MobiPlace: Mobility-Aware Controller Placement in Software-Defined Vehicular Networks

Ilora Maity, Student Member, IEEE, Ravi Dhiman, and Sudip Misra, Senior Member, IEEE

***Abstract*—In this paper, we propose a mobility-aware scheme, named MobiPlace, to address the controller placement problem (CPP) at the Road Side Units (RSUs) in Software-Defined Vehicular Networks (SDVNs). MobiPlace places local controllers at the selected RSUs to reduce the operational delay experienced in traditional SDVN architecture, where controllers are placed at the cloud. Additionally, we consider the effect of dynamic road traffic and propose dynamic adjustment of the placement of the controller with minimal changes. In contrast to the existing literature, we infuse traffic monitoring and traffic prediction strategies to ensure accurate delivery of control messages and data packets. We formulate an Integer Linear Program (ILP) and propose a solution approach consisting of three modules — *mobility management*, *controller placement*, and *controller selection*. The mobility management module uses Markov predictor to predict vehicle movement and reduce controller synchronization overhead when a vehicle moves to a different controller's coverage area. The controller placement module applies a simulated annealing-based algorithm to select potential RSUs that serve as local controllers. The controller selection module determines the preferable controller location for processing the service requests. Simulation results depict that MobiPlace reduces the average flow setup delay by 13.85% compared to the existing state-of-the-art.**

***Index terms*—** SDN, SDVN, Controller Placement, Simulated Annealing, Markov Predictor.

I. Introduction

Vehicular ad-hoc networks (VANETs) are evolving charismatically together with the evolution of SDN [24] [3], which adds flexibility and programmability to VANETs [26] [12]. In conventional SDVNs, the control plane is located at the cloud server. The centralized control plane manages Vehicle-to-Everything (V2X) communications. Some of the recent works [34] [36] [13] in SDVNs suggest placing local controllers at the RSU level to reduce communication latency. However, the location and number of local controllers affect the end-to-end processing delay of V2X communications, which are primarily latency-sensitive [11]. Moreover, road traffic conditions are highly dynamic, involving vehicles with high mobility. Therefore, there exists a need for a mobility-aware controller placement scheme for SDVNs, which considers the effects of traffic variation and vehicle mobility.

SDVN is a promising network architecture to support the Intelligent Transportation System (ITS) [6], which includes several smart applications [8] [4], including collision avoidance, emergency notification, detection of a traffic rule violation, dynamic speed limit, and toll collection. However, the data demand for ITS applications is thriving, and future SDVN architecture should be scalable to address this upsurge of traffic data [20]. In SDVN, the control plane is the primary module that takes routing decisions. Therefore, the optimization of the SDVN control plane enhances the scalability of the overall network. For network scalability, a distributed control plane with multiple controllers is preferable to a centralized control plane. However, the problem of controller placement in SDVN is different from the same in traditional SDN, because of the inherent limitations of vehicular networks such as latency-sensitivity of V2X communications, frequent traffic variations, and high mobility of vehicles [29].

ITS applications related to traffic safety are latency-sensitive, and high flow setup delay violates the Quality of Service (QoS) demands of the latency-sensitive applications [33] [28]. Sudheera *et al.* [34] proposed a secondary control plane at the RSU level to reduce the flow setup delay. However, the authors did not consider the effects of frequent traffic variations in vehicular networks. Toufga *et al.* [36] proposed a controller placement strategy that reckons the traffic variations and places local controllers at the RSU level. However, the change in controller placement is inefficient if a traffic variation lasts for a very short duration. On the other hand, high-speed vehicles frequently move from one RSU's coverage area to another. Controllers need to anticipate this change of location for the correct delivery of Flow-Mod messages and proper route generation for V2X communications. In another work, Kaur *et al.* [18] proposed a controller placement scheme for load balancing and energy management in SDN-based Internet of Autonomous Vehicles (SD-IoAV). However, the proposed method does not consider dynamic data traffic, which is evident in ITS due to the vehicles' high mobility.

In contrast to the existing state-of-the-art, in this work, we monitor traffic for a pre-defined duration to avoid unnecessary changes in the placement of local controllers. Additionally, we propose a mobility prediction module to pre-determine the vehicles' locations for smooth transmission of Flow-Mod messages and data packets. Mobility prediction also helps in proactive synchronization among controllers involved with the nodes in a communication path. Proactive controller synchronization allows proactive flow-rule installation to all nodes in the routing path and reduces flow setup delay.

In this work, we propose a scheme, named MobiPlace, for mobility-aware controller placement in SDVN, to place local controllers at selected RSUs to optimize the flow setup delay, by considering varying traffic conditions. The primary *contributions* of this work are as follows:

- We formulate an ILP to represent the local controller

Ilora Maity and Sudip Misra are with the Department of Computer Science and Engineering, Indian Institute of Technology Kharagpur, India (Email: imaity@iitkgp.ac.in; smisra@cse.iitkgp.ac.in).

Ravi Dhiman is with the Advanced Technology Development Centre, Indian Institute of Technology Kharagpur, India (Email: rd@iitkgp.ac.in).

For personal use only

placement problem.
- We design an algorithm to solve the ILP and assign RSUs to the local controllers with minimal changes in the existing controller placement status.
- We propose a Markov predictor-based mobility management approach to reduce the data loss due to vehicle mobility.

II. Related Work

In this section, we review the existing literature on SDVN, SDN controller placement in static networks, and SDN controller placement in mobile networks [11] [23].

A. SDVN

Several research works utilize the flexible network management option provided by SDN for V2X communication. Sadio *et al.* [29] designed a fog-based SDVN framework and proposed a routing strategy for Vehicle-to-Infrastructure (V2I) and Vehicle-to-Vehicle (V2V) communications. Misra and Bera [26] proposed a task offloading scheme for SDVN. In this approach, OBUs can offload tasks to selected fog-enabled RSUs. The proposed method considers the impact of vehicle mobility to reduce the fog-to-OBU task download delay. Ghafoor *et al.* [15] proposed a routing algorithm for the Software-Defined Internet of Vehicles (SDIoV) that routes the data traffic based on link connectivity and reliability. Liu *et al.* [21] designed a fog-enabled data scheduling framework for V2X communications in an SDVN. The authors construct a graph considering vehicle mobility, coverage of the transmitting nodes, and heterogeneous transmission rates. Based on the constructed graph, the authors propose a Clique Searching based Scheduling (CSS) algorithm for data dissemination. Dalgkitsis *et al.* [10] proposed a deep learning-based approach that predicts vehicle mobility and relocates services accordingly to minimize latency. However, high training time affects the QoS requirements of the traffic flows.

B. SDN Controller Placement in Static Networks

Heller *et al.* [16] coined CPP and formulated an optimization problem considering the average-case control path latency, the worst-case control path latency, and the maximum number of switches assigned to each controller. Hock *et al.* [17] proposed a resiliency-aware controller placement that aims to balance control plane load and minimize latency if any controller fails. Ksentini *et al.* [19] proposed a bargaining game-based controller placement approach to address the trade-off between control plane load, switch-to-controller latency, and inter-controller latency. Huque *et al.* [37] proposed a controller placement technique for large-scale sparse and dense network. The proposed approach aims to increase controller utilization.

C. SDN Controller Placement in Mobile Networks

Llerena and Gondim [22] presented a queueing theory-based analysis on the controller response time for Device-to-Device (D2D) communication in an SDN-enabled Long Term Evolution (LTE) network. The authors proposed an Ant Colony Optimization (ACO)-based algorithm to generate an optimal solution to the CPP that minimizes the control traffic response time. Kaur *et al.* [18] proposed an energy-aware controller placement scheme for SD-IoAV. The authors designed a greedy heuristic approach that finds a locally optimal solution based on the SDN switches' energy consumption. However, the proposed method does not consider dynamic data traffic generated by moving vehicles. Sudheera *et al.* [34] selected RSUs to place local controllers to reduce control traffic latency. The architecture proposed by the authors also includes global controllers placed at the cloud. In this work, the authors modeled the CPP as a p-median facility location problem. However, this approach fails to address the variant traffic load in vehicular networks. Therefore, as an extension of the work proposed by Sudheera *et al.*, Toufga *et al.* [36] proposed a dynamic controller placement strategy that changes the placement of the controllers according to change in traffic conditions. Additionally, the authors introduced a replacement cost metric to estimate the overhead due to incremental controller placement. However, this approach does not consider the inter-controller communication cost due to handover between controllers when a vehicle moves to the coverage area of a different controller.

Synthesis. We infer that the existing works on controller placement in SDVN do not consider the overhead due to synchronization between controllers when a vehicle selects a new controller. Moreover, frequent changes in controller-switch assignments cause service disruption. Therefore, in this work, we propose a controller placement scheme that performs proactive controller synchronization and minimizes the number of placement changes considering the varying traffic conditions. Additionally, we use a prediction-based approach to address service disruption due to vehicle mobility. The proposed mobility prediction approach is different from existing mobility prediction approaches because it predicts a vehicle's location after a specific interval when a service request is ready to be propagated to the vehicle.

III. System Architecture

A. Network Planes

In contrast to the traditional SDVN architecture consisting of data and control planes only, we consider an additional plane, named *hybrid plane*, that includes both the data and control elements. We introduce the hybrid plane to place local controllers and reduce the flow setup delay. Figure 1 shows the SDVN architecture considered for MobiPlace.

1) Data Plane: The data plane consists of vehicles equipped with Global Positioning System (GPS) and On Board Units (OBUs). An OBU enables a vehicle to communicate with an RSU using Dedicated Short Range Communication (DSRC) and with a Base Station (BS) using LTE. In this work, we consider the vehicles as a type of SDN switches. Therefore, the OBU stores forwarding decisions for V2V communications in the form of flows-rules. Each OBU broadcasts the current location of the vehicle periodically. Let V be the set of vehicles. In this work, we assume that each vehicle moves with a uniform random velocity [31]. Additionally, we consider a two-way highway, where vehicles move in both directions.

For personal use only

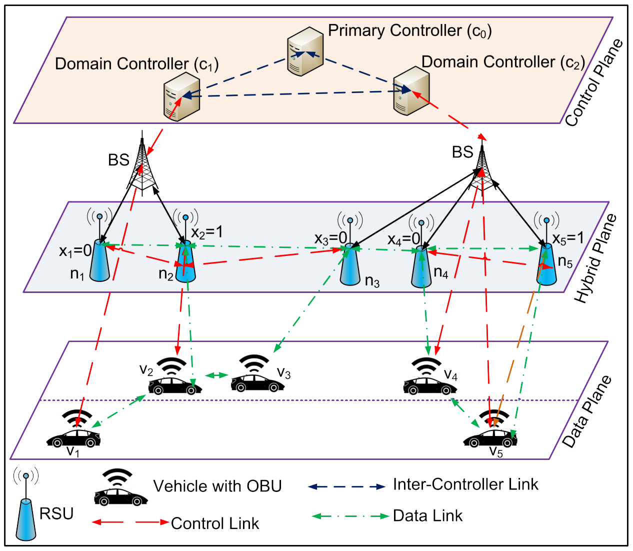

Fig. 1: SDVN Architecture

2) Hybrid Plane: The hybrid plane consists of RSUs that serve as SDN switches. RSUs are fog-based entities that store flow-rules to support V2I communications. In this work, we do not consider any specific placement strategy of the RSUs. We consider that some RSUs serve as local controllers to other RSUs and vehicles. We term an RSU that serves as a local controller as *controller-RSU (C-RSU)*. The deployment of C-RSUs reduces flow setup time for the vehicles and other non-controlling RSUs by bringing the control plane closer to the data plane. Moreover, communication with a controller located at the cloud is costly as the cellular spectrum is licensed. Let N denote the set of RSUs placed in the network. We define a binary variable to express placement of local controllers as:

$$x_j = \begin{cases} 1 & \text{if } n_j \in N \text{ is C-RSU,} \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

3) Control Plane: The control plane consists of multiple domain controllers and a primary controller c_0. A domain controller maintains global network view of a specific region. Let C be the set of domain controllers. RSUs and OBUs communicate with a domain controller via a cellular BS. The association between a C-RSU n_j and an RSU n_k is given by:

$$y_{jk} = \begin{cases} 1 & \text{if } n_k \text{ is assigned to } n_j, \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

B. Traffic Model

In SDVN, vehicles request different services such as real-time traffic information, parking space availability, road congestion information, and road accident notification. Each service request corresponds to a data flow with a source and destination. The source is the requesting vehicle itself. However, the destination is either an RSU or another vehicle. The set of data flows is denoted by F. Let D_a^{max} denote the maximum allowable delay for data flow $f_a \in F$.

C. Communication Model

1) V2V Communication: Each vehicle v_i can send data flow to another, which in the communication range of v_i. Therefore, using multi-hop V2V communication, a vehicle can communicate with a distant vehicle without involving the network infrastructure such as RSUs, C-RSUs, and BS. V2V communications use DSRC-based transmission, which has a low data transmission rate. Let r_v denote the data transmission rate of V2V communications.

2) V2I Communication: V2I communication involves vehicle to RSU communication and vehicle to domain controller communication. We assume that vehicle to RSU communication use existing DSRC wireless technology and vehicle to domain controller communication use LTE. Let r_f and r_c denote the data transmission rate for a vehicle to RSU/C-RSU communication and vehicle to domain controller communication, respectively. A vehicle communicates directly with the domain controller if no RSU or C-RSU is present within the vehicle's coverage area.

For simplicity, we assume a uniform data transmission rate for all vehicles. However, in practice, the data transmission rate varies with the vehicle's transmission power, noise power, and interference power [26].

D. Delay Model

If the route to the destination of a data flow is known to the OBU, the latter transmits the data flow. Otherwise, it offloads the request to a C-RSU or a domain controller by sending a Packet-In request [1]. We estimate the flow setup delay as the interval between when an OBU generates a Packet-In message and when the source OBU receives the Flow-Mod message. We define a binary variable to denote whether a Packet-In request for data flow f_a is processed by a domain controller or by a C-RSU. Mathematically,

$$\alpha(f_a) = \begin{cases} 1 & \text{if } f_a \text{ is processed by a C-RSU,} \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

1) Flow Setup Delay for OBU to C-RSU Communication: The flow setup delay for processing a Packet-In request at the hybrid plane comprises of six components:

- The uplink transmission delay from OBU to an RSU within the coverage of the OBU
- The propagation delay from the RSU to a C-RSU
- The queueing delay at the C-RSU
- The processing delay at the C-RSU
- The propagation delay from the C-RSU to the OBU
- The downlink transmission delay for downloading the new flow-rule at the OBU

The uplink transmission delay for transmitting a Packet-In message with size b from v_i to an RSU n_k is $D_{tran}^{local} = \frac{b}{r_f}$. The propagation delay from n_k to a C-RSU n_j with $y_{jk}(t) = 1$ is $D_{prop}^{local} = \sum_{e_{ab} \in E_{kj}} \delta_{ab}$, where E_{kj} is the set of links that forms the shortest path from n_k to n_j, and δ_{ab} denotes the propagation delay of the link $e_{ab} \in E_{kj}$. A C-RSU receives a control message from either an OBU or an RSU. Therefore, we consider the arrival rate of control messages at the C-RSU as a Poisson process [25] and model the queue of the C-RSU n_j as an $M/M/1$ queue. Let λ_j and μ denote the arrival rate and service rate of control messages at n_k, respectively. Hence, the queueing delay at the C-RSU n_j is $D_{que}^{local} = \frac{1}{\mu - \lambda_j}$.

For personal use only

The processing delay at n_j is $D^{local}_{proc} = \frac{\omega_i}{\Omega_j}$, where Ω_j is the processing capacity of C-RSU n_j and ω_i is the number of CPU cycles required for processing of the Packet-In message. After processing the Packet-In message, the C-RSU generates a Flow-Mod message that consists of the new flow-rule. Let $b^{'}$ be the size of the Flow-Mod message. The C-RSU sends the Flow-Mod message to the requesting OBU in vehicle v_i. Let n_l be the RSU that covers v_i at present. If $n_k \neq n_l$ and n_l is assigned to a C-RSU other than n_j, n_j synchronizes with the C-RSU associated with n_l. The propagation delay for the downlink communication from the C-RSU to the OBU is $D^{local}_{prop^{'}} = \sum_{e_{ab} \in E^{'}_{jl}} \delta_{ab}$, where $E^{'}_{jl}$ is the set of links that forms the shortest path from n_j to n_l. The downlink transmission delay is negligible because of the high transmission power and channel bandwidth. Therefore, the flow setup delay for processing a control message generated by an OBU at a C-RSU is $D^u = D^{local}_{tran} + D^{local}_{prop} + D^{local}_{que} + D^{local}_{proc} + D^{local}_{prop^{'}}$.

2) Flow Setup Delay for OBU to Domain Controller Communication: An OBU can send the control message directly to the domain controller of the corresponding region via a cellular link. However, cellular communication is expensive than DSRC communication. We model the queue of a domain controller placed at the cloud as a $M/M/1$ queue with infinite queue length. The flow setup delay for processing a control message generated by v_i at c_p is $D^c = D^{domain}_{tran} + D^{domain}_{prop} + D^{domain}_{que} + D^{domain}_{proc} + D^{domain}_{prop^{'}}$, where $D^{domain}_{tran} = \frac{b}{r_c}$, $D^{domain}_{prop} = \delta^{'}$, $D^{domain}_{que} = \frac{1}{\mu^{'} - \lambda^{'}_p}$, $D^{domain}_{proc} = \frac{\omega_i}{\Omega^{'}_p}$, $D^{domain}_{prop^{'}} = \delta^{''}$, $\delta^{'}$ is the propagation delay for V2C communication before processing of the control message, and $\delta^{''}$ denotes the propagation delay for V2C communication after processing of the control message. Therefore, the flow setup delay for a flow f_a is $D(f_a) = \alpha(f_a) D^u + (1 - \alpha(f_a)) D^c$.

***Definition* 1** (Hybrid Plane State)**.** *The state of the hybrid plane is defined as $\zeta = \{x, y\}$.*

The hybrid plane state is modified based on the changes in road traffic conditions. Accordingly, some C-RSUs change back to normal RSUs acting as SDN switches, and some RSUs may be upgraded to C-RSUs. However, newly placed C-RSUs need to synchronize with other C-RSUs for acquiring the global network view required to serve as a local controller. Therefore, significant changes in the hybrid plane state compared to the previous hybrid plane state are not desirable.

***Definition* 2** (State Migration Overhead)**.** *The overhead for the migration from a hybrid plane state $\zeta^{'}$ to another state ζ defined as $\beta(\zeta^{'}, \zeta) = \sum_{n_j \in N} x_j(1 - x^{'}_j) + x^{'}_j(1 - x_j)$.*

IV. Problem Formulation

This work aims to place a minimum number of C-RSUs in the hybrid plane to minimize the flow setup delay as well as the state migration overhead. Minimizing the number of C-RSUs reduces the Operation Expenditure (OPEX) of the network and the cost of synchronization between the C-RSUs. However, fewer C-RSUs increases the load on individual C-RSUs. Consequently, we formulate the CPP for SDVN as:

$$\underset{\zeta,\alpha}{\text{Minimize}} \quad \sum_{n_j \in N} x_j + \beta(\zeta^{'}, \zeta) \tag{4}$$

subject to

$$\lambda_j < \mu_j, \text{ where } x_j(t) = 1, \forall n_j \in N, \tag{5}$$

$$\lambda_p < \mu_p, \forall c_p \in C, \tag{6}$$

$$\sum_{n_j \in N} y_{jk} = 1, \forall n_k \in N, \tag{7}$$

$$D(f_a) \leq D^{max}_a, \forall f_a \in F, \tag{8}$$

where $\zeta^{'}$ is the previous state of the hybrid plane. Equations (5) and (6) state the controller capacity constraint for C-RSUs and domain controllers, respectively. Equation (7) ensures that each RSU is assigned to precisely one controller. Equation (8) ensures that the flow setup delay for each flow does not exceed the maximum allowable delay. The optimization problem in Equation (4) is a ILP and computationally expensive for large-scale networks. Therefore, we propose a heuristic algorithm to place the C-RSUs in the hybrid plane dynamically.

V. MobiPlace: The Proposed Scheme

MobiPlace consists of three modules — *mobility management*, *controller placement*, and *controller selection*. The mobility management module reduces the flow setup delay and the controller synchronization cost by using a mobility prediction-based approach. The controller placement module dynamically generates the set of C-RSUs based on changes in the road traffic volume. Finally, based on the current controller placement status, the controller selection module selects the preferable controller for processing the control messages.

A. Mobility Management

For addressing a service request, a vehicle contacts a C-RSU through the nearest RSU within the coverage of the vehicle. The corresponding C-RSU returns the Flow-Mod messages to the requesting vehicle. However, due to mobility, the vehicle may move near to a different RSU. If the C-RSU is not aware of this change, the previous RSU receives the Flow-Mod message. However, this RSU cannot find the vehicle in its coverage area and reports a 'Host Not Found' error. Adopted from the work of Misra and Bera [26], we use an order-m Markov predictor to determine the RSU that should receive the Flow-Mod message. Markov predictor is a popular location predictor that predicts the future location of mobile entities based on m most recent location data. We use the Markov predictor because it consumes less space and performs better than other location predictors [27] for low values of m [32].

Each vehicle periodically broadcasts its location [7] to the nearby RSUs. The C-RSUs collect this data and sends it to the domain controller regularly. The domain controller uses Markov predictor to compute the next RSU that connects with the vehicle and the time of the change in the connected RSU.

For each vehicle v_i, the input to the Markov predictor is $H^i_h = \{L^i, E^i, W^i, P^i\}$, where $L^i = \{l^i_1, l^i_2, \ldots, l^i_h\}$ is the set of locations, $E^i = \{e^i_1, e^i_2, \ldots, e^i_h\}$ is the set of arrival time at each location, $W^i = \{w^i_1, w^i_2, \ldots, w^i_h\}$ is the set of durations

For personal use only

of stay at each location, and P^i is the set of transition probabilities from one location to another. The context for an order-m Markov predictor is $g = \{l^i_{h-m+1}, l^i_{h-m+2}, \ldots, l^i_{h-1}, l^i_h\}$.

For each data flow f_a generated by a vehicle v_i, the Markov predictor calculates the probability that a v_i moves to location l within Δ time after the current elapsed time $\tau^{'}$. The value of Δ is given by:

$$\Delta = \alpha(f_a)\left(D^{local}_{prop} + D^{local}_{que} + D^{local}_{proc}\right) + (1-\alpha(f_a))\left(D^{domain}_{prop} + D^{domain}_{que} + D^{domain}_{proc}\right) \quad (9)$$

For a given context g and the current elapsed time $\tau^{'}$, the probability that v_i moves to l within Δ time is given by:

$$P(l|g,\tau^{'}) = P(l)P_l(\tau^{'} \le w < \tau^{'} + \Delta|g,\tau^{'}), \quad (10)$$

where $P(l)$ is the transition probability for the next location l and it is estimated as:

$$P(l^i_{h+1} = l|L^i) \approx \widehat{P}(l^i_{h+1} = l|L^i) = \frac{\gamma(gl, L^i)}{\gamma(g, L^i)}, \quad (11)$$

where $\gamma(gl, L^i)$ signifies the number of occurrences of gl in the set L^i. Therefore, the output of the Markov Predictor which is the most likely next location of v_i is given by:

$$l^i_{h+1} = \underset{l \in L^i}{\operatorname{argmax}}\, P(l^i_{h+1} = l) \quad (12)$$

If $\gamma(g, L^i) = 0$, the $O(m)$ Markov predictor fails to return a result. Therefore, we use fallback Markov predictor, which backtracks to an $O(m-1)$ Markov predictor whenever an $O(m)$ Markov predictor fails to return any value. The $O(0)$ Markov predictor yields the location that occurs most frequently in the location history set L^i. Finally, the domain controller selects the RSU nearest to l^i_{h+1} for transmitting the Flow-Mod message. The location prediction also initiates proactive controller synchronization required for RSUs/OBUs in the computed flow-path that belongs to different controllers. Accordingly, flow-rules are installed proactively at all RSUs/OBUs at the flow-path. We select proactive rule installation to reduce the end-to-end delay.

B. Controller Placement

We consider that each RSU has a *traffic monitor*, which estimates the traffic volume for τ seconds based on the received requests from OBUs. The observation duration τ should not be very less because a change in controller placement is not feasible for traffic variation that lasts for a brief duration. Let ξ^k_{cur} denote the current traffic volume at RSU r_k. If ξ_{cur} is less than the minimum threshold ξ_{min} or higher than the maximum threshold ξ_{max}, the current hybrid plane state is modified by changing the placement of C-RSUs. The minimum and maximum traffic volume thresholds are defined as $\xi_{min} = Z_1\tau\mu$ and $\xi_{max} = Z_2\tau\mu$, where $Z_1, Z_2 \in [0,1]$ are pre-defined constants and $Z_1 < Z_2$. For an estimated traffic volume, the minimum number of C-RSUs required is calculated as $\Gamma = \left\lceil \frac{\sum_{n_j \in N} \xi_{cur}}{\tau\mu} \right\rceil$. The C-RSU placement module computes the optimal hybrid plane state using a Simulated Annealing (SA)-based approach based on the current hybrid plane state and the road traffic data. SA is a meta-heuristic algorithm which generates locally optimal solution in limited iterations [30]. In contrast to other optimization techniques such as hill climbing and gradient descent, SA does not get trapped in local optima because less accurate solutions are accepted based on the acceptance probability. Therefore, SA is used for a wide range of optimization problems, including control engineering, signal processing, and production scheduling [2]. The optimization problem defined in Equation (4) has a high probability of getting stuck at local optima. This is because a minimal number of controllers may address all the service requests with less controller placement overhead. However, the global optima may be considerably different from the local optima in terms of the average flow setup delay. Therefore, we use SA to achieve a more accurate solution for the CPP formulated in Equation (4). For the placement of C-RSU, we assign priority values to each RSU, including the existing C-RSUs. The priority of an RSU r_k depends on the following parameters:

1) **Location**: RSUs placed at popular locations such as road intersections are potential candidates to serve as C-RSUs as more vehicles can communicate with the C-RSUs in this case. Let $u_k \in [0,1]$ denote the popularity of the location of r_k.
2) **Traffic Volume**: The priority of an RSU r_k is directly proportional to the estimated traffic volume ξ^k_{cur}. This is because high traffic volume generates high control traffic that should be processed locally to avoid communication with the domain controllers placed at the cloud.
3) **Number of Neighbors**: Placing the local controller module at an RSU with a high number of neighbors ensures that a minimum number of C-RSU manages all RSUs. Let q_k denote the number of neighbors of n_k.
4) **Duration of Stay**: Let W_k denote the average duration a vehicle remains connected to the RSU r_k before connecting to a different RSU. This information is available from the mobility data that the vehicles broadcast periodically. An RSU with low W_k value is an inferior choice as a C-RSU location because it increases the controller synchronization overhead.

***Definition* 3** (RSU Priority)**.** *The priority of an RSU r_k is defined as $\Psi_k = u_k + \frac{\xi^k_{cur}}{\sum_{n_j \in N} \xi^j_{cur}} + \frac{q_k}{|N|} + \frac{W_k}{\tau}$.*

***Definition* 4** (Controller Placement Cost)**.** *Given a hybrid plane state ζ, the cost of controller placement strategy is defined as:*

$$cost(\zeta) = \frac{\beta(\zeta^{'}, \zeta) \sum_{n_k \in N} x_k}{\sum_{n_k \in N} x_k \Psi_k}, \quad (13)$$

where $\zeta^{'}$ is the previous state of the hybrid plane.

Algorithm 1 shows the steps of the SA-based C-RSU Placement Algorithm (CPA). The inputs of CPA include the initial state of the hybrid plane $\zeta^{'}$, initial temperature T_0, the rate of cooling z, Markov chain length M, and acceptance probability p. The initial temperature determines the convergence time of the algorithm. A high value of T_0 signifies that the algorithm takes more time to reach an optimal solution, and a low T_0 may direct the algorithm to a less accurate solution in less time. The cooling rate determines the amount of decrease

For personal use only

Algorithm 1 C-RSU Placement Algorithm (CPA)

```
INPUTS: $\zeta^{'}$, $T_0$, $z$, $M$, $p$
OUTPUT: $\zeta$                                          ▷ Final state
PROCEDURE:
 1: $T \leftarrow T_0$                                   ▷ Current temperature
 2: Sort RSUs in descending order of $\Psi$
 3: $N^{'} \leftarrow$ The first $\Gamma$ RSUs
 4: Set $x_j \leftarrow 1$ for each RSU in $N^{'}$
 5: for all $n_k \in N$ do
 6:     Set $y_{jk} \leftarrow 1$ if $n_j$ is the nearest RSU with $x_j = 1$
 7: end for
 8: $\zeta \leftarrow \{x, y\}$                           ▷ Current state
 9: while $T > 0$ do
10:     while $M > 0$ do
11:         $\zeta^{next} \leftarrow GetNextState(\zeta)$       ▷ Next state
12:         if $exp\left(\frac{cost(\zeta)-cost(\zeta^{next})}{T}\right) > p$ then
13:             $\zeta \leftarrow \zeta^{next}$
14:         end if
15:         $M \leftarrow M - 1$
16:     end while
17:     $T \leftarrow z \times T$
18: end while
19: return $\zeta$
```

in the temperature, and the algorithm terminates when the temperature reaches 0. The length of the Markov chain signifies the maximum number of iterations before cooling the temperature. The acceptance probability determines whether a solution is acceptable or not. CPA aims to find the optimal RSUs to place the local controller modules. Initially, the current state is formed by selecting Γ high priority RSUs as C-RSUs and assigning each RSU to the nearest C-RSU. Subsequently, CPA modifies the initial state to reach an optimal final state. Maximum I iterations are performed for each value of the current temperature. In each iteration, CPA generates the next state using the *GetNextState* method, as shown in Algorithm 2. Equation (13) calculates the controller placement cost for the current state and the next state of the hybrid plane. The next state is selected as the current state if $exp\left(\frac{cost(\zeta)-cost(\zeta^{next})}{T}\right) > p$. After the completion of M iterations, the current temperature is reduced. In this work, we use an exponential function as the cooling method and set the new temperature as $T \leftarrow z \times T$. Algorithm 2 shows

Algorithm 2 GetNextState

```
INPUT: $\zeta$                                           ▷ Current state
OUTPUT: $\zeta^{next}$                                    ▷ Next state
PROCEDURE:
 1: $\zeta^{next} \leftarrow \zeta$
 2: Randomly select an RSU $n_k \in N$ and set $x_k \leftarrow (1 - x_k)$
 3: for all $n_k \in N$ do
 4:     if $n_j$ is the nearest RSU with $x_j = 1$ then
 5:         $y_{jk} \leftarrow 1$
 6:     end if
 7: end for
 8: return $\zeta^{next} \leftarrow \{x, y\}$
```

the steps of computing the next state given a current state. We alter the C-RSU placement status of a randomly selected RSU. The status of a single RSU is modified to reduce the state migration overhead. Subsequently, each RSU selects the nearest C-RSU as the serving local controller. Figure 2 shows the basic framework of MobiPlace. The domain controllers perform multiple operations, including periodic synchronization with other controllers, flow-rule management, mobility prediction based on the mobility history of the vehicles, and the placement of controller modules at selected RSUs. The RSUs monitor current traffic volume, maintain flow-tables, and collect mobility data from the connected vehicles. A C-RSU has an additional module for serving as a local controller.

C. Controller Selection

Based on current controller placement status, a vehicle can send a control message to either a C-RSU or a domain controller. If the computed flow setup delay less for processing the control message at a C-RSU than that for processing the message at a domain controller, the message is processed locally at a C-RSU. Therefore, for a flow f_a, the value of α is redefined as:

$$\alpha(f_a) = \begin{cases} 1 & \text{if } D^u \leq D^c, \\ 0 & \text{otherwise,} \end{cases} \tag{14}$$

The time complexity of the mobility management module is $O(m^2)$, where m is the order of the Markov Predictor. The time complexity of the controller placement module depends on the values of T_0 and z. A high T_0 and low z increases the possibility of finding an optimal solution at the cost of time complexity. Therefore, latency-sensitive networks can set T_0 to a low value for faster processing. The controller selection module takes constant time to determine the value of $\alpha(f_a)$.

VI. Performance Evaluation

A. Simulation Settings

We analyze the performance of MobiPlace using MATLAB-based simulation. We use scale-free Barabasi-Albert topology [5] as the network topology of the RSUs due to the non-availability of suitable topology-based datasets for SDVN. For the simulation of road traffic, we use the traffic data of January, 2020, on the M18 motorway, Ireland [14]. Figure 3 shows the average volume of daily traffic. We select current the location of the vehicles randomly. For each vehicle, we generate the location history based on the Gauss-Markov mobility model [9]. Additionally, we set the traffic observation duration τ to 600 seconds. The simulation parameters are shown in Table I.

B. Benchmark Schemes

We compare the performance of MobiPlace with greedy placement, and dynamic placement [36]. The greedy placement approach selects Γ high priority RSUs as C-RSUs. We select the greedy approach as a benchmark to show that high priority RSUs are not the optimal choice for the placement of local controllers, and state migration overhead is an essential parameter for C-RSU placement. The dynamic controller placement approach proposed by Toufga *et al.* [36] considers the number of local controllers, RSU to local controller latency, controller load balancing, and the number of placement changes as parameters for the placement of

For personal use only

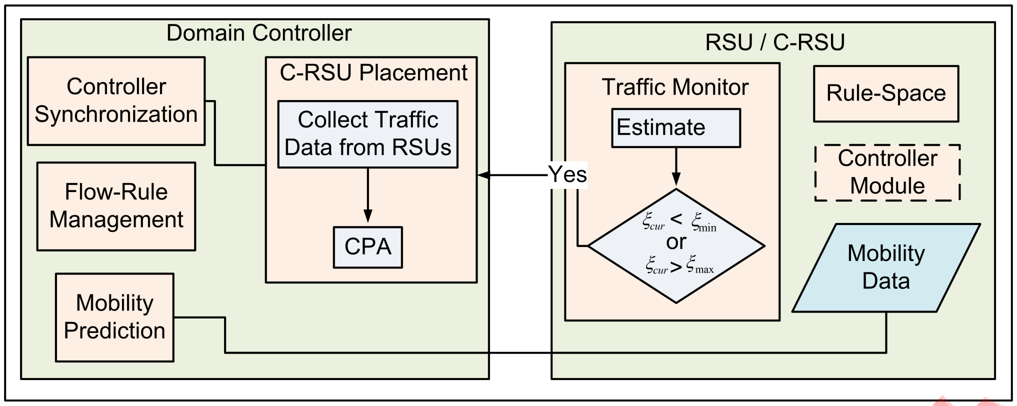

Fig. 2: Proposed Framework

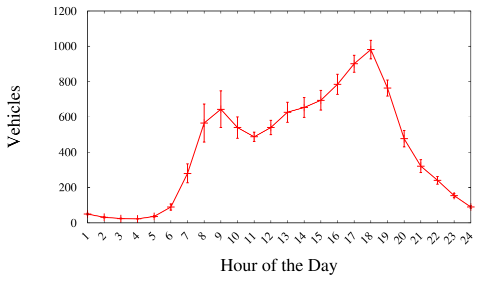

Fig. 3: Daily Traffic Volume

controllers and triggers controller placement periodically or based on pre-defined events. On the other hand, MobiPlace considers individual mobility status of the vehicles and triggers changes in controller placement considering the number of placement changes and several RSU-specific metrics including location, traffic volume, number of neighbors, and average duration stay of the vehicles at the RSU as parameters for the placement of local controllers. We select the dynamic placement proposed by Toufga *et al.* [36] as a benchmark because similar to MobiPlace this approach modifies the placement of local controllers based on traffic variations.

C. Performance Metrics

We evaluate the performance of MobiPlace based on the following metrics:

- **Flow setup delay**: Improper and inadequate local controller placement increases the average flow setup delay due to high queueing delay at the controllers and non-availability of nearby controller. Therefore, we use flow setup delay as a performance metric for the placement of local controllers.
- **End-to-End delay**: End-to-end delay of processing the data flows should not exceed the maximum allowable delay. We use this metric to estimate the QoS violation of the data flows.
- **State migration overhead**: High state migration overhead is not desirable because it increases the controller synchronization cost. We evaluate this metric to estimate the amount of additional inter-controller communications due to local controller placement.
- **Number of Local Controllers**: Minimal number of local controllers reduces the OPEX and the controller synchronization overhead. Therefore, we use the number of local controllers as a performance metric.

TABLE I: Simulation Parameters

Parameter	Value
Number of RSUs	50 – 200
Simulation duration	24 hours
Data transmission rate	6 Mbps (DSRC), 100 Mbps (Ethernet), 100 Gbps (Internet) [34]
Propagation speed	3×10^8 m/s (wirless channel), 2×10^8 m/s (wired channel), [34]
Size of Packet-In message	32 bytes [1]
Size of Flow-Mod message	56 bytes [1]
Packet size	400 bytes [34]
CPU frequency at OBU	$10-30$ MHz (OBU), $2.9-4.2$ GHz (C-RSU), $2.9-4.2$ GHz (domain controller) [26]
Service rate	50 packets/s (C-RSU), 0.02 million packets/s (domain controller) [35]
Request rate of an OBU	0 – 10 flows/s [34]
Maximum allowable delay	100 – 200 ms [34]
Initial temperature for SA	90 [38]
Cooling rate	0.97 [38]
Length of Markov chain	200 [38]
Acceptance probability	0.85 [38]

D. Result and Discussion

1) Flow Setup Delay: We measure the average flow setup delay based on the current hybrid plane state and the traffic volume in each hour. Figure 4 shows the average flow setup delay for hourly traffic. From simulation results, we observe that the average flow setup delay for MobiPlace is less than

For personal use only

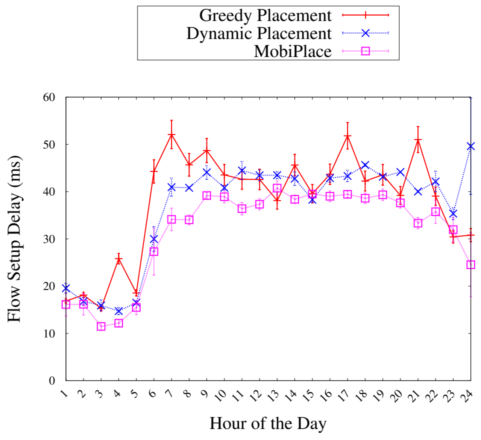

Fig. 4: Flow Setup Delay with 200 RSUs

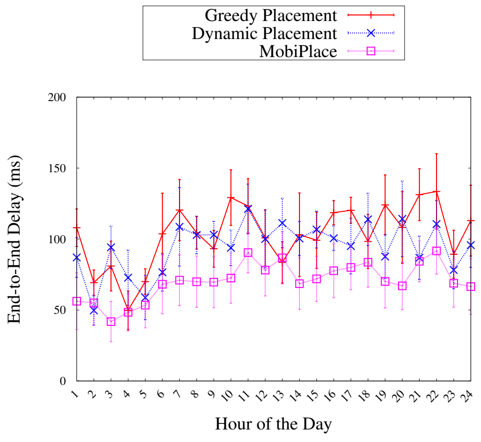

Fig. 5: End-to-End Delay with 200 RSUs

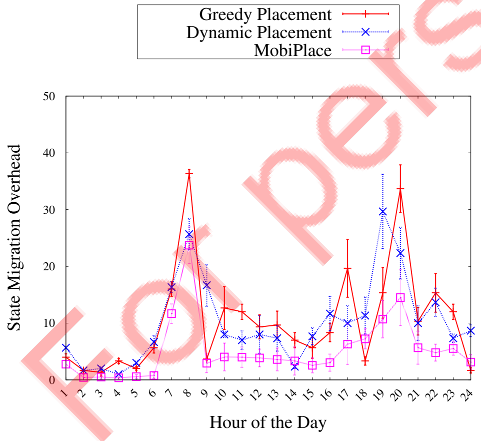

Fig. 6: State Migration Overhead with 200 RSUs

the benchmark schemes. In particular, at the 18th hour, the flow setup delay for MobiPlace is 8.62% and 15.44% less than greedy placement and dynamic placement, respectively. Moreover, we observe that the average flow setup delay for MobiPlace increases gradually for a sharp rise in traffic than the benchmarks. Figure 8 shows the flow setup delay for different RSU count with 1000 vehicles and a simulation duration of 1 hour. The flow setup delay for MobiPlace with

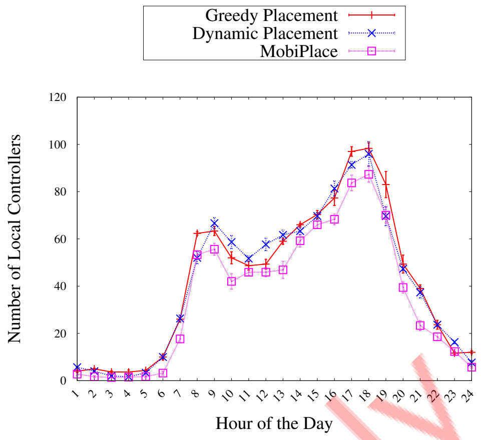

Fig. 7: Number of Local Controllers with 200 RSUs

200 RSUs is 5.15% more than that with 50 RSUs.

Inference: The flow setup delay in MobiPlace is less because MobiPlace avoids reactive controller synchronization after the Flow-Mod fails to reach the communicating vehicle. Instead of reactive controller synchronization, MobiPlace proactively performs controller synchronization using vehicle mobility prediction. Additionally, MobiPlace well-distributes the C-RSUs based on changing traffic load so that the vehicles prefer C-RSUs to domain controllers for processing their requests. Therefore, the flow setup delay does not increase rapidly for excessive change in the traffic load. From the simulation results, we conclude that MobiPlace places local controllers at optimal locations so that the time required for the installation of new flow-rules is minimized. Additionally, with more RSU count, the flow setup delay increases because each local controller handles requests forwarded by more RSUs.

2) End-to-End Delay: The end-to-end delay is the time interval between the generation of a Packet-In request and the reaching of the last packet of data flow at the destination. Figure 5 depicts that the average end-to-end delay for MobiPlace is less than the benchmark schemes. In particular, at the 18^{th} hour, the end-to-end delay for MobiPlace is 14.90% and 26.52% less than that of greedy placement and dynamic placement, respectively. Figure 9 shows the end-to-end delay for different RSU count with 1000 vehicles and a simulation duration of 1 hour. The end-to-end delay for MobiPlace with 200 RSUs is 6.42% more than that with 50 RSUs.

Inference: The end-to-end delay for MobiPlace is less because MobiPlace performs proactive rule installation and controller synchronization. From the simulation results, we infer that the probability of QoS violation is low in MobiPlace due to the low processing time of the data flows. Moreover, with more RSU count, the end-to-end delay increases because of more RSUs in each flow path.

3) State Migration Overhead: We measure the state migration overhead for each hour by comparing the current controller placement with that of the previous hour. Figure 6 shows that the average state migration overhead for MobiPlace over the 24 hours of the day is 49.62% and 48.28% less than that of greedy placement and dynamic placement, respectively. Additionally, we observe that the state migration overhead for

For personal use only

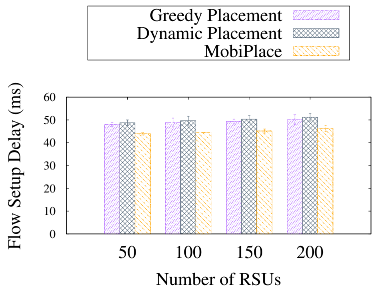

Fig. 8: Flow Setup Delay with 1000 Vehicles and Different RSU Count

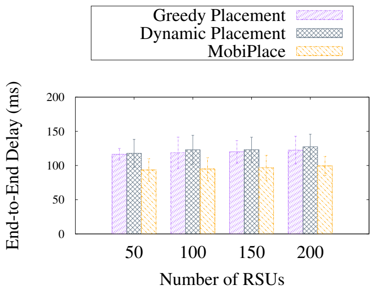

Fig. 9: End-to-End Delay with 1000 Vehicles and Different RSU Count

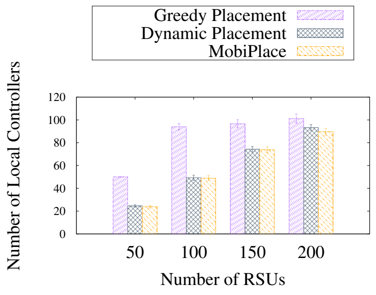

Fig. 10: Number of Local Controllers with 1000 Vehicles and Different RSU Count

MobiPlace does not increase rapidly for high traffic load.

Inference: As stated in Equation (13), MobiPlace places new C-RSUs based on traffic demand by prioritizing existing C-RSU placement. MobiPlace observes the traffic volume at the RSUs for τ seconds before initiating a change. Therefore, the state migration overhead is less in MobiPlace. On the other hand, the dynamic placement scheme reconsiders the placement of local controllers periodically. Therefore, local controllers are added or removed, even for short-term changes in traffic load. Therefore, the state migration overhead in dynamic placement is higher than that of MobiPlace. The average number of local controllers in the greedy placement approach is fixed to Γ. From the simulation results, we infer that MobiPlace incurs less inter-controller traffic because it does not prefer frequent changes in the hybrid plane state.

4) Number of Local Controllers: For each hour, we estimate the number of controllers placed at the RSU-level. From Figure 7, we observe that the average number of local controllers is less for MobiPlace as compared to the benchmark schemes. In particular, at the 18^{th} hour, the number of local controllers for MobiPlace is 11.19% and 9.89% less as compared to the greedy placement and dynamic placement, respectively. Figure 10 shows the number of local controllers for different RSU count with 1000 vehicles and a simulation duration of 1 hour. For 50 RSUs, the number of local controllers for MobiPlace is 52.40% and 2.98% less than the greedy placement and dynamic placement, respectively. On the other hand, for 200 RSUs, the number of local controllers for MobiPlace is 11.41% and 3.96% less as compared to the greedy placement and dynamic placement, respectively.

Inference: MobiPlace aims to manage the traffic load with a minimal number of local controllers. The number of local controllers in the greedy placement is high because the greedy placement scheme considers only RSU priorities for local controllers' placement and does not aim to minimize the number of controllers. The number of local controllers for the dynamic placement is higher than that of MobiPlace because the dynamic placement places more local controllers to reduce the RSU to local controller latency. Moreover, the dynamic placement scheme adds a new controller after a periodic time interval, even when the traffic increase lasts for a short duration. On the other hand, MobiPlace places an additional controller only if the traffic changes last for a considerable amount of time. From the simulation results, we conclude that MobiPlace reduces the OPEX and controller synchronization overhead by placing a minimal number of local controllers. Additionally, for MobiPlace, the number of local controllers is less even with low RSU count because MobiPlace optimizes the C-RSU placement considering the number of C-RSUs as a metric as mentioned in Equation (13).

VII. Conclusion

In this paper, we presented a mobility-aware controller placement scheme for SDVN. The proposed scheme, MobiPlace, places local controller modules at RSU level to reduce the vehicle to controller communication latency. However, static controller placement is not suitable for SDVN as road traffic conditions change over time. Therefore, MobiPlace estimates traffic conditions for a pre-defined duration and changes the placement status of local controllers dynamically with minimal state migration overhead. Additionally, MobiPlace uses mobility prediction to reduce data loss and controller synchronization overhead during flow setup. Simulation results exhibited that MobiPlace reduces the flow setup delay by 13.85% and the state migration overhead by 48.28% compared to the existing dynamic placement scheme. The limitation of the proposed scheme depends on the processing capability and resource availability of the RSUs because usage of an RSU as a C-RSU requires additional computation and storage.

The future extension of the work includes caching popular flow-rules at selected RSUs to reduce the flow setup delay further. Additionally, we plan to evaluate the proposed scheme with more traffic traces.

References

[1] OpenFlow Switch Specification (Version 1.5.1): Open Networking Foundation. Mar. 2015.

[2] H. M. E. Abdelsalam and H. P. Bao. A simulation-based optimization framework for product development cycle time reduction. *IEEE Trans. Eng. Manag.*, 53(1):69–85, 2006.

[3] Muhammad Anan, Ala Al-Fuqaha, Nidal Nasser, Ting-Yu Mu, and Husnain Bustam. Empowering networking research and experimentation through Software-Defined Networking. *Journal of Network and Computer Applications*, 70:140 – 155, 2016.

[4] G. Araniti, I. Bisio, M. De Sanctis, F. Rinaldi, and A. Sciarrone. Joint Coding and Multicast Subgrouping Over Satellite-eMBMS Networks. *IEEE J. Sel. Areas Commun.*, 36(5):1004–1016, May 2018.

[5] Albert-László Barabási and Réka Albert. Emergence of Scaling in Random Networks. *Science*, 286(5439):509–512, 1999.

[6] J. Bhatia, P. Kakadia, M. Bhavsar, and S. Tanwar. SDN-enabled Network Coding Based Secure Data Dissemination in VANET Environment. *IEEE Internet Things J.*, pages 1–1, 2019. doi: 10.1109/JIOT.2019.2956964.

For Personal Use only

[7] I. Bisio, C. Garibotto, F. Lavagetto, and A. Sciarrone. Computational Complexity Closed-Form Upper Bounds Derivation for Fingerprint-Based Point-of-Interest Recognition Algorithms. *IEEE Trans. Veh. Technol.*, 69(8):9083–9096, August 2020.

[8] I. Bisio, C. Garibotto, F. Lavagetto, A. Sciarrone, and S. Zappatore. Blind Detection: Advanced Techniques for WiFi-Based Drone Surveillance. *IEEE Trans. Veh. Technol.*, 68(1):938–946, January 2019.

[9] T. Camp, J. Boleng, and V. Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless Communications and Mobile Computing*, 2(5):483–502, Aug. 2002.

[10] A. Dalgkitsis, P. Mekikis, A. Antonopoulos, and C. Verikoukis. Data Driven Service Orchestration for Vehicular Networks. *IEEE Trans. Intell. Transp. Syst.*, pages 1–10, 2020. doi: 10.1109/TITS.2020.3011264.

[11] T. Das, V. Sridharan, and M. Gurusamy. A Survey on Controller Placement in SDN. *IEEE Commun. Surveys Tuts.*, 22(1):472–503, 2020.

[12] G. de Biasi, L. F. M. Vieira, and A. A. F. Loureiro. Sentinel: Defense Mechanism against DDoS Flooding Attack in Software Defined Vehicular Network. In *Proc. IEEE ICC*, pages 1–6, 2018.

[13] A. Di Maio, R. Soua, M. R. Palattella, and T. Engel. ROADNET: Fairness- and Throughput-Enhanced Scheduling for Content Dissemination in VANETs. In *Proc. IEEE ICC Workshops*, pages 1–6, 2018.

[14] Drakewell Ltd. *C2-Cloud Traffic Data*. Accessed: July, 2020.

[15] K. Z. Ghafoor, L. Kong, D. B. Rawat, E. Hosseini, and A. S. Sadiq. Quality of Service Aware Routing Protocol in Software-Defined Internet of Vehicles. *IEEE Internet Things J.*, 6(2):2817–2828, 2019.

[16] Brandon Heller, Rob Sherwood, and Nick McKeown. The Controller Placement Problem. In *Proc. ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking (HotSDN)*, pages 7–12, New York, NY, USA, 2012. ACM.

[17] D. Hock, M. Hartmann, S. Gebert, M. Jarschel, T. Zinner, and P. Tran-Gia. Pareto-optimal resilient controller placement in SDN-based core networks. *Proc. ITC*, pages 1–9, Sep. 2013.

[18] K. Kaur, S. Garg, G. Kaddoum, N. Kumar, and F. Gagnon. Sdn-based internet of autonomous vehicles: An energy-efficient approach for controller placement. *IEEE Wireless Commun.*, 26(6):72–79, 2019.

[19] A. Ksentini, M. Bagaa, T. Taleb, and I. Balasingham. On using bargaining game for Optimal Placement of SDN controllers. In *Proc. IEEE ICC*, pages 1–6, May 2016.

[20] Constantine Kyriakopoulos, Petros Nicopolitidis, Georgios Papadimitriou, and Emmanouel Varvarigos. Exploiting IP-layer traffic prediction analytics to allocate spectrum resources using swarm intelligence. *International Journal of Communication Systems (Wiley)*, 33(14):e4516, 2020. doi: 10.1002/dac.4516.

[21] K. Liu, K. Xiao, P. Dai, V. Lee, S. Guo, and J. Cao. Fog Computing Empowered Data Dissemination in Software Defined Heterogeneous VANETs. *IEEE Trans. Mobile Comput.*, pages 1–1, 2020. doi: 10.1109/TMC.2020.2997460.

[22] Y. P. Llerena and P. R. L. Gondim. SDN-Controller Placement for D2D Communications. *IEEE Access*, 7:169745–169761, 2019.

[23] J. Lu, Z. Zhang, T. Hu, P. Yi, and J. Lan. A Survey of Controller Placement Problem in Software-Defined Networking. *IEEE Access*, 7:24290–24307, 2019.

[24] Ashraf Mahmoud, Ahmad Abo Naser, Marwan Abu-Amara, Tarek Sheltami, and Nidal Nasser. Software-defined networking approach for enhanced evolved packet core network. *International Journal of Communication Systems (Wiley)*, 31(1):33–39, 2018.

[25] I. Maity, A. Mondal, S. Misra, and C. Mandal. CURE: Consistent Update With Redundancy Reduction in SDN. *IEEE Trans. Commun.*, 66(9):3974–3981, Sep. 2018.

[26] S. Misra and S. Bera. Soft-VAN: Mobility-Aware Task Offloading in Software-Defined Vehicular Network. *IEEE Trans. Veh. Technol.*, 69(2):2071–2078, 2020.

[27] P. Nicopolitidis N. Garg, S.K. Dhurandher and J.S. Lather. Efficient Mobility Prediction Scheme for Pervasive Network. *International Journal of Communication Systems (Wiley)*, 2017.

[28] N. Nasser, A. Hasswa, and H. Hassanein. Handoffs in fourth generation heterogeneous networks. *IEEE Commun. Mag.*, 44(10):96–103, October 2006.

[29] O. Sadio, I. Ngom, and C. Lishou. Design and Prototyping of a Software Defined Vehicular Networking. *IEEE Trans. Veh. Technol.*, 69(1):842–850, 2020.

[30] B. K. Saha, S. Misra, and S. Pal. SeeR: Simulated Annealing-Based Routing in Opportunistic Mobile Networks. *IEEE Trans. Mobile Comput.*, 16(10):2876–2888, 2017.

[31] R. Shahidi and M. H. Ahmed. Probability Distribution of End-to-End Delay in a Highway VANET. *IEEE Commun. Lett.*, 18(3):443–446, 2014.

[32] S. Sigg, D. Gordon, G. v. Zengen, M. Beigl, S. Haseloff, and K. David. Investigation of Context Prediction Accuracy for Different Context Abstraction Levels. *IEEE Trans. Mobile Comput.*, 11(6):1047–1059, Jun. 2012.

[33] A. Singh, G. S. Aujla, and R. S. Bali. Intent-Based Network for Data Dissemination in Software-Defined Vehicular Edge Computing. *IEEE Trans. Intell. Transp. Syst.*, pages 1–9, 2020. doi: 10.1109/TITS.2020.3002349.

[34] K. L. K. Sudheera, M. Ma, and P. H. J. Chong. Controller placement optimization in hierarchical distributed software defined vehicular networks. *Computer Networks*, 135:226 – 239, 2018.

[35] Amin Tootoonchian, Sergey Gorbunov, Yashar Ganjali, Martin Casado, and Rob Sherwood. On Controller Performance in Software-defined Networks. In *Proc. USENIX Conf. Hot Topics Manage. Internet, Cloud, Enterprise Netw. Services*, pages 1–6, 2012.

[36] Soufian Toufga, Slim Abdellatif, Hamza Tarik Assouane, Philippe Owezarski, and Thierry Villemur. Towards Dynamic Controller Placement in Software Defined Vehicular Networks. *Sensors*, 20(6):1701, Mar 2020.

[37] M. T. I. ul Huque, W. Si, G. Jourjon, and V. Gramoli. Large-Scale Dynamic Controller Placement. *IEEE Trans. Netw. Service Manag.*, 14(1):63–76, Mar. 2017.

[38] D. Wu, J. Yan, H. Wang, and R. Wang. User-centric Edge Sharing Mechanism in Software-Defined Ultra-Dense Networks. *IEEE J. Sel. Areas Commun.*, pages 1–1, 2020.

For personal use only