allocated by SBS b to VU i
$\mathcal{K}$	subtask set for the vehicular application	$\omega_{k,q,i}$	input data size of subtask k of VU i using application parameter q
k	index of the subtask	$\omega_{(k+1),q,i}$	the output data size of subtask k of VU i using application parameter q
$T_{k,i}$	computation delay of subtask k	$T_{tx,i}$	transmission delay for transmiting the required data for offloading
d_i	end-to-end delay of VU i	a_i	accuracy of the vehicular application of VU i
QoS_i	QoS utility of VU i	E_R	Energy consumption of the SRSU
E_S	Energy consumption of the VEC server	E_B	Energy consumption of the SBS
E_c	Energy consumed for executing the offloaded subtasks	E_t	the current available amount of solar energy
$c_{l,i}$	CPU-GPU configuration of VLC node of VU i	c_e	CPU-GPU configuration of the VEC server
y_i	offloading strategy of VU i	q_i	compression level of Vu i
$\mathcal{Q}$	the set of available compression levels for VU	$\mathcal{C}$	the set of available CPU-GPU configurations for the VEC server
h	number of VUs using *Full Offloading*	n	number of total offloading VUs

computing and communication resources. Although these task partitioning methods consider both the computing capacities in VLC nodes and VEC servers during decision making, they are not applicable to VEC servers whose operations are not only constrained by the limited communication and computing resources, but energy availability.

The authors in [22] propose a joint task offloading and user association strategy for the multi-user mobile edge computing system to minimize the overall energy consumption of the users and edge server. However, firstly, the proposed method does not consider task dependency as they assume multiple mutually independent tasks in each user. Secondly, the challenges of minimizing the SRSU's energy consumption is different from the challenges of operating the SRSU under limited energy availability, as the computing and communication resources of SRSU will also be constrained due to the lack of energy. Both [23] and [24] consider renewable energy powered edge server. In [23], task partitioning and offloading as well as the utilization of renewable energy is determined online using Lyapunov optimization to maximize the number of offloaded tasks. The authors in [24] propose an online learning technique for partitioning and offloading of the incoming tasks as well as autoscaling of the computing capacity for the edge servers to jointly minimize the application delay, battery deprecation, and back up power usage. The learning technique is used to predict the system's long-term channel rate and workload states. However, these techniques do not consider the task dependency graph, and the corresponding transmitted data size is linear to the partitioned load. In practice, the computation loads of subtasks in a task graph are discrete and do not possess such linearity relationship with the input data. Therefore, the proposed theoretical approaches can not be applied to our problem.

To the best of our knowledge, this is the first study to consider not only the compute and communication-intensive, delay-sensitive dependency-aware task partitioning and offloading with the collaboration of VLC node and VEC server, but the challenges of utilizing limited communication, computing, and energy resources of an SRSU.

III. Systems Overview

In this section, we introduce an overview of the SRSU-assisted VEC system, including vehicular applications, solar energy-driven communication, and computing paradigms. For ease of reference, we list the key notations in Table I.

A. Network and Channel Models

We consider an SRSU-assisted VEC system consisting of one serving SRSU b and multiple served VUs. The SRSU is equipped with a communication module SBS and a computation module VEC server. The operation time is divided into multiple time slots. Note that the following modellings and discussions are within a single time slot t, for simplicity, we do not attach superscript t for each variable. We denote the duration of time slot t as τ. For each time slot, there exists a set of VUs $\mathcal{I} = \{1, 2, \ldots, I\}$ in the coverage area of the SRSU b and the VUs' locations will vary in different time slots due to the mobility of the vehicles. For each VU $i \in I$, we denote $\eta_{b,i} = \frac{\rho_i * g_{b,i}}{N_0}$ as the current signal-to-noise ratio (SNR) of uplink transmission from VU i to the SBS of SRSU b. ρ_i is the transmit power of VU i, $g_{b,i}$ is the uplink channel gain, and N_0 is the noise level. The uplink transmission rate from VU i to SBS b can be represented as,

$$r_{b,i} = W_i * log_2(1 + \eta_{b,i}) \quad (1)$$

where W_i is the bandwidth allocated by SBS to VU i and the interference from other VUs is negligible with the use of Orthogonal Frequency-Division Multiplexing (OFDM) technology. We ignore the inter-cell interference by assuming it is mitigated by inter-cell interference coordination (ICIC) technologies, e.g. Fractional Frequency Reuse (FFR) [25]. We assume the wireless communication between the SBS and the VUs use C-V2X protocols [26] and the available bandwidths are evenly distributed among the VUs which require uplink transmission. We model the uplink channel gain, $g_{b,i}$, by using B1 Manhattan grid layout [27] as the pathloss and slow fading, and the Nakagami-m distribution [28] as the fast fading, which have been widely used

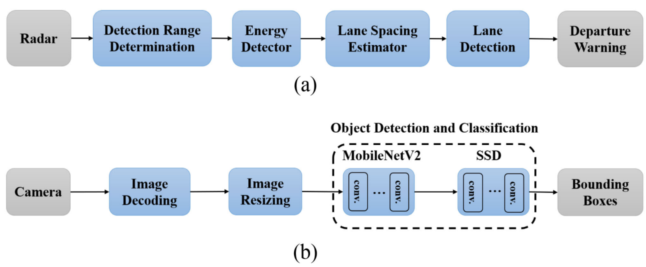


Fig. 1. Task dependency graphs of (a). Radar signal-based lane departure warning system (b). DNN-based object detection and classification application for the cameras.

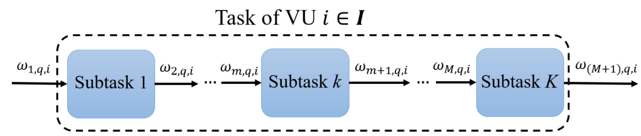


Fig. 2. Subtask breakdown of a vehicular application.

by the industry [29], [30] and are shown to be sufficient to model vehicular communication channels [28].

Compared to the uplink data of the vehicular applications, e.g., the captured images or radar point clouds, whose data sizes are usually more than several KBytes (images) or even several MBytes (point clouds) [31], the data sizes of the vehicular applications' computation results are very small. The computation results, such as the bounding boxes for object detection, classification, and fusion as well as basic safety messages (BSM) for collision detection, are less than 1 KBytes in terms of data size. Moreover, the downlink data rate is usually higher than the uplink data rate due to the higher transmission power [32]. Therefore, we ignore the impact of downlink data transmission in our study.

We assume the duration of time slot, τ, to be small enough so that $r_{b,i}$ is unchanged within one time slot [5]. Note that $r_{b,i}$ will still change across different time slots due to the mobility of VU.

B. Vehicular Task Model

Most vehicular applications involve computation tasks that can be expressed as a dependency graph of sequential subtasks. For example, Fig. 1 shows two dependency graphs of radar and camera-based vehicular applications. In Fig. 1(a), the detection range determination block decides the range of distance to perform the energy detection on the radar signal. Lane interval estimator and lane detector are applied if enough signal energy is detected. If a lane is present and the application detects possible departure due to the vehicle's speed, a departure warning will be sent to the driver [33]. In Fig. 1(b), the image captured by the camera will be decided and resized to be the input feature for the DNN-based object detection and classification, where we use SSD-MobileNetV2 [7] as an example. In Fig. 1(b), conv. is the convolution layer, which is the most common layer in SSD-MobileNetV2. If any object is detected, the application will return the coordination of the bounding boxes for the detected object.

We assume that at each time slot, every VU generates a computation task that consists of a set of $\mathcal{K} = \{1, 2, \ldots, K\}$ sequentially dependent subtasks, as shown in Fig. 2. That is, the data input of subtask k depends on the data output of subtask $k+1$. Therefore, subtask $k+1$ can start only after the completion of subtask k. For each subtask $k \in \mathcal{K}$ of VU i, we assume $\omega_{k,q,i}$ is the input data size and $\omega_{(k+1),q,i}$ is the output data size, where q is an application adaptation parameter. For example, if the considered vehicular application is vehicular machine vision, such as object detection and classification, q can be the encoding bitrate of the input image.

Each subtask can be executed locally in the VLC node or, offloaded and executed at the VEC server. In such cases of computation offloading to edge, data at the task-splitting point, e.g. subtask k', needs to be transmitted over the wireless communication channel, such that the first $\{1, 2, .., k'\}$ subtasks are executed at the VLC node and the remaining $\{k'+1, \ldots, K\}$ subtasks are executed at the VEC server.

C. Performance Metrics

Herein, we consider the performance metrics for object detection, which is a critical component in various complex vehicular applications as mentioned above. Therefore, we primarily focus on two object detection performance metrics - the end-to-end delay and the accuracy.

End-to-end delay: The end-to-end delay in our study is defined as the summation of the computing delay of each subtask in $\mathcal{K}$ and the communication delay of transmitting the required data for computation offloading. Therefore, the end-to-end delay of VU i can be represented as,

$$d_i = \sum_{k=1}^{K} T_{k,i} + T_{tx,i} \tag{2}$$

where $T_{k,i}$ is the computing delay of subtask k. $T_{tx,i}$ is the transmission delay for offloading subtask k' and its subsequent subtasks to the VEC server, that is, for transmitting the input of subtask k' with data size $\omega_{k,'q,i}$. Therefore, $T_{tx,i}$ can be defined as $\frac{\omega_{k,'q,i}}{r_{b,i}}$. $T_{tx,i} = 0$ if all the subtasks of VU i are executed locally.

Note that as we focus on optimizing the end-to-end delay in this study, without loss of generality, we assume the data is processed frame-by-frame in the vehicular application, that is, subtask 1 starts processing the next input data after subtask K, which is the last subtask in $\mathcal{K}$, finishes processing the previous input. Therefore, queuing delay is negligible in the network.

Accuracy: The accuracy of the object detection a_i of VU i can be represented as a function of the application level adaption parameter q, namely, $a_i = a(q_i)$, where q_i is the parameter q used by VU i. In this paper, we take compression level of the input image (i.e. the encoding bitrates of the jpeg compressed image) as an example of the application adaptation parameter. We measure the accuracy in terms of the intersection over union (IoU). IoU is the intersection over union of the areas of the bounding boxes of the detected objects in the input image with respect to the result of the uncompressed base image.

Assume for image m, the bounding box area of the detected objects in the base image (i.e., at $q = 1$, the lowest possible compression level) is N_1^m, and the bounding box area of the detected objects in the corresponding input image (with compression level q') frame is $N_{q'}^m$. The accuracy of this input image is defined as: $(N_1^m \cap N_{q'}^m)/(N_1^m \cup N_{q'}^m)$. Without loss of generality, we define the overall accuracy of images at image quality q' by averaging the accuracy over M different frames,

$$a(q') = \frac{1}{M} \sum_{m=1}^{M} \frac{(N_1^m \cap N_{q'}^m)}{(N_1^m \cup N_{q'}^m)} \tag{3}$$

where M is a large whole number. Note that $(0 < a(q') < 1)$.

QoS Utility: The quality of service (QoS) of vehicular applications aims to have lower delay and higher accuracy. However, higher accuracy is usually achieved by larger data size, that potentially impacts the end-to-end delay, which is a function of data transmission time and computation delay at the VLC node and VEC server as expressed in (2). Therefore, the system needs to consider a trade-off between end-to-end delay and accuracy. In this paper, we define a joint performance metric, QoS utility, which we represent as a weighted function of the end-to-end delay and accuracy. For each VU i, the QoS utility is defined as,

$$QoS_i = \alpha \frac{d_n}{d_i} + (1 - \alpha) * a(q_i) \tag{4}$$

and the average QoS utility of all the current VUs is,

$$Q\hat{o}S = \frac{1}{I} \sum_{i \in \mathcal{I}} QoS_i \tag{5}$$

where d_n is the term used to normalize d_i to the same range of $a(q_i)$. For example, if the value of $a(q_i)$ is between 0 and 1, d_n will be determined as the smallest possible value of d_i. α $(0 < \alpha < 1)$ is the trade-off factor between the end-to-end delay and accuracy, and is determined by the service provider. Higher value of α means the QoS utility emphasizes more on the performance of end-to-end delay and vice versa. For example, for the SRSUs deployed along a highway, where end-to-end delay is critical to driving experience due to high vehicle speed, the service provider can choose higher value of α to focus more on reducing the end-to-end delay than very high accuracy. Meanwhile, because the moving patterns of vehicles on the highway are stable and easy to track across multiple consecutive frames, detection accuracy can be compensated by object tracking techniques so that the impact of the trade-off in accuracy will not affect the driving safety.

D. Energy Consumption and Harvesting at SRSU

The energy consumption of the SRSU E_R consists of the energy consumed by its VEC server and SBS. We denote E_S and E_B be the energy consumed by the VEC server and SBS, respectively. Therefore, $E_R = E_S + E_B$. Note that E_S depends on the load and CPU-GPU configuration c_e of VEC server. The load of VEC server is a function of the offloaded subtasks, therefore, E_S for the current time slot can be represented as,

$$E_S = E_{S,idle} + E_c \tag{6}$$

where $E_{S,idle}$ is the idle energy consumption and E_c is the energy consumed for executing the offloaded subtasks.

On the other hand, the energy consumption of the SBS can be represented as the following,

$$E_B = E_{B,idle} + \sum_i P(r_{b,i}) * T_{tx,i} \tag{7}$$

where $P(r_{b,i})$ is the base-band signal processing power consumption at SBS for uplink transmission at datarate $r_{b,i}$. $T_{tx,i}$ is the transmission time (i.e. the time when SBS is actively processing the uplink signal at datarate $r_{b,i}$).

At the beginning of each time slot, we let E_t be the amount of energy harvested from the solar panel of SRSU b and can be immediately used by the SRSU. Therefore, the energy consumption of VEC server and SBS should satisfy,

$$E_R = E_B + E_S \leq E_t \tag{8}$$

In the following section, we build the empirical system models for the vehicular tasks, performance metrics and energy consumption with real-world control parameters, e.g., application adaptation parameter, computing configurations and load, that instills the non-linear behaviors in the aforementioned system models.

TABLE II
CHOSEN CPU AND GPU CONFIGURATIONS TO EMULATE THE COMPUTING CAPACITY OF VLC NODE AND VEC SERVER

Configuration name	VLC_ config1	VLC_ config2	VEC_ config1	VEC_ config2
CPU cores	2	1	6	6
CPU frequency (MHz)	960	652	{960, 1267, 1574}	{652, 824, 960}
GPU cores	1	1	1	1
GPU frequency (MHz)	114	114	{725, 1300}	{725, 1300}

IV. EMPIRICAL SYSTEM MODEL

We emulate the SRSU-assisted VEC system by using a setup of Nvidia Jetson TX2 boards which are power-efficient embedded AI computing devices [8], and use NI USRP B210 radios for communications. We operate the Nvidia Jetson TX2s at different CPU-GPU configurations to emulate the different computing capacities of the VEC server and VLC nodes. We list the corresponding hardware configurations in Table II, including CPU and GPU frequencies and the number of available CPU and GPU cores. Note that the CPU and GPU frequencies listed in Table II are chosen from the available frequencies allowed by the Nvidia Jetson TX2 board.

We mimic different computing capacities of the VEC server with two configurations, VEC_config1 and VEC_config2, by choosing two sets of available CPU-GPU configurations of the Nvidia Jetson TX2 board. Each set consists of a collection of CPU and GPU frequencies that the VEC server can tune

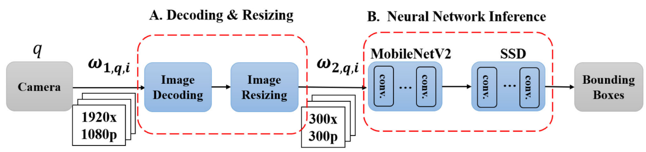


Fig. 3. Task dependency graph of object detection using SSD-MobileNetV2, showing data size and compression level.

to operate at depending on current performance and energy requirements.

Similarly, for VLC nodes, we emulate their computing capacity by choosing two different CPU-GPU configurations, VLC_config1 and VLC_config2. For the disparity of computing capacity between the edge and local computing devices, we choose the lowest available GPU frequency for both VLC_config1 and VLC_config2. Additionally, we assume VLC_config1 and VLC_config2 can only access two and one CPU cores, respectively, on the Nvidia Jetson TX2 board, while both edge configurations can access six CPU cores. Finally, for the CPU frequency of VLC_config1, we choose the lowest frequency listed among the available CPU frequencies of VEC_config1, and likewise, we determine the CPU frequency of VLC_config2 as the lowest CPU frequency listed for VEC_config2.

As mentioned earlier, while we use object detection using a vehicle camera as an example of the real-time vehicular application, our work can be easily extended to other types of vehicular applications as well. We use SSD-MobileNetV2 [7] for object detection due to its lightweight computations, favorable for real-time applications. In this section, first we provide the task model for object detection with SSD-MobileNetV2 and show the impact of corresponding application adaptation parameter, i.e. compression level, on the data size in the task pipeline and accuracy. Second, we will present the empirical model for the computing delay at the VLC node and VEC server w.r.t. different system settings. Third, we will demonstrate the empirical model for energy consumption of the SRSU, including energy consumption at the SBS and VEC server, by extending the theoretical models described in the previous section, considering different system configurations and load conditions.

A. Object Detection Task Model and Impact of Compression

The task graph considered for object detection is shown in Fig. 3. After the vehicle camera captures a 1080p image, the image is decoded and resized into a 2-dimensional 300 by 300 matrix input and forwarded to the neural network of SSD-MobileNetV2 for object detection. For the sake of simplicity, we choose a combination of functional blocks as one subtask, as shown in the dotted boxes A and B (i.e., the following two subtasks, Decoding & Resizing and Neural Network Inference), for the rest of this study. However, note that our approach is not limited to two subtasks and is scalable for more subtask scenarios.

As the application adaptation parameter, we use compression level $q \in \mathcal{Q}$, which is applied to camera, to control the encoding bitrate and hence the data size of images. $\mathcal{Q}$ is the set of available compression levels. The impact of compression level is two-fold. Higher compression level reduces the size of the input image, thus reducing the dataflow transmission time to forward to the next node in the task pipeline, but it also affects the accuracy of the object detection.

TABLE III
INPUT DATA SIZE OF EACH SUBTASKS AT DIFFERENT COMPRESSION LEVEL

Compression level q	1	2	3	4
Encoding bitrate level	100%	75%	50%	25%
$\omega_{1,q,i}$ (KByte)	92	79	55	27
$\omega_{2,q,i}$ (KByte)	270	270	270	270

TABLE IV
ACCURACY AT DIFFERENT COMPRESSION LEVELS

q	1 (Base image)	2	3	4
Accuracy	1.0	0.97	0.93	0.9

Table III shows the data size along the processing flow of the object detection using a compressed 1080p jpeg image, under different values of q. Note that $\omega_{2,q,i}$ is the decoded 300x300 pixels image, therefore, its size is not impacted by the encoding bitrate compression level.

Table IV shows the corresponding impact on accuracy for different values of q for a set of 70 image frames of 1080p resolution. We choose the base image when the compression level is 1 with 100% encoding bitrate. Thus the accuracy for $q = 1$ is 1.0 based on Eq. 3 and it decreases with the increasing values of q. Note that the lowest accuracy in Table IV (i.e. $q_i = 4$), is just 10% less than the accuracy of the base image. However, even with a 10% decrease, the accuracy of SSD-MobileNetV2 is still higher than some of the other object detectors, e.g., YOLOV2 [34], which has been largely used for autonomous vehicles as mentioned in literature [35], [36]. Therefore, our study can still meet the same driving safety performance as other vehicular object detection studies even with the trade-off in accuracy.

B. Computing Delay and Impact of Computing Capacity

Here, we empirically model the computing delay for the object detection using the different computing capacities of VLC node and VEC server. Note that VLC node will run the application for a single VU and thus its computing capacity is impacted by CPU-GPU configurations only as mentioned in Table II. However, the VEC server runs applications for multiple VUs and thus is impacted by both its CPU-GPU configurations and the load in terms of the number of application instances. Now, based on Fig. 3, there are two subtasks in the task graph. Therefore, $K = 2$, and T_1 is the DR delay (execution delay of Decoding and Resizing subtask) and T_2 is the inference delay (execution delay of neural network inference), which together constitutes the computing delay.

1) Computing Delay at the VLC Node: We implement the object detection subtasks in Fig. 3 on the Nvidia Jetson TX2 board and measure T_1 and T_2. Table V shows the observed

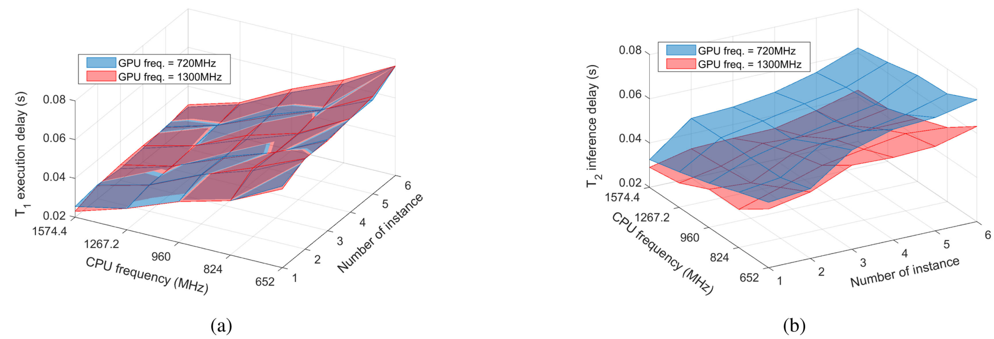


Fig. 4. (a) DR delay T_1 and (b) Inference delay T_2, under different number of running instances and VEC server's computing capacities.

TABLE V
BREAKDOWN OF THE END-TO-END DELAY OF OBJECT DETECTION USING DIFFERENT VLC COMPUTING CAPACITIES

Computing delay (s)	VLC_config1	VLC_config2
T_1	0.036	0.085
T_2	0.095	0.103
Overall	0.131	0.188

computing delay of each subtask for processing an image frame under the VLC configurations listed in Table II. Note that the minimum computing delay of the object detection at the VLC node is higher than 0.130 s, which is not fast enough for the 0.1 s requirements for vehicular applications to react to the fast changing traffic condition [37]. Therefore, offloading some of the subtasks to the VEC server, which has higher computing capacity than VLC node, can reduce the computing delay, and thus, potentially reduce the end-to-end delay. In the next subsection, we demonstrate the empirical model of the computing delay at the VEC server.

2) Computing Delay at the VEC Server: We model the computing delay of subtasks on the VEC server empirically by observing T_1 and T_2 under different VEC server load conditions. Fig. 4 shows the DR delay T_1 and the inference delay of neural network inference T_2, under the conditions of different VEC server capacities and different number of instances of DR and the neural network subtasks. Note that the CPU and GPU frequencies used in Fig. 4 are listed in Table II. In Fig. 4(a), the DR delay does not change between different GPU frequencies because the execution of DR does not use any GPU resource. On the other hand, Fig. 4(b) shows that the relationship between the inference delay and the increasing CPU-GPU frequencies as well as the number of application instances is not easy to represent by simple linear and quadratic models. Therefore, such knowledge of nonlinear correlation between delay and computing capacity shown in Fig. 4 is necessary for accurate delay performance optimization.

C. Energy Consumption Model

1) Energy Consumption at the VEC Server: In our empirical study, we observe the major factors that impact the energy consumption at the VEC server are the server's CPU-GPU configuration, the number of offloading VUs (i.e. the running application instances), and the offloaded computation loads.

Fig. 5(a) shows the energy consumed per second of a Jetson TX2 board as VEC server while a number of VUs offload both DR and the neural network inference simultaneously. The measurement is taken under different CPU-GPU configurations. The energy consumption is linearly increasing with the CPU frequency and number of offloading VUs. However, the increasing rate varies when the Jetson board is operated with different GPU frequencies. In reference to Fig. 4(a), we can see that although higher CPU and GPU frequencies lead to less computing delay, the corresponding energy consumption will be higher. Under the condition when the SRSU lacks of available energy, VEC server needs to reduce its operating CPU-GPU frequency while sacrificing the computing delay of the offloaded subtasks. Fig. 5(b) shows the energy consumed per second while multiple VUs offload only the neural network subtask. While it shows similar trend of increasing energy consumption as Fig. 5(a), its absolute value is less than Fig. 5(a) under fixed CPU-GPU frequency settings and number of instances, because only one of the subtasks (i.e. neural network inference) is executed.

2) Energy Consumption at the SBS: To measure the energy consumed by the wireless communication at the SRSU, we use the same experimental settings as in [38], with one Jetson board and one NI USRP B210 radio to emulate the SBS. The wireless channel is established by srsLTE tool [39], which is used to create an LTE link between SRSU and VU. We create different values of uplink data-rate using *iperf* and measure the corresponding energy consumption on the Jetson TX2 board and the NI USRP B210 radio. The result of the consumed energy per second is reported in Fig. 5(c). Note that due to hardware limitations, the maximum uplink datarate achievable over LTE by our experimental setting is 6 Mbps. Therefore, we use curve fitting approach for the energy consumption model of the SBS at high data rate conditions. It can be observed that the energy consumption $P(r_{b,i})$ is linear to the uplink data rate $r_{b,i}$, that is, $P(r_{b,i}) = 0.14 r_{b,i}$, with the idle power = 4.5 W. Based on (7), we consider the following energy consumption model for E_B.

$$\begin{aligned} E_B &= 4.5\tau + 0.14 \sum_i r_{b,i} * \frac{\omega_{k,'q,i}}{r_{b,i}} \\ &= 4.5\tau + 0.14 \sum_i \omega_{k,'q,i} \qquad (9) \end{aligned}$$

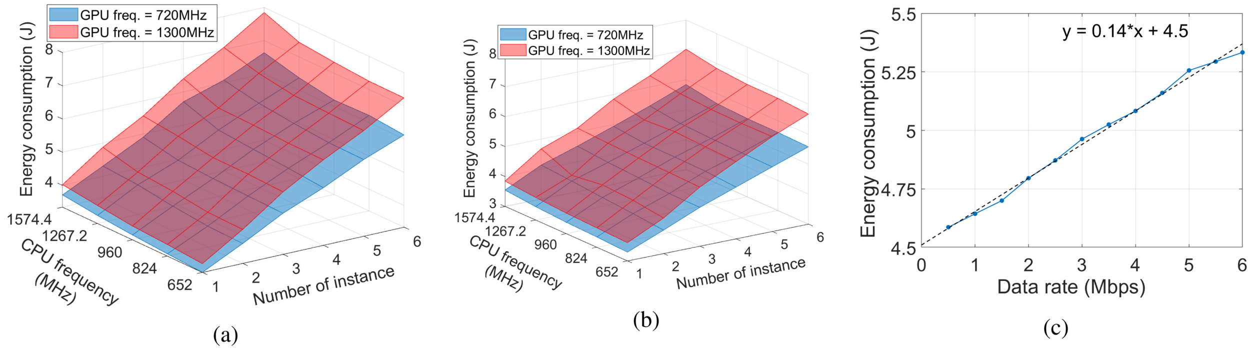


Fig. 5. Energy consumption per second of (a) left, VEC server executes instances of both DR and the neural network inference and (b) center, VEC server executes various instances of the neural network inference under different VEC server's computing capacities, and (c) right, SBS under different uplink data rates.

where $\omega_{k,'q,i}$ is the data size per frame required to be transmitted corresponding to splitting point of subtasks for offloading.

V. Overall Approach and Problem Formulation

A. Task Partitioning and Offloading

From the above real-world system models, we can observe that to optimize a single VU's end-to-end delay, we can offload all of its subtasks to the more powerful VEC server. However, an inferior wireless channel quality can potentially increase the communication delay and thus increase the end-to-end-delay resulting in low QoS utility. Hence, partitioning a task at a point that reduces the data size is desirable, in order to reduce the communication delay. Moreover, when multiple VUs try to offload all their subtasks to one VEC server simultaneously, the resource constraint at the VEC server may increase the average computing delay and can potentially violate the energy constraint in (8). Therefore, we need an optimal offloading strategy that allows VUs to selectively offload part of their subtasks based on their transmission rate, local computing capacity, and current VEC server load, as well as energy constraint.

In this paper, we consider the following three partitioning and offloading strategies denoted by y_i, for a VU with the considered object detection application (1) $y_i = 1$: *Full Offloading*, (2) $y_i = 2$: *Partial Offloading*, and (3) $y_i = 3$: *Encoded Partial Offloading*. We also denote $y_i = 0$ as the *Local Only* strategy, where all the subtasks are executed at the VLC node. The high level block diagrams of these strategies are shown in Fig. 6, where the blocks represent each subtask. Blue blocks indicate the subtask is executed locally and green blocks indicate the subtask is executed at the VEC server. Red dash arrow indicates where the wireless uplink data transmission between the VU and the VEC server happens.

For *Full Offloading*, strategy, VU will transmit the captured image (i.e. with size $\omega_{1,q,i}$) to the SRSU, and hence, offload both of the DR and neural network inference to the VEC server. On the other hand, for *Partial Offloading* strategy, VU will first execute DR subtask at VLC node, then offload the decoded as well as resized 2-dimensional input image features (i.e. with size $\omega_{2,q,i}$) to SRSU, and let the VEC server execute the neural network inference. However, note that the data size after decoding is several times larger than the encoded image, which is not feasible for transmission in real-time unless the transmission rate is very high. Therefore, in this paper, we propose another partial offloading strategy: *Encoded Partial Offloading*.

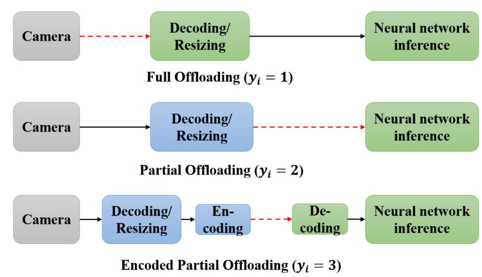


Fig. 6. Possible task partitioning and offloading strategies in object detection application using SSD-MobileNetV2.

TABLE VI
Encoded Data Size for Transmission of *Encoded Partial Offloading* Strategy With Different Compression Levels

Compression level q	1	2	3	4
Encoding bitrate level	100%	75%	50%	25%
$\omega^c_{2,q,i}$ (KByte)	21	18	12.5	6.2

In *Encoded Partial Offloading* strategy, at VLC node, VU will encode the resized image feature again to a jpeg image with the same resolution as the smaller resized image (i.e. 300x300 pixels in the studied example) before transmission. Subsequently, the VEC server will decode the received image to the 2-dimensional image feature and then send to the neural network inference for object detection. Compared to *Partial Offloading* strategy, *Encoded Partial Offloading* strategy incurs overhead of execution of extra encoding at the VLC node and extra decoding at the VEC server, with the trade-off for a high gain in reduction of communication delay due to highly reduced data size. The data size of the 300x300 resized image feature after encoding is shown in Table VI, where $\omega^c_{2,q,i}$ is the encoded data size of $\omega_{2,q,i}$.

1) Impacts of Offloading Strategies to End-to-End Delay: Previously we have separately modeled T_1, T_2 at the VLC node and the VEC server, and T_{tx} under different data rates and transmitted data sizes. In this section we model the end-to-end delay combining the offloading strategies and the above empirical models. While all of the offloading strategies (i.e. $y_i > 0$) offload the neural network inference to the VEC server, only *Full Offloading* strategy offloads the DR subtask. Therefore, we model T_1 for DR as a function of the number of *Full Offloading* users h and T_2 for the neural network inference as a function of the number of total offloading users n, where $h = \sum_{i:y_i=1} 1$ and $n = \sum_{i:y_i>0} 1$.

We use $T_1^l(c_{l,i})$ and $T_1^e(c_e, h)$ to denote DR delay on VLC node and VEC server, respectively, given h, VLC node configuration $c_{l,i}$ and VEC server configuration $c_e \in \mathcal{C}$. $\mathcal{C}$ is the set of available CPU-GPU configurations for the VEC server. Similarly, $T_2^l(c_{l,i})$ and $T_2^e(c_e, n)$ denote the inference delay, respectively, given n, VLC node configuration $c_{l,i}$ and VEC server configuration c_e. Therefore, the end-to-end delay of VU i using y_i offloading strategy can be modeled as,

$$d_i(y_i, q_i, n, h, c_e, c_{l,i}) =$$
$$\begin{cases} T_1^l(c_{l,i}) + T_2^l(c_{l,i}); & \text{if } y_i = 0 \\ T_1^e(c_e, h) + T_2^e(c_e, n) + \frac{\omega_{1,q_i,i}}{r_{i,b}}; & \text{if } y_i = 1 \\ T_1^l(c_{l,i}) + T_2^e(c_e, n) + \frac{\omega_{2,q_i,i}}{r_{i,b}}; & \text{if } y_i = 2 \\ T_1^l(c_{l,i}) + T_2^e(c_e, n) + T_3^l(c_{l,i}) + T_4^e + \frac{\omega_{2,q_i,i}^c}{r_{i,b}}; & \text{if } y_i = 3 \end{cases} \tag{10}$$

where $T_3^l(c_{l,i})$ is the encoding delay for a smaller resized 300x300 pixels jpeg image given the VLC node configuration $c_{l,i}$, and T_4^e is the decoding delay for the same resized image at the VEC server. Based on our observation, T_3 is 0.003 s and 0.007 s, respectively, for VLC configuration VLC_config1 and VLC_config2. We have measured that given the worst VEC server configuration, T_4^e is 0.003 s, which is very small compared to T_1 (i.e. decoding and resizing for 1080p jpeg image). Therefore, we set T_4^e to 0.003 s and ignore the impact of c_e to the value of T_4^e.

Similarily, with the above offloading strategies, based on (8) and 9, we model the empirical total energy consumption of SRSU E_R as,

$$\begin{aligned} E_R(\omega,' n, h, c_e) &= E_S(n, h, c_e) + E_B(\omega') \\ &= \frac{\tau}{n}(h\mathcal{E'}_1(n, c_e) + (n-h)\mathcal{E'}_2(n, c_e)) + 4.5\tau + 0.14\omega' \end{aligned} \tag{11}$$

where ω' is the summation of the data size that needs to be transmitted depending on the decisions of y_i and q_i, $\forall i \in \mathcal{I}$. $\mathcal{E'}_1$ is the energy consumption of the VEC server shown in Fig. 5(a), and $\mathcal{E'}_2$ is the energy consumption shown in Fig. 5(b).

Since not all of the n VUs will offload both of the subtasks simultaneously, for the sake of simplicity, we assume the overall energy consumption is the interpolation of the corresponding energy consumption values when all of the n VUs offload both subtasks (i.e. $\mathcal{E'}_1(n, c_e)$) and when all of the n VUs offload just the neural network (i.e. $\mathcal{E'}_2(n, c_e)$). The above empirical models are specifically for SSD-MobileNetV2-based object detection applications. For other types of applications, once the action space of y_i is defined and the delay, accuracy, and energy consumption models are established correspondingly, our work can be applied to those vehicular applications.

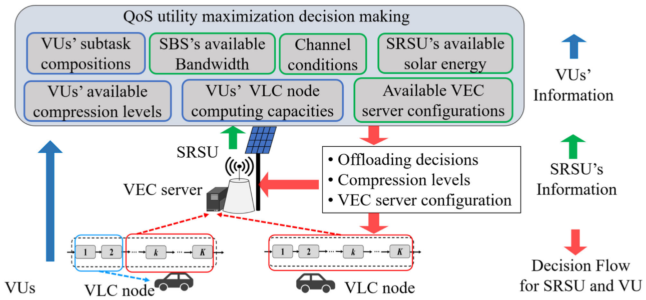


Fig. 7. Overview of the SRSU-assisted VEC system, including offloading request and decision flows.

B. Overall Approach and Problem Formulation

We assume that at each time slot, each VU will send an offloading request for this application. The request will include information of the local computing capacity $c_{l,i}$, available compression levels $\mathcal{Q}$, and the subtask composition of $\mathcal{K}$. The SRSU will take the above information, along with the available solar energy E_t, bandwidth W, and VEC server configurations $\mathcal{C}$, and make optimal offloading decision y_i as well as compression level q_i for each VU i. We assume the SRSU already knows the delay and accuracy models like Table IV and Fig. 4. The decisions will be sent to each VU by the SBS as the offloading instruction. In the meantime, SRSU will need to decide the VEC server's CPU-GPU configuration c_e for operation.

Fig. 7 depicts the whole process. In Fig. 7, blue arrow shows the flow of offloading requests from VUs to the SRSU, with the included information listed in blue boxes; green arrow shows the flow of SRSU's information, which is listed in green boxes including channel conditions, bandwidth, solar energy, and VEC server's computing availabilities; red arrows indicate the flow of decisions for the SRSU and VUs. The objective of this paper is to determine in real-time the VEC server's operating configuration c_e and the optimal offloading strategy y_i as well as the compression level q_i for each VU i to maximize the average QoS utility of all the VUs in $\mathcal{I}$ at any given time slot. The decision is made at the beginning of the time slot. The optimization problem for each time slot can, therefore, be formulated as,

$$\underset{y_i, q_i: i \in \mathcal{I}, c_e}{\text{maximize}} \quad Q\hat{o}S \tag{12a}$$
$$\text{subject to} \quad E_R \leq E_t \tag{12b}$$
$$W_i = W_j, \forall i, j : y_i > 0, y_j > 0 \tag{12c}$$
$$\sum_{i:y_i>0} W_i \leq W \tag{12d}$$
$$\sum_{i:y_i>0} 1 \leq N_{VEC} \tag{12e}$$

where Constraint 12b is the energy consumption constraint of SRSU. Constraint 12c states that the utilized bandwidth is evenly

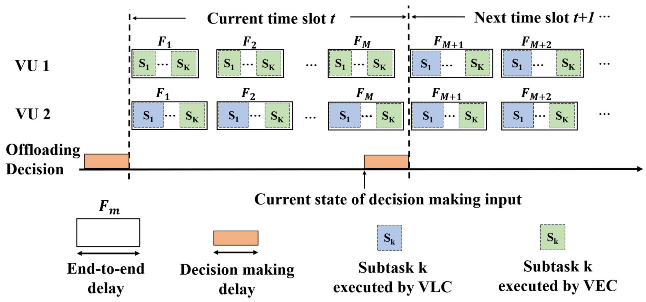


Fig. 8. Overview of the time domain flow of both application execution frames and task partitioning as well as offloading decisions in the SRSU-assisted VEC system.

distributed among the VUs who are offloading. Constraint 12d ensures that the overall utilized bandwidth does not exceed the available bandwidth of the system W. Constraint 12e shows that the total number of offloading VUs can not exceed the maximum capacity of the VEC server, which is also equivalent to the maximum number of application instance N_{VEC} that the VEC server can run simultaneously due to the computing capacity and computer memory limitations. For example, $N_{VEC} = 6$ for the Jetson TX2 board running SSD-MobileNetV2-based object detection applications.

Fig. 8 shows the time domain flow of the SRSU-assisted VEC system. For each VU, F_m represents the execution of the m^{th} frame of the application to process the m^{th} input camera image in the current time slot t, and the width of the F_m box shows its end-to-end delay. The small colored blocks in each F_m box are the subtasks for the application, where blue and green blocks indicate that the corresponding subtask is executed locally at the VLC or offloaded to the VEC, respectively. The orange boxes represent the execution of the offloading decision making. The offloading decision making for the next time slot takes the current states of VUs' request information and SRSU's resource capacities, then returns the optimal offloading decisions as well as compression levels at the beginning of the next time slot.

Since the partitioning and offloading decisions are made at the beginning of a time slot, the duration of the current time slot, τ, should be determined by the following equation,

$$\tau \geq \max\{\max_{i\in\mathcal{I}} d_i, T_{decision}\}, \tag{13}$$

so that no computation task will occupy any computing and communication resources when the next time slot begins. Note that for VU i, d_i is upper bounded by $d_i(0, q_{\max}, 0, 0, 0, c_{l,i}) = \sum_{k=1}^{K} T_k^l(c_{l,i})$, which is the local execution delay ($y_i = 0$) in (10), where $c_{l,i}$ is the VLC configuration and k is the index of each subtask. $T_{decision}$ is the delay of making optimal partitioning and offloading decisions, which depends on the VEC computing capacity and complexity of the decision making. We will show with experimental result in latter section that τ is mostly bounded by the $T_{decision}$ of our proposed decision algorithm with reasonable size of VLC and VEC computing capacities, and can be small enough to ensure an unchanged data rate, $r_{b,i}$.

VI. Solution Methodology

Note that in our problem formulation 12, the decision variables $y_i, q_i, \forall i \in \mathcal{I}$ and c_e are integers while $Q\hat{o}S$ is a nonlinear function of the above variables. Therefore, problem 12 is an NP-hard nonlinear integer programming problem [40]. The complexity of exhaustively listing all the possible values of y_i, q_i and c_e, and tracking for the solution which gives the maximum objective value is $O(Y^I Q^I)$. Where Y is the number of possible choices for partitioning and offloading (including *Local Only* strategy), Q is the number of compression levels for application level adaption of each VU, and I is the total number of VUs. If there are only a few VUs in the area, exhaustively search can provide optimal solution with a low time-complexity. However, the complexity of this problem grows exponentially with the number of VUs. Moreover, since we are considering vehicular users, the number of VUs changes over time, and it is very likely that there are tens of vehicles in the coverage area of an SRSU (e.g. during the peak hour of a highway). This leads to prohibitively expensive time-complexity for the exhaustive search approach. Therefore, in this work, we propose a dynamic programming-based heuristic algorithm to solve problem 12.

For a given instance of problem 12, and assuming $\sum_{i:y_i>0} 1 = N'$, $c_e = c'$, where both N' and c' are fixed integers, we consider a matrix f with dimension $I * N' * N' * N' * N'$. $f(i, n, h, u, v)$ represents the maximum average QoS utility achievable considering VU set $\{1, 2, \ldots, i\}$ and allowing n VUs offloading. On the other hand, h, u, and v are the numbers of the offloading VUs using *Full Offloading*, *Partial Offloading*, and *Encoded Partial Offloading* strategies, respectively. The core formula of this dynamic programming strategy is in (14),

$$f(i,n,h,u,v) = \begin{cases} 0; & \text{if } n \neq h+u+v \\ 0; & \text{if } i = 0 \\ 0; & \text{if } i, h, u, \text{ or } v < 0 \\ \max(A_y, \quad \forall y \in \mathcal{Y}); & \text{otherwise} \end{cases} \tag{14}$$

where $\mathcal{Y}$ is the set of all the possible values of y_i, and A_y is defined in the following. For $y = 0$,

$$A_0 = P_i^l + f(i-1, n, h, u, v) \tag{15}$$

is the QoS utility when including VU i in the considered VU set while VU i is not offloading any subtask, where P_i^l is the QoS utility using VLC node of VU i. For $y > 0$,

$$A_y = \begin{cases} \max_q \ P_{i,y,q} + f(i-1, n-1, h-\mathbb{1}_{y=1}, u-\mathbb{1}_{y=2}, v-\mathbb{1}_{y=3}); \\ \text{if} \quad E_R(w'(i-1, n-1, h-\mathbb{1}_{y=1}, u-\mathbb{1}_{y=2}, v-\mathbb{1}_{y=3}) \\ +\,\omega_{y,q,i}, n, h, c') < E_t \quad 0; \text{otherwise} \end{cases} \tag{16}$$

is the QoS utility while including VU i using $y_i = y$ ($y = 1, 2,$ or 3 in the considered use case) with compression level q_i^*,

Algorithm 1: Dynamic programming Algorithm for Fixed Offloading VU and System configuration (DAFOS).

Input: $N', c', \mathcal{I}, p_i^l, r_{b,i}, c_{l,i}, \forall i \in \mathcal{I}$
Output: $f', \hat{\mathbf{y}}, \hat{\mathbf{q}}$

1 $f, \psi, \pi \leftarrow zeros(I, N', N', N', N')$;
2 **for** $i \leftarrow 1$ **to** I **do**
3   **for** $n \leftarrow 0$ **to** N' **do**
4   **for** $h \leftarrow 0$ **to** N' **do**
5   calculate $P_{i,y_i,q_i} = \alpha * d_n / d_i(y_i, q_i, n, h, c_e, c_{l,i}) + (1-\alpha) * a(q_i) \quad \forall y_i \in \mathcal{Y}, q_i \in \mathcal{Q}$;
6   **for** $u \leftarrow 0$ **to** N' **do**
7   **for** $v \leftarrow 0$ **to** N' **do**
8   update $f(i, n, h, u, v)$ by Eq. 14 ;
9   update $\psi(i, n, h, u, v)$, $\pi(i, n, h, u, v)$, and $w'(i, n, h, u, v)$;
10 $f' \leftarrow \max_{h,u,v} f(I, N', :, :, :)$;
11 $h^*, u^*, v^* \leftarrow \text{argmax}_{h,u,v} \quad f(I, N', :, :, :)$;
12 $\hat{N} \leftarrow N'$;
13 $\hat{\mathbf{y}}, \hat{\mathbf{q}} \leftarrow zeros(I)$;
14 **for** $i \leftarrow I$ **to** 1 **do**
15   $\hat{\mathbf{y}}[i] \leftarrow \psi(i, N', h^*, u^*, v^*)$;
16   $\hat{\mathbf{q}}[i] \leftarrow \pi(i, N', h^*, u^*, v^*)$;
17   **if** $\hat{\mathbf{y}}[i] > 0$ **then**
18     $\hat{N} \leftarrow \hat{N} - 1$;
19   **if** $\hat{\mathbf{y}}[i] == 1$ **then**
20       $u^* \leftarrow h^* - 1$;
21   **else if** $\hat{\mathbf{y}}[i] == 2$ **then**
22       $v^* \leftarrow u^* - 1$;
23   **else**
24       $z^* \leftarrow v^* - 1$;
25 return $f', \hat{\mathbf{y}}, \hat{\mathbf{q}}$;

which gives the maximum QoS utility $P_{i,y,q}$ among all the possible q. Note that $P_{i,y,q} = \alpha * d_n / d_i(y, q, n, h, c_e, c_{l,i}) + (1-\alpha) * a(q)$. $\mathbb{1}_{y=j}$ is an indicator function whose value is 1 if $y =$ j, otherwise, its value is 0. The $E_R < E_t$ inequality is used to ensure the energy constraint, where $w'(i-1, n-1, h - \mathbb{1}_{y=1}, u - \mathbb{1}_{y=2}, v - \mathbb{1}_{y=3})$ is the total data size required to be transmitted for VU 1 to $i-1$. $\omega_{y,q,i}$ is the data size required to be transmitted by VU i when using $y_i = y$ and $q_i = q$.

On the other hand, we use $\psi(i, n, h, u, v)$ and $\pi(i, n, h, u, v)$ to record the optimal offloading decision y_i^* and compression level q_i^* that correspond to the value of $f(i, n, h, u, v)$ for VU i. Matrices f, ψ, π are initialized as zero matrices. We then recursively calculate the elements in f for v from 0 to N', u from 0 to N', h from 0 to N', n from 0 to N', i from 0 to I, until all the elements in f are updated. The optimal cumulative QoS utility for VU set $\mathcal{I}$ considering $\sum_{i:y_i>0} 1 = N'$ and $c_e = c'$ is then the maximum elements among $f(I, N,' :, :, :)$. We then calculate the optimal offloading and compression level decisions for each VU iteratively from $i = I$ to $i = 1$ by using ψ and π. We list the steps for updating elements in f in Algorithm 1, which we name as Dynamic programming Algorithm for Fixed Offloading VU and System configuration (DAFOS). Steps 2 to 9 execute the core function of dynamic programming and steps 14 to 24 retrieve the recorded optimal offloading and compression level decisions in ψ and π.

Note that DAFOS returns the heuristic solution of problem 12 under the condition that $\sum_{i:y_i>0} 1 = N'$ and $c_e = c'$. To obtain the solution of problem 12, all the possible values of N' and c' need to be considered. Therefore, we propose the following System and Application aware Multiple User Offloading Algorithm (SAMOA), which executes DAFOS on different N' and c' and returns the maximum QoS utility, the corresponding offloading strategies as well as compression levels for each VU. The steps of SAMOA are listed in Algorithm 2. In SAMOA, steps 1 to 3 calculate the QoS utility of *Local Only* for each VU. We start to execute DAFOS on different N' and c' and pick the maximum possible optimal solution between steps 6 to 9. We use $\hat{\mathbf{P}}[N,' c']$ to record the returned optimal QoS utility. We then append the corresponding offloading strategies and compression levels $\{\hat{\mathbf{y}}, \hat{\mathbf{q}}\}$ to $\hat{\mathbf{L}}[N,' c']$. After all the possible sets of N' and c' are iterated, in steps 10 to 12, SAMOA will return the maximum elements in $\hat{\mathbf{P}}$ as the optimal QoS utility and its corresponding offloading strategies and compression levels of each VU.

Algorithm 2: System and Application aware Multiple User Offloading Algorithm (SAMOA).

Input: $\mathcal{I}, \mathcal{C}, N_{VEC}, \mathcal{Y}, \mathcal{Q}, r_{ib}, c_{l,i}, \forall i \in \mathcal{I}$
Output: L^*, p^*

1 $p^l \leftarrow zeros(N)$;
2 **for** i *in* I **do**
3   $p_i^l \leftarrow \frac{\alpha * d_n}{d_i(i, 0, q_{max}, 0, 0, c_{l,i})} + (1-\alpha) * a(q_{max})$;
4 $\hat{\mathbf{P}} \leftarrow zeros(N_{VEC}, \mathcal{C})$;
5 $\hat{\mathbf{L}} \leftarrow list()$;
6 **for** $N' \leftarrow 1$ **to** N_{VEC} **do**
7   **for** c' *in* $\mathcal{C}$ **do**
8     $\hat{\mathbf{P}}[N', c'], \hat{\mathbf{y}}, \hat{\mathbf{q}} \leftarrow DAFOS(N', c', \mathcal{I}, p_i^l, r_{ib}, c_{l,i}, \forall i \in \mathcal{I})$;
9   append $\{\hat{\mathbf{y}}, \hat{\mathbf{q}}\}$ in $\hat{\mathbf{L}}$
10 $N^*, c_e^* \leftarrow \underset{N', c'}{\text{argmax}}\ \hat{\mathbf{P}}[N', c']$;
11 $p^* \leftarrow \hat{\mathbf{P}}[N^*, c_e^*]$;
12 $L^* \leftarrow \hat{\mathbf{L}}[N^*, c_e^*]$;

Note that for DAFOS, the variables n, h, u, v need to iterate N_{VEC} times and (14) requires iteration of all the Q compression levels and Y offloading decisions in the worst case. Therefore, the complexity of DAFOS is $O(I * N_{VEC}^4 * Q * Y)$. On the other hand, in SAMOA, DAFOS is the component that has the largest complexity and DAFOS is executed $N_{VEC} * C$ times, where C is the number of possible VEC server configurations. Therefore, the complexity of SAMOA is $O(I * N_{VEC}^5 * Q * C * Y)$. Since N_{VEC}, Q, and C are constant, the time complexity of SAMOA is $O(I)$, where I is total number of VUs. Hence, as validated with our experimental results reported in the next section, SAMOA can be executed in real-time for reasonable size of VU set $\mathcal{I}$.

VII. Performance Evaluation

We first show how SAMOA performs under different resource conditions. Then, we present the online trace-driven simulation framework and demonstrate the performance comparison of SAMOA with existing approaches. Finally, we show how SAMOA can be applied to more dense VU scenarios.

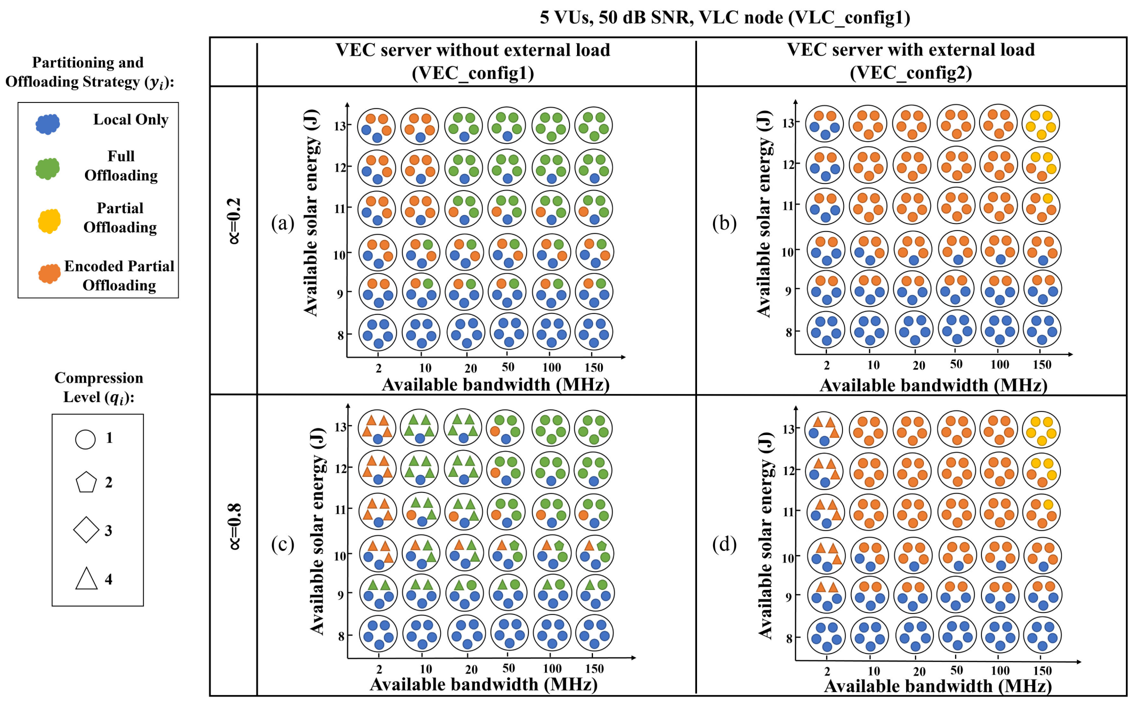


Fig. 9. Partitioning and Offloading strategy with compression levels for individual VU under different resource availability.

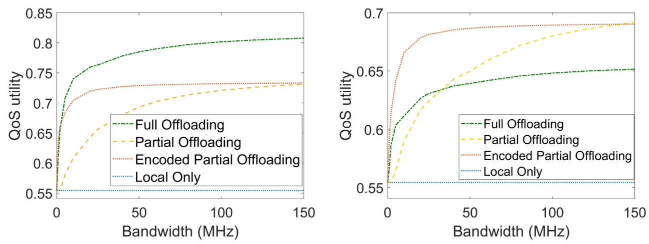
A. SAMOA Performance Evaluation

Herein, we present and analyze how our algorithm decides the optimal decisions in a single time slot. In Fig. 9(a), (b), (c), and (d), we show the evolution of offloading strategies y_i as well as compression level q_i of each VU determined by SAMOA in different resource availability and system parameter regions. The time slot duration τ is set to 1 s. For simplicity, we assume there are 5 identical VUs in $\mathcal{I}$, all of them have an VLC node with VLC_config2 configuration (i.e. listed in Table V) and SNR value 50 dB. We consider scenarios with different bandwidth, energy, and VEC computation capacities as well as different trade-off factors α. We mimic the impact of computing load caused by other applications sharing the VEC server by reducing the CPU-GPU resources of the Jetson TX2 board. *VEC server without external load* (i.e. using VEC_config1 configuration) represents the condition when the VEC server is not busy computing other applications and *VEC server with external load* (i.e. using VEC_config2 configuration) means the VEC server is simultaneously executing other applications. On the other hand, vertically we vary the trade-off factor value α from 0.2 to 0.8. The color and shape of each circle show the y_i and q_i decisions, respectively, to each VU. Blue, yellow, red, and green colors represent *Local Only*, *Partial Offloading*, *Encoded Partial Offloading*, and *Full Offloading* strategies, respectively. Circle, pentagon, diamond, and triangle shapes, respectively, show the compression levels 1 to 4 (i.e. listed in Table VI).

Fig. 9(a) considers the scenario where $\alpha = 0.2$ and *VEC without external load*, where y_i changes from *Local Only* to *Encoded Partial Offloading*, then to *Full Offloading* strategy when the available solar energy and bandwidth increases. It is because compared to *Full Offloading* strategy, (1) *Encoded Partial Offloading* strategy needs less bandwidth as it transmits the encoded image after resizing, (2) *Encoded Partial Offloading* strategy executes the DR subtask in VLC node and transmits lesser number of bits, hence requires less energy consumption in VEC server. Therefore, in the regions where SRSU lacks of either bandwidth or solar energy, *Encoded Partial Offloading* outperforms *Full Offloading* strategy in terms of QoS utility.

On the other hand, with $\alpha = 0.2$ and when VEC server *has external load* (i.e. the Fig. 9(b)), the *Encoded Partial Offloading* strategy dominates when the available bandwidth is below 100 MHz and solar energy is below 13 J. After the bandwidth reaches 150 MHz, we can observe some VUs use *Partial Offloading* strategy. This is because *Partial Offloading* strategy transmits the resized image without encoding, while the reduction in computing delay dominates the growth of transmission delay only when the transmission rate is very high. Also, there is no *Full Offloading* strategy observed because the computing capacity at VEC server is low because of load. Thus offloading with VLC node executing DR subtask first can achieve higher average QoS utility.

In Fig. 9(c), we can observe the optimal decision involves different compression levels because of higher α which indicates more importance of delay sacrificing some accuracy. We can observe that VUs offload at the highest compression level (i.e. lowest image quality, the triangle shape) when both the bandwidth and solar energy are in lower availability. We also observe some VUs transmit at the compression level 2 (e.g. the pentagon shape at 10 J, 50 MHz) when the bandwidth is higher than 50 MHz and available solar energy is in medium region (i.e. 10 J). In Fig. 9(d), we can also observe that VUs offload at the highest compression level when the bandwidth availability is low. After the available bandwidth exceeds 10 MHz, VUs



(a) (b)

Fig. 10. Impact of different offloading decisions on QoS utility under different bandwidth resource availability when available solar energy is 13 J for 1 s time slot and VEC server is (a) left, without external load and (b) with external load.

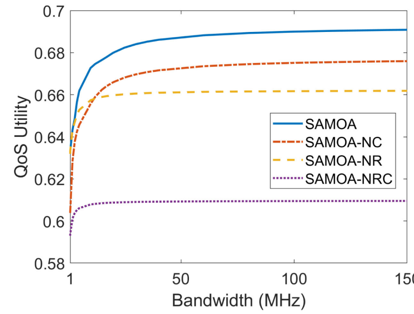


Fig. 11. QoS utility comparison of SAMOA, SAMOA-NC, SAMOA-NR, SAMOA-NRC under varying bandwidth availabilities at solar energy 10 J and VEC without external load.

offload at the lowest compression level because the resulting transmission delay will be small enough such that high accuracy can be achieved.

For the sake of consistency in the granularity of dimensions of the labels in Fig. 9, we did not show the condition when our algorithm chooses compression level 3. Actually, in Fig. 9(c), compression level 3 will be chosen at 10 J, 25 MHz with *Full Offloading* strategy for 3 VUs while the rest 2 VUs use *Local Only* strategy with compression level 1. Overall, Fig. 9 demonstrates how VEC server's computing capacity and different choices of α impact the optimal partitioning and offloading strategy for each VU under different resource availabilities. When the algorithm emphasizes more on optimizing the end-to-end delay, we observe some higher compression levels used by VUs for the trade-off between accuracy and end-to-end delay.

While Fig. 9 shows the optimal offloading decisions by SAMOA, Fig. 10 shows the resulting average QoS utility that drives the decision for the above 5 VUs. We show two scenarios (VEC with and without load) under various bandwidth conditions with 13 J of available solar energy and $\alpha = 0.8$, and show the QoS utilities for various offloading decisions. Note that, a VU can always use *Local Only* strategy, if other available strategies are not feasible in a parameter region. Fig. 10(a) shows the results for VEC server *without external load*. The red curve (i.e. *Encoded Partial Offloading*) tops the blue curve (*Full Offloading*) when bandwidth availability is low (<5 MHz), while the green curve (i.e. *Full Offloading* strategy) dominates the others afterward. The observation matches the results in Fig. 9(c), where the *Full Offloading* strategy dominates at high solar energy and bandwidth regions. On the other hand, the result for VEC server *with external load* is shown in Fig. 10(b). We can observe the red curve dominates other curves until bandwidth reaches 150 MHz, where the yellow curve (i.e. *Partial Offloading* strategy) tops the red one. Also, the green curve is always lower than either red or yellow curves. The observation conforms with the results in Fig. 9(d).

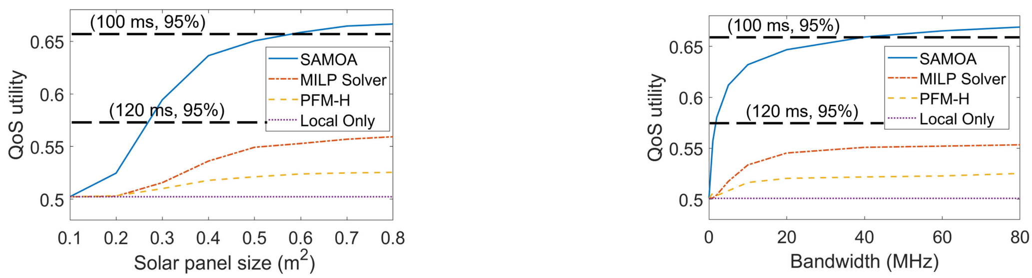
Impact of system and application level adaption: Next we present results to show the benefit of using system level as well as application level (i.e. compression levels) adaptions. Fig. 11 shows the results for the scenario when the solar energy is 10 J for a time slot with $\tau = 1\ s$ and VEC server does not have external load. In this figure, SAMOA-NC denotes the SAMOA algorithm with no additional compression (i.e. lowest fixed compression level 1), SAMOA-NR denotes the SAMOA algorithm with no reconfiguration (i.e., fixed VEC server configuration with highest possible CPU-GPU frequencies in VEC_config1), and SAMOA-NRC denotes the SAMOA algorithm with no additional compression and reconfiguration, i.e., fixed compression level as SAMOA-NC and the fixed VEC server configuration as SAMOA-NR.

Including the compression and reconfiguration, the gain in the performance of SAMOA is apparent. When bandwidth is above 60 MHz, the average QoS utility of SAMOA is 2%, 4%, and 13% higher than SAMOA-NR, SAMOA-NC, and SAMOA-NRC. The difference between SAMOA and SAMOA-NRC is $> 10\%$, shows the importance of joint system and application level adaptation to improve the QoS utility performance.

B. Real-World Trace Driven Simulation

Next, we present the online performance of SAMOA using a simulator we have developed [41], which allows creation of realistic trace driven movements, topology, location, and channel condition for each VU at every time slot. The tool simulates the vehicle's trace in a 1000x800 m^2 rectangular neighborhood in Brooklyn, New York City based on historical vehicular traffic data obtained from [42]. With the street topology and traces of vehicles, the tool generates the SNR values from each VU to the 20 SRSUs located in the area. The SNR is generated by assuming VU's transmit power, ρ_i, is 23 dbm and using B1 Manhattan grid layout [27] as the pathloss and slow fading, and the Nakagami-m distribution [28] as the fast fading for the uplink channel model.

At each time slot, we assume each VU is associated to the SRSU which corresponds to the highest signal strength. For the following experiment, we pick one of the SRSUs in this area to demonstrate the simulation result. We assume each VU will have 50% of probability to have a VLC node with capacity VLC_config1 and 50% of probability to have a VLC node with capacity VLC_config2, which are listed in Table V. On the other hand, at each time slot, we assume VEC server will have 50% of probability to be *without external load* (VEC_config1) and 50% to be *with external load* (VEC_config2), with the specific available CPU-GPU configurations as specified in Fig. 9.



(a) (b)

Fig. 12. QoS utility of 4 algorithms under various scenarios of (a). left, solar panel size (b) right, bandwidth availability.

1) Compared Algorithms: We compare the performance of SAMOA with two other relevant algorithms, MILP Solver [19] and PFH-M [21], which are the two closest approaches to SAMOA as they both allow task dependency aware partitioning and offloading with limited VEC communication and computing resources.

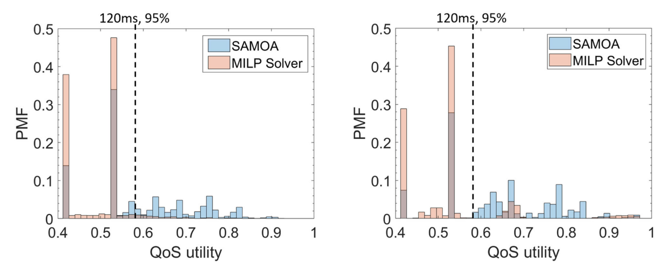
MILP Solver: In [19], the authors model the partitioning and offloading problem as a Mixed Integer Linear Programming (MILP) problem. They then propose to use existing standard MILP software packages to find the optimal solution and minimize the end-to-end delay of an DNN application.

PFM-H Algorithm: In [21], the authors address the challenges of maximizing the throughput under limited edge computing and communication resources using optimal task partitioning and bandwidth allocation decisions. They formulate the problem to a variant of Knapsack Problem and propose to find the heuristic solution by using Performance Function Matrix based Heuristic (PFM-H) algorithm.

Although the above two approaches consider the constrained communication and computing capacities in VEC server, they do not consider energy constraint. Therefore, we impose an energy constraint check point after these approaches return their offloading and partitioning solution. If the resulting energy consumption violates the constraint, we ask all the VUs to execute their tasks locally. We also present the performance of the naive strategy, *Local Only*, which only allows VUs to execute their tasks locally using VLC nodes.

2) Trace Driven Online Simulation Result: In this experiment, we run the simulation for 1 h, starting from 9 AM. The duration of each time slot is 1 s, namely, $\tau = 1$ s. Fig. 12(a) and (b) demonstrate the average QoS utility over the total simulation time for all the 4 algorithms under different solar panel sizes and bandwidth, respectively. The average QoS utility of the total simulation time is defined as the average of the QoS Utility of every VU instance in every time slot during the total simulation time. In the simulated neighborhood area of Brooklyn, because vehicles are dense and vehicle speed is high, the end-to-end delay is very critical to driving experience. Therefore, we set $\alpha = 0.8$, which make SAMOA emphasizes more on the end-to-end delay.

Impact of solar panel size: In Fig. 12(a), the x-axis shows the different solar panel sizes vary from 0.1 to 0.8 m^2. The bandwidth of the SRSU is 20 MHz and equally distributed among the offloading VUs. When the solar panel size is 0.5 m^2, it is shown that the average QoS utility of SAMOA is the best



(a) (b)

Fig. 13. PMF of the QoS utility for each individual VU using SAMOA and MILP solver, with 20 MHz bandwidth and solar size equals (a). left, 0.3 m^2 (b) right, 0.8 m^2.

among all the algorithms and is 18.4%, 24.8%, and 29.5% better than MILP Solver, PFM-H, and *Local Only*, respectively. On the other hand, the dash lines in Fig. 12(a) shows the end-to-end delay and accuracy values corresponding to the specific average QoS utility values. Note that except SAMOA, none of the above algorithms can achieve the 120 ms end-to-end delay and 95% accuracy simultaneously. SAMOA achieves the average QoS utility of 120 ms end-to-end delay and 95% accuracy when solar panel size is around 0.3 m^2. Moreover, when the solar panel size is higher than 0.55 m^2, SAMOA delivers an average QoS utility of 100 ms end-to-end delay and 95% accuracy.

Impact of bandwidth availability: In Fig. 12(b), the x-axis shows the different available bandwidth varies from 0 to 80 MHz and the solar panel size of SRSU is 0.5 m^2. When the bandwidth is 40 MHz, the average QoS utility of SAMOA is the best among all the algorithms and is 16.6%, 26.3%, and 31.6% better than MILP Solver, PFM-H, and *Local Only*, respectively. On the other hand, except SAMOA, none of the above algorithms can achieve the 120 ms end-to-end delay and 95% accuracy simultaneously. SAMOA achieves an average QoS utility of 120 ms end-to-end delay and 95% accuracy when the available bandwidth is around 2.5 MHz. Moreover, when available bandwidth is higher than 35 MHz, SAMOA achieves an average QoS utility of 100 ms end-to-end delay and 95% accuracy.

Empirical probability mass function (PMF) of the QoS: In Fig. 13, we show the empirical probability mass function (PMF) of the individual QoS utility for the VUs. To clearly demonstrate the gap between SAMOA and others, we compare SAMOA with the second best algorithm, MILP Solver, in Fig. 13. In Fig. 13(a), solar panel size is 0.3 m^2 and bandwidth is 20 MHz. Even though the energy availability is low, 45% of the VU instances can still

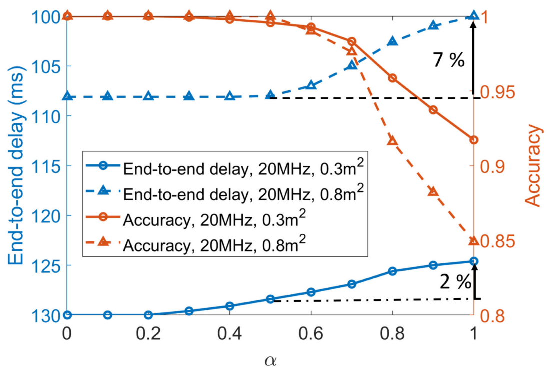


Fig. 14. Impact of different α values on the end-to-end delay and accuracy performance.

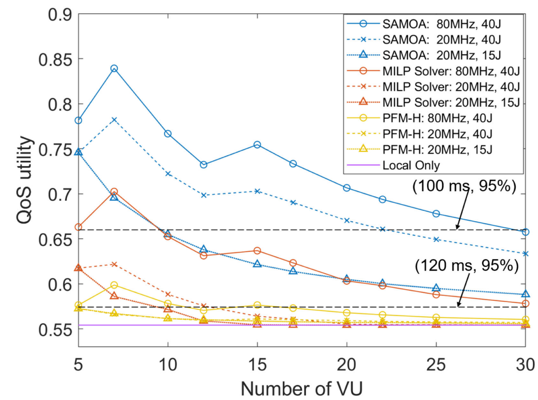


Fig. 15. QoS utility performance of the 4 algorithms with distributive computing capacity expansion under different number of VUs.

achieve the QoS utility corresponds to 120 ms end-to-end delay and 95% accuracy by using SAMOA algorithm while only 6% of VUs achieves the same QoS utility by using MILP Solver. When the solar panel size increases to 0.8 m^2, in Fig. 13(b), the VU instances that achieve the same QoS utility increases to 65% by using SAMOA algorithm, which is 3 times larger than using MILP Solver.

Impact of α values in delay and accuracy performance: In Fig. 14, we show how the average end-to-end delay and accuracy will change when α value changes from 0 to 1. For consistency with Fig. 13, we also choose 20 MHz bandwidth with 0.3 m^2 and 0.8 m^2 solar panel size, respectively, for comparison. When α increases, the accuracy is reduced in exchange for the decreased end-to-end delay. On the other hand, while the accuracy starts to decrease at $\alpha = 0.5$ when solar panel size is 0.8 m^2, it starts decreasing earlier at $\alpha = 0.2$ when solar panel size is 0.3 m^2. Although larger solar panel size leads to a lesser tunable range for α values, the delay improvement is better. For example, the end-to-end delay reduces by 7% when α increases from 0.5 to 1 for 0.8 m^2 solar panel size while the delay reduces just by 2% for 0.3 m^2 solar panel size within the same range of α. With results like Fig. 14, the SRSU operators or service providers can jointly decide the optimal solar panel size and α value during the SRSU deployment based on the desired delay and accuracy performance.

C. Scalability of SRSU

Note that, in the empirical model and the experiment setup, the maximum available number of offloading VUs N_{VEC} is 6. Herein, we demonstrate the performance of SAMOA when both the capacity of SRSU and the number of served VUs are scaled up. We emulate the scaled up computing capacity of SRSU by adding additional Jetson TX2 boards to the SRSU. The bandwidth and energy availabilities are scaled up in terms of Hz and Joule, respectively. At each time slot, which has duration 1 s, for a given instance which has more than 6 VUs, we execute the following VU distribution algorithm before executing SAMOA. VU distribution algorithm will first calculate the required number of Jetson boards x,

$$x = min\left(\left\lceil \frac{I}{N_{VEC}} \right\rceil, \left\lfloor \frac{E_t}{10} \right\rfloor\right) \tag{17}$$

where I is the total number of VU, N_{VEC} is 6 in our scenario, and E_t is the current available energy. We use $\lfloor \frac{E_t}{10} \rfloor$ to ensure each active board has at least 10 J of energy for operation within the time slot. Then the algorithm will sort VUs by their SNR values and evenly distribute them by the sorted order into x groups. Finally, the distribution algorithm will assign each group to one Jetson board and execute SAMOA respectively for VUs in that group. For performance comparison, we use the same VU distribution algorithm for MILP Solver and PFM-H. Fig. 15 shows the numerical result of these three algorithms using the distribution algorithm under different values of I. We consider all the VUs are using VLC node configuration VLC_config2 and VEC servers (i.e. Jetson boards) *without external load*. The average QoS utility is calculated after 10 rounds of simulations, in which we randomly and uniformly generate the SNR values between 10 to 50 dB for each VU in set $\mathcal{I}$.

Fig. 15 shows that with the same available bandwidth and energy, SAMOA performs the best compared to the other two algorithms and *Local Only* even when the number of VUs is high. For example, when there are 20 VUs, SAMOA performs 17.1%, 24.4%, and 27.5% better than MILP Solver, PFM-H, and *Local Only* approaches, respectively, with 80 MHz bandwidth and 40 J solar energy. In low resource availability when there are 20 MHz bandwidth and 15 J solar energy, SAMOA's capability will be constrained, but still performs 9.1%, 8.6%, and 9.1% better than MILP Solver, PFM-H, and Local Only approaches, respectively, for 20 VUs.

Fig. 15 also shows that when the number of VUs exceeds 10, only SAMOA can achieve the average QoS utility of 100 ms end-to-end delay and 95% accuracy even when the available bandwidth and energy resources are high (i.e. 80 MHz and 40 J). SAMOA achieves the average QoS utility of 120 ms end-to-end delay and 95% accuracy at lower resource availability (20 MHz and 15 J) for up to 30 VUs. However, MILP Solver requires higher available resources to achieve the same average QoS utility for up to 30 VUs and PFM-H can only achieve the same performance for up to 17 VUs.

The results in Fig. 15 clearly demonstrate the advantage of SAMOA over other approaches. Moreover, the above trade-off analysis between QoS utility and different resource availability

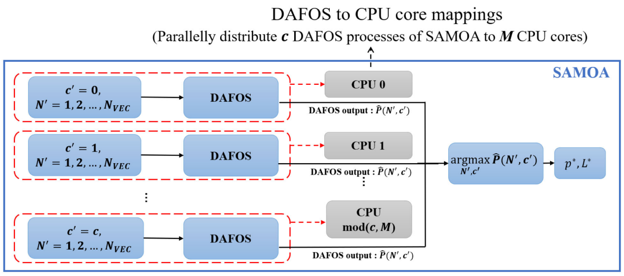


Fig. 16. Multi-core parallel processing of SAMOA using Nvidia Jetson TX2.

will enable the the service providers to identify the best SRSU configurations given expected solar generation and VU density.

Run-time analysis: To measure the run-time complexity of SAMOA, we implement SAMOA and the VU distribution algorithm using Python on the Nvidia Jetson TX2 board. Since the Nvidia Jetson TX2 platform allows parallel processing on multiple cores, we have developed an efficient implementation of SAMOA with parallel multi-core processing, as shown in Fig. 16, where we parallelly distribute and execute all the c DAFOS processes of a SAMOA algorithm to the M available CPU cores on the edge computing platform. Note that c is decided by the number of available CPU-GPU configurations. In our experimental setup, $c = 6$ and $M = 6$.

The resulting average execution time of SAMOA, $T_{decision}$ is 250 ms, allowing SAMOA-based task partitioning decision to be made as frequently as every 250 ms. Note that the end-to-end delays for local execution of the vehicular object detection application are 131 ms and 188 ms, respectively, using VLC_config1 and VLC_config2. Hence with reasonable size of VEC and VLC configurations, this experimental result shows that the duration of a time slot τ can be defined as small as 250 ms, which makes the assumption of the constant data rate within a time slot more realistic while ensuring the completion of all the vehicular computation tasks. Note that the application's end-to-end delay is independent of the execution time of SAMOA. SAMOA is executed before a time slot starts, and the resulting decision is used by each VU to partition and offload the application tasks for multiple subsequent executions of the application during the decision time slot, till the next execution of SAMOA and resulting change in partitioning decision.

VIII. Conclusion

In this paper, we propose a real-time system and application adaptive task partitioning and offloading algorithm, SAMOA, to support the computation intensive applications of the vehicles using solar-powered RSU. The algorithm jointly minimizes the end-to-end delay and maximizes the object detection accuracy, which we jointly define as QoS utility, based on the communication bandwidth, computing, and energy resources availabilities at the SRSU, as well as the computing capacity at the VLC nodes. We establish empirical models for the computation and communication capacities as well as energy consumption of SRSU. With the empirical model-based simulation, we show that SAMOA significantly maximizes the average QoS utility compared to existing techniques under various resource availability and VU density. As dense deployment of RSUs takes place in our cities and neighborhoods in the next several years, our research results will help service providers and city planners to adopt solar energy based RSUs to avoid additional impact on carbon footprint. Moreover, they will be able to use SAMOA and the simulation and analysis tools we provide to identify adequate SRSU designs, with appropriate computing, communication and solar capacities, for the expected vehicular traffic load and desired delay-accuracy performance. In future work, we plan to investigate the addition of other RE sources (e.g., wind energy) and battery to ensure energy diversity and guarantee service availability in adverse weather conditions.

References

[1] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, Jan. 2019.

[2] K. Zhang, Y. Mao, S. Leng, Y. He, and Y. Zhang, "Mobile-edge computing for vehicular networks: A promising network paradigm with predictive off-loading," *IEEE Veh. Technol. Mag.*, vol. 12, no. 2, pp. 36–44, Jun. 2017.

[3] G.-S. Initiative and Deloitte, "Digital with purpose: Delivering a smarter 2030," Accessed: Nov. 16, 2020. [Online]. Available: https://gesi.org/research/gesi-digital-with-purpose-full-report.

[4] M. H. Alsharif, J. Kim, and J. H. Kim, "Green and sustainable cellular base stations: An overview and future research directions," *Energies*, vol. 10, no. 5, p. 587, Apr. 2017.

[5] Y. Ku and S. Dey, "Sustainable vehicular edge computing using local and solar-powered roadside unit resources," in *Proc. IEEE 90th Veh. Technol. Conf.*, 2019, pp. 1–7.

[6] S. Baidya, Y.-J. Ku, H. Zhao, J. Zhao, and S. Dey, "Vehicular and edge computing for emerging connected and autonomous vehicle applications," in *Proc. 57th ACM/IEEE Des. Automat. Conf.*, San Francisco, CA, USA: IEEE, 2020, pp. 1–6.

[7] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, 2018, pp. 4510–4520.

[8] Nvidia, "Nvidia jetson tx2 series system-on-moduledatasheet v1. 2, 2014," Accessed: Apr. 9, 2020. [Online]. Available: https://developer.nvidia.com/embedded/jetson-tx2.

[9] G. Qiao, S. Leng, K. Zhang, and Y. He, "Collaborative task offloading in vehicular edge multi-access networks," *IEEE Commun. Mag.*, vol. 56, no. 8, pp. 48–54, Aug. 2018.

[10] Y. Zhou *et al.*, "Distributed scheduling for time-critical tasks in a two-layer vehicular fog computing architecture," in *Proc. IEEE Annu. Consum. Commun. Netw. Conf.*, 2020, pp. 1–7.

[11] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2092–2104, Feb. 2020.

[12] B. Gu and Z. Zhou, "Task offloading in vehicular mobile edge computing: A matching-theoretic framework," *IEEE Veh. Technol. Mag.*, vol. 14, no. 3, pp. 100–106, Sep. 2019.

[13] M. Li, J. Gao, L. Zhao, and X. Shen, "Deep reinforcement learning for collaborative edge computing in vehicular networks," *IEEE Trans. Cogn. Commun. Netw.*, vol. 6, no. 4, pp. 1122–1135, Dec. 2020.

[14] C. Yang *et al.*, "Efficient mobility-aware task offloading for vehicular edge computing networks," *IEEE Access*, vol. 7, pp. 26 652–26 664, 2019.

[15] S. Mao, S. Leng, and Y. Zhang, "Joint communication and computation resource optimization for noma-assisted mobile edge computing," in *Proc. IEEE Int. Conf. Commun.*, Shanghai, China: IEEE, 2019, pp. 1–6.

[16] L. P. Qian, B. Shi, Y. Wu, B. Sun, and D. H. K. Tsang, "NOMA-enabled mobile edge computing for Internet of Things via joint communication and computation resource allocations," *IEEE Internet Things J.*, vol. 7, no. 1, pp. 718–733, Jan. 2020.

[17] J. Mao, X. Chen, K. W. Nixon, C. Krieger, and Y. Chen, "MoDNN: Local distributed mobile computing system for deep neural network," in *Design, Automation & Test in Europe Conference & Exhibition (DATE)*, Lausanne, Switzerland: IEEE, 2017, pp. 1396–1401.

[18] Z. Zhao, K. M. Barijough, and A. Gerstlauer, "DeepThings: Distributed adaptive deep learning inference on resource-constrained IOT edge clusters," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 37, no. 11, pp. 2348–2359, Nov. 2018.

[19] M. Gao, W. Cui, D. Gao, R. Shen, J. Li, and Y. Zhou, "Deep neural network task partitioning and offloading for mobile edge computing," in *Proc. IEEE Glob. Commun. Conf.*, Waikoloa, HI, USA: IEEE, 2019, pp. 1–6.

[20] S. Teerapittayanon, B. McDanel, and H.-T. Kung, "Distributed deep neural networks over the cloud, the edge and end devices," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, Atlanta, GA, USA, 2017, pp. 328–339.

[21] L. Yang, J. Cao, Z. Wang, and W. Wu, "Network aware multi-user computation partitioning in mobile edge clouds," in *Proc. 46th Int. Conf. Parallel Process.*, 2017, pp. 302–311.

[22] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12 313–12325, Dec. 2018.

[23] W. Chen, D. Wang, and K. Li, "Multi-user multi-task computation offloading in green mobile edge cloud computing," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 726–738, Sep./Oct. 2019.

[24] J. Xu and S. Ren, "Online learning for offloading and autoscaling in renewable-powered mobile edge computing," in *Proc. IEEE Glob. Commun. Conf.*, 2016, pp. 1–6.

[25] N. Saquib, E. Hossain, and D. I. Kim, "Fractional frequency reuse for interference management in LTE-advanced hetnets," *IEEE Wireless Commun.*, vol. 20, no. 2, pp. 113–122, Apr. 2013.

[26] S. Chen *et al.*, "Vehicle-to-everything (V2X) services supported by LTE-Based systems and 5G," *IEEE Commun. Standards Mag.*, vol. 1, no. 2, pp. 70–76, Jul. 2017.

[27] P. Kyosti, "WINNER II D1.1.2 V1.0 WINNER II Channel Models," *Eur. Commission*, Tech. Rep. IST-4-027756, Sep. 2007.

[28] L. Cheng, B. E. Henty, D. D. Stancil, F. Bai, and P. Mudalige, "Mobile vehicle-to-vehicle narrow-band channel measurement and characterization of the 5.9 GHz dedicated short range communication (DSRC) frequency band," *IEEE J. Sel. Areas Commun.*, vol. 25, no. 8, pp. 1501–1516, Oct. 2007.

[29] T. 3rd Generation Partnership Project, "Study on LTE-based V2X services," 3GPP-REF-36885, rel. 14, Accessed: Jan. 19, 2018. [Online]. Available: http://www.3gpp.org/.

[30] C. Huang, Y. P. Fallah, R. Sengupta, and H. Krishnan, "Intervehicle transmission rate control for cooperative active safety system," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 3, pp. 645–658, Sep. 2011.

[31] K. Bansal, K. Rungta, S. Zhu, and D. Bharadia, "Pointillism: Accurate 3D bounding box estimation with multi-radars," in *Proc. 18th Conf. Embedded Networked Sensor Syst., Ser. SenSys'20*, 2020, pp. 340–353.

[32] T. 3rd Generation Partnership Project, "Small cell enhancements for E-UTRA and E-UTRAN - physical layer aspects," 3GPP-REF-36872, rel. 12, Accessed: Jan. 7, 2018. [Online]. Available: http://www.3gpp.org/.

[33] D.-H. Kim, "Lane detection method with impulse radio ultra-wideband radar and metal lane reflectors," *Sensors*, vol. 20, no. 1, p. 324, Jan. 2020.

[34] M. M. William *et al.*, "Traffic signs detection and recognition system using deep learning," in *Proc. 9th Int. Conf. Intell. Comput. Inf. Syst.*, 2019, pp. 160–166.

[35] X.-Y. Ye *et al.*, "A two-stage real-time YOLOV2-based road marking detector with lightweight spatial transformation-invariant classification," *Image Vis. Comput.*, vol. 102, 2020, Art. no. 103978.

[36] J. Zheng, K. Okamoto, and P. Tsiotras, "Vision-based autonomous vehicle control using the two-point visual driver control model," 2019. [Online]. Available: https://arxiv.org/abs/1910.04862

[37] 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.*, 2018, pp. 751–766.

[38] Y.-J. Ku, S. Sapra, S. Baidya, and S. Dey, "State of energy prediction in renewable energy-driven mobile edge computing using CNN-LSTM networks," in *Proc. IEEE Green Energy Smart Syst. Conf.*, Long Beach, CA, USA: IEEE, 2020, pp. 1–6.

[39] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "srsLTE: An open-source platform for LTE evolution and experimentation," in *Proc. 10th ACM Int. Workshop Wireless Netw. Testbeds Exp. Eval. Characterization*, 2016, pp. 25–32.

[40] R. Hemmecke, M. Köppe, J. Lee, and R. Weismantel, "Nonlinear integer programming," in *50 Years of Integer Programming 1958–2008*. Berlin, Germany: Springer, 2010, pp. 561–618.

[41] Y. J. Ku, P. H. Chiang, and S. Dey, "Real-time QoS optimization for vehicular edge computing with off-grid roadside units," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 11 975–11991, Oct. 2020.

[42] Department of Transportation, *NYS Traffic Data Viewer*. New York, NY, USA: Department of Transportation, 2018. [Online]. Available: https://www.dot.ny.gov/tdv

Yu-Jen Ku (Student Member, IEEE) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, in 2014, and the M.S. degree in electrical and computer engineering from the University of California San Diego, San Diego, CA, USA, where he is currently working toward the Ph.D. degree in electrical and computer engineering. His research interests include wireless communication standards research, green communication, mobile edge computing, and time series forecasting.

Sabur Baidya (Member, IEEE) received the Ph.D. degree in computer science from the University of California Irvine, Irvine, CA, USA, in 2019 and the M.S. degree in computer science from the University of Texas at Dallas, Richardson, TX, USA, in 2013. He is currently an Assistant Professor of computer science and engineering with the J.B. Speed School of Engineering, University of Louisville, Louisville, KY, USA. Prior to that, he was a Postdoctoral Scholar with the Electrical and Computer Engineering Department, University of California San Diego, San Diego, CA, USA. He was also a Visiting Summer Researcher with WINLAB, Rutgers University, New Brunswick, NJ, USA, and has had prior working experience with IBM, Cisco Systems, Huawei Research Lab, and Nokia Bell Labs. His research interests include the areas of the Internet of Things, wireless networks, intelligent and autonomous systems, and edge-cloud computing. He is a Member of the ACM and IEEE Communication Society.

Sujit Dey (Fellow, IEEE) received the Ph.D. degree in computer science from Duke University, Durham, NC, USA, in 1991. Prior to joining the University of California San Diego (UCSD), San Diego, CA, USA, in 1997, he was a Senior Research Staff Member with NEC C&C Research Laboratories, Princeton, NJ, USA. He was the Chair of the Advisory Board of Zyray Wireless till its acquisition by Broadcom, in 2004, and an Advisor to multiple companies, including ST Microelectronics and NEC. In 2004, he founded Ortiva Wireless, where he has served as its Founding CEO and later as the CTO and the Chief Technologist till its acquisition by Allot Communications, in 2012. He was the Faculty Director of the von Liebig Entrepreneurism Center, from 2013 to 2015, and the Chief Scientist in mobile networks with Allot Communications, from 2012 to 2013. In 2017, he was appointed as an Adjunct Professor with the Rady School of Management and the Jacobs Family Chair in Engineering Management Leadership. He is currently a Professor with the Department of Electrical and Computer Engineering and the Director of the Center for Wireless Communications and the Institute for the Global Entrepreneur, UCSD. He heads Mobile Systems Design Laboratory, developing innovative technologies in mobile cloud computing, adaptive multimedia and networking, green computing and communications, and predictive and prescriptive analytics to enable future applications in connected health, immersive multimedia, and smart transportation. He has coauthored more than 250 publications and a book on low-power design. He holds 18 U.S. and two international patents, resulting in multiple technology licensing and commercialization. He was the recipient of the seven IEEE/ACM Best Paper Awards and has chaired multiple IEEE conferences and workshops.