Accepted for publication in IEEE Trans. on Vehicular Technology, DOI: 10.1109/TVT.2019.2958740, For Personal use only

Soft-VAN: Mobility-Aware Task Offloading in Software-Defined Vehicular Network

Sudip Misra, *Senior Member, IEEE,* and Samaresh Bera, *Student Member, IEEE*

***Abstract*—In this paper, we propose a mobility-aware task offloading scheme, named as *Soft-VAN*, with an aim to minimize task computation delay in a software-defined vehicular network. The proposed scheme consists of two phases — fog node selection and task offloading. In the first phase, we formulate an integer linear program (ILP), and solve the problem to get *optimal* number of fog nodes required for a given network. In the task offloading phase, we formulate an optimization problem to minimize overall delay in task computation, while considering associated constraints. As finding *optimal* solution to the problem is NP-hard, we propose a greedy heuristic approach in two phases — task offloading and computed task downloading — to solve it in polynomial time. The greedy solution for offloading takes into account network delay, flow-rule capacity, and link utilization. On the other hand, the solution for computed task downloading considers vehicle's mobility in addition to the parameters associated with the offloading decisions. Experimental results show that the proposed scheme, Soft-VAN, is capable of enhancing the performance approximately by 30%, 45%, and 50% in terms of delay compared to state-of-the-art schemes — Detour, DAGP, and SD2O, respectively.**

***Index Terms*—Software-defined networks, VANET, Fog computing, Task offloading, Optimization**

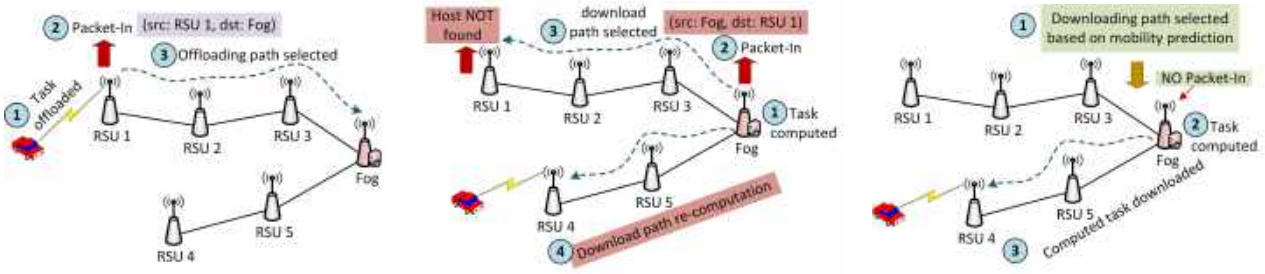
I. Introduction

The rapid growth in vehicular applications supported by next generation networks demands unprecedented network capacity and stringent quality of service (QoS) in connected vehicular networks [1]. To meet such requirements, vehicular networks should support advanced communication technologies and dynamic data forwarding mechanisms in real-time, in order to improve safety and efficiency, and reduce traffic congestion in the transportation system. Recent studies on edge computing suggest that vehicular tasks can be offloaded to nearby fog computing facilities (such as road-side units) to meet the stringent QoS requirements of vehicular applications with heavy computations and hard-deadlines [2], [3]. However, the current *best-effort* Internet technology limits the flexibility of dynamic decision making on task offloading [4]. To address the limitations, researchers proposed task offloading schemes in software-defined networks (SDN) [5], [6], in which a centralized controller takes task offloading decisions. Further, the software-defined approach allows the controller to manage the network in a flexible and simplified manner. Consequently, in this work, we focus on software-defined task offloading in a vehicular network in the context of fog computing. Further, the compute, storage and networking resources in the network are deployed at a few *selected* road-side units (RSUs), known as fog nodes, while providing abstracted global view of the entire network.

In SDN, rule-based packet forwarding is followed at the devices in contrast to the traditional networks. The existing SDN-based approaches [5], [6] considered the delay involved in task offloading from an end-device to a fog node, and overlooked the complexity involved in downloading the computed task from the fog node to end-device from the context of SDN. Further, presence of mobile devices in the network is a crucial factor to be considered while offloading tasks to fog nodes. Figure 1 depicts a software-defined task offloading scenario and associated complexities/issues present in a vehicular network. As shown in Figure 1(a), a vehicle offloads its task to nearby RSU. On receiving the offloading request, the RSU generates a `Packet-In` to the controller with packet meta-data [7]. According to the `Packet-In`, the controller selects the forwarding path and places flow-rules at the associated RSUs/fog nodes. Once the task is computed at the selected fog node, it generates another `Packet-In` to the controller with packet meta-data, as depicted in Figure 1(b). Similar to the offloading, the controller decides the path for downloading the computed task and places flow-rules. However, due to the mobility, the vehicle is no longer associated with the previous RSU (RSU 1 in this case, refer to Fig. 1(b)). As a result, RSU 1 generates another message to the controller as 'Host Not Found'. Finally, the controller decides the correct path based on vehicle's association to RSU, as depicted in Figure 1(b), Step 4. Consequently, the computed task experiences additional control overhead and delay, which may violate required QoS of the application. In contrast, as depicted in Figure 1(c), the issues with downloading the computed task can be avoided by placing flow-rules in a proactive manner, while considering vehicle's mobility in the network.

Motivated by the above mentioned facts, in this work, we propose a mobility-aware task offloading scheme in software-defined vehicular networks. The proposed scheme consists of two phases — fog node selection and task computation. In the first phase, we formulate an ILP to select optimal number of RSUs required to be selected a fog nodes for a given network. In the second phase, we formulate a non-linear optimization problem to minimize delay in task computation, while considering associated constraints. As the task computation problem is NP-hard to solve, we divide the problem into two sub-problems — task offloading and computed task downloading. We propose two greedy-heuristic approaches for task

S. Misra and S. Bera are with the Computer Science and Engineering Department, Indian Institute of Technology, Kharagpur, 721302, India, Email: smisra@sit.iitkgp.ac.in, s.bera.1989@ieee.org



(a) Task offloading in SDN (b) Existing Work: computed task downloading (c) Proposed Work: computed task downloading

Fig. 1: Motivating scenario: a) task offloading in software-defined vehicular network; b) Issues due to mobility of vehicles and reactive approach; c) Proactive scheme (proposed)

offloading and computed task downloading, while considering vehicles' mobility in the network. Extensive experiment results show that the proposed task offloading scheme outperforms the existing schemes in terms of delay and control overhead in the network. In brief, the contributions are as follows:

- We formulate an ILP-based optimization problem to select optimal number of fog nodes for a given network, in order to minimize capital expenditure and operational cost, i.e., CAPEX and OPEX, in the network.
- We formulate a non-linear optimization problem to minimize task computation delay, while considering associated constraints. As the problem is NP-hard, we propose greedy heuristic approaches — delay minimization in task offloading and computed task downloading — to solve the problem.
- Experimental results show that the proposed scheme is capable of minimizing task computation delay by 30%, 45%, and 50% compared to existing schemes — Detour [6], DAGP [5], and SD2O [8], respectively.

The rest of the paper is organized as follows. Section II discusses existing works on task offloading and their limitations. Section III presents the selection of fog nodes and detailed system model for a given network. Section IV presents the optimization problem for task offloading in software-defined vehicular network. Section V presents the proposed task offloading scheme to minimize the task computation delay. The performance evaluation of the proposed scheme is presented in Section VI. Finally, Section VII concludes the work with some future research directions.

II. Related Work

In this section, we discuss state-of-the-art works, which focused on task offloading in order to minimize task computation delay [1], [5], [6], [8]–[16]. Recently, Tan and Hu [1] proposed an edge caching and computing scheme, while considering vehicle's mobility. They showed that task offloading to the edge devices is useful in a vehicular network in order to minimize associated cost. Yousefpour et al. [10] studied the task offloading problem while considering benefits of inter-fog communication and load sharing to minimize service delay. Similarly, Jiang and Tsang [11] proposed delay-aware task offloading scheme in a shared fog network. The authors considered three types of tasks – delay-sensitive, delay-tolerant, and delay-insensitive. The delay-sensitive tasks are prioritized over the delay-tolerant and delay-insensitive tasks.

Huang et al. [12] proposed a mobile edge computing (MEC) framework for SDN-based LTE networks. The proposed framework abstracts *radio API* to collect network information, such as topology, bandwidth, and signal strength, from radio access network (RAN). Tran and Pompili [13] studied task offloading and resource allocation as a joint optimization problem. The authors formulated two sub-problems – task offloading and resource allocation – in order to minimize overall service delay. Huang et al. [15] studied a vehicle-to-vehicle (V2V) data offloading scheme while utilizing the benefits of SDN. In such a scheme, the centralized SDN controller manages end-to-end connection between two communicating vehicles. Further, Li et al. [16] proposed a mobile edge computing framework based on SDN to improve scalability and response time of the network. Chen and Hao [5] proposed a task offloading scheme in software-defined mobile edge computing network, in order to minimize task computation delay. Recently, Misra and Saha [6] proposed a similar approach for task offloading in software-defined fog network. In contrast to the other works, the authors considered that a fog node may be situated at multi-hop distance from end-device.

Synthesis: Table I summarizes the existing works. Detailed analysis of the existing works reveals that they either considered one-hop distance between end-device and fog node or overlooked the impact of mobility of the end-devices, while offloading tasks in a software-defined network. Further, delay involved in downloading the computed task from fog node to vehicle in a multi-hop scenario is an important concern to be considered, in order to avoid additional control overhead and delay, as depicted in Figure 1. Consequently, we propose a task offloading scheme in software-defined vehicular network, while considering multi-hop path and impact of mobility of the vehicles.

III. System Model

We consider a software-defined architecture of fog-based vehicular network consisting of a sets of RSUs, fog nodes, and vehicles, as depicted in Figure 2. The vehicular network is modeled as a directed graph $\mathcal{G} = (\mathcal{N}, \mathcal{L})$, where $\mathcal{N}$ and $\mathcal{L}$ denote the set of all RSUs and links between the RSUs, respectively. Further, the set of RSUs is denoted as $\mathcal{N} =$

TABLE I: Differences between the proposed and state-of-the-art works

Work	Energy	Delay-offload	Delay-download	Multi-hop	Mobility	SDN	Flow-rule
Tan and Hu [1]	✓	✗	✗	✗	✓	✗	✗
Yousefpour et al. [10], Jiang and Tsang [11]	✓	✗	✗	✗	✗	✗	✗
Tran et al. [13]	✓	✓	✗	✗	✗	✗	✗
Huang et al. [12], Chen and Hao [5]	✓	✓	✗	✗	✗	✓	✗
Li et al. [16], Huang et al. [15]	✗	✓	✗	✓	✓	✓	✗
Misra and Saha [6]	✓	✓	✗	✓	✗	✓	✓
Soft-VAN (Proposed scheme)	✓	✓	✓	✓	✓	✓	✓

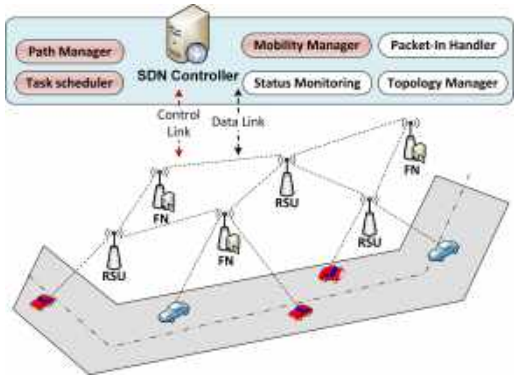


Fig. 2: Software-defined architecture of fog-based VANET

$\{N_1, N_2, \ldots, N_n\}, n \in \mathbb{Z}^+$. The set of links is represented as $\mathcal{L} = \{(i,j), \forall i, \forall j \in \mathcal{N}, i \neq j\}$. Further, the set of vehicles in the network is represented as $\mathcal{V} = \{v_1, v_2, \ldots, v_n\}, n \in \mathbb{Z}^+$. In the subsequent sections, we discuss the selection of fog nodes and detailed system model considered in this work.

A. Selection of Fog Nodes

For a given network $\mathcal{G}$, our objective is to select *optimal* number of fog nodes, which is denoted by a set $\mathcal{F}$, from the set of all RSUs $\mathcal{N}$. We consider hop-count to select *optimal* number of fog nodes in the network. In other words, for dense fog node deployment, hop-count between an RSU and at least one fog node can be considered as one. Whereas for sparse deployment of fog nodes, hop-count between fog node and RSU is large. Accordingly, we formulate an ILP to select *optimal* number of RSUs that need to be configured as fog node as follows:

$$\text{Minimize} \sum_{j=1}^{|\mathcal{N}|} w_j$$

$$\text{subject to} \sum_{j} \beta_{i,j}^{h} \geq 1, \forall i \in \mathcal{N}, \tag{1}$$

$$\beta_{i,j}^{h} \leq w_j, \forall i \text{ and } j \in \mathcal{N} \tag{2}$$

where w_j denotes whether j^{th} RSU is selected as a fog node. Therefore, the optimization problem determines minimum number of RSUs required to be deployed as fog nodes in the network. In other words, the optimization problem expresses the selection of optimal number of RSUs that need to be deployed as fog nodes in the network, which is unknown a priori. Equation (1) ensures that there exists at least one fog node for each RSU within the desired hop-count h[1]. $\beta_{i,j}^{h}$ captures the availability of such fog node from an RSU. Mathematically,

$$\beta_{i,j}^{h} = \begin{cases} 1, & \text{if there exists a fog node} \\ & \text{from RSU within hop-count } h \\ 0, & \text{Otherwise} \end{cases} \tag{3}$$

Equation (2) restricts on the number of selected fog nodes in order to minimize the overall objective function. We solve this optimization problem using existing ILP solver[2]. Therefore, the fog nodes are the subset of RSUs having additional task processing capacity. Consequently, the set of fog nodes is represented as $\mathcal{F} = \{F_1, F_2, \ldots, F_m\}$, $m \leq n$. We denote the set of *general* RSUs as $\mathcal{R} = \{R_1, R_2, \ldots, R_{|\mathcal{N}|-|\mathcal{F}|}\}$.

It is noteworthy that a vehicle can compute its task locally or can offload to nearby RSU, while considering associated delay and energy consumption. A task, θ, associated with a vehicle $v \in \mathcal{V}$, is represented as a tuple of task size, s_v, and CPU cycles required to compute the task, w_v. Mathematically, a task is represented as $\theta = \{s_v, w_v\}$. Therefore, the set of tasks from all vehicles is represented as $\Theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$, $n \in \mathbb{Z}^+$. Similarly, a computed task is denoted as $\theta^{'} = \{s_v^{'}\}$, where $s_v^{'}$ denotes the data size after computation. In subsequent sections, we discuss the delay and energy consumption associated with local computation and offloading a task.

B. Delay Model

The delay associated with task computation is calculated as the delay involved in local computation and offloading, as described below.

1) Delay in Local Computation: Delay associated with local computation is mathematically represented as $\Delta_{\theta}^{loc} = \frac{w_v}{\zeta_v^{loc}}$, where ζ_v^{loc} denotes the task computation capacity of a vehicle $v \in \mathcal{V}$.

2) Delay in Offloading: In contrast to the local computation, delay in task offloading depends on delay involved in task uploading and downloading the computed task. Further, delay in uploading a task comprises of transmission delay, propagation delay, queuing delay, and processing delay. On the other hand, the delay involved in downloading a computed task comprises of propagation delay and transmission delay.

[1]The value of h is user-defined and application-specific.
[2]https://www.gnu.org/software/glpk/

We present each of these delay models associated in task offloading.

A vehicle transmits the task to nearby RSU/fog. Therefore, the transmission delay for task offloading is calculated as $\Delta_\theta^{tran} = \frac{s_v}{r_{v,i}}, v \in \mathcal{V}, i \in \mathcal{N}$, where $r_{v,i}$ denotes the upload data rate between vehicle v and RSU/fog i. Mathematically, the upload data rate is calculated as $r_{v,i} = Blog_2(1 + p_v g_{v,i}/(\sigma^2 + I_{v,i}))$, where B denotes the channel bandwidth, and p_v and σ^2 denote the transmission power and noise power of the vehicle, respectively. The symbols $g_{v,i}$ and $I_{v,i}$ denote the channel gain and interference power between the vehicle and RSU/fog, respectively. Each offloaded task experiences a propagation delay from RSU to fog node at which the task is finally computed. Mathematically, it is represented as $\Delta_\theta^{prop} = \sum_{(i,j)\in\mathcal{L}} \delta_{i,j} x_{i,j}^\theta$, where $\delta_{i,j}$ denotes the propagation delay of a link $(i,j) \in \mathcal{L}$, and $x_{i,j}^\theta$ denotes whether link (i,j) is selected for offloading task θ. Further, offloaded tasks, $\theta \in \Theta$, arrive at each fog node following different paths. Therefore, queuing delay at a fog node depends on task arrival and execution rates of the fog node. It is noteworthy that a task can be offloaded from a vehicle and an intermediate RSU. Therefore, the task arrival rate at a particular fog node is modeled as Poisson process [17]. Consequently, task arrival rate at a fog node $j \in \mathcal{F}$ is mathematically denoted as $\alpha_j = \frac{1}{T}\sum_{\theta\in\Theta}\sum_{i\in\mathcal{R}} x_{i,j}^\theta$, where T denotes the total time period, and x_j^θ is a binary variable used to denote whether fog j is selected to serve task θ. Considering multi-threaded model, the queuing delay experienced by the task θ at the fog node j is mathematically represented as $\Delta_\theta^{que} = \frac{1}{\epsilon_j - \alpha_j}$ [18], where ϵ_j and α_j denote the task execution and arrival rate at the fog node, respectively. Processing delay experienced by the task θ at the fog node depends on the size of the task and the processing capacity of the fog node. Mathematically, it is denoted as $\Delta_\theta^{proc} = \frac{\sum_\theta s_v x_j^\theta}{\zeta_j}$, where ζ_j denotes the processing capacity of the fog node.

Similarly, each computed task experiences propagation and transmission delay. The propagation delay, $\Delta_{\theta'}^{prop}$, depends on the link delay through which the computed task is sent back to the vehicle. On the other hand, transmission delay between vehicle and edge RSU/fog, to which the vehicle is associated with, depends on the transmission power of the RSU/fog. It is noteworthy that the transmission delay for computed task downloading is negligible as transmission power and channel bandwidth is high for downloading [5]. However, the transmission delay for task offloading is considered. Therefore, the total delay involved in task offloading is represented as $\Delta_\theta^{off} = (\Delta_\theta^{que} + \Delta_\theta^{tran} + \Delta_\theta^{prop} + \Delta_\theta^{proc} + \Delta_{\theta'}^{prop})$.

C. Energy Consumption Model

We adopt the energy consumption of vehicles for task offloading from the existing works [5], [13]. Energy consumption for local computation depends on the CPU cycles required to complete the task. Mathematically, it can be represented as $E_v^{loc} = \sigma_v w_v$, where σ_v denotes the power coefficient of the vehicle $v \in \mathcal{V}$. On the other hand, energy consumption of a vehicle related to task offloading depends on the transmission power, p_v, task size, s_v, and upload data rate between the vehicle and RSU or fog, $r_{v,i}$, and it is represented as $E_v^{off} = \frac{p_v s_v}{r_{v,i}}, i \in \mathcal{N}$.

IV. Problem Formulation

The objective is to select *optimal* fog node to offload a task, so that overall delay associated with task offloading and computed task downloading is minimized. The overall optimization problem is mathematically formulated as follows:

$$\underset{y_\theta, x_{i,j}^\theta, x_j^\theta, x_{i,j}^{\theta'}}{\text{Minimize}} \sum_{\theta\in\Theta} \Big[\overbrace{y_\theta \Delta_\theta^{off}}^{\text{task offloaded}} + \overbrace{(1-y_\theta)\Delta_\theta^{loc}}^{\text{task computed locally}} \Big]$$

s. t.

$$\sum_{\theta\in\Theta} w_v x_{i,j}^\theta + \sum_{\theta'\in\Theta'} s_v^{'} x_{i,j}^{\theta'} \le C_{i,j}, \forall (i,j) \in \mathcal{L}, v \in \mathcal{V} \quad (4)$$

$$\sum_{\theta\in\Theta} s_v x_j^\theta \le \zeta_j, \forall j \in \mathcal{F}, v \in \mathcal{V} \quad (5)$$

$$\sum_{\theta\in\Theta} r_i^\theta + \sum_{\theta'\in\Theta'} r_i^{\theta'} \le R_i^{max}, \forall i \in \mathcal{N} \quad (6)$$

$$(1-y_\theta)E_v^{loc} < E_v^{avl} \text{ and } y_\theta E_v^{off} < E_v^{avl} \quad (7)$$

In the optimization problem, Equation (4) ensures that the link utilization is always less than or equal to the link capacity. Equation (5) denotes that the assigned tasks to a fog node for processing is always within the capacity of the fog node. Equation (6) ensures that number of flow-rules present at the RSUs and fog nodes is always less than or equal to their maximum flow-rule capacity. Further, Equation (7) denotes that the energy consumed in local computation and offloading is always less than the available energy of the vehicle. y_θ is a binary variable used to denote whether a task is computed locally or offloaded to a fog node. Mathematically,

$$y_\theta = \begin{cases} 1, & \text{if task is offloaded} \\ 0, & \text{if task is computed locally} \end{cases}$$

The above mentioned delay optimization problem is a non-linear programming problem, and it is NP-hard [19].

V. Proposed Task Offloading Scheme

To solve the problem in polynomial time, we divide the optimization problem into two sub-problems — task offloading and computed task downloading. In the task offloading problem, we propose a greedy heuristic approach to decide whether to offload the task and to select *optimal* fog node if the task is offloaded. On the other hand, we propose another greedy heuristic approach for selecting the path for downloading computed task, while considering vehicle's mobility and associated RSU.

We redefine the decision variable y_θ to decide whether to offload a task as follows:

$$y_\theta = \begin{cases} 1, & \text{if } \frac{\Delta_\theta^{off} - \Delta_\theta^{loc}}{\Delta_\theta^{off}} \le 0 \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

where y_θ is always 1 when the delay in local computation is more than the delay in offloading. Further, we define a cost

function for selecting a fog node to which the task has to be offloaded as follows:

$$C_{\theta,j}^{fog} = \alpha \Delta_j^{que} + (1-\alpha)\frac{w_v}{\zeta_j} \tag{9}$$

where α is a predefined constant to denote relative importance of queuing delay and processing delay. In this work for experiment, we give equal importance on queuing delay and processing delay, i.e., α = 0.5. We define another cost function to select path through which the task has to be offloaded to the selected fog node as follows:

$$C_{i,j}^{link} = \beta^{prop}\frac{\delta_{i,j}}{\max\limits_{i,j}\delta_{i,j}} + \beta^{link}\frac{L_{i,j}^{util}}{L_{i,j}} + \beta^{rule}\frac{R_j^{util}}{R_j^{max}} \tag{10}$$

where $L_{i,j}^{util}$ and R_j^{util} denote the link- and rule utilization, and $L_{i,j}$ and R_j^{max} denote the link capacity and maximum rule capacity, respectively. Constants β^{prop}, β^{link}, and β^{rule} denote the relative importance of propagation delay, link utilization, and rule utilization, respectively. It is noteworthy that the propagation delay, link utilization, and rule utilization are normalized, in order to get the forwarding path cost. We also give equal importance on these constants. Algorithm 1

Algorithm 1 Algorithm for task offloading

Inputs: Set of RSUs, $\mathcal{R}$; Set of fog nodes, $\mathcal{F}$; Set of links $\mathcal{L}$, Set of link-delays, $\mathcal{D}$

Output: Selection of fog node $j \in \mathcal{F}$ for offloading task $\theta \in \Theta$

```
for task θ ∈ Θ do
    Calculate y_θ according to Equation (8)
    if y_θ == 1 then
        for fog node j ∈ F do
            Calculate C^fog_{θ,j} according to Equation (9)
        Select fog node k = arg min_j C^fog_{θ,j}
        Calculate path P to offload task θ to fog node k
         according to Equation (10)
    else
        Compute task θ locally
```

presents the proposed greedy algorithm for task offloading.

For downloading the computed task from fog node to vehicle, *optimal* path is required to be selected, in order to minimize the propagation delay, as mentioned in the optimization problem. Further, vehicle's mobility also needs to be considered, while selecting the downloading path. Therefore, the SDN controller decides the downloading path from fog node to the associated RSU[3], after predicting location of the vehicle. In our previous work [20], we showed that order-k Markov predictor is useful to predict locations of mobile nodes, in order to control end-device and access-point association in a centralized manner. We adopt a similar approach to predict locations of vehicles in the network, which is briefly discussed in this paper.

The location predictor calculates the probability of hand-off that will occur in the next Δ time period, where $\Delta = (\Delta_\theta^{prop} + \Delta_\theta^{que} + \Delta_\theta^{proc})$. Further, the predictor calculates conditional probability that the vehicle will move to a location s within Δ time after the current elapsed time t. Consequently, for a given location context c and elapsed time t, the probability of each vehicle moving to each possible location s within Δ time is calculated as follow:

$$P(s|c,t) = P(s)P_s(t \le z < t+\Delta|c,t) \tag{11}$$

where $P(s)$ is the transition probability of every possible next location s, which can be calculated as follows:

$$P(s_{t+\Delta} = s|\mathcal{H}) \approx \hat{P}(s_{t+\Delta} = s|\mathcal{H}) = \frac{N(cs,\mathcal{H})}{N(c,\mathcal{H})} \tag{12}$$

where $N(cs,\mathcal{H})$ denotes the number of occurrences of cs in the movement history set $\mathcal{H}$. Accordingly, the Markov predictor predicts the most likely location s will be visited at $t+\Delta$ time as follows:

$$s_{t+\Delta} = \arg\max_{s\in\mathcal{S}}(P(s_{t+\Delta} = s)) \tag{13}$$

The SDN controller decides the RSU to be associated with the vehicle based on the predicted location $s_{t+\Delta}$. It is noteworthy that the association between RSU and vehicle can be controlled in a centralized manner using SDN framework [21]. Therefore, the controller determines path for downloading computed task from the selected fog node (refer to Algorithm 1) to the associated RSU at $t+\Delta$ time. Algorithm 2 presents the proposed downloading path selection scheme. It is also noteworthy that both the Algorithms 1 and 2 are executed for all vehicles, i.e., for all tasks, in the network.

Algorithm 2 Algorithm for selecting downloading path

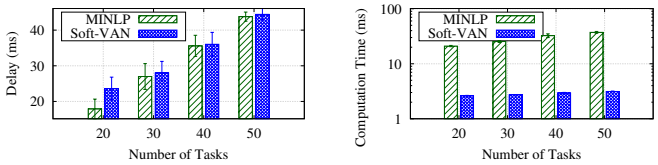
Inputs: Movement history set, $\mathcal{H}$; Set of RSUs, $\mathcal{R}$; Set of fog nodes, $\mathcal{F}$; Set of links, $\mathcal{L}$

Output: Selection of downloading path from fog node $f \in \mathcal{F}$ to associated RSU $R \in \mathcal{R}$

```
Calculate probability of next possible locations s according to Equation (11)
Predict next location s_{t+Δ} according to Equation (13)
Select an RSU R ∈ R to be associated with the vehicle based on s_{t+Δ}
Calculate path between fog node f ∈ F and the RSU R using Equation (10)
```

Time complexity of the proposed scheme is analyzed by two steps — complexity in task offloading and downloading. Algorithm 1 includes selection of fog nodes, which runs in $O(|\mathcal{F}|)$ time, and path selection for offloading based on Dijkstra's shortest path principle, which runs in $O(|\mathcal{R}|+|\mathcal{F}|^2)$. Therefore, Algorithm 1 runs in $O(|\mathcal{F}| + (|\mathcal{R}| + |\mathcal{F}|^2)) \approx O(|\mathcal{R}| + |\mathcal{F}|^2)$. On the other hand, Algorithm 2 includes location prediction and path selection. The order-k Markov predictor runs in $O(k^2)$, where k is the number of input sequences of locations of vehicles. Similar to Algorithm 1, path selection runs in $O(|\mathcal{R}| + |\mathcal{F}|^2)$. Therefore, the running time complexity of the proposed scheme is $O(|\mathcal{R}| + |\mathcal{F}|^2 + k^2)$, where value of k is user-dependent. We solve the optimization problem formulated in Section IV with small number of tasks using

[3]The vehicle is associated with an RSU.



(a) Average delay (b) Execution time

Fig. 3: Comparison between MINLP and proposed greedy approach

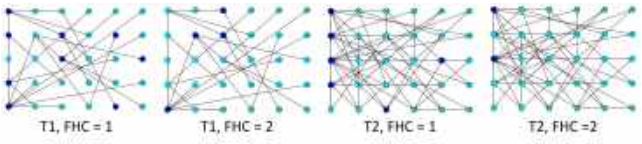


Fig. 4: Network topology and fog node selection using different hop-count (fog nodes in blue color)

APMonitor-GEKKO (http://apmonitor.com/). Figure 3 shows the comparison between the MINLP problem and proposed greedy approach for task offloading scheme. We see that the proposed greedy method yields competitive results compared to MINLP, while having very small computation time.

VI. Performance Evaluation

We evaluate the performance of the proposed scheme using python-based simulations. The performance is carried out on a Intel i5 2.2 GHz PC with 4GB RAM and running Linux kernel 4.15. Table II shows the parameters and their values used in the experiment. To create network topology consisting of RSU, we use scale-free Barabasi-Albert topology [22], as topology traces of software-defined vehicular network are difficult to obtain. The RSUs and vehicles are deployed within the area in a uniform random manner. Vehicles move in the network following the Gauss-Markov mobility model, as considered in our previous work [20]. Figure 4 shows the topology and deployment of fog nodes based on the optimization problem formulated in Section III-A. We consider two different topologies and vary the hop-count to fog node from any RSU as 1 and 2. Accordingly, we present the results for four different scenarios — T1-FHC = 1, T1-FHC = 2, T2-FHC = 1, and T2-FHC = 2, as presented in Section VI-A. Further, we plot the results using 95% confidence interval.

We consider three existing schemes — Detour [6], DAGP [5], SD2O [8] — to show the effectiveness of the proposed scheme. In all schemes, Soft-VAN, Detour, DAGP, and SD2O, fog nodes are selected based on the fog node selection optimization problem, and the topologies are considered, as depicted in Figure 4. In Detour [6], DAGP [5], and SD2O [8], the offloading decisions are taken based on associated delay in uploading the tasks to a fog node. Further, Detour follows a utility function to take offloading decisions. Therefore, in Detour, the fog node with the highest utility is selected to serve the offloaded tasks. On the other hand, DAGP follows hop-count to fog nodes in order to take offloading decisions. Therefore, in DAGP, fog node with the lowest hop-count is selected to serve the offloaded tasks. In SD2O, the fog node with the lowest delay, i.e., with the highest utility, is selected to offload the task irrespective of the offloading path. In contrast, the proposed scheme, Soft-VAN, takes offloading decisions based on delay associated to uploading the tasks to a fog node and downloading the computed tasks from the fog node, while predicting vehicles' locations in the network. Further, Soft-VAN also considers the cost function (refer to Equations (9) and (10)), while selecting a fog node to serve the offloaded tasks. In rest of the work, we use Soft-VAN, Detour, DAGP, and SD2O to denote the proposed scheme and existing schemes, respectively. It is also noteworthy that the predicted location may not be always true in real-time, which, in turn, results increased delay and control overhead compared to the predicted ones. We limit our discussion on the impact of wrong location prediction in this work. Interested readers may refer to Mobi-Flow [20].

TABLE II: Simulation parameters

Parameter	Value
Number of total RSUs	30
Number of tasks	200 – 1500
Vehicle CPU frequency	10 – 30 MHz
Vehicle transmit power	20 dBm [13]
Fog CPU frequency	2.9 – 4.2 GHz [6]
Task computation amount	1500 – 2500 megacycles [13]
Average task size	450 KB [13]
Channel noise power	-100 dB [13]
Channel bandwidth	20 MHz [13]

A. Results and Discussion

We consider three different performance metrics — delay, energy consumption, and `Packet-In` — to show the results.

1) Delay: We measure the reduction in delay per task while offloading a task compared to local computation. Figure 5 shows the reduction in delay using the proposed scheme, Soft-VAN, compared to the existing schemes, Detour, DAGP, and SD2O. It is evident that the Soft-VAN is capable to reducing task computation delay approximately by 30%, 45%, and 50% compared to Detour, DAGP, and SD2O, respectively. In Soft-VAN, the SDN controller takes offloading decisions based on delay associated to uploading and downloading the task, link capacity, rule capacity of the RSUs, and locations of vehicles. On the other hand, Detour, DAGP, and SD2O did not consider the delay associated to downloading the computation task, while taking offloading decisions. Further, locations of vehicles are not considered, which leads to incorrect forwarding of the computation task, as discussed in Section I. Consequently, the existing schemes, Detour, DAGP, and SD2O, yield degraded performance compared to the proposed scheme, Soft-VAN. It is also noteworthy that the performance of the proposed scheme is slightly degraded with an increase in the number of tasks in the network. This is due to the fact the more number of tasks are offloaded to the fog nodes, which, in turn, increases queuing delay in the network. However, it is always better than the existing schemes. Further, we see that significant

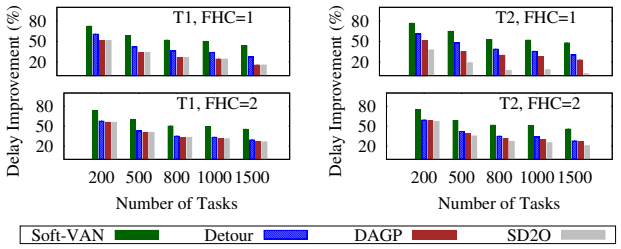


Fig. 5: Reduction in delay with different tasks

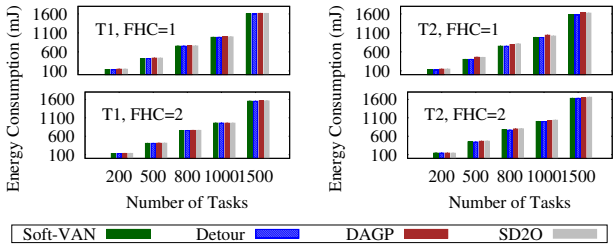


Fig. 6: Energy consumption per vehicle with different tasks

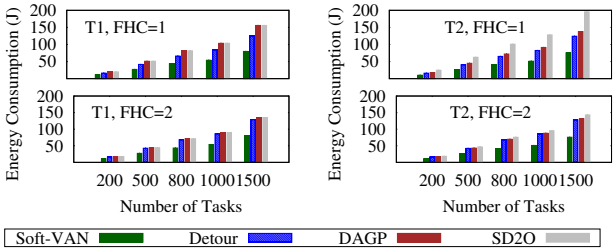


Fig. 7: Energy consumption per RSU with different tasks

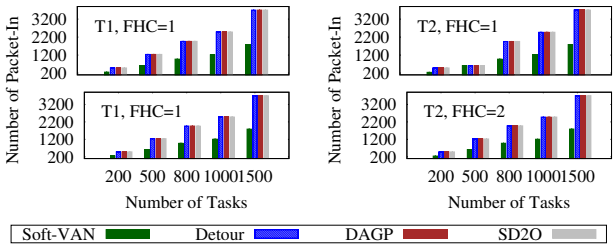


Fig. 8: Number of `Packet-In` with different tasks

performance degradation in terms of delay with increasing number of hops to fog nodes for all the schemes. This is also due to the fact that the queuing delay at the fog nodes increases with decreasing number of fog nodes in the network. Similar to number of tasks, the proposed scheme, Soft-VAN, yields better performance in terms of delay compared to Detour, DAGP, and SD2O with different network topologies.

2) Energy Consumption: Energy consumption of the vehicles in the network is also taken into consideration, while taking decisions on whether to offload or to locally compute the task. Accordingly, we measure the average energy consumption per vehicle in the network using Soft-VAN, Detour, DAGP, and SD2O schemes, as depicted in Figure 6. We see the all the schemes are equally competitive in term of vehicle's energy consumption with different number of tasks. This is due to the fact that all the schemes prefer to offload the task instead of local computation. Consequently, in all schemes, vehicle's transmission energy is consumed to offload the task.

To show the effects of SDN, we also measure the energy consumption at the RSUs. We consider energy consumption at RSUs for `Packet-In`, `rule-matching`, and `Actions` based on Open vSwitch [23]. We do not consider the base energy consumption as it is the same for all schemes. Further, energy consumption for flow-table lookup is considered from the study by Congdon et al. [24]. Figure 7 shows average energy consumption per RSU in the network. In contrast to Figure 6, we see that the proposed scheme, Soft-VAN, is capable of minimizing energy consumption at RSUs approximately by 30%, 40%, 42% compared to the existing schemes, Detour, DAGP, and SD2O, respectively. In Soft-VAN, the task offloading and downloading decisions are taken by the controller, while considering link delay, bandwidth, flow-rule utilization of RSUs, and locations of the vehicles. Accordingly, fog nodes are selected to serve the offloaded task. On the other hand, Detour did not consider the locations of vehicles that leads to incorrect forwarding path, which, in turn, increases the energy consumption at RSUs. In contrast, DAGP only considered delay to offload the task, due to which it incurs more energy consumption at RSUs compared to Soft-VAN and Detour. Similarly, SD2O also considered delay of the fog nodes in order to take offloading decisions. Therefore, SD2O incurs more energy consumption compared to Soft-VAN and Detour. We also see that energy consumption at RSUs increases with increasing number of tasks, as more number of tasks need to be forwarded and processed at the RSUs and fog nodes, respectively.

3) Packet-In Message: Control overhead is a major concern in SDN while placing flow-rules at the forwarding devices. Consequently, we measure the number of `Packet-In` messages generated by the RSUs/fog nodes to the controller, as depicted in Figure 8. It is also evident that the proposed scheme, Soft-VAN, is capable of minimizing number of `Packet-In` in the network approximately by 50% compared to the existing schemes, Detour, DAGP, and SD2O. In Soft-VAN, the SDN controller proactively places flow-rules at the RSUs/fog nodes, while predicting locations of vehicles in the network. On the other hand, in Detour, DAGP, and SD2O, the controller places flow-rules at RSUs/fog nodes in reactive manner. Further, due to the change in locations of the vehicles, incorrect forwarding path leads to more `Packet-In` to the controller. As a result, the existing schemes, Detour, DAGP, and SD2O, generate more number of `Packet-In` compared to the proposed scheme, Soft-VAN. It is also noteworthy that the Soft-VAN incurs a few additional `Packet-In` due to wrong location prediction (refer to [20]). However, it is always better than the existing schemes.

B. Use-Case Scenario: Practical Aspects

We discuss a use-case scenario of the proposed scheme from practical aspects of smart grid. In smart grid, plug-in electric vehicles (PHEVs) play a key role in real-time energy management [25]. The PHEVs can charge/discharge their batteries to reduce imbalance in energy supply-demand in the grid. In order to take the charging/discharging decisions, the PHEVs exchange real-time information with the grid which needs to be computed in real-time, in order to take the decisions. However, due to the resource-constrained nature of the vehicles, computational data can be offloaded to access devices in the smart grid network to minimize the delay. In such a scenario, the proposed task offloading scheme is beneficial to minimize the task computation delay for PHEVs, while considering the vehicles' mobility in the network.

VII. Conclusion

In this paper, we proposed a task offloading scheme in software-defined vehicular network, while considering mobility of vehicles in the network. A centralized controller takes offloading decisions from three different aspects — whether to offload, where to offload, and path to offload the task, while utilizing global view of the network. We formulated an optimization problem and solve it to get *optimal* number of fog nodes to be deployed in the network. Further, we proposed a greedy heuristic-based solution approach to take offloading decisions while considering locations of vehicles, as finding global optimization is NP-hard. Experimental results showed that the proposed scheme is capable of reducing task computation delay compared to the existing schemes, while minimizing the control overhead in the network.

In this work, we considered that the fog nodes are determined during network deployment stage and they remain static in the network. However, selection of fog nodes can be done dynamically in real-time in the presence of mobile fog nodes in addition to the vehicles in the network. Therefore, we plan to study the task offloading scheme in the presence of mobile fog nodes as a future extension of the work. Further, in the proposed scheme, we did not consider vehicle-to-vehicle (V2V) communication that may be useful to minimize the energy consumption of vehicles and overall delay in task-offloading. Therefore, we also plan to study the impact of V2V communication on task-offloading in vehicular network.

References

[1] L. T. Tan and R. Q. Hu, "Mobility-aware edge caching and computing in vehicle networks: A deep reinforcement learning," *IEEE Trans. on Vehicular Technology*, vol. 67, no. 11, pp. 10 190–10 203, Nov. 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. on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, Jun. 2016.

[3] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Comm. Surveys & Tutorials*, vol. 19, no. 3, pp. 1628–1656, 2017.

[4] B. A. A. Nunes, M. Mendonca, X.-N. Nguyen, K. Obraczka, and T. Turletti, "A survey of software-defined networking: past, present, and future of programmable networks," *IEEE Comm. Surveys & Tutorials*, vol. 16, no. 3, pp. 1617–1634, 2014.

[5] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE J. on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, Mar. 2018.

[6] S. Misra and N. Saha, "Detour: dynamic task offloading in software-defined fog for IoT applications," *IEEE J. on Selected Areas in Communications*, vol. 37, no. 5, pp. 1159–1166, May 2019.

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. R. andScott Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," in *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, Apr. 2008, pp. 69–74.

[8] G. S. Aujla, R. Chaudhary, N. Kumar, J. J. P. C. Rodrigues, and A. Vinel, "Data offloading in 5G-enabled software-defined vehicular networks: A stackelberg game-based approach," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 100–108, 2017.

[9] J. Liu, J. Wan, B. Zeng, Q. Wang, H. Song, and M. Qiu, "A scalable and quick-response software defined vehicular network assisted by mobile edge computing," *IEEE Communications Magazine*, vol. 55, no. 7, pp. 94–100, 2017.

[10] A. Yousefpour, G. Ishigaki, R. Gour, and J. P. Jue, "On reducing IoT service delay via fog offloading," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 998–1010, Apr. 2018.

[11] Y. Jiang and D. H. K. Tsang, "Delay-aware task offloading in shared fog networks," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4945–4956, Dec. 2018.

[12] A. Huang, N. Nikaein, T. Stenbock, A. Ksentini, and C. Bonnet, "Low latency MEC framework for SDN-based LTE/LTE-A networks," in *Proc. of the IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[13] T. X. Tran and D. Pompili, "Joint task offloading and resource allocation for multi-server mobile-edge computing networks," *IEEE Trans. on Vehicular Technology*, vol. 68, no. 1, pp. 856–868, Jan. 2019.

[14] K. Wang, H. Yin, W. Quan, and G. Min, "Enabling collaborative edge computing for software defined vehicular networks," *IEEE Network*, vol. 32, no. 5, pp. 112–117, 2018.

[15] C.-M. Huang, M.-S. Chiang, D.-T. Dao, W.-L. Su, S. Xu, and H. Zhou, "V2V data offloading for cellular network based on the software defined network (SDN) inside mobile edge computing (MEC) architecture," *IEEE Access*, vol. 6, pp. 17 741 – –17 755, 2018.

[16] M. Li, P. Si, and Y. Zhang, "Delay-tolerant data traffic to software-defined vehicular networks with mobile edge computing in smart city," *IEEE Trans. on Vehicular Technology*, vol. 67, no. 10, pp. 9073–9086, 2018.

[17] D. P. Bertsekas and R. G. Gallager, *Data networks*. Englewood Cliffs, New Jersey 07632: Prentice Hall, 1992.

[18] D. Gross, J. F. Shortle, J. M. Thompson, and C. M. Harris, *Fundamentals of queueing theory*. Wiley, 2008.

[19] Y. Pochet and L. A. Wolsey, *Production planning by mixed integer programming*. Springer, 2006.

[20] S. Bera, S. Misra, and M. S. Obaidat, "Mobi-Flow: mobility-aware adaptive flow-rule placement in software-defined access Network," *IEEE Trans. on Mobile Computing*, vol. 18, no. 8, pp. 1931–1842, 2019.

[21] L. Suresh, J. Schulz-Zander, R. Merz, A. Feldmann, and T. Vazao, "Towards programmable enterprise WLANS with Odin," in *Proc. of the ACM Workshop on HotSDN*, Helsinki, Finland, Aug. 2012, pp. 115–120.

[22] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, no. 5439, pp. 509–512, 1999.

[23] F. Kaup, S. Melnikowitsch, and D. Hausheer, "Measuring and modeling the power consumption of OpenFlow switches," in *Proc. of International Conference on Network and Service Management (CNSM) and Workshop*, Rio de Janeiro, Brazil, Nov. 2014.

[24] P. T. Congdon, P. Mohapatra, M. Farrens, and V. Akella, "Simultaneously reducing latency and power consumption in OpenFlow switches," *IEEE/ACM Trans. on Networking*, vol. 22, no. 3, pp. 1007–1020, Jun. 2014.

[25] S. Misra, S. Bera, and T. Ojha, "D2P: distributed dynamic pricing policy in smart grid for PHEVs management," *IEEE Trans. on Parallel & Distributed Systems*, vol. 26, no. 3, pp. 702–712, 2015.