CORE: Prediction-Based Control Plane Load Reduction in Software-Defined IoT Networks

Ilora Maity, Student Member, IEEE,
Sudip Misra, Senior Member, IEEE, and Chittaranjan Mandal, Senior Member, IEEE

***Abstract*—In this paper, we propose a scheme to address the problem of load management in the control plane of Software-Defined Internet of Things (SDIoT) networks. In SDIoT, multiple controllers are deployed to enhance network scalability. With the growth of IoT, the number of devices is increasing rapidly. The management of control plane load is an essential issue for IoT networks because of the dynamic traffic characteristics. IoT traffic is highly dynamic due to the heterogeneity of IoT devices in terms of mobility, activation model, Quality of Service (QoS) demand, and flow generation rate. The challenge is to prevent controller overload and distribute traffic optimally under the consideration of heterogeneous IoT devices. The proposed scheme estimates control plane load based on the mobility and activation model of IoT devices. For mobility prediction, we use Order-m fallback Markov Predictor as it consumes less space and performs efficiently even for small values of m. Based on the prediction results, we implement a traffic-aware rule-caching mechanism and a master controller assignment scheme to reduce the control plane load. Simulation results show that the proposed scheme reduces the peak intensity of the control traffic by 23.08% and 16.67%, as compared to the considered benchmark schemes.**

***Index Terms*—oftware-Defined Networking (SDN), Caching, Markov Predictor, IoT, Branch and Bound.oftware-Defined Networking (SDN), Caching, Markov Predictor, IoT, Branch and Bound.S**

I. Introduction

SDN is an evolving networking paradigm that separates control decisions from data forwarding [1]. The global network view provided by SDN makes it an attractive choice for use in large-scale IoT deployment applications. However, SDIoT network requires multiple controllers to handle heterogeneous traffic from IoT devices [2]. In SDIoT, overloaded control plane is a bottleneck that disrupts the network performance by increasing the response time. The control plane load is directly linked to controller-switch assignments. Therefore, dynamic assignment of switches to controllers is required to balance control plane load and provide uninterrupted network services in IoT networks where fluctuation of traffic is frequent.

With the rising demand for IoT applications, SDIoT networks encounter additional challenges due to the presence of a large number of wireless smart devices. The primary challenge involves handling a large number of Packet-In messages, which are control messages initiated by a switch to its master controller. For each new flow generated from a device, an ingress switch transmits a Packet-In request to the master controller. The master controller formulates a new flow-rule and sends it to the corresponding switch in the form of a Flow-Mod message [3]. Therefore, the load of a controller depends on the set of associated switches and Packet-In messages generated by the associated switches. If these Packet-In messages are not handled in a timely manner, the response time of the control plane increases. In the worst case, the incoming packets get stalled in the queue of the switches and are dropped once the queue is full. This case is not desirable for IoT devices because IoT devices are low-power, resource-constrained, and loss-sensitive.

In addition, the bursty nature of IoT traffic is challenging for the control plane due to the sudden rise of control messages. IoT devices follow different activation models [4], which is the cause of the bursty traffic. Randomly activated devices generate data flow bursts for a short duration. On the other hand, periodically activated devices generate traffic at a fixed interval. Consequently, a large number of control messages are generated during the peak times of the day and overloads some of the controllers.

Moreover, IoT networks involve devices having heterogeneous QoS requirements [5]. For example, IoT applications generating audio or video traffic are more latency-sensitive than the applications generating image-based traffic. Similarly, IoT applications providing emergency services are also latency-sensitive. Therefore, an overloaded controller is a bottleneck for these latency-sensitive IoT applications.

Additionally, device mobility is an essential factor of IoT networks [6]. For example, IoT-based health monitoring applications include mobile IoT devices to track the status of the patients. Usually, these mobile IoT devices or wearables move from one location to another due to human mobility. The load of the corresponding controllers changes when a device moves from one controller's domain to another.

Hence, a high number of devices, bursty traffic, different QoS demand, and device mobility are crucial metrics for control plane load management in SDIoT network. The majority of the existing solutions consider single parameter such as QoS or dynamic traffic [7], [2]. This work aims to manage the workload of the controllers in SDIoT, under the consideration of the essential metrics of IoT networks. The proposed control plane load reduction approach, named CORE, consists of three modules — *mobility prediction*, *rule-caching*, and *master controller assignment*. The *mobility prediction* module uses a Markov Predictor [8] to predict device-switch associations based on device mobility. Subsequently, the *rule-caching* module caches the popular rules based on the QoS demand and traffic pattern of IoT devices. Finally, the *master controller assignment* module computes optimal controller-switch assignments to prevent control plane overload. The specific *contributions* of this work are as follows:

- **We formulate an integer linear problem (ILP) to minimize control plane load under the consideration of controller capacity, rule-space limit, device-specific QoS demands as constraints.**
- **We propose a master controller assignment scheme that identifies optimal controller-switch assignments to minimize the control plane load. The proposed scheme uses Markov Predictor to predict device-switch associations based on device mobility. In contrast to existing schemes, the proposed scheme considers the bursty nature of IoT traffic due to the different activation schedules of the devices.**
- **Based on the prediction results, we propose a device-aware**

The authors 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; chitta@iitkgp.ac.in).

rule-caching approach to reduce the controllers' load. This approach considers device-specific parameters such as QoS demand and flow generation rate.

II. Related Work

A. SDIoT

SDN is emerging as a promising platform for IoT solutions. The integration of SDN and IoT is beneficial in terms of intelligent traffic engineering, real-time decision making, and provision for global view of the network [9]. Muñoz *et al.* [10] proposed a SDIoT architecture to manage IoT traffic dynamically and avoid link congestion. Bellavista *et al.* [11] presented a SDIoT architecture involving Fiber-Wireless (FiWi) access networks for better Quality of Service (QoS).

B. Control Plane Load Management in SDN

Existing approaches in this field are categorized in two parts — controller placement-based and switch migration-based.

1) Controller Placement-Based: Controller placement-based schemes select the number and locations of the controllers to manage the load. Hock *et al.* [12] considered the maximum control link latency and the number of switches attached to a controller in order to place the controllers and stabilize the load. However, the authors assume static traffic between switches and controllers. Therefore, this approach is not preferable for large-scale networks, including IoT networks. Ksentini *et al.* [13] proposed a controller placement technique based on Nash bargaining game. The authors considered control link latency and equal load distribution to the controllers as the major objectives. However, this approach does not consider the master and slave roles of an SDN controller. Huque *et al.* [2] proposed LiDy+, which places the controller modules based on data plane traffic prediction. The load is distributed evenly among the controllers in each module. However, frequent activation and deactivation increases control plane overhead in terms of messages.

2) Switch Migration-Based: Switch migration-based schemes migrate switches from a highly-loaded controller's domain to the domain of a lightly-loaded controller. Dixit *et al.* [14] proposed a switch-migration scheme that migrates switches from an overloaded controller to a controller with less load. The proposed approach includes the addition and removal of controllers, as required, in the presence of dynamic traffic. However, this approach ignores switch-to-controller latency. Bari *et al.* [7] proposed Dynamic Controller Provisioning with Simulated Annealing (DCP-SA), which dynamically activates and deactivates controllers to reduce the flow setup cost and the overhead for communication. In this work, the authors used two heuristics based on simulated annealing and greedy knapsack. The proposed heuristics periodically reassign switches to controllers for addressing load imbalance at the control plane. The proposed approach does not consider queueing delay at the controller. Sahoo *et al.* [15] proposed an Efficient Switch Migration technique for Load Balancing (ESMLB) scheme to balance the control plane load in SDIoT. The proposed approach identifies the overloaded controllers and the switches which generate the maximum Packet-In requests to each overloaded controller. Each selected switch is migrated to a lightly-loaded target controller, which is selected based on multiple criteria such as hop count, memory usage, and bandwidth. However, the proposed approach does not consider the effects of uneven traffic distribution.

***Synthesis*:** From the detailed study of the existing literature, we infer that there exists a lacuna in the research literature addressing the problem of control plane load management for large-scale SDN, including SDIoT, where the heterogeneous attributes of IoT devices have a major impact on the control plane load. However, existing solution approaches do not consider device heterogeneity while dealing with the dynamic workload. Moreover, existing solution approaches ignore the bursty nature of IoT traffic due to different activation models of IoT devices. Therefore, in this work, we propose a prediction-based master controller assignment scheme for control plane load reduction in SDIoT while considering heterogeneous attributes of IoT devices such as mobility, activation model, and QoS demand.

III. System Model

A. System Architecture

The SDIoT architecture considered in our work is depicted in Figure 1 [16]. The architecture comprises three layers — application, network, and perception.

1) Application Layer: The application layer consists of IoT applications, which perform services requested by the users based on the data collected from the network layer.

2) Network Layer: The network layer consists of data plane and control plane. Let $G = (S, E)$ represent the data plane topology, where S denotes the set of SDN switches and E denotes the set of data links. We assume that a switch stores up to R_{max} flow-rules. Each flow-rule r_e has a timeout duration T_e seconds. Let C represent the set of controllers. We consider that each switch s_i is attached to single master controller and one or multiple slave (read-only) controllers [3] during a time-slot. Let, ϵ seconds be the duration of each time-slot. The master controller for switch s_i is expressed as:

$$x_{ij}(t) = \begin{cases} 1 & \text{if } c_j \text{ is the master controller of } s_i, \\ 0 & \text{otherwise}. \end{cases} \quad (1)$$

The slave controller for switch s_i is expressed as:

$$y_{ij}(t) = \begin{cases} 1 & \text{if } c_j \text{ is the slave controller of } s_i, \\ 0 & \text{otherwise}. \end{cases} \quad (2)$$

A controller cannot be both master and slave for the same switch at the same time-slot. Therefore, $x_{ij}(t) + y_{ij}(t) \leq 1, \forall s_i \in S, \forall c_j \in C$.

***Definition* 1** (Controller Capacity)**.** *The capacity of a controller c_j is the maximum number of Packet-In requests the controller handles in a time-slot and is denoted by Ω_j.*

3) Perception Layer: The perception layer contains static and mobile IoT devices that are heterogeneous in terms of QoS requirements. The flows generated by the IoT devices are transmitted over the wireless channel to switches via access points (APs) having different radio access capabilities such as WiFi, WiMax, Bluetooth, 3G, 4G, Zigbee, mmWave, and TV White Space. In this work, we assume that each IoT device is capable of communicating via more than one radio access technique. For a time-slot t, $D(t)$ denotes the set of IoT devices. For simplicity, we assume that all the flow-rules are exact-match flow-rules [3], where the mapping between flow-rule and flow type is one-to-one. Further, we assume that each device generates only single type of flow. However, if an IoT device generates multiple types of flows, multiple instances of that device can be considered. Let Q_k be the number of flows generated by a device $d_k(t) \in D(t)$ per second. We assume that the controllers record device specific parameters such as the flow generation rate, the mapped flow type, and QoS requirement for each time-slot. The association between an IoT device and an SDN switch is expressed as:

$$z_{ik}(t) = \begin{cases} 1 & \text{if } d_k(t) \text{ is associated with } s_i, \\ 0 & \text{otherwise}. \end{cases} \quad (3)$$

We consider that a device is associated with single switch only, as each AP sends data to a specific SDN switch. Therefore, $\sum_{i=1}^{|S|} z_{ik}(t) = 1, \forall d_k(t) \in D(t)$. Based on the type of IoT application, each IoT device $d_k(t)$ activates/deactivates following either — (1) random activation model or (2) periodic activation model [4]. For example, devices used for IoT applications such as

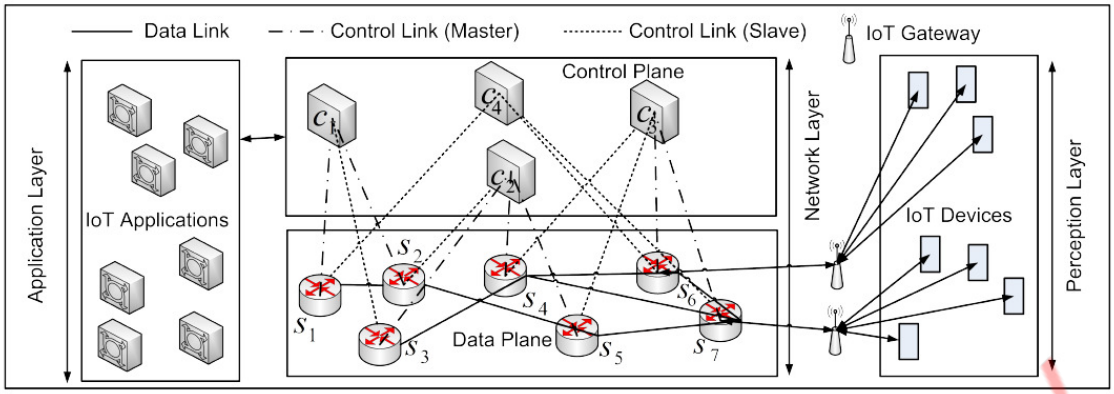


Fig. 1: SDIoT Architecture

traffic monitoring, and connected health follow random activation model. On the other hand, devices for IoT applications such as smart home, smart city, and environment monitoring require to report data periodically. Let t_0 be the start time of time-slot t. A device $d_k(t)$ following random activation model activates at the τ^{th} second, where $\tau \in [t_0, t_0+\epsilon]$, according to the beta distribution with shape parameters β_1, and β_2 [4], which is expressed as $f_k(\tau) = \frac{(\tau-t_0)^{\beta_1-1}(t_0+\epsilon-\tau)^{\beta_2-1}}{\epsilon^{\beta_1+\beta_2-1}\int_0^1 \tau^{\beta_1-1}(1-\tau)^{\beta_2-1}d\tau}$. On the other hand, a device $d_k(t)$ following periodic activation model activates repeatedly after a fixed duration τ_k seconds. Therefore, the probability that a device $d_k(t)$ following periodic activation model activates at time instant $\tau \in [t_0, t_0+\epsilon]$ given by:

$$f_k(\tau) = \begin{cases} 1 & \text{if the interval between } \tau \text{ and the last active} \\ & \text{time of } d_k(t) \text{ is more than } \tau_k, \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

The maximum number of Packet-In messages generated by $d_k(t)$ in time-slot t is $M_k(t) = \int_{t_0}^{t_0+\epsilon} f_k(\tau) Q_k d\tau$.

B. Mobility Model

We consider a network which has both static or mobile IoT devices. Examples of some mobile IoT devices are smart wearables, cameras, and AR/VR glasses [17]. During each time-slot, SDN controllers collect device locations using Simple Network Management Protocol (SNMP) via southbound APIs [18]. We use this collected data as a history data set to predict device-switch associations.

C. Caching Model

Each flow-entry has a default timeout duration [3]. However, an IoT device usually generates similar flow requests for a particular time duration. The interval of the arrival of such similar flows may be greater than the timeout duration of the corresponding flow-rule. In this case, a Packet-In message is re-generated, and an expired rule is re-installed. Rule-caching is one of the measures to reduce the number of Packet-In requests. However, as the cache size increases, the rule-space required for storing new flow-rules decreases, and the number of Packet-in requests increases, eventually. Therefore, CORE considers that each SDN switch caches maximum $R_{cache} < R_{max}$ flow-rules. To express whether a switch s_i caches a flow-rule for $d_k(t)$ during time-slot t we define a binary variable as:

$$w_{ik}(t) = \begin{cases} 1 & \text{if } s_i \text{ caches flow-rule that maps to the flow} \\ & \text{type of } d_k(t), \\ 0 & \text{otherwise.} \end{cases} \quad (5)$$

D. Delay Model

Let $\delta_k(t)$ denote the delay of an IoT flow of type f_k. The delay $\delta_k(t)$ has three components — a) device to AP communication delay $\delta_k^1(t)$, b) AP to switch communication delay $\delta_k^2(t)$, and c) flow setup delay $\delta_k^{ex}(t)$ at the switch. Mathematically, $\delta_k^1(t) = \Delta^1(t) + \frac{g_k(t)}{G^1 E^1}$ and $\delta_k^2(t) = \Delta^2(t) + \frac{g_k(t)}{G^2 E^2}$, where $\Delta^1(t)$ is the transmission delay from device to AP, $\Delta^2(t)$ is the transmission delay from from AP to switch, $g_k(t)$ represents the number of bytes sent by $d_k(t)$ in time-slot t, G^1 is the bandwidth of the wireless channel from device to AP, G^2 represents the bandwidth of the wireless channel from AP to switch, E^1 and E^2 represent the channel overheads of the corresponding wireless channels. The flow setup delay $\delta_k^{ex}(t)$ is $\delta_k^{ex}(t) = \sum_{i=1}^{|S|}\sum_{j=1}^{|C|} z_{ik}(t)x_{ij}(t)\left(2\delta_{ij}(t) + \delta_j^{que}(t)\right)$, where $\delta_{ij}(t)$ is the transmission delay associated with the control link and $\delta_j^{que}(t)$ is the queueing delay at controller c_j. A controller stores the Packet-In requests in its queue and processes the requests in a First-Come-First-Serve (FCFS) order. Motivated by the work by He *et al.* [19], in this work, we consider each request as an individual and independent Poisson process. Moreover, we assume that each controller is single-threaded. Therefore, we model controller queue as a $M/M/1$ queue. The service rate of this queueing model is controller capacity Ω_j. The maximum request arrival rate is:

$$\lambda_j(t) = \sum_{i=1}^{|S|}\sum_{k=1}^{|D(t)|} x_{ij}(t)z_{ik}(t)(1-w_{ik}(t))M_k(t) \quad (6)$$

Here, (6) considers the associated devices which have no rules cached and estimates the maximum number of Packet-In requests based on the active duration of the devices for each controller c_j in time-slot t. Using Little's law, we get the queueing delay at the controller c_j as $\delta_j^{que}(t) = \frac{1}{\Omega_j - \lambda_j(t)}$.

E. Cost Model

Control plane cost has two components — 1) controller-switch communication cost and 2) inter-controller communication cost due to device mobility. The controller-switch communication cost at c_j is the traffic intensity $\rho_j(t) = \frac{\lambda_j(t)}{\Omega_j}$. Controllers collect global network status data by synchronizing with other controllers at regular intervals. We assume that each controller completes this synchronization process at the beginning of a time-slot. Additionally, there exist two cases when a controller synchronizes with another controller.

- **Case 1: Change in Controller-Switch Association**
  At time-slot t, each controller c_j records the switches to which c_j served as a slave controller for time-slot $t-1$

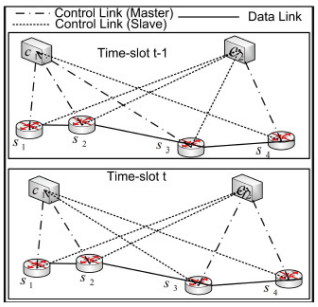


Fig. 2: Case 1: Change in Controller-Switch Association

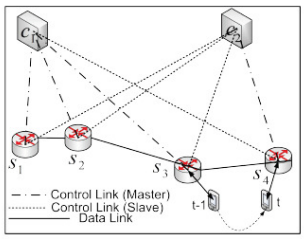


Fig. 3: Case 2: Change in Device-Switch Association

before changing its role to a master controller. In this case, c_j needs to synchronize with the former master controller(s) of the switches. Figure 2 shows an example where the master controller of switch s_3 changes from c_1 to c_2 at time-slot t. Therefore, for seamless handover, c_2 collects unfinished session data and flow information from c_1. For each controller c_j, the number of master-slave role changes during a time-slot is $\chi_c^j(t) = \sum_{i=1}^{|S|} |x_{ij}(t) - x_{ij}(t-1)|$.

- **Case 2: Change in Device-Switch Association**
  At time-slot t, each controller c_j records the mobile IoT devices which are newly associated with the switches assigned to c_j. If the old switches have different master controller(s), c_j needs to synchronize with the master controller(s) of the old switches. Figure 3 shows an example in which a mobile device changes the associated switch from s_3 to s_4 at time-slot t. As s_3 and s_4 have different master controllers c_1 and c_2, controller synchronization is required for seamless handover. For each controller c_j, the number of such changes where controller synchronization is required is $\chi_s^j(t) = \sum_{i=1}^{|S|} \sum_{k=1}^{|D(t)|} \xi_k^j(t)$, where $\xi_k(t)$ is expressed as:

$$\xi_k^j(t) = \begin{cases} 1 & \text{if } x_{ij}(t) z_{ik}(t) = x_{i'j'}(t-1) \\ & z_{i'k}(t-1) = 1,\ i \neq i',\ j \neq j', \\ 0 & \text{otherwise.} \end{cases} \quad (7)$$

Therefore, the total inter-controller communication cost for a controller c_j is $\Gamma_j(t) = \chi_c^j(t) + \chi_s^j(t)$.

F. Problem Formulation

The objective of CORE is to determine optimal controller-switch assignments to minimize the control plane cost. Therefore, we formulate the cache-enabled minimum cost master controller assignment (CMCA) problem as:

$$\underset{x(t), w(t)}{\text{Minimize}} \quad \alpha \sum_{j=1}^{|C|} \rho_j(t) + (1-\alpha) \sum_{j=1}^{|C|} \Gamma_j(t) \quad (8)$$

subject to

$$\lambda_j(t) \leq \Omega_j, \forall c_j \in C, \quad (9)$$

$$\sum_{j=1}^{|C|} x_{ij}(t) = 1, \forall s_i \in S, \quad (10)$$

$$\sum_{k=1}^{|D(t)|} w_{ik}(t) \leq R_{cache}, \forall s_i \in S, \quad (11)$$

$$x_{ij}(t) = x_{ij}(t-1) + y_{ij}(t-1), \quad \forall s_i \in S, \forall c_j \in C \quad (12)$$

$$\sum_{i=1}^{|S|} w_{ik}(t) \leq 1, \forall d_k(t) \in D(t), \quad (13)$$

$$\delta_k(t) \leq \delta_k^{max}, \forall d_k(t) \in D(t) \quad , \quad (14)$$

where $\alpha \in [0,1]$ is a weighting factor to control the relative importance of controller-switch communication cost and inter-controller communication cost. The relation in (9) ensures that none of the controllers is overloaded. As mentioned in (6), the value of $\lambda_j(t)$ is estimated based on the probability of activation considering random or periodic activation model. The truth that each switch belongs to a single master controller is presented in (10). Additionally, (11) ensures that the number of cached flow-rules in a switch does not exceed the maximum allowable limit R_{cache}. The relation in (12) ensures that a controller can be assigned with the master role for a switch in time-slot t if and only if it is the master or slave controller for that switch in the previous time-slot. Moreover, (13) ensures that a device can have cached rule only in single switch as each switch has limited rule storage capacity. Finally, (14) expresses the delay constraint for each device, where δ_k^{max} is the maximum allowable delay for $d_k(t)$.

Theorem 1. *The cache-enabled minimum cost master controller assignment (CMCA) problem is NP-hard.*

Proof: **Let us consider a particular instance of the CMCA problem by excluding the rule caching at switches. In this case, we have $|C|$ controllers and $|S|$ switches. Each controller-switch association increases traffic intensity at the corresponding controller. In addition, each controller has a maximum capacity. For example, a switch s_i can be associated with a master controller c_j only if $\lambda_j(t) < \Omega_j$. A feasible solution ensures completeness constraint in (10) that each switch is assigned to exactly one master controller. The goal of the problem is to find a feasible solution that minimizes the total control traffic intensity. This is in the form of a generalized assignment problem [20], which has been proved as NP-hard. Hence, the CMCA problem is also NP-hard.** ■

As the optimization problem in (8) is NP-hard, it is difficult to obtain a solution in reasonable time. Therefore, we propose a master controller assignment scheme based on the branch and bound technique [21] that can efficiently determine near-optimal solutions.

IV. CORE: The Proposed Scheme

CORE contains three modules for the purpose of — (a) mobility prediction, (b) rule-caching, and (c) master controller assignment. The mobility prediction module analyzes mobility history of IoT devices to predict device-switch association information. Thereafter, the selected flow-rules are cached by the rule-caching module to reduce the control plane load. Finally, the master controller assignment module determines optimal controller-switch associations.

A. Mobility Prediction

We determine the control plane load based on the number of control messages it handles during a time-slot. However, the number of control messages or the number of new flows depends on the devices associated with the switches at a time-slot. Consequently, we need to predict the device-switch associations, while considering the movement history of the devices. Subsequently, we use this prediction data to cache device specific flow-rules in switches and determine optimal controller-switch associations. To predict future positions of the IoT devices, we use the existing Markov Predictor [8].

1) Markov Predictor: **Several location predictors are present in existing literature [8]. Markov Predictor is one of the most popular location prediction algorithms to predict the future location of a mobile device based on its mobility history. An $O(m)$ Markov Predictor considers m most recent locations of a mobile device and predicts the next location. Markov Predictor consumes less space and performs better than other popular**

predictors for small values of m [22]. Therefore, we use order-m ($O(m)$) Markov Predictor to determine the future location of each device. Therefore, for predicting the location of a device $d_k(t)$, the components of an order-m ($O(m)$) Markov Predictor are:

- ***Input***: The input set $H_k(t) = \{\{L_{t,k}, \mathcal{T}_{t,k}, V_{t,k}\}, P_{t,k}\}$ represents the mobility history of $d_k(t)$ at time-slot t, where $L_{t,k} = \{l_{tk1}, l_{tk2}, \ldots, l_{tkn}\}$ is the set of locations or meaningful places that the device visits, $\mathcal{T}_{t,k} = \{\tau_{tk1}, \tau_{tk2}, \ldots, \tau_{tkn}\}$ denotes the set of arrival times at the locations in $L_{t,k}$, $V_{t,k} = \{v_{tk1}, v_{tk2}, \ldots, v_{tkn}\}$ is the set of durations of stay at each location in $L_{t,k}$, and $P_{tkij} \in P_{t,k}$ represents the transition probability from location l_{tki} to location l_{tkj}, $i \neq j$.
- ***Output***: The output $l_{t+1,k} \in L_{t,k}$ represents the predicted location of device $d_k(t)$ in time-slot $t+1$.
- ***Context***: The context is formulated as h=$L_{t,k}(n-m+1, n)$=$\{l_{tk(n-m+1)}, l_{tk(n-m+2)}, \ldots, l_{tk(n-1)}, l_{tkn}\}$.

Markov Predictor extracts the context h from the input set $H_k(t)$ and examines the duration of stay V_l at a location l that follows h. Mathematically,

$$V_l = \{v_{tki} | v_{tki} = \tau_{tk(i+1)} - \tau_{tki}, \text{ where } L_{t,k}(i-m+1, i+1) = hl\} \quad (15)$$

From each V_l, we compute the conditional probability $P_l(\tau \leq v < \tau + \Delta\tau | h, \tau)$ that the device shifts to location l within $\Delta\tau$ time beyond the current elapsed time τ. We consider $\Delta\tau$ as the remaining time of the current time-slot. Therefore, for a given context h and elapsed time τ, the probability that a device moves to a possible location l within $\Delta\tau$ time is:

$$P(l|h,\tau) = P(l) P_l(\tau \leq v < \tau + \Delta\tau | h, \tau), \quad (16)$$

where $P(l)$ denotes the transition probability of every possible next location l which is calculated as $P(l_{tk(n+1)} = l | L_{t,k}) \approx \widehat{P}(l_{tk(n+1)} = l | L_{t,k}) = \frac{N(hl, L_{t,k})}{N(h, L_{t,k})}$, where $N(hl, L_{t,k})$ signifies the number of occurrences of hl in the set $L_{t,k}$. Therefore, the output of the Markov Predictor which is the most likely next location of $d_k(t)$ is given by:

$$l_{tk(n+1)} = \underset{l \in L_{t,k}}{\operatorname{argmax}} P(l_{tk(n+1)} = l) \quad (17)$$

If $N(h, L_{t,k}) = 0$, the $O(m)$ Markov Predictor fails to return a result. Therefore, we use fallback Markov Predictor [8] which backtracks to an $O(m-1)$ Markov predictor whenever an $O(m)$ Markov Predictor fails to return a result. The $O(0)$ Markov Predictor yields the location that occurs most frequently in the location history set $L_{t,k}$.

Algorithm 1 Mobility Prediction Algorithm

INPUTS: $H_k(t-1)$, h
OUTPUT: $z(t)$
PROCEDURE:
1: Extract $L_{t-1,k}$ from $H_k(t-1)$
2: Compute V_l at possible locations l using (15)
3: Calculate $P(l|h,\tau)$ using (16)
4: Predict the next location using (17)
5: Select the nearest AP which covers the predicted location and matches the radio access capability of the $d_k(t)$
6: Set $z_{ik}(t) \leftarrow 1$ if s_i is associated with the selected AP

Algorithm 1 presents the steps required for mobility prediction of a device $d_k(t)$ and the formulation of device-switch association $z_{ik}(t)$. The Mobility Prediction Algorithm (MPA) is executed for each device $d_k(t)$. An $O(m)$ Markov Predictor returns a location where the device is predicted to be present in time-slot t. We consider that $d_k(t)$ associates with the nearest AP that covers the predicted location and matches its radio access capability. Let s_i be the switch associated with the selected AP. Therefore, MPA predicts the device-switch association $z_{ik}(t) = 1$.

B. Rule-Caching

To estimate rule popularity, rule-caching module sorts the flow-rules in each switch in descending order of the received packet count. Let R_i be the set of flow-rules in s_i. Therefore, the rule popularity is denoted by $\Theta = \{\theta_1, \theta_2, \theta_3, \ldots, \theta_{|R_i|}\}$, where $\theta_j \in [0,1]$ is the probability that an incoming flow matches with the j^{th} flow-rule. In this work, we assume that rule popularity satisfies the Zipf distribution [23]. Therefore, the popularity of the j^{th} ordered flow-rule is given by:

$$\theta_j = \frac{\frac{1}{j^\gamma}}{\sum_{a=1}^{|R_i|} \frac{1}{a^\gamma}}, \quad (18)$$

where $\gamma \in [0,1]$ denotes the skewness of the rule popularity. The value $\gamma = 0$ signifies uniform popularity distribution. Whereas, a larger γ implies more uneven rule popularity. Algorithm 2

Algorithm 2 Rule-Caching Algorithm

INPUTS: R_i, γ, $z(t)$
OUTPUT: $w(t)$
PROCEDURE:
1: Compute popularity of the rules in R_i using (18)
2: Sort the flow-rules in descending order of popularity
3: **for each** rule r_e in the sorted list **do**
4: Select the device $d_k(t)$ whose flow type maps to r_e
5: **if** $z_{ik}(t) == 1$ and r_e not cached **then**
6: Delete the least popular rule from cache if cache is full
7: $T_e \leftarrow \frac{1}{Q_k(t-1)} + (T_0 - \delta_k^{max})$, $w_{ik}(t) \leftarrow 1$
8: **end if**
9: **end for**

presents the steps of the proposed greedy solution for caching rules in each switch s_i. For each switch s_i, the Rule-Caching Algorithm (RCA) sorts the flow-rules present in the rule-space of the switch based on the rule popularity calculated using (18). For each flow-rule r_e which maps to the flow type of $d_k(t)$, RCA checks whether $z_{ik}(t) == 1$ from the output of MPA. In addition, RCA checks whether the rule is already cached by s_i. If the cache size reaches its maximum limit R_{cache}, RCA deletes the least popular rule from the cache by setting its timeout as the default timeout T_0. For caching r_e, RCA sets the timeout value as $T_e = \frac{1}{Q_k(t-1)} + (T_0 - \delta_k^{max})$. This timeout value ensures that latency-sensitive flows are prioritized over other flows as a larger timeout value signifies lower chance of flow-table miss.

C. Master Controller Assignment

We derive the optimization problem for minimum cost master controller assignment from the joint optimization problem of cache-enabled minimum cost master controller assignment stated in (8). Hence, for a given caching policy $w(t)$, the optimization problem P_0 for minimum cost master controller assignment

(MCA) is given by:

$$\underset{x(t)}{\text{Minimize}} \quad \alpha \sum_{j=1}^{|C|} \rho_j(t) + (1-\alpha) \sum_{j=1}^{|C|} \Gamma_j(t) \tag{19}$$

subject to

$$\lambda_j(t) \leq \Omega_j, \forall c_j \in C, \tag{20}$$

$$\sum_{j=1}^{|C|} x_{ij}(t) = 1, \forall s_i \in S, \tag{21}$$

$$x_{ij}(t) = x_{ij}(t-1) + y_{ij}(t-1), \quad \forall s_i \in S, \forall c_j \in C \tag{22}$$

$$\sum_{i=1}^{|S|} w_{ik}(t) \leq 1, \forall d_k(t) \in D(t), \tag{23}$$

$$\delta_k(t) \leq \delta_k^{max}, \forall d_k(t) \in D(t) \tag{24}$$

The MCA problem is non-convex because of the presence of binary decision variable. Therefore, we use the branch and bound technique for solving the MCA problem [21]. The branch and bound technique defines a common structure to solve a wide range of non-convex optimization problems. It follows the divide and conquer paradigm and starts with the original optimization problem. Subsequently, the algorithm computes lower bound (LB) of the global optimum by applying the lower-bounding method to the original problem with the complete feasible region. The branch and bound technique uses a branching rule to divide the feasible region into multiple disjoint subregions when no optimal solution is achieved. Each subregion signifies a subproblem having the same objective function and constraints as the original problem.

For each subproblem, the LB is computed. A subproblem is discarded if its LB is greater than the optimal LB of the original problem. Otherwise, the subproblem is partitioned further if no optimal solution is found. Hence, a search tree with the original problem as root is generated. The process continues until all nodes have been visited, or an optimal solution is achieved. Therefore, the master controller assignment scheme for the MCA problem has two significant components — 1) branching method and 2) lower-bounding method.

1) Branching Method: **Let P_0 denote the MCA problem stated in (19). The branching method starts with P_0 as the root of the search tree. The total number of levels in the tree is $|S|+1$ starting from level 0. Each level corresponds to the choice of a master controller for each switch s_i. For example, level 1 corresponds to the choice of a master controller for switch s_1. Therefore, each node at a level denotes a subproblem. At each level, we partition the leaves or subproblems. Each child node of a node P_v at level l corresponds to a feasible master controller for s_{l+1}. Let C_v be the set of feasible master controllers for s_{l+1}. From constraint (22), we find that a controller $c_j \in C$ is a member of C_v if $x_{l+1,j}(t-1) + y_{l+1,j}(t-1) = 1$. Therefore, the number of children of P_v is $|C_v|$.**

2) Lower-Bounding Method: **Initially, we construct a lower bound for the original MCA problem. To find the initial lower bound, we construct a relaxed problem MCA-R by removing the controller capacity constraint in (20). Therefore, each switch freely selects the master controller so that the control plane cost is minimum. For a given switch s_i, the cost for the assignment to a master controller c_j is $U_{ij} = \alpha\rho_j(t) + (1-\alpha)\Gamma_j(t)$, where $x_{ij} = 1$ and $x_{ij'} = 0$ for all $j \neq j'$. Therefore, the cost of a minimum cost controller-switch association for a given switch s_i is expressed as $U_{ij^i} = \min_{\forall c_j}\{U_{ij}\}$. Hence, the LB for problem P_0 is $LB_0 = \sum_{i=1}^{|S|} U_{ij^i}$. Subsequently, we find the LB for each subproblem P_v where $v \neq 0$. Let $x^v(t)$ be the allocation matrix for the branch ending at a node P_v at level l. Therefore, the initial value of LB is $LB_v^0 = \sum_{\forall c_j \in C, s_i \in S} U_{ij} x^v(t)$. $S' = s_{l+1}, s_{l+2}, \ldots, s_{|S|}$ denotes the set of unassigned switches for the current branch. For each switch $s_i \in S'$, we find the minimum cost controller-switch association that satisfies the constraints (20), (21), (22) and (24). Therefore, the LB of P_v is given by:**

$$LB_v = LB_v^0 + \sum_{s_i \in S'} \min_{\forall c_j}\{U_{ij}\} \tag{25}$$

Algorithm 3 Master Controller Assignment Algorithm

INPUTS: P_0, C, S, $z(t)$, $w(t)$
OUTPUT: $\{x^*(t), u^*\}$
PROCEDURE:

```
P ← {P_0}, u* = +inf, x*(t) = 0
while P ≠ φ do
    Select a node P_v ∈ P
    P ← P − {P_v}
    Apply branching method to P_v and generate subproblems P_{v_1}, P_{v_2}, ..., P_{v_|C_v|}
    for each P_{v_a} do
        Compute LB_{v_a} using (25)
        if LB_{v_a} > u* then
            Delete P_{v_a}
        else if P_{v_a} gives a complete solution {x'(t), u'} then
            u* ← u', x*(t) ← x'(t)
        else
            P ← P ∪ {P_{v_a}}
        end if
    end for
end while
```

3) Master Controller Assignment Algorithm: **Algorithm 3 shows the branch and bound procedure to solve the MCA problem. The Master Controller Assignment Algorithm (MCAA) initializes the values of optimal solution $x^*(t)$ and the optimal objective value u^*. In addition, MCAA adds the root node P_0 to the set of live nodes P. For each live node $P_v \in P$, MCAA applies branching method to generate child nodes or subproblems. A subproblem is deleted if it has a LB greater than the optimal objective value u^*. The values $x^*(t)$ and u^* are updated when a subproblem generates a complete solution with each switch assigned to a master controller. Otherwise, the subproblem is added to the set of live nodes P. The output of MCAA signifies an optimal master controller assignment $x^*(t)$ for time-slot t. At time-slot $t-1$, we compute $x^*(t)$ and change the controller-switch assignments accordingly by using Role-Change messages [3]. Additionally, we deactivate the controllers which have no assigned switches.**

The proposed scheme applies to any controller placement strategy. We explore control plane load balancing in networks where controllers are already placed, and load imbalance occurs due to the dynamic nature of IoT traffic. In this scenario, we propose an approach to distribute the control plane load optimally across the available controllers as the installation of additional controllers is not possible always due to budgetary constraints. In each time-slot, CORE determines the optimal controller-switch associations and makes changes only if the existing association is not optimal in terms of the current load.

V. Performance Evaluation

A. Simulation Settings

We evaluate the performance of CORE by implementing a discrete event simulator in MATLAB. For the simulation, we consider wearable IoT devices with speed $1-2$ m/s. The

simulation parameters are depicted in Table I. Additionally, we consider random controller placement. In addition, we consider an equal number of randomly and periodically activated devices. We conduct two sets of experiments with $\alpha = 0.8$ for the performance evaluation. In the first experiment, we consider that 80% devices generate high traffic. This experiment evaluates the performance of the proposed scheme in the presence of high IoT traffic volume. In the second experiment, we set the percentage of latency-sensitive devices as 80% to analyze the performance for time-critical IoT applications. Table II shows the specific parameters considered for categorizing high traffic generating and latency-sensitive devices.

TABLE I: Simulation Parameters

Parameter	Value
Network topology	$8-$pod Fat-tree [24]
Simulation Area	500 m $\times$ 500 m
Mobility model	Gauss-Markov [25]
Number of IoT devices $\lvert D(t)\rvert$	$200 - 2500$
Speed of IoT devices	$1 - 2$ m/s [26]
Number of switches $\lvert S\rvert$	20
Flow-rule default timeout T_0	10 s
Number of controllers $\lvert C\rvert$	5
Controller capacity Ω	$7200-10800$ K req/time-slot [27]
Average packet size	$94 - 234$ bytes [28]
Mean data rate	$462 - 11388$ bytes/s [28]
Maximum allowable delay δ^{max}	$0.001 - 1$ s [5]
Time-slot duration ϵ	1 hour
Skewness of rule popularity γ	0.56
Shape parameter β_1	3 [4]
Shape parameter β_2	4 [4]
Weighing factor α	$0.2 - 0.8$

B. Benchmark Schemes

We compare CORE with existing switch migration-based schemes — DCP-SA [7] and ESMLB [15]. DCP-SA considers flow setup delay and inter-controller communication in the presence of dynamic traffic for controller placement and switch migration. ESMLB considers the control traffic generated by the switches as primary criteria for switch migration-based load balancing in SDIoT control plane. On the other hand, CORE considers flow setup delay, inter-controller communication, dynamic network traffic, device mobility, and heterogeneous QoS demands to determine feasible controller-switch assignment.

C. Performance Metrics

The performance metrics considered for evaluating the proposed scheme are as follows:

- **Prediction accuracy:** Prediction accuracy shows the correctness of mobility prediction for the IoT devices.
- **Control plane cost:** Control plane cost is the cumulative cost of controller-switch communication cost and inter-controller communication cost, as mentioned in (8). We evaluate this metric to estimate the load on the control plane as a high controller load increases the cost.
- **Peak traffic intensity:** We calculate the peak traffic intensity across all controllers to analyze the distribution of control traffic. Mathematically, the peak traffic intensity is given as $max\left(\rho_j(t)\right), \forall c_j \in C$.
- **QoS violated flows:** QoS violated flows are the flows which do not satisfy end-to-end delay requirement of the flow type. We evaluate this metric to show the efficiency of CORE in terms of QoS.

D. Observations and Results

1) Prediction Accuracy: For the simulation, we fix the order of the Markov predictor as $k = 3$. We use a Twitter dataset [29] involving $200-1000$ devices to analyze the prediction accuracy of the Markov predictor. Figure 4 shows that the average prediction accuracy is 83.72%. From the simulation, we infer that CORE is capable of correctly predicting the device locations for a significant number of cases, although the mobility pattern and speed of the devices are highly dynamic. Additionally, based on the dataset values, we observe that the prediction accuracy is almost uniform for varying device count. In particular, the prediction accuracy with 1000 devices is 0.05% less than that with 800 devices.

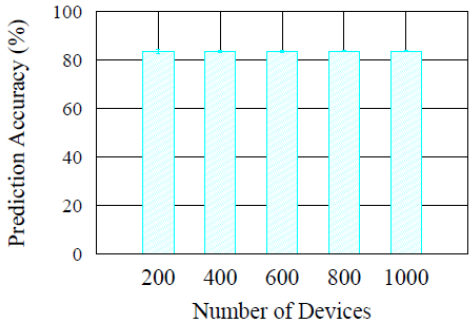
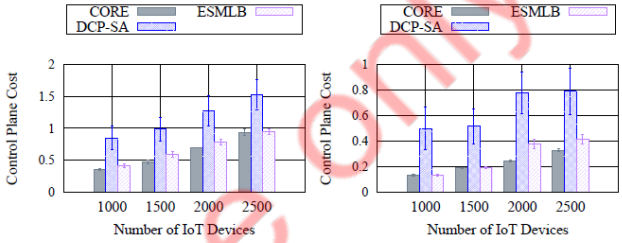


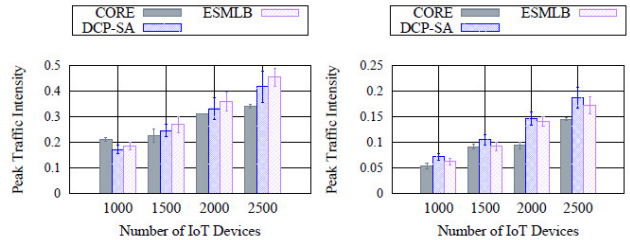
Fig. 4: Prediction Accuracy



(a) With 80% High Traffic Generating Devices (b) With 80% Latency-Sensitive Devices

Fig. 5: Control Plane Cost

2) Control Plane Cost: We analyze the average control plane cost for different number of IoT devices. Figure 5a shows the performance of CORE compared to DCP-SA and ESMLB for high traffic load. CORE achieves 46.94% and 9.82% reduction in control plane cost as compared to DCP-SA and ESMLB, respectively. Figure 5b shows the average control plane cost when the majority of the devices are latency-sensitive. In this case, CORE achieves 65.63% and 20.14% reduction in control plane cost as compared to DCP-SA and ESMLB, respectively.



(a) With 80% High Traffic Generating Devices (b) With 80% Latency-Sensitive Devices

Fig. 6: Peak Traffic Intensity

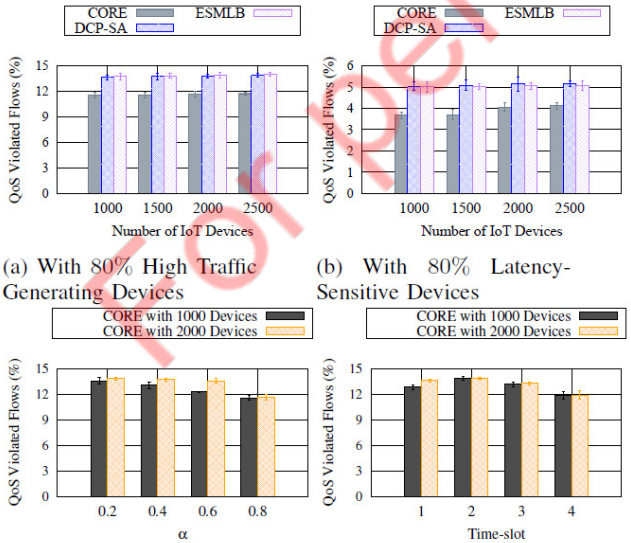
3) Peak Traffic Intensity: We analyze the peak traffic intensity across the controllers. Figure 6a shows the peak traffic intensity for the first experiment. We observe that CORE

TABLE II: Device Category

Category	Average packet size (bytes)	Mean data rate (bytes/s)	Maximum allowable delay (s)
High traffic generating	234 [28]	11388 [28]	$0.001 - 1$ [5]
Latency-sensitive	94 [28]	462 [28]	$0.001 - 0.25$ [5]

performs better than the benchmark schemes in terms of peak traffic intensity. In particular, for 2500 devices, the peak traffic intensity of CORE is 18.66% and 25.27% less as compared to DCP-SA and ESMLB, respectively. Figure 6b shows the peak traffic intensity for the second experiment. We observe that CORE achieves 23.08% and 16.67% reduction in peak traffic intensity as compared to DCP-SA and ESMLB, respectively.

4) QoS Violated Flows: **We evaluate the percentage of flows that violate respective QoS demands due to delay higher than the maximum allowable delay. Figure 7a reports the performance of CORE compared to DCP-SA and ESMLB when the majority of the devices generate high traffic. We observe that the percentage of QoS violated flows is less for CORE even when the number of devices is high. In particular, for 2500 IoT devices, CORE achieves 15.64% and 16.33% decrease in QoS violation as compared to DCP-SA and ESMLB, respectively. Figure 7b shows the number of QoS violated flows with high number of latency-sensitive devices. For this experiment, CORE achieves 23.73% and 22.82% better performance as compared to DCP-SA and ESMLB, respectively. Figure 7c shows that the QoS violation for CORE decreases with increasing α. This is because minimizing the control plane cost with a high α implies more reduction of the traffic intensity. Therefore, each controller handles fewer Packet-In requests, and the queueing delay at the controller as well as the end-to-end delay decrease. Figure 7d shows the percentage of QoS violated flows for a simulation duration of one hour with 80% high traffic generating devices, $\alpha = 0.8$, and $\epsilon = 900s$. From simulation results, we observe that the QoS violation increases initially and reduces in later time-slots. Although the traffic load varies based on the different activation models followed by the IoT devices, CORE optimizes controller-switch assignments based on the existing assignments and present traffic load. Therefore, more optimal assignments are obtained in later time-slots.**



(a) With 80% High Traffic Generating Devices

(b) With 80% Latency-Sensitive Devices

(c) Effect of α with 80% High Traffic Generating Devices

(d) Variation with Time for 80% High Traffic Generating Devices and $\alpha = 0.8$

Fig. 7: QoS Violated Flows

E. Discussion

From the simulation result, we observe that CORE significantly outperforms the benchmark schemes, especially for the IoT environment. The majority of the IoT flows are latency-sensitive, and CORE has low control plane cost for a high number of latency-sensitive devices. This is because the rule-caching module prioritizes latency-sensitive flows and reduces controller-switch communication. It is noteworthy that with less number of IoT devices, the peak traffic intensity of CORE is similar to the benchmark schemes. This is because, at a lower load, the control traffic is well-distributed across the controllers. However, IoT networks expect the presence of a massive number of IoT devices, and CORE reduces the peak traffic intensity for a high number of IoT devices. Therefore, we deduce that CORE is more suitable for reducing control plane load in the IoT environment than the benchmark scheme.

VI. Conclusion

In this paper, a prediction-based scheme to reduce the control plane load in SDIoT was presented. We proposed rule-caching and master controller assignment schemes under the consideration of heterogeneous attributes of IoT devices. We observe that the proposed scheme, CORE, ensures appropriate distribution of control traffic across the controllers while minimizing the control plane cost and QoS violation. Simulation results indicate that CORE reduces the average control plane cost under both varying traffic load and varying QoS demand. Specifically, for high traffic load, the average control plane cost decreased approximately by 46.94% and 9.82% as compared to DCP-SA and ESMLB, respectively.

The future extension of this work includes the implementation in a real testbed. Additionally, we plan to increase prediction accuracy further. We also plan to modify CORE, under the consideration of the placement of the controllers.

References

[1] I. Maity, A. Mondal, S. Misra, and C. Mandal, "CURE: Consistent Update With Redundancy Reduction in SDN," *IEEE Trans. Commun.*, vol. 66, no. 9, pp. 3974–3981, Sep. 2018.

[2] M. T. I. ul Huque, W. Si, G. Jourjon, and V. Gramoli, "Large-Scale Dynamic Controller Placement," *IEEE Trans. Netw. Service Manag.*, vol. 14, no. 1, pp. 63–76, Mar. 2017.

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

[4] M. Mozaffari, W. Saad, M. Bennis, and M. Debbah, "Mobile Unmanned Aerial Vehicles (UAVs) for Energy-Efficient Internet of Things Communications," *IEEE Trans. Wireless Commun.*, vol. 16, no. 11, pp. 7574–7589, Nov. 2017.

[5] S. F. Abedin, M. G. R. Alam, S. M. A. Kazmi, N. H. Tran, D. Niyato, and C. S. Hong, "Resource Allocation for Ultra-Reliable and Enhanced Mobile Broadband IoT Applications in Fog Network," *IEEE Trans. Commun.*, vol. 67, no. 1, pp. 489–502, Jan. 2019.

[6] W. Ayoub, A. E. Samhat, F. Nouvel, M. Mroue, and J. Prévotet, "Internet of Mobile Things: Overview of LoRaWAN, DASH7, and NB-IoT in LPWANs standards and Supported Mobility," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1561–1581, Oct. 2019.

[7] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba, "Dynamic Controller Provisioning in Software Defined Networks," *Proc. IEEE CNSM*, pp. 18–25, Oct. 2013.

[8] S. Bera, S. Misra, and M. S. Obaidat, "Mobi-Flow: Mobility-Aware Adaptive Flow-Rule Placement in Software-Defined Access Network," *IEEE Trans. Mobile Comput.*, vol. 18, no. 8, pp. 1831–1842, Aug. 2019.

[9] K. Sood, S. Yu, and Y. Xiang, "Performance Analysis of Software-Defined Network Switch Using $M/Geo/1$ Model," *IEEE Commun. Lett.*, vol. 20, no. 12, pp. 2522–2525, Dec. 2016.

[10] R. Muñoz, R. Vilalta, N. Yoshikane, R. Casellas, R. Martínez, T. Tsuritani, and I. Morita, "Integration of IoT, Transport SDN, and Edge/Cloud Computing for Dynamic Distribution of IoT Analytics and Efficient Use of Network Resources," *Journal of Lightwave Technology*, vol. 36, no. 7, pp. 1420–1428, Apr. 2018.

[11] P. Bellavista, C. Giannelli, T. Lagkas, and P. Sarigiannidis, "Quality Management of Surveillance Multimedia Streams Via Federated SDN Controllers in Fiwi-Iot Integrated Deployment Environments," *IEEE Access*, vol. 6, pp. 21 324–21 341, 2018.

[12] 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*, pp. 1–9, Sep. 2013.

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

[14] A. Dixit, F. Hao, S. Mukherjee, T. V. Lakshman, and R. Kompella, "Towards an Elastic Distributed SDN Controller," *Proc. ACM SIGCOMM*, vol. 43, no. 4, pp. 7–12, Aug. 2013.

[15] K. S. Sahoo, D. Puthal, M. Tiwary, M. Usman, B. Sahoo, Z. Wen, B. P. S. Sahoo, and R. Ranjan, "ESMLB: Efficient Switch Migration-Based Load Balancing for Multicontroller SDN in IoT," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5852–5860, 2020.

[16] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey," in *Proc. IEEE*, vol. 103, no. 1, Jan. 2015, pp. 14–76.

[17] D. Wu, X. Nie, E. Asmare, D. I. Arkhipov, Z. Qin, R. Li, J. A. McCann, and K. Li, "Towards Distributed SDN: Mobility Management and Flow Scheduling in Software Defined Urban IoT," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 6, pp. 1400–1418, Jun. 2020.

[18] K.-K. Yap, M. Kobayashi, R. Sherwood, N. Handigol, T.-Y. Huang, M. Chan, and N. McKeown, "OpenRoads: Empowering research in mobile networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 1, pp. 125–126, 2010.

[19] M. He, A. Varasteh, and W. Kellerer, "Toward a Flexible Design of SDN Dynamic Control Plane: An Online Optimization Approach," *IEEE Trans. Netw. Service Manag.*, vol. 16, no. 4, pp. 1694–1708, 2019.

[20] L. Õzbakir, A. Baykasoğlu, and P. Tapkan, "Bees algorithm for generalized assignment problem," *Applied Mathematics and Computation*, vol. 215, no. 11, pp. 3782 – 3795, 2010.

[21] A. Jarray and A. Karmouch, "Decomposition Approaches for Virtual Network Embedding With One-Shot Node and Link Mapping," *IEEE/ACM Trans. Netw.*, vol. 23, no. 3, pp. 1012–1025, Jun. 2015.

[22] 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.*, vol. 11, no. 6, pp. 1047–1059, Jun. 2012.

[23] A. F. Tayel, S. I. Rabia, and Y. Abouelseoud, "An Optimized Hybrid Approach for Spectrum Handoff in Cognitive Radio Networks With Non-Identical Channels," *IEEE Trans. Commun.*, vol. 64, no. 11, pp. 4487–4496, Nov. 2016.

[24] M. Al-Fares, A. Loukissas, and A. Vahdat, "A Scalable, Commodity Data Center Network Architecture," *Proc. ACM SIGCOMM*, pp. 63–74, 2008.

[25] T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Communications and Mobile Computing*, vol. 2, no. 5, pp. 483–502, Aug. 2002.

[26] R. W. Bohannon, "Comfortable and maximum walking speed of adults aged 20-79 years: reference values and determinants," *Age and Ageing*, vol. 26, no. 1, pp. 15–19, Jan. 1997.

[27] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On Controller Performance in Software-defined Networks," *Proc. USENIX Conf. Hot Topics Manage. Internet, Cloud, Enterprise Netw. Services*, pp. 1–6, 2012.

[28] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Characterizing and classifying IoT traffic in smart cities and campuses," in *Proc. IEEE INFOCOM Workshops*, May 2017, pp. 559–564.

[29] X. Liu, Y. Liu, Y. Chen, and L. Hanzo, "Trajectory Design and Power Control for Multi-UAV Assisted Wireless Networks: A Machine Learning Approach," *IEEE Trans. Veh. Technol.*, vol. 68, no. 8, pp. 7957–7969, Aug. 2019.