Zhu, Chao; Chiang, Yi-Han; Mehrabi, Abbas; Xiao, Yu; Yla-Jaaski, Antti; Ji, Yusheng

Chameleon

Published in:
IEEE Transactions on Vehicular Technology

DOI:
10.1109/TVT.2019.2924911

Published: 01/09/2019

Document Version
Publisher's PDF, also known as Version of record

Chameleon: Latency and Resolution Aware Task Offloading for Visual-Based Assisted Driving

Chao Zhu, Yi-Han Chiang, Abbas Mehrabi, Yu Xiao, Antti Ylä-Jääski, and Yusheng Ji, *Senior Member, IEEE*

***Abstract*—Emerging visual-based driving assistance systems involve time-critical and data-intensive computational tasks, such as real-time object recognition and scene understanding. Due to the constraints on space and power capacity, it is not feasible to install extra computing devices on all the vehicles. To solve this problem, different scenarios of vehicular fog computing have been proposed, where computational tasks generated by vehicles can be sent to and processed at fog nodes located for example at 5G cell towers or moving buses. In this paper, we propose Chameleon, a novel solution for task offloading for visual-based assisted driving. Chameleon takes into account the spatiotemporal variation in service demand and supply, and provides latency and resolution aware task offloading strategies based on partially observable Markov decision process (POMDP). To evaluate the effectiveness of Chameleon, we simulate the availability of vehicular fog nodes at different times of day based on the bus trajectories collected in Helsinki, and use the real-world performance measurements of visual data transmission and processing. Compared with adaptive and random task offloading strategies, the POMDP-based offloading strategies provided by Chameleon shortens the average service latency of task offloading by up to 65% while increasing the average resolution level of processed images by up to 83%.**

***Index Terms*—Vehicular fog computing, task offloading, assisted driving, POMDP.**

I. Introduction

EMERGING visual-based assisted driving applications, such as see-through and cooperative lane-change, involve time-critical and data-intensive computational tasks, such as real-time object recognition and scene understanding from images/video. Obviously, processing visual data demands for a lot more computing power, compared with other sensor data like GPS fixes and motion data. Due to the space, weight, and cost constraints [1], computing capacity of most vehicles may not be high enough to handle such tasks. On the other hand, offloading these tasks to the cloud is not applicable, due to the remarkable transmission delay.

To provide low-latency processing for assisted driving, a novel computing paradigm called *vehicular fog computing* (VFC) [2]–[6] has been proposed. Its key idea is to offload computational tasks from the client vehicles where data is generated to *fog nodes* located at for example 5G cell towers or buses with extra computing power. In either scenario, only one-hop communication is required, which greatly shortens the transmission delay [7]. Moreover, turning moving vehicles like buses into fog nodes enables on-demand fog computing, and reduces service migration between fog nodes when client vehicles are traveling along with fog nodes [8].

Due to the mobility of vehicles, the density of client vehicles and therefore the amount of tasks generated by client vehicles vary with time and place. Meanwhile, the availability of fog nodes carried by vehicles, called vehicular fog nodes in this paper, depends on the driving routes of the carriers. The spatiotemporal variation in both supply and demand of computing services adds a layer of complexity to the scheduling of task offloading from client vehicles to fog nodes.

Depending on the availability of computing resources, in order to complete tasks within latency constraints, a widely adopted approach is to decrease the quality of data to be processed with the amount of computing resources. In case of visual-based assisted driving, the quality of data can be measured with image resolution. More computing power is required for processing images with higher resolution, which are expected for providing more accurate scene understanding.

To address these challenges, we propose *Chameleon*, a task offloading scheme that reduces service latency while increasing the supported quality levels of visual data to be processed within application specific latency constraints, taking into account the mobility of vehicles and the impact of data quality (e.g. resolution) on processing delay. Chameleon is applicable to the scenarios where both infrastructural (e.g. 5G cell towers) and vehicular (e.g. bus-carried) fog nodes are available. It solves the above-mentioned challenges from two perspectives. Firstly, it analyzes the spatiotemporal variation in the workload of fog nodes based on vehicular traffic patterns in urban areas. Secondly, it allows each client vehicle to decide whether to offload based on the received information about fog node workload.

Manuscript received January 23, 2019; revised April 28, 2019; accepted June 15, 2019. Date of publication July 22, 2019; date of current version September 17, 2019. This work was supported in part by the Academic of Finland under Grant 317432 and Grant 297892, in part by the European Union's Horizon 2020 Research, Innovation Program under Grant 815191 and Grant 825496, in part by JSPS KAKENHI under Grant JP18KK0279, and in part by the Nokia Center for Advanced Research and Technology Industries of Finland Centennial Foundation. The review of this paper was coordinated by Prof. J. Ren. *(Corresponding author: Yu Xiao.)*

C. Zhu and Y. Xiao are with the Department of Communications and Networking, Aalto University, 02150 Espoo, Finland (e-mail: chao.1.zhu@aalto.fi; yu.xiao@aalto.fi).

Y.-H. Chiang and Y. Ji are with the Information Systems Architecture Science Research Division, National Institute of Informatics, Tokyo 101-8430, Japan (e-mail: yhchiang@nii.ac.jp; kei@nii.ac.jp).

A. Mehrabi and A. Ylä-Jääski are with the Department of Computer Science, Aalto University, 02150 Espoo, Finland (e-mail: abbas.mehrabi davoodabadi@aalto.fi; antti.yla-jaaski@aalto.fi).

Digital Object Identifier 10.1109/TVT.2019.2924911

Chameleon formulates the optimization of task offloading as a partially observable Markov decision process (POMDP) and solves the problem through stochastic dynamic programming approach [9].

To evaluate the effectiveness of Chameleon, we simulate the scenario of assisted driving in VFC using the real-world vehicular traffic data and applications of image-based object recognition. We take an area of 67 square kilometers in Helsinki city center for case study. Compared with the existing solutions, including adaptive and random task offloading strategies [10], the POMDP based task offloading strategies provided by Chameleon reduce the average service latency by up to 65% and increase the average resolution level of processed images by up to 83%.

The key contributions of this work are summarized below:

- We analyze real-world traffic data collected from *HERE* maps [11] and discover the spatiotemporal patterns shown in vehicular traffic density.
- We develop Chameleon, a novel task offloading scheme that tries to process higher resolution images within application specific latency constraints, taking into account the spatiotemporal variation in vehicular traffic density and the impact of image resolution on processing delay.
- We prove the effectiveness of Chameleon through large-scale simulation, using the profiles of an image-based object recognition application and real-world bus trajectories as input.

The rest of the paper is organized as follows. Section II discusses the related works. Section III describes the system. The variation of fog node workload is analyzed in Section IV. The formulation of POMDP is presented in Section V. Section VI discusses the evaluation configuration and final results. We discuss the limitation of our work and present the future plan in Section VII before we conclude in Section VIII.

II. Related Work

A. Task Offloading

In the past few years, plenty of works [12]–[15] have been devoted to the studies of task offloading in mobile edge/fog computing. Based on the trade-off between reduced computation cost and increased transmission cost, edge/fog nodes or sometimes the clients decide whether to offload, where to offload and what to offload.

Beside reducing computation latency, offloading computational tasks to edge/fog servers can reduce the energy consumption of user equipments (UEs). In [12], Cao *et al.* divided tasks into non-offloadable and offloadable types, and then proposed an optimal adaptive algorithm to minimize energy consumption subject to delay constraints. In [13], Deng *et al.* evaluated the dependency of tasks and proposed a particle swarm optimizer based algorithm for energy conservation while satisfying strict delay requirements. These papers [12], [13] are limited to single-UE scenarios. In [14], You *et al.* assumed a multi-UEs scenario where time is divided into slots and partial tasks generated by UEs may be offloaded according to channel quality, local computing energy consumption, and fairness among the UEs. Similarly, in [15], Munoz *et al.* provided a trade-off analysis between the energy consumption and the execution delay for the partial task offloading decision in the multi-UEs scenario.

The targets of all aforementioned papers on task offloading is to minimize UE's energy within latency constrains.Compared with UEs like mobile devices, vehicles have relatively sufficient power supply. Therefore, instead of energy consumption, we take a different aspect of task offloading in assisted driving to focus on the trade-off between service latency and quality of data.

B. Task Offloading for Assisted Driving

The evolution of assisted driving applications comes along with various compute-intensive and time-critical tasks, which have limited its usages on vehicles lacking computing resources. VFC, which enables vehicles to offload computational tasks to fog nodes within one-hop communication, has provided a promising approach to addressing the intensive computation burden. A few existing works have investigated the potentials of task offloading for assisted driving applications in VFC. In [16], Chen *et al.* proposed a resource cognitive intelligent architecture based on the learning of network contexts to manage computing and communication resources for low-latency task offloading. In [17], Xiao *et al.* reduced the service response time and improved the efficiency of power usage in fog computing by balancing the workload of fog nodes. However, differently from Chameleon, none of the previous works has considered the spatiotemporal variation in vehicular traffic density and the impact of fog node workload on task processing latency.

III. System Description

In this section, we present the related terms, and introduce the procedure of task offloading.

A. Terminology

Fog nodes: In Chameleon, we consider two types of fog nodes: 1) *infrastructural fog nodes*—the computing nodes co-located with network infrastructures (e.g. 5G cell towers), and 2) *vehicular fog nodes*—the computing nodes carried by moving commercial fleets (e.g. buses) with on-board opportunistic communication modules (e.g. Wi-Fi and DSRC).

Tasks: An assisted driving application consists of a set of tasks. For example, assisted lane changing can be implemented with real-time object recognition from images collected from neighboring vehicles. In Chameleon, each task is considered as the basic unit for offloading. One task will be offloaded as a whole to one fog node.

Fog node workload: In this paper, we assume that all the client vehicles run the same visual processing tasks and generate service demand at the same speed. Thus, the workload of a fog node is directly affected by the number of neighboring client vehicles, which is closely related to the density of vehicle traffic on the road. From the analysis of real-world traffic message channel (TMC) data (including location, average speed and traffic jam factor) collected from [11], we find that the variation in

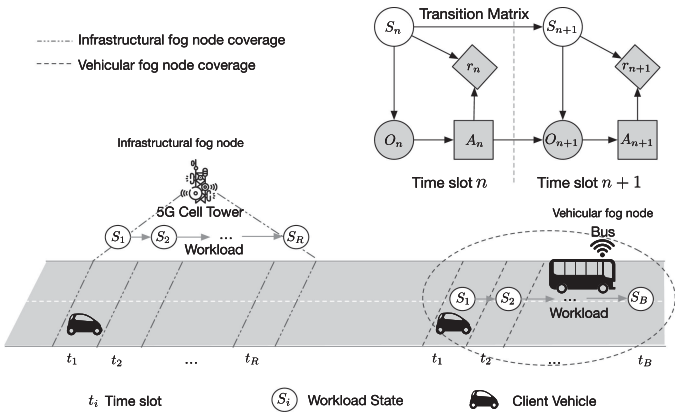


Fig. 1. System architecture.

the number of client vehicles which fall into the communication range of a specific fog node and the corresponding workload generated by these client vehicles changes on a weekly basis. We will analyze the variation of the workload of fog nodes in details in Section IV.

Observation of fog node workload: Instead of broadcasting the workload information, we propose that after completing one offloaded task, the fog node would send the information about its current workload together with the processed results to the client vehicle. The client vehicle can get a workload observation only when a task offloading completes. Otherwise, the client vehicle is not aware of the workload of fog node.

Time bucket: Within a certain time period, we find that the vehicular traffic density changes regularly. We define these time periods as *time buckets*.

Task utility: The utility for processing a task, which decays along with the service latency and increases with the image resolution. In Chameleon, task utility unifies service latency and image resolution, and can be tuned according to application specific requirements. For example, if the application is more latency sensitive, the task utility will be more heavily weighted in favor of service latency, and vice versa.

B. Process of Chameleon

In vehicular network, frequent communication with a single point or central controller equipped on a specific fog node may not be feasible due to the high mobility of vehicles. Therefore, we design a distributed task offloading scheme, where each client vehicle makes the task offloading strategy by itself. Fig. 1 illustrates the procedure of task offloading in Chameleon. Assume that tasks are generated continuously with a fixed arrival rate. Accordingly, a time bucket can be equally divided into a sequence of time slots. We assume that the variation in the fog node workload in a specific time bucket evolves as a Markov chain. We record the workload state of the fog node in each time slot and use a probability transition matrix to describe the transition of the Markov chain from S_n to S_{n+1}, where S_n denotes the workload state of fog node at time slot n. The workload probability transition matrix is updated on a daily or weekly basis. In Chameleon, we aim at finding a sequence of task offloading actions to maximize the cumulative task utility during each time bucket. The whole process consists of 4 steps as shown below.

- *Fog node selection:* In the initial stage, a client vehicle needs to figure out which fog nodes are located within its communication range. It broadcasts one-hop probe messages and collects responding messages from fog nodes. The responding messages contain the information about the location and the workload probability transition matrix of fog nodes in current time bucket. According to the responding messages, the client vehicle can choose one fog node for offloading. In this paper, we just adopt a simple fog node selection scheme, in which the client vehicle would select the fog node with the shortest communication distance for task offloading [6], [18].
- *Offloading service initialization:* We illustrate the process of seeking the task offloading strategy for a certain time bucket in Fig. 1. Assume the fog node workload is S_n and the client vehicle receives an observation O_n at time slot n. Based on O_n, the client vehicle selects an offloading action A_n, receives a reward r_n, and the fog node workload changes into S_{n+1}, where S_{n+1} depends only on the transition matrix and S_n. The client vehicle then receives an observation O_{n+1}, which is dependent on S_{n+1} and A_n. According to the received observation, the task offloading strategy instructs the client vehicle to offload or locally process the task generated in each time slot. In Chameleon, the POMDP based task offloading strategy tries to achieve the maximum cumulative reward r_n in the current time bucket.
- *Task offloading:* During the task offloading procedure, the client vehicle will continuously get workload observations from the fog node. By mapping the observations into the task offloading strategy, the vehicle can get a sequence of optimal offloading decisions (i.e., selecting an appropriate image resolution for task offloading at each time bucket). According to the task offloading strategy and the received workload observations, tasks generated in one time bucket would be decided to be either processed locally or offloaded with appropriate image resolution.
- *Service interruption and termination:* In vehicular network, the connection between a client vehicle and a fog node may be lost due to the mobility of client vehicles and/or fog nodes. To avoid task interruption, fog nodes need to monitor the quality of the communication channel and stop task offloading service once the channel quality falls below a certain threshold. For instance, when client vehicle is moving out from the coverage of the connected fog node, the fog node will inform the client vehicle to stop offloading new tasks and the service running on the corresponding fog node will be terminated once the remaining tasks are processed.

IV. Fog Node Workload

In this section, we analyze the fog node workload variation based on the real-world datasets. Specifically, we explore the

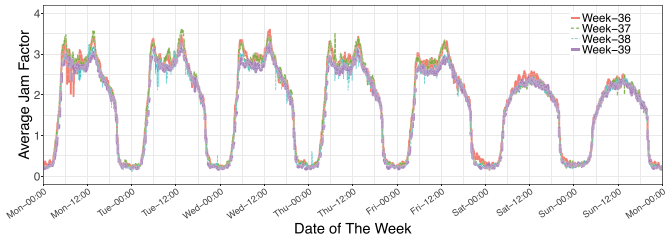
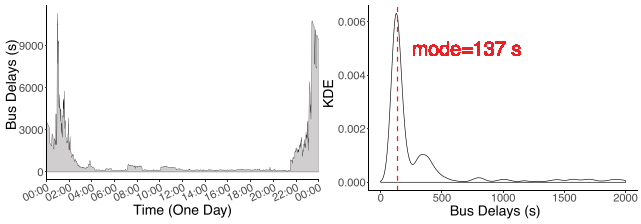


Fig. 2. Temporal variation of traffic jam in Helsinki during 3rd Sep. and 6th Oct., 2018.



(a) Variation in Bus Delays (b) Distribution of Bus Delays

Fig. 3. Bus delays.

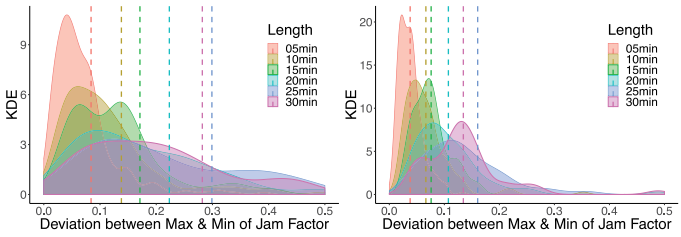
spatiotemporal variation in vehicular traffic density by analyzing the TMC data, and examine the variation of the workload of vehicular fog nodes by investigating the bus trajectories. Furthermore, we explore the influence of the time buckets length on the deviation of the vehicular traffic density.

A. Temporal Variation in Vehicle Density

To identify the temporal variation of vehicular traffic density, we collected the TMC data in a region of 67 square kilometers in Helsinki using *Here Traffic API* [11] for about one month from 3rd Sep. to 6th Oct., 2018. The TMC data contains the traffic reports received from vehicles, including the location, direction, average speed of the traffic flow. From the TMC data, we can get the *jam factor*, which represents the expected quality of travel [11]. The jam factor is a number between 0 and 10. The larger the jam factor is, the higher the vehicular traffic density is. The temporal variation of the jam factor is illustrated in Fig. 2. We find that the vehicle density changes dramatically within one day but remains stable during certain time periods. For example, suppose the length of a time period is one hour. As illustrated in Fig. 2, the jam factor changes within the range from 2.6 to 2.8 during 11:00∼12:00, compared with a range from 2.7 to 3.3 in the time period 12:00∼13:00. In addition, the temporal variation of the vehicular traffic density exhibits a repetitive pattern on a weekly basis.

B. Availability of Vehicular Fog Node

Due to the mobility of vehicular fog nodes, whether they show time- and place-varying workload should be validated in the first place. To identify the workload pattern of vehicular fog nodes, we collected bus trajectories by using the open high-frequency positioning (HFP) API provided by *HSL* [19] in Helsinki region



(a) Weekday (b) Weekend

Fig. 4. Jam factor deviation under different length of time buckets.

during the same period as mentioned previously. Fig. 3a illustrates the variation of bus delays during one day. We can see that the buses are most of time punctual (except early moring and late night). We analyze the kernel density estimation (KDE) of bus delays in Fig. 3b. From the figure, we can see that the delays of most buses were limited to 500 seconds and the mode of bus delays was 137 seconds. Since the buses have daily-fixed trajectories, the supply of vehicular fog computing follows a regular pattern according to pre-defined time tables.

C. Length of Time Bucket

To explore the impact of the length of time bucket on the deviation of the density of vehicle traffic, we select the TMC information in two days (03/09/2018, Monday and 09/09/2018, Sunday) for analysis and illustrate the deviation between the maximum and the minimum of jam factor under different length of time buckets in Fig. 4.

The smaller the deviation, the more stable the fog node workload within one time bucket. As shown in Fig. 4a, the deviation between the max and the min is around 0.3 when the length of time buckets is 30 minutes in weekdays. However, with the refinement of the time buckets, the deviation reduces to 0.09 when the length of time bucket drops to 5 minutes. Furthermore, as illustrated in Fig. 4b, the length of time bucket can be relaxed to 15 minutes to keep a similar deviation (under 0.1) for weekends. By exploring the spatiotemporal variation in the density of vehicular traffic, the length of time buckets can be customized for different days of each week.

V. POMDP Formulation

In this section, we introduce the formulation of POMDP. As listed in Table I, we first present the notations of POMDP and explain their coincidence to the related terms in task offloading. Then, we give the POMDP's recursion function and illustrate the methodology for seeking the optimal offloading strategy for task offloading.

A. Notations of POMDP

1) States: Assume the fog node workload in specific time bucket evolves as an M-state Markov chain $\mathcal{L} = \{l_1, l_2, ..., l_M\}$, where $l_i \in \mathbb{Z}_+$ and with the interpretation that the greater the index, the heavier the fog node workload $l_1 < l_2 < \cdots < l_M$.

2) Transition Probability: We use a probability transition matrix R to denote the transition of states, where $R_{ij} = P\{L_n =$

TABLE I
NOTATIONS AND DEFINITIONS

Notations	Definitions
$l_i, \mathcal{L}$	fog node workload state, set
M	number of states
$q_i, \mathcal{Q}$	image resolution level, set
K	number of image resolution levels
L_n	fog node workload state at time slot n
A_n	task offloading action at time slot n
$\mathcal{A}$	set of task offloading actions
O_n	workload observation at time slot n
$\mathcal{O}$	set of workload observations
$\mathbf{p}_n$	probability distribution of workload states at time slot n
R	probability transition matrix
a	task utility decay per second
b	task utility grow per level of image resolution
$t(l_i, q_i)$	transmission latency when fog node workload state is l_i and image resolution level is q_i
$p(l_i, q_i)$	processing latency when fog node workload state is l_i and image resolution level is q_i
$\mathcal{U}_r$	unified task utility for offloading task to fog node
$\mathcal{U}_l$	unified task utility for processing task locally
γ	discount factor weighing future task utility decaying
δ, Δ	task offloading strategy, set
δ^*	optimal task offloading strategy
ϕ	latency sensitivity parameter

$l_j|L_{n-1} = l_i\}$ means that the probability of the state L_n at time slot n is l_j given that at the last time slot it is l_i. For brevity, we denote the i^{th} row of R by R_i.

Regarding the variation of the fog node workload, the workload probability transition matrix within a time bucket can be learned from the historical records with *Monte Carlo* method. For example, suppose the time bucket is 5 minutes and the task arrival rate is 1 task/second. Then, the time bucket can be divided into 300 time slots. Given a fog node, we record its workload state at each time slot during the same day of each week. For example, to get the workload transition matrix of a fog node within the first time bucket on Mondays, we can record the workload state of the fog node on every second during 00:00∼00:05 on Mondays. From a large amount of historical records, the probability transition matrix R of the time bucket of the fog node can be learned as following:

$$R_{ij} = P\{L_n = l_j | L_{n-1} = l_i\} = \frac{N(e(l_i \text{ to } l_j))}{N(e(l_i))}, \quad (1)$$

where $e(l_i)$ and $e(l_i \text{ to } l_j)$ represents the events that the workload state of the fog node stays at l_i and transfers from l_i to l_j, respectively. N is a function that counts how often the event occurs.

3) Actions: For each application, we suppose K-level image resolutions can be chosen for task offloading, which is $\mathcal{Q} = \{q_1, q_2, ..., q_K\}$ and $\{q_i \in \mathbb{Z}_+\}$. Assume $q_1 < q_2 < \cdots < q_K$, with the interpretation that the greater the index, the higher the image resolution.

Due to the limited computing capacity of client vehicles, we assume that client vehicles can only process tasks with the lowest level of image resolution. Thus, there are $|\mathcal{Q}| + 1$ actions for task offloading and are denoted by $\mathcal{A} = \{0\} \cup Q$, where $A_n \in \mathcal{A} = q_i > 0$ means offloading the task with image resolution level q_i at time slot n, and $A_n = 0$ means processing the task locally with image resolution level q_1.

4) Observations: In Chameleon, we assume that there is no extra connection between fog node and client vehicle unless a task offloading occurs. Thus, the state of the fog node workload at a time slot will be made available to the client vehicle at the beginning of the next time slot, only if a task offloading is attempted by the client vehicle at the current time slot [20]. Otherwise, the client vehicle has no idea about the fog node workload at the current time slot. Let $\mathcal{O} = \{\ominus\} \cup \mathcal{L}$ be the *observation* set. We get the workload observation O_n at time slot n as:

$$O_n = \begin{cases} L_{n-1} & \text{if } A_{n-1} > 0 \\ \ominus & \text{if } A_{n-1} = 0, \end{cases} \quad (2)$$

where $\ominus$ indicates there is no workload observation.

5) Rewards: The higher the image resolution, the higher probability of detecting small objects. We regard the rewards of task offloading as the task utility, which decays along with the service latency and grows along with the image resolution. According to [21], we use a linear function to model the task utility for offloading an image with resolution level q_i when the fog node workload state is l_i:

$$\mathcal{U}_r(l_i, q_i) = \phi a(t(l_i, q_i) + p(l_i, q_i)) + (1 - \phi) b q_i, \quad (3)$$

where the task utility grows a utility per second and b utility per resolution level. $t(l_i, q_i)$ and $p(l_i, q_i)$ denote the transmission and processing latency of task offloading, respectively. The parameter $\phi \in [0, 1]$ models the latency sensitivity. If the application is more sensitive to the service latency, ϕ can be tuned to be larger, and vice versa.

For simpilification, we set the value of a to -1 and b to 1. Additionally, the client vehicle can process the task locally by itself. The task utility for local processing is:

$$\mathcal{U}_l = -\phi p(\ominus, q_1) + (1 - \phi) q_1 \quad (4)$$

B. Recursion Function and Solution

Due to the uncertainty of fog node workload state, we use an *information vector* $\mathbf{p}_n = \langle p_n(1), p_n(2), ..., p_n(M) \rangle$ to denote the conditional probability distribution of the fog node workload state at time slot n [9]. Given the initial information vector $\mathbf{p}_0$, the past offloading decisions $A_0, A_1, ..., A_{n-1}$ and the past and present observations $O_0, O_1, ..., O_n$, the probability of staying at state l_i at time slot n can be calculated by using Bayes' formula as following:

$$p_n(i) = P\{L_n = l_i | \mathbf{p}_0; A_{n-1}, \ldots, A_0; O_n, \ldots, O_0\}, \quad (5)$$

where $i = 1, ..., M$.

According to Eq. (2), the probability distribution at time slot n is updated as below,

$$\mathbf{p}_n = \begin{cases} R_i & \text{if } A_{n-1} > 0, \quad O_n = l_i \\ \mathbf{p}_{n-1} R & \text{if } A_{n-1} = 0. \end{cases} \quad (6)$$

According to Eq. (3), the expected task utility at time slot n^{th} is described as below,

$$U_r(\mathbf{p}_n, A_n) = E[-\phi(t(l_i, A_n) + p(l_i, A_n))\mathbf{p}_n + (1-\phi)A_n]. \tag{7}$$

For simplification, we use U_l to denote the task utility for local processing. According to Eq. (4), the expected task utility at time slot n can be calculated as:

$$U(\mathbf{p}_n, A_n) = \begin{cases} U_r(\mathbf{p}_n, A_n) & \text{if} \quad A_n > 0, \\ U_l & \text{if} \quad A_n = 0. \end{cases} \tag{8}$$

We use $\delta = \{A_0, A_1, ..., A_i, ...\}$ and $\delta^* = \{A_0^*, A_1^*, ..., A_i^*, ...\}$ to denote an arbitrary task offloading strategy and the optimal one, respectively. According to [9], the maximum of expected cumulative discounted task utility under optimal offloading strategy δ^* can be calculated as:

$$\mathcal{V}(\mathbf{p}) = \sum_{n=0}^{\infty} [\gamma^n U(\mathbf{p}_n, A_n) | \mathbf{p}_0, \delta^*], \tag{9}$$

where $\mathbf{p}_0$ is an initial probability distribution on $\mathcal{L}$ and $\gamma \in (0, 1)$ is a discount factor weighing present costs more heavily than future costs.

We use $\Delta = \{\delta\}$ to denote the set of task offloading strategies. The optimal task offloading strategy δ^* can be calculated as:

$$\delta^* = \arg\max_{\delta \in \Delta} \sum_{n=0}^{\infty} [\gamma^n U(\mathbf{p}_n, A_n) | \mathbf{p}_0, \delta]. \tag{10}$$

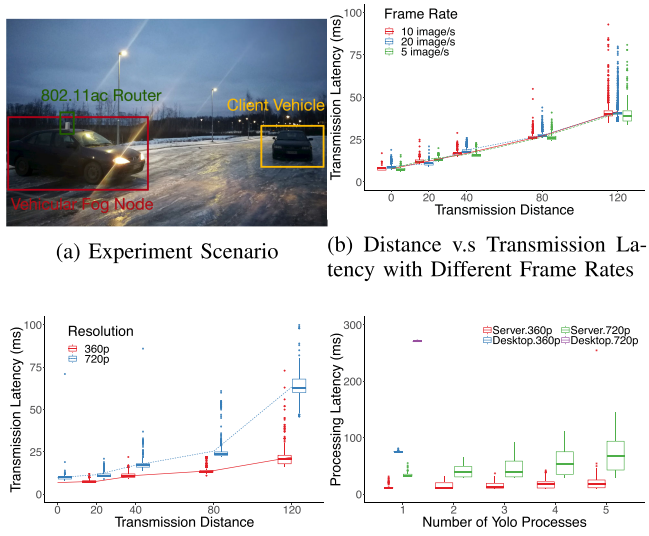
According to [20], the problem of seeking optimal task offloading strategy can be formulated as a POMDP. For an infinite horizon optimization problem, the recursion function of the maximum accumulative discounted task utility is given as [9], [20]:

$$\mathcal{V}(\mathbf{p}) = \max\{U_l + \gamma \mathcal{V}(\mathbf{p}R), \max_{A \in Q}\{\mathcal{U}_r(\mathbf{p}, A)\} + \gamma \sum_{j \in \mathcal{L}} \mathcal{V}(R_j) p(j)\}. \tag{11}$$

According to [20], we obtain the stationary and time-invariant optimal task offloading strategy by utilizing *incremental pruning* [22].

VI. Performance Evaluation

In this section, we evaluate the effectiveness of Chameleon. We first investigate the profiles of an image-based object recognition application, and see how image resolution influences the transmission and processing latency in case of task offloading. Then, we give the simulation settings regarding the real-world TMC data and bus trajectories, and observe the temporal variation in the task offloading service demand and supply. Finally, we illustrate the achieved performance of the POMDP based offloading strategies, and compare it with two existing task offloading strategies.



(a) Experiment Scenario

(b) Distance v.s Transmission Latency with Different Frame Rates

(c) Distance v.s Transmission Latency with Different Resolutions

(d) Fog Node Worload v.s. Processing Latency

Fig. 5. Experiments of image transmitting and processing.

A. Application Profiling

1) Transmission Latency: To profile the transmission latency in task offloading, we implement an image transmitting application, where a client vehicle is continuously transmitting images to a fog node. As shown in Fig. 5a, we parked a vehicle in the center of an open-parking ground and let it act as a fog node. Meanwhile, we drive another vehicle through the open-parking ground in both directions. While the client vehicle is moving, images with different resolutions are being transmitted from various communication distances. The length of the open-parking ground is about 300 m and the transmission latency is recorded every 20 m. Due to the limitation of the site, the speed of the client vehicle is very low, which is under 2.5 m/s. We place a laptop with *Airport Extreme* wireless card in each vehicle and install an 802.11ac router *Huawei b618* on the top of the fog node.

To evaluate the impact of frame rate on the transmission latency, we fix the image resolution to 720p and select 3 frame rates for experiments, which are 5, 10 and 20 image/s, respectively. As shown in Fig. 5b, the transmission latency increases with the extension of communication distance. When the communication distance exceeds 120 m, we cannot measure the transmission latency, due to the lost of connection. By comparing the transmission latency of different frame rates, we can see that the frame rate does not have an obvious impact on transmission latency due to the large bandwidth of 802.11 ac radio access technology.

To further evaluate the impact of image resolution, two image resolutions (i.e., 360p and 720p) are selected for transmitting and the frame rate is fixed to 5 image/s. As shown in Fig. 5c, with the increase of communication distance, the transmission latency of 360p images stays stable at a certain level (less than 25 ms) while that of 720p images increases rapidly (from 20 ms to 65 ms). In VFC scenario, it is hard to predict the variation in communication distance between two vehicles. For reliability

concern, we set the transmission latency of images with 360p resolution to 25 ms and that of images with 720p resolution to 65 ms over 802.11 ac radio access technology.

According to [7], the average units per transaction (UPT) of LAA (License Assisted Access) standards of 5G radio access technologies can achieve 100mbps within 90 m coverage radius. Accordingly, for infrastructural fog nodes with 5G radio access technology, we set the transmission latency of 360p images (with 125 KB datasize) to 10 ms and that of 720p images (with 250 KB datasize) to 20 ms, respectively.

2) Processing Latency: To profile the processing latency of the images under different levels of fog node workload, we use a real-time object detection application – YOLO [23] for image processing. We assume the computing devices equipped on the client vehicles are kind of computers with normal GPU cards and those on the fog nodes are kind of GPU servers. We collected 1000 driving-scene images with 360p and 720p resolution from a dash-cam and ran YOLO [23] on a Linux-OS desktop (with Quadro K2200 CPU, 4 GB memory) and a GPU server (NVIDIA Corporation GV100, 16 GB memory) respectively, to recognize the objects in the images. Due to the memory limitation, up to 5 YOLO processes on the GPU servers and only 1 YOLO process on the desktop can be simultaneously operated. According to Fig. 5d, the processing latency of the image processing in the fog node outperforms that in the client vehicle for both 360p and 720p images. However, when the fog node workload is heavy (more than 5 processes), the new coming task needs to wait for the access to GPU memory until the YOLO processes being processed are completed. Regarding the specification of assisted driving applications, suppose that the service latency cannot exceed 200 ms [24]. According to Fig. 5c and Fig. 5d, each fog node is restricted to have maximum 10 connections to client vehicles in parallel. Because in this case, the worst situation is that one task with 720p image is offloaded to the fog node when there are already 5 tasks using the fog node memory. Then the service latency of that task is the summation of the delay for transimitting, waiting for GPU memory access and processing, which is $60 + 65 + 65 = 190$ ms (the worst case is that one of the tasks in the GPU memory is with 720p image).

We define two states of fog node workload, which are *Busy* and *Idle* to denote whether the fog node GPU memory is fully occupied or not. For simplification, we use the set of $\mathcal{L} = \{0, 1\}$ to denote the two states, respectively. According to Fig. 5d, we set the workload state to 1 when there are more than 5 client vehicles connected to a fog node. Otherwise, we suppose the fog node workload state is 0. For a task with 360p image, we set the processing latency of the task in the client vehicle to 75 ms and that in the fog node to 25 ms when its workload state is 0. When the fog node workload state is 1, the new coming task needs to wait for at most 65 ms for the GPU memory access. Thus, it takes 90 ms for the fog node to process the task with 360p image and 130 ms to process the task with 720p image. As illustrated in the Fig. 5d, it takes more than 280 ms for the client vehicle to process one task with 720p image, which is intolerable for the assisted driving applications. Thus, we just consider three kinds of task offloading actions, which are processing task locally with 360p image and offloading task with 360p image and 720p image respectively, and use set $\mathcal{A} = \{0, 1, 2\}$ to denote them. We summarize the application profiles in Table II.

TABLE II
IMAGE TRANSMISSION AND PROCESSING LATENCY

Resolution	Transmission Latency (ms)		Processing Latency (ms)		
	WiFi	5G	Idle	Busy	Local
360p	20	10	25	90	75
720p	60	20	65	130	280

(a) Time Bucket I (b) Time Bucket II

Fig. 6. Visualization of fog nodes. Each color represents the traces of a specific fog node. However, due to the limitation of color palette, colors may not be distinguished from each other.

B. Simulation Scenario

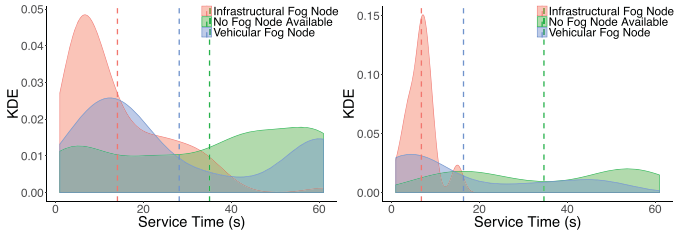
1) Area and Time: To evaluate the impact of the temporal features of vehicular traffic, we select an area of 1 square kilometers in Helsinki region. The latitude of that area ranges from $24°53'16''$ to $24°54'25''$, while the longitude ranges from $60°12'16''$ to $60°12'49''$. According to Fig. 4 and Fig. 2, we set the length of time bucket to 5 minutes and select two time buckets to represent the time periods with high and low vehicular traffic density as following.

* *Time Bucket I:* $08:00 \sim 08:05$, September 17, 2018
* *Time Bucket II:* $20:00 \sim 20:05$, September 17, 2018

We collect the bus trajectories and TMC data within the area in the above two time buckets.

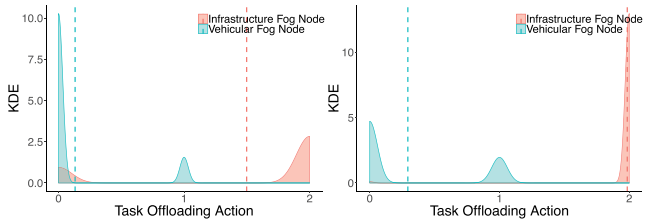
2) Fog Nodes and Client Vehicles: We visualize the bus trajectories in the selected area in two time buckets in Fig. 6. The route of each bus is represented by a specific color. As shown in the Fig. 6, the number of buses appeared in Time Bucket I is larger. In details, 25 buses appeared in the selected area in the first time bucket, compared with 17 buses in the latter. Besides the vehicular fog nodes, ten 5G cell towers are evenly placed in the selected area in the simulation and act as the infrastructural fog nodes. We suppose vehicular and infrastructural fog nodes utilize 802.11ac and 5G access technologies for communication, respectively. According to the measurements mentioned previously in Section VI-A, we set the radial coverage of vehicular fog nodes to 120 m and that of infrastructural fog nodes to 90 m [7].

From the analysis of the TMC data, we get that the average speed of vehicular traffic flows in these two time buckets are 28 km/h and 36 km/h, respectively. From the correlation between



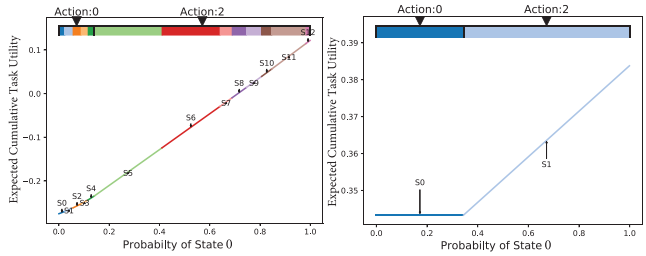
(a) Time Bucket I (b) Time Bucket II

Fig. 7. Service time of different types of fog nodes.



(a) Time Bucket I (b) Time Bucket II

Fig. 8. Distribution of task offloading actions with different types of fog nodes when $\phi = 0.95$.



(a) R_I (b) R_{II}

Fig. 9. Maximum cumulative task utility vs. WVP matrix.

the average speed and vehicular traffic density learned from real-world traffic flows [25], 40% and 15% of the road are occupied by vehicles in the Time Bucket I and II, respectively. Suppose the total length of road in that area is about 3 km and the length covered by a vehicle is about 5m, the number of vehicles appeared in Time Bucket I and II can be calculated accordingly, which are 240 and 90, respectively. We generate 240 and 90 vehicle routes in these two time buckets in SUMO [26], following the method used in [27]. These vehicles act as client vehicles and continuously generate images for processing with the rate of 5 images/s.

3) Fog Node Selection: In Chameleon, we adopt the fog node selection method presented in [8]. This method assigns the fog node which is closest to the client vehicle for task offloading. As introduced in Section III, the service from the fog node would be interrupted if the client vehicle is not in its coverage. We define the *service time* as the duration from a connection is established until the connection is disconnected. We illustrate the KDE of the service time of fog nodes in Fig. 7. From Fig. 7a and Fig. 7b, we can see that the service time of vehicular fog nodes is longer than that of the infrastructural fog nodes in both two time buckets. This is because compared with infrastructural fog nodes, vehicular fog nodes could travel along with client vehicles. Furthermore, the service time of the infrastructural fog nodes is shorter in Time Bucket II compared with that in Time Bucket I. This is because when vehicular traffic becomes more congested, the average speed of vehicular traffic flows will decrease and client vehicles will stay longer in the coverage of the infrastructural fog nodes.

C. POMDP-Based Offloading Strategy

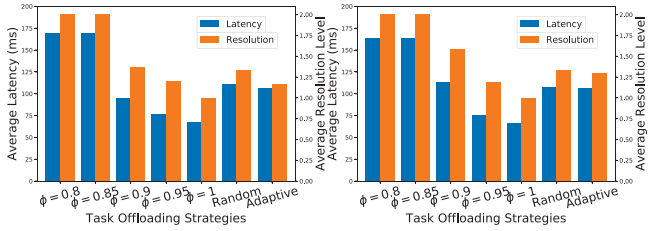
The larger the value of ϕ, the higher the sensitivity to latency. Notably, the POMDP based task offloading strategies achieve a same performance with $\phi \in [0, 0.8]$. To analyze the POMDP based strategies in details, we set the latency sensitivity parameter to 0.95. Fig. 8 illustrates the KDE of task offloading actions with infrastructural and vehicular fog nodes in the two time buckets. From Fig. 8a, we can see that the client vehicles tend to offload processing tasks of 720p images if the fog node is an infrastructural one, process images locally or offload but rather 360p images if the fog node is a vehicular one. This is because compared with Wi-Fi, 5G access radios equipped on infrastructural fog nodes are more advanced and the transmission latency of task offloading would be greatly reduced. Compared with Fig. 8a and Fig. 8b, we can see that client vehicles are more likely to process tasks locally when the vehicular traffic becomes congested. This is because local processing would avoid increasing the workload of the fog nodes.

By Monto-Carlo method, we can calculate the probability transition matrix of each fog nodes in each time bucket. To illustrate the internal principle of POMDP offloading strategy, we select one infrastructural fog node located in the center of the area (24°53′48″, 60°12′30″) and present their probability transition matrices, R_I and R_{II}, in time bucket I and II as following:

$$R_I = \begin{bmatrix} 0.967 & 0.033 \\ 0.034 & 0.966 \end{bmatrix}, R_{II} = \begin{bmatrix} 0.999 & 0.001 \\ 0.999 & 0.001 \end{bmatrix}.$$

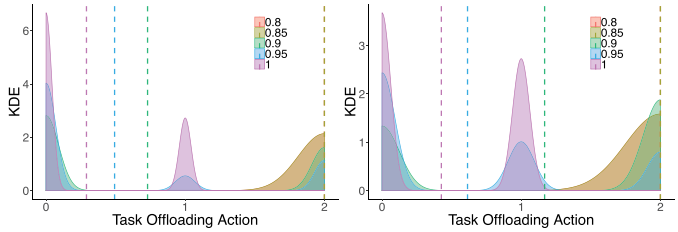
Compared with the first rows of two matrices, we can see that there is a high probability that the fog node workload would keep at state 0 at the next time slot if the fog node workload is state 0 at current time slot in Time Bucket II. However, the probability of the fog node workload state changing from 0 to 0 in Time Bucket I is lower than that in Time Bucket II. This is because the density of vehicular traffic is higher in Time Bucket I and the state of fog node workload would be more likely change to 1 in the future.

Fig. 9 illustrates the expected cumulative task utility in the two time buckets with $\phi = 0.9$. Compared with Fig. 9a and Fig. 9b, we can see that the expected cumulative task utility in Time Bucket II is larger than that in Time Bucket I. This is because the service demand of task offloading is lower in Time Bucket II and therefore the tasks have shorter processing time. Besides, Fig. 9 shows the optimal task offloading strategies in these two time buckets. The client vehicle would estimate the state probability distribution of the fog node workload based on the workload



(a) Time Bucket I (b) Time Bucket II

Fig. 10. Performance of various offloading strategies.



(a) Time Bucket I (b) Time Bucket II

Fig. 11. Distribution of task offloading actions with different task offloading strategies.

probability transition matrix and get the optimal task offloading actions by mapping the received workload observations to the task offloading strategies. For example, in Time Bucket I, if the initial state probability distribution of the fog node workload $\mathbf{p}_0 = \langle 0.8, 0.2 \rangle$, which means the probability of the fog node workload stays at state 0 is 0.8 and state 1 is 0.2, then the best task offloading action for the client vehicle is to offload the task with 720p image (Action: 2) to the infrastructural fog node. Notably, the same task offloading action will be chosen for the client vehicle when the probability of the fog node workload falls into the range from 0.75 to 0.82 (described as S9 in Figure 9a).

D. Comparison Between Task Offloading Strategies

For comparison, we implement two reference task offloading strategies – Random and Adaptive [10]. With Random strategy, the action of each task offloading are randomly chosen from the set $\mathcal{A} = \{0, 1, 2\}$, which means the task would be randomly decided to be processed locally or offloaded with a random resolution image without the awareness of fog node workload. According to our knowledge, the Adaptive strategy is the best solution that we can adopt for the comparison in our system model, which uses a probability distribution method to scale the offloading actions linearly with the number of processes running on the fog node. For instance, if the maximum number of tasks that can be simutaneously processed on a fog node is 10 and the number of tasks run on the fog node is 1, then the probabilities that the client vehicle offloads the task with 360p and 720p resolution are 10% and 90%, respectively.

Thus, we tune the value of latency sensitivity parameter ϕ from 0.8 to 1 to generate different POMDP based task offloading strategies. Fig. 10a illustrates the performance of different task offloading strategies in Time Bucket I. From the figure, we can see that the tasks achieve the highest image resolution approach with the POMDP offloading strategy with $\phi = 0.8$. More specifically, the POMDP based task offloading strategy with $\phi = 0.8$ increases the average resolution level of processed images by 66% and 83%, as compared with Random and Adaptive task offloading strategies, respectively. When $\phi = 1$, tasks would be offloaded without taking into account the image resolution benefits. In this case, the POMDP based task offloading strategy shortens the average latency by 65% and 58%, as compared with Random and Adaptive task offloading strategies, respectively. Meanwhile, the average resolution level of processed images decreases by 34% and 30%. This indicates that the POMDP based task offloading strategies can reduce the service latency or increase the resolution level of image processed depending on the specific demand of application. Moreover, the POMDP based task offloading strategy improves both latency and resolution metrics in case $\phi = 0.9$. Specifically, at this point, it reduces the average service latency by 14% and 10% while increasing the average resolution level of processed images by 2% and 17% compared with Random and Adaptive task offloading strategies, respectively. As shown in Fig. 10b, in Time Bucket II, the POMDP based task offloading strategies increase the average resolution level of processed images by 60% with $\phi = 0.8$ and shorten the average latency by 52% with $\phi = 1$ compared with Random task offloading strategy, which are less than the achievements in Time Bucket I. This indicates that the POMDP based task offloading strategies achieve better performance in reducing service latency and increasing resolution level of processed images when the vehicular traffic is more dense.

Fig. 11 compares the KDE of task offloading actions between different values of ϕ. We can see that when the sensitivity parameter is less than 0.85, the tasks are more likely to be offloaded with resolution set to 720p. With the increase of the latency sensitivity parameter, the tasks are more likely to be offloaded with resolution set to 360p. According to Fig. 11a and Fig. 11b, we can see that when $\phi = 0.9$ or $\phi = 0.95$, more tasks with 720p images are offloaded in Time Bucket II. This is because when the vehicular traffic is not congested, more computing resources of the fog nodes are available for the individual client vehicle and the client vehicle would be more likely to offload tasks.

In summary, the POMDP based task offloading strategies provided by Chameleon moderate the resolution level of images for task offloading, taking into account the variation in fog node workload. Chameleon could reduce the service latency or increase the resolution level of processed images according to the specific demands of assisted driving applications.

VII. Limitation and Future Work

In this section, we discuss the limitation of our work and present the future plan. In our simulation, we estimate the workload of the fog nodes by the vehicular traffic-load and this could cause the overestimation problem because not all vehicles are running visual applications in real life. However, it is hard to exactly estimate the number of client vehicles (i.e., service demand of the assisted driving applications). In the future work, we will study the development of assisted driving

applications and build a more finer-grained model to estimate their service demand. Besides, in this article, we treat the task as the basic unit and assume that tasks are independent from each other. However, in real-world assisted driving applications, the involved tasks may have various inter-logical relationships, and then influence the efficiency of task offloading. For the future work, we will make more effort to explore the dependency of the tasks and design a more efficient task offloading scheme, with taking into account the internal complexity of assisted driving applications.

In our experiment, we have limited the maximum number of tasks processed simultaneously at a fog node to 5, due to the memory capacity of the server. It is interesting to further confirm the performance of our system through experiment with servers having larger memory capacity. Furthermore, our task utility objective function takes into account only the latency and resolution parameters, while it provides the platform to integrate further performance criteria, such as mobility-aware caching into the optimization problem and to make more sophisticated task offloading models. It is also worth mentioning that some learning algorithms can be designed to predict the workload of the fog nodes in some particular regions (with known mobility trajectories) and hence alleviates the explicit communication between the vehicle and server before the task offloading. This will create an autonomous task offloading system for the next generation of vehicular networking.

From the system scalability point of view, the client vehicle makes the task offloading decisions by solving the local task utility optimization problem at the beginning of every time bucket after receiving the workload probability transition matrix of the selected fog node. Since in practice the workload status of the fog nodes at all time buckets are not known beforehand, the proposed task offloading approach makes the task offloading decisions in online manner which makes it suitable for practical implementation at large-scale VFC environments.

In reality, the assisted driving applications and vehicle manufacturers may prefer everything computed locally rather than offloading computation to edge. However, with the data aggregation and analytics in edge, the assisted driving applications will achieve collective perception for safety driving by offloading tasks to the fog nodes. Furthermore, with the data fusion in the backhaul network, the fog nodes connected to multiple networks can perceive the traffic situation in a larger range, which will benefit the route navigation applications involving traveling time prediction and congestion avoidance. In the future work, we will develop an information sharing and system coordination mechanism to enable "crowdsourcing" to assist individual vehicle to gain better understanding of complex traffic situation and driving environments.

In the proposed system, only the processing capability of the fog nodes is utilized, while, the edge caching [28] at the fog node can be also leveraged in order to further reduce the operational latency for the client vehicles. However, the main challenging issue would be designing efficient caching strategy that can intelligently recognize the similar processing tasks and cache them for the future access of the vehicles. Furthermore, the collaborative caching and processing among the neighboring fog nodes is expected to further improve the system performance although careful investigation is required to validate this expectation.

From the communication perspectives, the vehicular fog nodes in our system are assumed to be moving buses which commute on specified trajectories. As an alternative solution particularly in situations when there are no regular buses available in the vicinity of cars, the moving drones can act as the mobile fog node which establish the communication between the vehicles on the road and the cloud server. In order to ensure the autonomous and continuous task offloading service, the system design for drones should take into account the limitation of high energy consumption. Recent research proposes RF/wireless energy transfer or the renewable energy harvesting as the promising solutions to overcome the energy consumption limitation of the drones in 5G/Beyond 5G wireless cellular networks [29], [30].

VIII. Conclusion

In this paper, we propose Chameleon, a latency and resolution aware task offloading scheme for assisted driving applications. It aims at finding task offloading strategies to reduce service latency and increase the resolution level of processed images according to the specific demands of assisted driving applications. We approach the problem of seeking the optimal task offloading strategy via formulation of a POMDP, taking into account the variation of fog node workload, and solve it through stochastic dynamic programming technologies. Compared with previous works, our solution reduces service latency by up to 65% and increases the average resolution level of processed image by up to 83%.

References

[1] S. Li *et al.*, "Joint admission control and resource allocation in edge computing for Internet of Things," *IEEE Netw.*, vol. 32, no. 1, pp. 72–79, Jan. 2018.

[2] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *IEEE Trans. Veh. Technol.*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.

[3] M. Satyanarayanan, "Edge computing for situational awareness," in *Proc. IEEE Int. Symp. Local Metrop. Area Netw.*, Jun. 2017, pp. 1–6.

[4] J. Feng, Z. Liu, C. Wu, and Y. Ji, "AVE: Autonomous vehicular edge computing framework with ACO-based scheduling," *IEEE Trans. Veh. Technol.*, vol. 66, no. 12, pp. 10660–10675, Jun. 2017.

[5] Y. Xiao and C. Zhu, "Vehicular fog computing: Vision and challenges," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. Workshops*, Mar. 2017, pp. 6–9.

[6] C. Zhu, G. Pastor, Y. Xiao, Y. Li, and A. Ylä-Jääski, "Fog following me: Latency and quality balanced task allocation in vehicular fog computing," in *Proc. Annu. IEEE Int. Conf. Sens., Commun., Netw.*, Jun. 2018, pp. 1–9.

[7] "Five trends to small cell 2020," Oct. 2018. [Online]. Available: http://www-file.huawei.com/-/media/CORPORATE/PDF/News/Five-Trends-To-Small-Cell-2020-en.pdf

[8] C. Zhu, G. Pastor, Y. Xiao, and A. Ylä-Jääski, "Vehicular fog computing for video crowdsourcing: Applications, feasibility, and challenges," *IEEE Commun. Mag.*, vol. 56, no. 10, pp. 58–63, Oct. 2018.

[9] G. E. Monahan, "State-of-the-art—A survey of partially observable Markov decision processes: Theory, models, and algorithms," *Manage. Sci.*, vol. 28, no. 1, pp. 1–16, Jan. 1982.

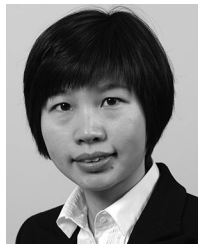
[10] C. Huang, Y. P. Fallah, R. Sengupta, and H. Krishnan, "Adaptive intervehicle communication control for cooperative safety systems," *IEEE Netw.*, vol. 24, no. 1, pp. 6–13, Jan./Feb. 2010.

[11] "Here API," Sep. 2018. [Online]. Available: https://developer.here.com/documentation/traffic
[12] S. Cao, X. Tao, Y. Hou, and Q. Cui, "An energy-optimal offloading algorithm of mobile computing based on HetNets," in *Proc. IEEE Int. Conf. Connected Veh. Expo*, Oct. 2015, pp. 254–258.
[13] M. Deng, H. Tian, and B. Fan, "Fine-granularity based application offloading policy in cloud-enhanced small cell networks," in *Proc. IEEE Int. Conf. Commun. Workshops*, May 2016, pp. 638–643.
[14] C. You, K. Huang, H. Chae, and B. Kim, "Energy-efficient resource allocation for mobile-edge computation offloading," *IEEE Trans. Wireless Commun.*, vol. 16, no. 3, pp. 1397–1411, Mar. 2017.
[15] O. Munoz, A. Pascual-Iserte, and J. Vidal, "Joint allocation of radio and computational resources in wireless application offloading," in *Proc. Future Netw. Mobile Summit*, Oct. 2013, pp. 1–10.
[16] M. Chen, Y. Qian, Y. Hao, Y. Li, and J. Song, "Data-driven computing and caching in 5G networks: Architecture and delay analysis," *IEEE Wireless Commun.*, vol. 25, no. 1, pp. 70–75, Feb. 2018.
[17] Y. Xiao and M. Krunz, "Distributed optimization for energy-efficient fog computing in the tactile Internet," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 11, pp. 2390–2400, Nov. 2018.
[18] C. Zhu *et al.*, "Folo: Latency and quality optimized task allocation in vehicular fog computing," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4150–4161, Jun. 2019.
[19] "HSL API," Sep. 2018. [Online]. Available: https://digitransit.fi/en/developers/apis/4-realtime-api/vehicle-positions
[20] D. Zhang and K. M. Wasserman, "Transmission schemes for time-varying wireless channels with partial state observations," in *Proc. 21st Annu. Joint Conf. IEEE Comput. Commun. Soc.*, Jun. 2002, pp. 467–476.
[21] N. Wang and J. Wu, "Opportunistic WiFi offloading in a vehicular environment: Waiting or downloading now?" in *Proc. 35th Annu. IEEE Int. Conf. Comput. Commun.*, Apr. 2016, pp. 1–9.
[22] A. R. Cassandra, M. L. Littman, and N. L. Zhang, "Incremental pruning: A simple, fast, exact method for partially observable Markov decision processes," in *Proc. Thirteenth Conf. Uncertainty Artif. Intell.*, Morgan Kaufmann Publishers Inc., 1997, pp. 54–61.
[23] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2017, pp. 7263–7271.
[24] A. Costa *et al.*, "Evaluating WiMAX for vehicular communication applications," in *Proc. IEEE Int. Conf. Emerg. Technol. Factory Autom.*, Sep. 2008, pp. 1185–1188.
[25] R. Sen, A. Cross, A. Vashistha, V. N. Padmanabhan, E. Cutrell, and W. Thies, "Accurate speed and density measurement for road traffic in India," in *Proc. 3rd ACM Symp. Comput. Develop.*, Jan. 2013, Art. no. 14.
[26] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO—Simulation of urban mobility," *Int. J. Adv. Syst. Meas.*, vol. 5, no. 3/4, pp. 128–138, Dec. 2012.
[27] L. Codeca, R. Frank, and T. Engel, "Luxembourg SUMO traffic (LuST) scenario: 24 hours of mobility for vehicular networking research," in *Proc. IEEE Veh. Netw. Conf.*, Dec. 2015, pp. 1–8.
[28] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Trans. Veh. Technol.*, vol. 67, no. 11, pp. 10190–10203, Nov. 2018.
[29] T. Long, M. Ozger, O. Cetinkaya, and O. B. Akan, "Energy neutral Internet of Drones," *IEEE Commun. Mag.*, vol. 56, no. 1, pp. 22–28, Jan. 2018.
[30] S. Sekander, H. Tabassum, and E. Hossain, "Multi-tier drone architecture for 5G/B5G cellular networks: Challenges, trends, and prospects," *IEEE Commun. Mag.*, vol. 56, no. 3, pp. 96–103, Mar. 2018.

Chao Zhu received the B.E. degree in computer science and technology from the Beijing University of Posts and Telecommunications, Beijing, China, in 2012, and the M.S. degree in computer technology from Tsinghua University, Beijing, China, in 2016. He is currently working toward the Ph.D. degree with the Department of Computer Science and the Department of Communications and Networking, Aalto University, Espoo, Finland. His research interests include mobile edge computing and vehicular networking.

Yi-Han Chiang received the Ph.D. degree from the Graduate Institute of Communication Engineering, National Taiwan University, Taipei, Taiwan, in 2017. He is currently a Project Assistant Professor with the National Institute of Informatics, Tokyo, Japan. His research interests include network greenness, wireless content caching, mobile edge computing, multi-armed bandits, and online and approximation algorithms.

Abbas Mehrabi received the B.Sc. degree in computer engineering from the Shahid Bahonar University of Kerman, Kerman, Iran, in 2008, the M.Sc. degree in computer engineering from Azad University, South Tehran, Iran, in 2010, and the Ph.D. degree from the School of Electrical Engineering and Computer Science, Gwangju Institute of Science and Technology, Gwangju, South Korea, in 2017. He is currently a Postdoctoral Researcher with the Department of Computer Science, Aalto University, Espoo, Finland. His research interests include quality of experience optimization and resource allocation in mobile edge computing environments, Internet of things, vehicular fog computing, and scheduling/planning problems in smart grids.



Yu Xiao received the doctoral degree in computer science from Aalto University, Espoo, Finland, in 2012. She is currently an Assistant Professor with the Department of Communications and Networking, Aalto University. Her current research interests include mobile crowdsensing, augmented reality, and edge computing.

Antti Ylä-jääski received the Ph.D. degree from ETH Zurich, Zürich, Switzerland, in 1993. He has been a Professor of telecommunications software with the Department of Computer Science, Aalto University, Espoo, Finland, since 2004. His current research interests include green ICT, mobile computing, and service architecture.

Yusheng Ji (M'94–SM'18) received the B.E., M.E., and D.E. degrees in electrical engineering from The University of Tokyo, Tokyo, Japan. She joined the National Center for Science Information Systems, Tokyo, Japan, in 1990. She is currently a Professor with the National Institute of Informatics, Tokyo, Japan, and Sokendai (The Graduate University for Advanced Studies), Hayama, Japan. Her research interests include network architecture, resource management in wireless networks, and mobile computing. She has served as a Board Member of Trustees of the Institute of Electronics, Information and Communication Engineers (IEICE), a Steering Committee Member of Quality Aware Internet SIG, an Expert Member of IEICE Technical Committees on Communication Quality, a Symposium Co-Chair of the IEEE GLOBECOM in 2012 and 2014, a Track Chair of IEEE Vehicular Technology Conference 2016 Fall and 2017 Fall, an Associate Editor for IEICE Transactions and IPSJ Journal. She is an Expert Member of the IEICE Technical Committees on Internet Architecture, a Steering Committee Member of Internet and Operation Technologies SIG of IPSJ, an Editor for the IEEE TRANSACTIONS OF VEHICULAR TECHNOLOGY, and a TPC member for IEEE International Conference on Computer Communications, IEEE International Conference on Communications, IEEE Global Communications Conference, IEEE Wireless Communications and Networking Conference, etc.