LS-SDV: Virtual Network Management in Large-Scale Software-Defined IoT

LS-SDV: Virtual Network Management in Large-Scale Software Defined IoT

He Li, Kaoru Ota, Mianxiong Dong
Department of Information and Electronic Engineering, Muroran Institute of Technology,
Muroran, Hokkaido, Japan.
E-mail: {heli, ota, mxdong}@mmm.muroran-it.ac.jp

***Abstract*—Internet of Things (IoT) becomes a very important area for providing various services on connected smart devices. For isolation of different services in IoT, software defined networking (SDN) based virtual networks will be a scalable and flexible solution. However, in a large-scale IoT, as smart devices will move long distance between different positions, virtual network management becomes very difficult in providing network services. In this paper, we propose LS-SDV, an efficient virtual network management framework in large-scale software defined IoT (SDIoT). In this framework, we design a two-layer distributed control plane to manage devices and virtual networks in a large-scale environment. To the best of our knowledge, LS-SDV is the first work to apply distributed control plane for virtual network management in softwarized networks. Moreover, based on the novel structure of LS-SDV, we also provide a solution for network flow scheduling through network analysis. We evaluate the performance of our framework and virtual network management by extensive simulation and experiment in an open SDN framework.**

***Index Terms*—Software Defined Networking, Network Virtualization, Internet of Things (IoT)**

I. Introduction

Internet of Things (IoT) is an important concept to provide various convenient services in human life through connected smart devices [1], [2], [3]. As large IoT infrastructures are usually expensive, IoT providers want to provide different services in a single network infrastructure [4], [5], [6]. Thus, network virtualization becomes a solution to support different services in a single IoT infrastructure with isolated virtual networks [7]. As software defined networking (SDN) provides a scalable and flexible programmable network management platform, SDN-based network virtualization is an emerging solution for providing virtual networks during forwarding network flows [8].

Usually, as the coverage of a single wireless IoT network is limited into a small area with several access points (APs), it is not very hard to manage all network flows by a centralized controller [9]. Meanwhile, as the moving area of smart devices is also limited, each forwarding device can store all flow control rules with some modifications to the wired SDN for device mobility [10], [11], [12]. However, as more and more IoT infrastructures begin to provide services in urban or even larger scale environments with heterogeneous networks, a single controller can hardly afford such complex network flow management with enough quality of service (QoS) for each virtual network.

A potential solution is to build a distributed control plane [13]. As SDN decouples the network control and forwarding function, the control plane is independent of the forwarding devices [14]. Therefore, as the distributed structure can provide better performance and scalability than the centralized structure, a distributed control plane can increase the capability of network control for large-scale networks. There are some solutions to the distributed SDN control plane that deploy multiple controllers for large-scale wired networks. For large-scale networks, the problem becomes difficult because of the device mobility and heterogeneous network structures [15], [16]. For example, in a wired network environment, as the end node has determined position and connections, the corresponding network controller can be easily determined [17]. In the mobile environment, the problem becomes to find the corresponding controller when the device moves from one position to another [18].

Thus, in this paper, we propose *large-scale software defined virtualized networking* (LS-SDV), a virtual network management framework in the large-scale software defined IoT (SDIoT) with the nature of the distributed control plane. In LS-SDV, we design a two-layer overlay structure to organize distributed controllers because of the hierarchical structure of network virtualization. Virtual devices can be used to enhance IoT services at the SDN-based network by enhancing the utilization of smart devices [19]. In IoT systems, the nodes are limited by their energy lifetime. Therefore, at the step of placing a controller, it is necessary to consider the cost of communicating with IoT nodes, in order to reduce the overall consumption. We introduce distributed hashing for addressing virtual devices to corresponding controllers. After analyzing the network status with the network information, we state the problem of flow scheduling problem in virtual networks and give a solution to optimize the network performance. Finally, we implement our framework in a simulator and test the performance with an open SDN framework. The evaluation results prove that LS-SDV performs better than other distributed or single control plane solutions.

The main contributions of this paper are summarized as follows.

- We first study the virtual networks in large-scale IoT and propose a two-layer overlay distributed control plane for virtual network management. We also study and exam the mobility and other issues in virtual network management.
- We study the problem of network flow scheduling in

virtual networks with LS-SDV. We propose a solution for a determined traffic and flow scheduling in order to optimize the network performance of each virtual network.

- We evaluate the performance of LS-SDV through extensive simulations and using an open SDN framework. We also compare our work with the other methods and the results show that our method performs better on network performance in the large-scale IoT environment.

The rest of this paper is organized as follows. Section II reviews the related work. The framework design and distributed mechanisms of LS-SDV are introduced in Section III. The network flow scheduling is proposed in Section IV. Section V gives the simulation results. Finally, Section VI concludes this paper and gives the future work.

II. Related Work

In this section, we first discuss some related works on SDIoT then we introduce some previous works focusing on the distributed control plane.

A. SDN and IoT

As IoT systems often choose wireless networks as the connection between devices and servers, we first list and discuss some works focusing on software defined wireless networks.

As SDN decouples the network control and data forwarding, it needs a programmable data plane for network control in the control plane. Thus, OpenRadio [20] is a programmable data plane specifically designed for wireless networks. OpenRadio decoupled processing and decision in wireless protocols and provided a programmable interface while hiding the execution in packet forwarding. It provides a fundamental solution of software defined wireless networks.

Multinetwork information architecture (MINA) [11] is a middleware solution for SDN-enable heterogeneous wireless networks. MINA introduced a tree-based overlay network to study the heterogeneous network view and proposed a centralized network scheduling method through the observed network view.

Loading balance method is required to respond to traffic burst and relieve the heavy load. Chen et. al [21] proposes a traffic-aware load balancing method to achieve various requirements by traffic identification and scheduling ways.

To upgrade the traditional IoT system into SDN-based one, the update cost is the main concern for administers. BLLC [22] is proposed to meet the demand on low cost network update. Such that, a new control rule is applied to compress several control packets into the new one.

Meanwhile, as a global environment for networking innovation, the global environment for networking innovation (GENI) [23] integrates software defined vertical handover to support mobile environments. Therefore, it is possible to provide a software defined network for IoT systems with these software defined wireless network solutions. For example, we have proposed an SDN-based radio access network virtualization for social IoT and optimized the capacity of a given radio access network [7]. Muno et. al [24] introduces an integrated framework of IoT, SDN and Edge systems. A scalable solution is proposed to implement both network and cloud layer resource management.

Network function virtualization is used to reduce the cost of the network, the combination of SDN and NFV is promising in IoT systems. SD-NFV based 6LoWPAN [25], [26] is an energy-efficient mechanism to enhance the lifetime of IoT nodes.

Chen et al. [27] is the first work moving SDN to Software-Defined-Mobile-Network (SDMN) architecture, which also brings some problems due to the hybrid properties. They focus on the security issues on three layers: data layer, control layer and application layer. Also, a method named STRIDE is proposed to avoid attacks in all three layers.

B. Distributed SDN Control Plane

Distributed SDN structure is a major solution for scalable SDN management. Onix [28] is the first work focusing on distributed control plane, which provides a general API for distributed network management. Further, Onix distributes the network view among multiple controllers.

HyperFlow [29] is also a distributed control plane for SDN, which brings good scalability with a customizable number of controllers for different size of networks. Meanwhile, HyperFlow is compatible with the standard OpenFlow protocols, which means it is easy to implement existing control applications with minor modifications.

Kandoo [30] is a framework that provides good scalability without modification to switches. Kandoo introduced a two-layer of controllers in which only the controller in the top layer maintains the entire network view while controllers in the bottom layer can only manage the local devices.

SCL [31] is a simplifying framework to deploy one controller network into distributed SDN systems. This coordination layer helps to seamlessly upgrade the original system to a distributed one in which the consistency mechanisms is complex and difficult to implement in practice.

Thus, locality becomes an important issue in the distributed control plane. Schmid et al. [32] studied the importance of the network view of local controllers. The authors found the optimized locality can improve the network performance and introduced a so-called supported locality model for a better match with the distributed control plane.

Meanwhile, as placement is a solution for locality optimization in distributed systems, Heller et al. [33] studied the control placement problem in the control plane and found the problem is relevant to the network topology.

ElastiCon [34] improves the distributed control plane with better scalability, in which the number of controllers is dynamically changed according to the network traffic. ElastiCon also focused on the load balance between controllers with varying network traffic.

The distributed control plane also has other issues even with better scalability. For example, Canini et al. [35] proposed a distributed SDN control plane which focuses on concurrent and robust policy implementation, which is very important

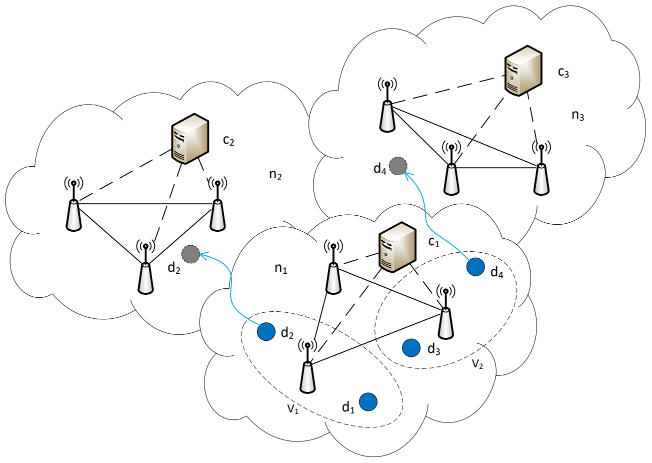


Fig. 1. Example of virtual network management in large-scale IoT

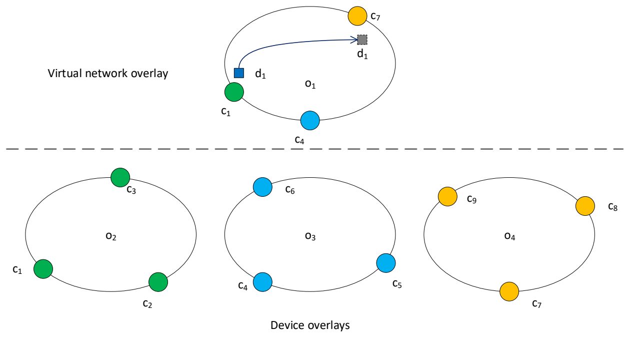


Fig. 2. Two layers overlay structure in LS-SDV

to the network management. DevoFlow [36] is a centralized alternative method of the distributed control plane that adds some load balancing strategies to offload a part of the workload from controllers to switches.

The distributed control plane is inevitable for SDN systems, the guaranteeing performance is also important for these systems. Xie et al. [37] introduces the validation problem of control plane performance and proposes a robust validation framework.

Above works provide examples and experiences on the distributed control plane and show that a distributed control plane is efficient for the SDN scalability especially in a large-scale environment. However, as these works have little consideration on mobility and wireless networks, we add SDN-based mobility management in the IoT network.

III. Framework Design

In this section, we first describe the scenario of virtual network management in large-scale SDIoT. Then, we introduce LS-SDV framework design, addressing in overlays, and device mobility management.

A. Scenario

We use an example to show the virtual network management in a large-scale IoT environment. As shown in Fig. 1, there are four virtual devices, d_1, d_2, d_3, and d_4, connected in a IoT network. We assume all virtual devices perform similar with physical devices and every virtual device is only allowed to connect one virtual network. Therefore, we use two virtual networks, v_1 and v_2, to group these four devices, while virtual network v_1 consists of virtual device d_1 and d_2, and virtual network v_2 consists of virtual device d_3 and d_4.

There are three physical networks, n_1, n_2, and n_3, connecting these virtual IoT devices. We assume these networks covering a large area and each network has one SDN controller for network control, in which controller c_1 manages network n_1, controller c_2 manages network n_2, and controller c_3 manages network c_3.

For supporting virtual networks, each controller needs to maintain the virtual network information and corresponding forwarding strategies. At the beginning, as all devices are connected to network n_1, controller c_1 needs to maintain the forwarding strategy of virtual network v_1 and v_2.

When device d_2 moves to the area of network n_2, as controller c_2 has no information about virtual network v_1, it is impossible to maintain the network communication of virtual network v_1. Similarly, when device d_4 moves to the area of network n_3, the network communication of virtual network v_3 will be interrupted.

A possible solution is that all controllers maintain the virtual network information and forwarding strategies. There are two important issues for maintaining virtual networks in all controllers. First, the update of virtual network information will be very complex. Since all controllers should maintain concurrent virtual network information and forwarding strategies, the information update needs to be spread to each controller instantly for guaranteeing concurrency. Second, maintaining entire virtual network information in all controllers wastes controller resources. Usually, as the moving area of members in a virtual network is limited to several physical networks, it is no need to maintain the virtual network information in other physical networks. To solve these two issues, we discuss our solution in the rest of this section.

B. Framework Design

We propose LS-SDV, a distributed control plane, to support virtual network management in large-scale IoT. The main principle of LS-SDV is that information of a given virtual network is maintained by only one specified controller and a given virtual device is also controlled by only one specified controller. Therefore, the concurrency issue during updating the information of each virtual network or each device in the distributed control plane is avoided.

We investigate how our virtual network and virtual devices are superior to physical devices. The virtual device and virtual network can be controlled by SDN controller from the global view, to provide the same services as physical devices. A physical device can reduce energy consumption as its virtual device is representing it in the network. In other words, the physical devices can be in an idle mode while their virtual duplicates are operating, which helps them reducing power consumption. Therefore, virtual devices and networks are suitable for tailoring specific service requirements.

We design a two-layer overlay distributed control plane to support the main principle in LS-SDV. Layered overlay SDN control network is a method using virtualized technology to create separate virtualized network layer on the top of a physical network in the traditional network. We proposed a layered network based on IoT network, which is not considered in prior works. The properties of IoT nodes and traditional nodes are different, which is the innovation of our work. In virtual network management, there are two layers for addressing a virtual device, the virtual network layer and the device layer. Therefore, we use a similar hierarchical structure in LS-SDV. As shown in Fig. 2, there are 9 controllers from c_1 to c_9 and one device d_1. We use a two-layer overlay structure to organize 9 controllers including the virtual network layer and the device layer.

In the virtual network layer, we use one virtual network overlay to organize controllers for virtual network management. As the number of virtual networks is much fewer than devices, we deploy virtual network management in parts of controllers. The parts of controllers are organized in a distributed hash table (DHT) like topology and each controller manages parts of virtual networks [38]. In Fig. 2, controller c_1, c_4 and c_7 are organized into a virtual network overlay o_1.

In the device layer, there are multiple device overlays for management of network flows between devices. LS-SDV assigns one controller for handling the network control of each virtual device and controllers are organized into multiple overlays. In Fig. 2, controller c_1, c_2 and c_3 are organized into overlay o_2, c_4, c_5 and c_6 are into o_3, and c_7, c_8 and c_9 are into o_4. The assigned controllers of all devices in a single virtual network are organized into one overlay. There is no virtual network in which the assigned controllers of devices belong to different overlay. LS-SDV organizes one device overlay for one controller in the virtual network overlay. Thus, all devices in the virtual networks managed by one controller are controlled by controllers in the same device overlay. For example, devices in all virtual networks managed by controller c_1 are controlled by controller c_1, c_2 and c_3 in overlay o_1.

C. Overlay Addressing

Then, we introduce the addressing mechanism of virtual networks and devices in LS-SDV. The addressing mechanism is based on the address structures of devices. As shown in Fig. 3(a), each virtual device has a global ID with $m + n$ bits. The value of m and n is elastic with the scale of virtual networks and devices. The high m bits are used for addressing the corresponding virtual network and low n bits are used for addressing the controller assigned for the device. The high m bits are also the ID of the virtual network and the low n bits are the ID of the virtual device in the virtual network. Meanwhile, each controller has a controller ID for identification. We use an example to describe how the address structure ID works in LS-SDV. When a virtual device d_1 connects to a switch in network n_2 and the switch will inform the global ID of the device to controller c_2. Then, controller c_2 will first resolve the ID of virtual network v_1 to find out the corresponding controller c_1 and inform controller c_1. Then, controller c_1 will resolve the ID of device d_1 to find the assigned controller c_3 and inform controller c_3.

LS-SDV organizes all controllers in DHT like overlay structures including the virtual network overlay and device overlays. We first describe the addressing mechanism of the virtual network overlay. In the virtual network overlay, each controller maintains a finger table containing up to k entries while k is the bit number of controller ID. The controller ID has p bits while the consistent hash function generates a p bits key of each virtual network ID. The ith entry in the table means the nearest controller to the queried key, of which controller ID is more than $l + 2^{(i-1)}$ where l is the controller ID of the original one. Each device overlay also has the similar structure with the virtual network overlay while the p bits key is generated by the consistent function of the device ID.

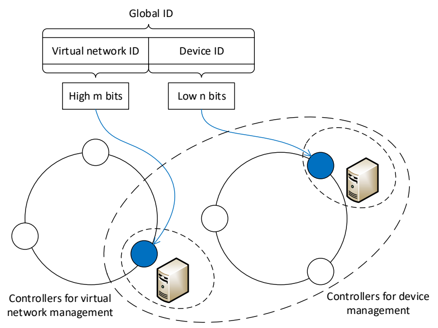
As shown in Fig. 3(b), we use the addressing procedure of device d_1 as an example. When controller $c(2)$ with ID=2 wants to find out the assigned controller of device d_1, it first finds out the virtual network ID is 102. With key=102, the consistent hashing function generates the value of 4. From the finger table of controller $c(2)$, we can find that the controller $c(4)$ manages virtual network $v(102)$. Therefore, controller $c(4)$ tries to find out the assigned controller of device d_1 from the device overlay. In device overly, controller $c(4)$ has another ID of 12. The hash value of key=119 is 5, which is more than $c'(12)+8 \mod 16$. From the finger table of controller $c'(12)$, the addressing mechanism will access controller $c'(4)$. After one step in controller $c'(4)$, the addressing mechanism can find out that controller $c'(5)$ is the assigned controller of device d_1.

Theorem 1: The time complexity of the addressing mechanism in LS-SDV is $2 \cdot \mathcal{O}(\log N)$ where N is the number of controllers.

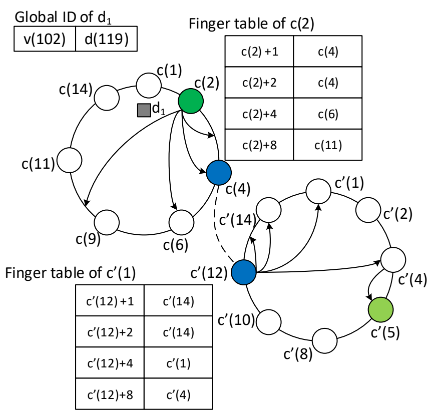
Proof: In the addressing mechanism, there are two steps of query. The first step is to find the controller managing the virtual network of the given device and the second step is to find the assigned controller. Each step is similar to Chord with a time complexity of $\mathcal{O}(\log n)$ where n is the number of the controllers in the overlay. Therefore, the entire time complexity is $\max(\mathcal{O}(\log n_1), \mathcal{O}(\log n_2))$ where n_1 is the number of controllers in the virtual network overlay and n_2 is the number of controllers in the device overlay. Since the maximum number of each overlay is N, the time complexity of the addressing mechanism in LS-SDV is $2 \cdot \mathcal{O}(\log N)$. ■

D. Device Mobility

The mobility in LS-SDV is a critical issue for network management of large-scale IoT. When a new virtual network with virtual devices joins LS-SDV, LS-SDV will assign a controller for the management of the virtual network and some controllers from the same device overly for the management of devices. Meanwhile, each virtual device also has a temporary controller for network connections. This temporary controller stores the session when the device connects the network. For example, in Fig. 1, the temporary controller of device d_2 is controller c_1 at first then c_2 at last.


Global ID
Virtual network ID
Device ID
High m bits
Low n bits
Controllers for virtual network management
Controllers for device management

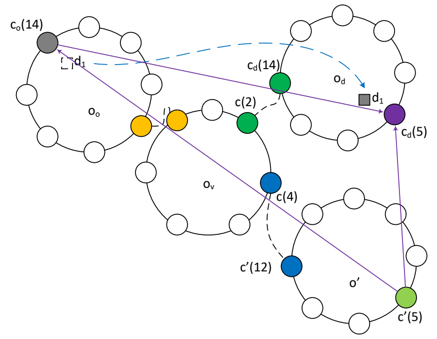
(a) Address structure



c(2) +1	c(4)
c(2)+2	c(4)
c(2)+4	c(6)
c(2)+8	c(11)

c'(12)+1	c'(14)
c'(12)+2	c'(14)
c'(12)+4	c'(1)
c'(12)+8	c'(4)

(b) Addressing



(c) Mobility management

Fig. 3. Addressing and mobility management in LS-SDV

Thus, after a device moves into a new network, the new temporary controller will ask the assigned controller of the device to find the previous temporary controller. Then, the previous temporary controller will transfer the session of the device to the new temporary controller. Thus, network connections of the moving device will remain. Therefore, for a given device, the assigned controller should maintain at least two positions, the previous temporary controller and current temporary controller.

As shown in Fig. 3(c), when device d_1 moves from one network to another network, LS-SDV will transfer the network session from controller $c_o(14)$ to controller $c_d(5)$ by assigned controller $c'(5)$. The steps are as follows. First, destination controller $c_d(5)$ will ask controller $c_d(14)$ which is the controller for virtual network management. Then, controller $c_d(14)$ will find the controller managing the virtual network of device d_1 through overlay o_v. As the procedures described in Section III-C, the assigned controller $c'(5)$ will be found by the addressing mechanism. Controller $c'(5)$ checks the current temporary controller of device d_1 which is controller $c_o(14)$. Then, controller $c'(5)$ will ask controller $c_o(14)$ to transfer the network session of device d_1 to controller $c_d(5)$ and set the current temporary controller of d_1 is $c_d(5)$. After transferring network session, LS-SDV finishes the mobility management of device d_1. Since both controller $c_o(14)$ In LS-SDV, the assigned controller is able to manage multiple positions of the given device.

Theorem 2: The time complexity of the mobility management for a given device in LS-SDV is $2 \cdot \mathcal{O}(\log N)$.

Proof: The time complexity of addressing the assigned controller is $2 \cdot \mathcal{O}(\log N)$ from Theorem 1. In mobility management, it needs two more processes for querying the controller for virtual network management and the previous temporary controller. Thus, the time complexity of the mobility management is $2 \cdot \mathcal{O}(\log N)$. ■

IV. Flow Scheduling

In this section, we present the flow scheduling for virtual networks in LS-SDV. We first describe the way to analyze the network status then introduce our flow scheduling algorithm in LS-SDV.

A. Network Analysis

It is necessary to understand the status of multiple networks before network flow scheduling. As controllers can collect most network information from software defined devices, it is possible to analyze the network view through network calculus [39]. In network calculus, a network flow can be modeled as a set of cumulative functions denoted by F, where $F(t)$ is the data transferred in the time period $[0, t)$. Functions in set F are increasing and non-negative while time domain is a set of non negative real numbers. We use D to denote the set of cumulative functions for the departure flow. For expressing resources in a given network, we use S to denote a set of service curves. From min-plus algebra [40], if $D(t) \geq F(t) \otimes S(t)$, the network can provide a minimal service for the give network flow in time period $[0, t)$.

Then, we define the service curve of the network. From a previous work [11], service curves in IoT can be defined by

$$S_i = \frac{w_i}{\sum_{j \neq i} w_j} B \cdot \max\{0, t - T\} \quad (1)$$

where i means the ith flow, w_i means the weight of flow i, $\sum_{j \neq i} w_j$ is the weight of all flows except flow i, B is the maximum transmission rate, and T is the delay.

As the IoT communication needs multi-hop transmission, the service curve for a given path will be summarized as $S(t) = S_1(t) \otimes S_2(t) \otimes ... \otimes S_n(t)$ where $1, 2, ..., n$ is the sequence number of hops in the communication.

For analyzing the transmission QoS of each packet in the network flow, we use a discrete model of the network traffic. We slight the traffic into packets and calculate the delay of each packet. From the transmission model, the packet delay in a given hop includes the transmission delay and the transmission delay of the waiting queue. The length of the queue can be monitored by the controller and transmission of each packet costs approximately the same time period. Thus, we can analyze the network status first from the information in each controller of LS-SDV.

B. AP Assignment Based Flow Scheduling

Since all devices are connected by APs in wireless networks, APs are usually the bottleneck in network transmission. Therefore, LS-SDV focuses on the network flow scheduling to optimize the AP assignment for each virtual network. As LS-SDV can obtain the network status through the network analysis, it is possible to find out the workload of each access point and assign them to devices in each virtual network. We use A to denote the set of APs in all networks and a_j to denote an AP in set A. We use V to denote the set of virtual networks and v_i to denote a virtual network in set V. We use a value X_{ij} to denote the set of devices in virtual network v_i connected by AP a_i and R_{ij} to denote the required bandwidth of set X_{ij}. We use an elastic function $U_i(\cdot)$ to denote the utility of the assignment of virtual network v_i and the problem can be formulated as

$$\begin{aligned} \max \quad & \textstyle\sum_{i=1}^{|V|}\sum_{j=1}^{|A|} U_i(X_{ij}) \\ s.t., \quad & \textstyle\sum_{i=1}^{|V|} R_{ij} \leq C_j \\ & \textstyle\bigcap_{j=1}^{|A|} X_{ij} = \varnothing \\ & \textstyle\bigcup_{j=1}^{|A|} X_{ij} = E_i \end{aligned} \tag{2}$$

where C_j is the available capacity of AP a_j , and E_i is the set of devices in virtual network v_i.

For solving (2), we design an AP assignment algorithm shown in Algorithm 1. The algorithm first finds an assignment set X^* to satisfy the minimal request of each virtual network. Then, the algorithm initials all X_{ij} as the output. The algorithm uses an iteration to find out the assignment of each AP in set X^*. Then, the algorithm tries to assign AP a_j to each virtual network. After assigning AP a_j to virtual network v_i, we use a value Q_{ij} to measure the efficiency of the assignment. We sort virtual networks with the non-decreasing Q_{ij} and assign AP a_j to the sorted virtual networks until AP a_j has no capacity for assignment. Obviously, since there is no duplicated device in different virtual networks, the time complexity of the proposed algorithm is $\mathcal{O}(M)$ where M is the total number of devices.

Theorem 3: The AP assignment algorithm for virtual networks is a 2-approximation for solving (2).

Proof: When the algorithm assigns AP a_j to virtual networks, there must be a space without assignment, which is $B_j - R_j$ where R_j is total required bandwidth of assigned virtual networks. If there is a fraction of a virtual network, the algorithm will match or exceed the optimal solution U_j^* by adding $\frac{B_j-R}{r_e} \cdot \Delta U_e$ where $\Delta U_e = U_e((X - X_{(e)j}) \cap X'_{(e)j})$. As $\sum_{i=1}^{e-1} R_{ij}\Delta U_i \geq \frac{1}{2}U_j^*$ or $\Delta U_e \geq \frac{B_j-R}{r_e} \cdot \Delta U_e \geq \frac{1}{2}U_j^*$, the approximation ratio for assignment of AP a_j is 2. For the entire assignment, as $\sum_{j=1}^{|A|} B_j - R_j \leq \sum_{j=1}^{|A|} U_j^*$, the AP assignment algorithm is a 2-approximation for solving (2). ■

We consider utility function $U_i(\cdot)$ as an elastic function for different network condition. As we focus on QoS guarantee of virtual networks, we choose a utility function based on network analysis. We choose a popular method to design the utility function, which is based on normalized weights. We consider there are two requirements of the network flow scheduling, including total bandwidth $\mathcal{B}$ and average delay $\mathcal{D}$ of the AP. We use two functions to define the utilities from these requirements, given by

Algorithm 1 AP Assignment for Virtual Networks

```
1: Find a set $X^*$ that meets the minimal request of the assignment;
2: for $i \leftarrow 1$ to $|V|$ do
3:     for $j \leftarrow 1$ to $A$ do
4:         $X_{ij} \leftarrow \varnothing$;
5:     end for
6: end for
7: for $j \leftarrow 1$ to $|A|$ do
8:     if $\exists i : X^*_{ij} \neq \varnothing$ then
9:         for $i \leftarrow 1$ to $|V|$ do
10:            $X'_{ij} \leftarrow X^*_{ij}$;
11:            for $k \leftarrow 1$ to $|E_i|$ do
12:                if $d_k \notin X^*_{ij}$ then
13:                    $X''_{ij} \leftarrow X'_{ij} \cup \{d_k\}$;
14:                else
15:                    $X''_{ij} \leftarrow X'_{ij} - \{d_k\}$;
16:                end if
17:                if $U_i((X^* - X_{ij}) \cup X''_{ij}) > U_i(X^*)$ then
18:                    $X'_{ij} \leftarrow X''_{ij}$
19:                end if
20:            end for
21:            $Q_{ij} \leftarrow \frac{U_i((X^* - X_{ij}) \cup X'_{ij})}{R_{ij}}$;
22:        end for
23:        Sort $V$ by $Q_{1j} \geq Q_{2j} \geq, ..., \geq Q_{|V|j}$;
24:        for $i' \leftarrow 1$ to $|V|$ do
25:            if $\sum_{i'' \leftarrow 1}^{i'} R_{i''j} > B_j$ then
26:                $e \leftarrow i'$;
27:                Break;
28:            end if
29:        end for
30:        for $i \leftarrow 1$ to $e - 1$ do
31:            $X_{ij} \leftarrow X'_{ij}$;
32:        end for
33:    end if
34: end for
```

$$U_i^{\mathcal{B}}(X_{ij}) = \log(1 + \frac{C(j)}{R_{ij}}), \text{ and} \tag{3}$$

$$U_i^{\mathcal{D}}(X_{ij}) = \log(1 + \frac{\mathcal{D}'}{\mathcal{D}}) \tag{4}$$

where $\mathcal{D}'$ is the average latency after assigning devices in X_{ij} to the networks through network analysis.

Therefore, the utility function $U_i(X_{ij})$ can be calculated with (3) and (4), given by

$$U_i(X_{ij}) = \omega^{\mathcal{B}} U_i^{\mathcal{B}}(X_{ij}) + \omega^{\mathcal{D}} U_i^{\mathcal{D}}(X_{ij}), \omega^{\mathcal{B}} + \omega^{\mathcal{D}} = 1 \tag{5}$$

where $\omega^{\mathcal{B}}$ and $\omega^{\mathcal{D}}$ are the normalized weights of function $U_i^{\mathcal{B}}(\cdot)$ and $U_i^{\mathcal{D}}(\cdot)$, respectively.

C. Online Assignment

We also focus on the online assignment to schedule network flows without having the entire virtual networks. Because

Algorithm 2 Online AP Assignment

```
Find $X_i^*$ that meets the minimal request of $v_i$;
for $j \leftarrow 1$ to $|A|$ do
    if $\exists i : X_{ij}^* \neq \varnothing$ then
        $X'_{ij} \leftarrow X_{ij}^*$;
        for $k \leftarrow 1$ to $|E_i|$ do
            if $d_k \notin X_{ij}^*$ then
                $X''_{ij} \leftarrow X'_{ij} \cup \{d_k\}$;
            else
                $X''_{ij} \leftarrow X'_{ij} - \{d_k\}$;
            end if
            if $U_i((X^* - X_{ij}) \cup X''_{ij}) > U_i(X^*)$ then
                $X'_{ij} \leftarrow X''_{ij}$;
            end if
        end for
        $\Phi(z_{ij}) \leftarrow (U^{\max} \cdot \mathrm{e}/U^{\min})^{z_{ij}}/(U^{\min}/\mathrm{e})$;
        if $\frac{U_i((X^*-X_{ij})\cup X'_{ij})}{R_{ij}} \geq \Phi(z_{ij})$ then
            $X_{ij} \leftarrow X'_{ij}$;
            Break;
        end if
    end if
end for
```

virtual networks will be updated and changed, the online assignment is very important for a dynamic environment.

We use a stream of $|V|$ virtual networks having utilities and required bandwidth $(U_i(X_{ij}), R_{ij})$, where i is the time when the virtual network joins into the IoT network. We also use an utility-to-bandwidth ratio of each virtual network as the efficiency. The goal of the online assignment is to assign APs, that is making a decision to maximize the total utility.

As shown in Algorithm 2, the algorithm assigns AP to virtual network v_i in an online fashion. When a virtual network v_i in time i joins into the IoT network, the algorithm first finds a solution set X_i^* to meet the minimal request of v_i. After that, the algorithm decides whether AP $a_j \in A$ is assigned to virtual network v_i. We design a function $\Phi(z_{ij})$ for the assignment decision, where z_{ij} is the remaining capacity of AP a_j in time i. In function Φz_{ij}, we use $U^{\max}$ and $U^{\min}$ to denote the upper bound and lower bound of utilities and e is the base of the natural logarithm. If the ratio of utility to the requirement is more than the value of function $\Phi(z_{ij})$, the algorithm assigns AP a_j to virtual network v_i.

Theorem 4: For a given virtual network sequence V, the online assignment algorithm can attain a competitive ratio as

$$OPT(V) \leq \mathcal{U}(V) \cdot (\ln(U^{\max}/U^{\min}) + 1) \qquad (6)$$

where $\mathcal{U}(V)$ is the utility of the online assignment, and $OPT(V)$ is the optimal utility.

Proof: Please see the appendix ■

V. Performance Evaluation

We use two different ways to evaluate the performance of LS-SDV. We first study the performance of LS-SDV in a given area then test the performance of an open SDN framework. We also compare the performance of LS-SDV to some existing methods as follows.

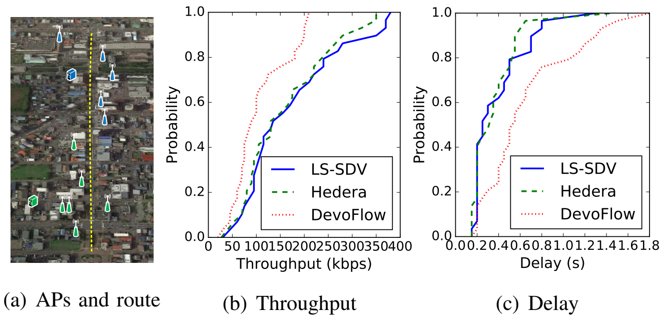


Fig. 4. Mobility management performance of LS-SDV

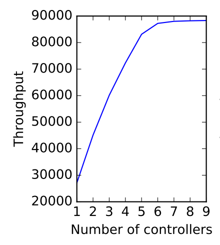
- Hedera [41] is a method focusing on scheduling elephant flows to reduce the problem of load balance. Hedera aims to reduce average flow completion time by scheduling elephant flows, because they occupy bandwidth in a long periods.
- DevoFlow [36] is also used to schedule the workload to meet high performance of networks. Both of them achieve good performance in flow scheduling. We test throughput and delay with [23], [36] with several timeslots which are both main targets in their work. Therefore, it is fair to compare with them.
- GENI [23] is an acknowledged open SDN framework for innovative network experiments, comparing with it is persuasive. In addition, we test throughput and delay factors, which are dominate concern in network performance, so it is fair to compare with GENI.

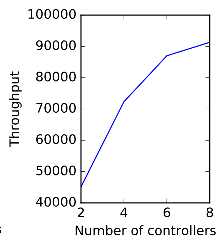
A. Mobility Management

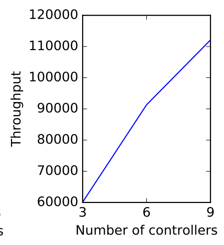
We first analyze the performance with device mobility in LS-SDV. All experiments are executed in OMNeT++ [42] and we add the LS-SDV framework to support mobility management. We choose an area near our university as a real scenario for the virtual network management in LS-SDV. The Nakajimacho is the downtown of Muroran city, in which many APs are deployed for tourists. As shown in Fig. 4(a), we get the positions of APs from service providers and divides all APs into two groups. One group near to the railway station has 6 APs, and another near to the shopping center has 5 APs. There APs are connected by two switches. Meanwhile, we deploy two servers for providing services. The bitrate of each AP is set to 12 Mbps with the 802.11g protocol and each AP has a 1 Gbps wired link with the switch. There are 4 controllers managing the network, in which 2 controllers manage the virtual networks.

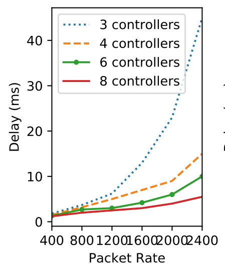
We add 30 IoT devices grouped in 6 virtual networks into the network, and each device has a network flow to servers. These devices move along the yellow path. We use a trace dataset of video streaming with 180 traces [43] and we randomly select 30 traces in the test. We also compare the performance of LS-SDV with DevoFlow [36] and Hedera [41].

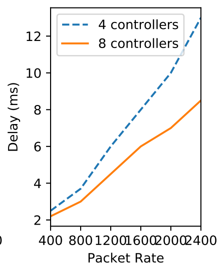
We test the throughput and delay from servers to devices during the movement of devices. LS-SDV and Hedera consider the capability of the entire network in network flow scheduling


(a) 1 controller in top


(b) 2 controllers in top


(c) 3 controllers in top


(d) 1 controller in top


(e) 2 controllers in top

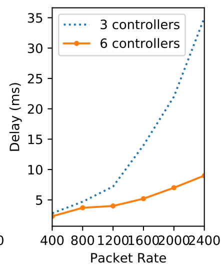
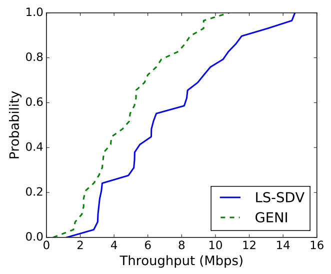

(f) 3 controllers in top

Fig. 5. Scalability of LS-SDV


(a) Throughput

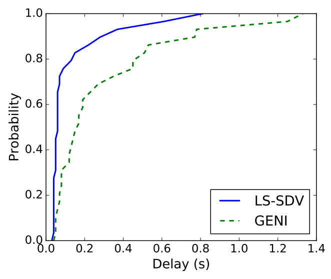
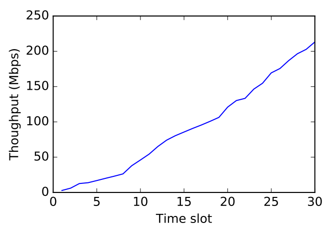

(b) Delay

Fig. 6. Open framework experiment results in ORBIT


(a) Total throughput

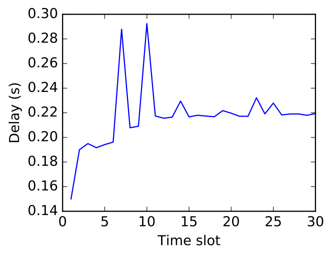

(b) Average delay

Fig. 7. Online assignment performance with the open framework experiment

while DevoFlow try to maximize the usage of each link. The packet loss of wireless links in the IoT environment is much more severe than wired links in the data center environment, the methodology of DeveFlow is not appropriate for a wireless environment. Therefore, as more packets will be dropped when the capacity of a wireless link exceeds a threshold, LS-SDV and Hedera provide higher server-to-device throughput than DevoFlow as shown in Fig. 4(b). Then, we test the delay in video streaming. As shown in Fig. 4(c), Hedera and LS-SDV still perform better than DevoFlow because of the packet loss in full assigned wireless links. Considering Hedera is a centralized structure with less delay in the control plane, LS-SDV still performs good enough with the distributed structure in the given scenario.

B. *Scalability of LS-SDV*

The scalability of LS-SDV framework is an important issue with the distributed structure. We also test the scalability of LS-SDV by adjusting the number of controllers in the control plane. From previous works on the evaluation of the SDN control plane [44], the Packet-In is a basic and frequently used message when a new flow joins the network. Thus, we test the performance of serving Packet-In messages with a different number of controllers. Since LS-SDV has a two-layer overlay structure, we adjust the number of controllers in different layers and test the system throughput.

As shown in Fig. 5(a), we use one controller for virtual network management and increase the number of controllers in the device overlay from 1 to 9 and add one controller in each step. The throughput increases with more controllers in the system. However, after the number of controllers is more than 6, the performance can not be increased obviously. From the results shown in Fig. 5(b) and 5(f), when we increase the number of controllers for virtual network management, the scalability of LS-SDV performs better than one controller in the top layer. When the number of controllers in the top layer is two, the scalability becomes a little worse when the number of controllers exceeds 6. While the number of controllers in the top layer is three, the scalability of LS-SDV maintains the same with 9 controllers.

We also test the average response delay for processing Packet-In messages. We set the packet rate from 400 to 2400 packets per second and increase 400 in each step. Each packet has a 16-bit random device ID and a 16-bits random virtual network ID. As shown in Fig. 5(d), we set the number of controllers in the top layer to one and test the average response delay with 3, 4, 6 and 8 controllers. With more controllers in the control plane, the response delay decreases obviously. From the results shown in Fig.5(e) and 5(f), as LS-SDV uses the two-layer overlay structure in the control plane, the additional addressing in the top layer increases the response delay. Meanwhile, as the more controllers in the virtual network overlay, the response delay with higher packet rate is better than the delay with one controller in the top layer. When the number of controllers in the top layer is set to 2, the response delay with 2400 packets per second is slightly shorter than the delay with one controller in the top layer. When the number of controllers in the top layer increases to 3, the response delay is also shorter than the delay with one controller in the top layer.

C. *Open Framework Experiments*

We also test LS-SDV with ORBIT which is an open SDN framework composed of 400 radio nodes. In our experiments, we use an ORBIT sandbox and set the number of APs as 12 and the number of controllers as 3. All controllers manage both the virtual networks and devices. For the mobility issue of IoT devices in experiments, we choose an AP access trace dataset collected from mobile devices [45] as movement events.

We compare the performance of LS-SDV with GENI which is an implementation of mobility management in the wireless environment. From the formulations of (2) and (5), the flow scheduling focuses on the QoS guarantee of virtual networks. Therefore, we use cumulative distribution probability (CDF) to reflect whether the QoS is guaranteed in the experiments.

We use the video streaming trace dataset as network flows and randomly select 30 traffic traces. For devices in experiments, we choose 30 most active traces in the AP access trace file. As shown in Fig. 6(a), we compare the server-to-device throughput of LS-SDV with GENI. The CDF plots show that the throughput of LS-SDV is near 20% more than GENI. The server-to-device delay results in Fig. 6(b) show that the server-to-device delay with LS-SDV is less than the delay with GENI. If we set 4 Mbps as the QoS requirement of throughput, the QoS satisfied ratio of LS-SDV is 75% while the ratio of GENI is only 52%. Meanwhile, the fluctuation of delay with LS-SDV is also lower than GENI, which means devices in LS-SDV have more stable communications. If the QoS requirement of delay is set to 0.2 s, the QoS satisfied ratio of LS-SDV is 87% while the ratio of GENI is only 64%. Since GENI applies a first-come-first-served flow scheduling in order to simplify the configuration, it is hard to guarantee the QoS with a heavy traffic. Meanwhile, GENI only uses a centralized controller without enough scalability. Therefore, our solution performs better than GENI with higher throughput, lower delay and higher QoS satisfied ratio.

Finally, we test the performance of the online assignment with ORBIT. For the online assignment, we set the number of time slots to 30 and add one virtual network in each time slot. We record the total throughput and average delay from the time slot 1 to 30. As shown in Fig. 7(a), the total throughput with the online assignment algorithm increases linearly with the time slots. Meanwhile, the delay performance in Fig. 7(b) shows the online assignment algorithm can provide a stable delay in scheduling network flows.

As a result, since LS-SDV introduces scalability and efficient flow scheduling into large-scale IoT, LS-SDV performs better than other solutions on both throughput and delay. Furthermore, LS-SDV shows good scalability of the SDN control plane with virtual network management in the large-scale IoT environment.

VI. Conclusion and Future Work

In this paper, we investigate the problem of virtual network management in the IoT environment. There are two important issues in maintaining virtual networks, updating complex virtual topology and storing massive data of all virtual network information. We propose LS-SDV to address these issues, which shows that a distributed structure brings much better scalability than the centralized structure in large-scale SDIoT networks. Meanwhile, LS-SDV introduces a delicate structure for virtual network management in large-scale IoT. The two-layer overlay structure avoids complex concurrency mechanism for management of virtual networks in the distributed environment. LS-SDV provides simple mobility management through assigning a specific controller for each device. Meanwhile, the efficient network analysis based network flow scheduling in LS-SDV also improves the network performance of each virtual network. From the experiment results, LS-SDV performs better than existing solutions in the mobile IoT environment.

In the future, we plan to introduce LS-SDV into the IoT cloud computing and implement an IoT cloud prototype with real world IoT devices. Meanwhile, an online flow scheduling strategy in large-scale SDIoT will be introduced for the virtual network management.

Acknowledgments

This work is supported by JSPS KAKENHI Grant Number JP16K00117, JP17K12669, JP19K20250, KDDI Foundation, and MEXT's Leading Initiative for Excellent Young Researchers (LEADER). Mianxiong Dong is the corresponding author.

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," *Computer Networks*, vol. 54, no. 15, pp. 2787 – 2805, 2010.

[2] D. Li, M. Dong, Y. Yuan, J. Chen, K. Ota, and Y. Tang, "SEER-MCache: A prefetchable memory object caching system for IoT real-time data processing," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3648–3660, oct 2018.

[3] J. Zhou, Y. Wang, K. Ota, and M. Dong, "AAIoT: Accelerating artificial intelligence in IoT systems," *IEEE Wireless Communications Letters*, vol. 8, no. 3, pp. 825–828, jun 2019.

[4] L. Atzori, A. Iera, G. Morabito, and M. Nitti, "The social internet of things (siot) – when social networks meet the internet of things: Concept, architecture and network characterization," *Computer Networks*, vol. 56, no. 16, pp. 3594 – 3608, 2012.

[5] L. Atzori, A. Iera, and G. Morabito, "From "smart objects" to "social objects": The next evolutionary step of the internet of things," *IEEE Communications Magazine*, vol. 52, no. 1, pp. 97–105, January 2014.

[6] J. Xu, K. Ota, M. Dong, A. Liu, and Q. Li, "SIoTFog: Byzantine-resilient IoT fog networking," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 12, pp. 1546–1557, dec 2018.

[7] H. Li, M. Dong, and K. Ota, "Radio access network virtualization for the social internet of things," *IEEE Cloud Computing*, vol. 2, no. 6, pp. 42–50, Nov 2015.

[8] D. Drutskoy, E. Keller, and J. Rexford, "Scalable network virtualization in software-defined networks," *IEEE Internet Computing*, vol. 17, no. 2, pp. 20–27, March 2013.

[9] W. H. Chin, Z. Fan, and R. Haines, "Emerging technologies and research challenges for 5g wireless networks," *IEEE Wireless Communications*, vol. 21, no. 2, pp. 106–112, April 2014.

[10] K. Pentikousis, Y. Wang, and W. Hu, "Mobileflow: Toward software-defined mobile networks," *IEEE Communications Magazine*, vol. 51, no. 7, pp. 44–53, July 2013.

[11] Z. Qin, G. Denker, C. Giannelli, P. Bellavista, and N. Venkatasubramanian, "A software defined networking architecture for the internet-of-things," in *2014 IEEE Network Operations and Management Symposium (NOMS)*, May 2014, pp. 1–9.

[12] M. Dong, H. Li, K. Ota, and J. Xiao, "Rule caching in sdn-enabled mobile access networks," *IEEE Network*, vol. 29, no. 4, pp. 40–45, July 2015.

[13] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Towards an elastic distributed sdn controller," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13. New York, NY, USA: ACM, 2013, pp. 7–12.

[14] K. Kirkpatrick, "Software-defined networking," *Commun. ACM*, vol. 56, no. 9, pp. 16–19, Sep. 2013.

[15] T. Mahmoodi and S. Seetharaman, "On using a sdn-based control plane in 5g mobile networks," in *Wireless World Research Forum, meeting*, vol. 32, 2014.

[16] J. Xu, K. Ota, and M. Dong, "Real-time awareness scheduling for multimedia big data oriented in-memory computing," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3464–3473, oct 2018.

[17] J. Liu, Y. Li, and D. Jin, "Sdn-based live vm migration across datacenters," in *Proceedings of the 2014 ACM Conference on SIGCOMM*, ser. SIGCOMM '14. New York, NY, USA: ACM, 2014, pp. 583–584.

[18] H. Ali-Ahmad, C. Cicconetti, A. de la Oliva, V. Mancuso, M. R. Sama, P. Seite, and S. Shanmugalingam, "An sdn-based network architecture for extremely dense wireless networks," in *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*, Nov 2013, pp. 1–7.

[19] H. Flores, P. Hui, S. Tarkoma, Y. Li, T. Anagnostopoulos, V. Kostakos, C. Luo, and X. Su, "Sensorclone: a framework for harnessing smart devices with virtual sensors," in *Proceedings of the 9th ACM Multimedia Systems Conference*. ACM, 2018, pp. 328–338.

[20] M. Bansal, J. Mehlman, S. Katti, and P. Levis, "Openradio: A programmable wireless dataplane," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 109–114.

[21] Y.-J. Chen, L.-C. Wang, M.-C. Chen, P.-M. Huang, and P.-J. Chung, "Sdn-enabled traffic-aware load balancing for m2m networks," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 1797–1806, 2018.

[22] W. Ren, Y. Sun, H. Luo, and M. Guizani, "Bllc: A batch-level update mechanism with low cost for sdn-iot networks," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1210–1222, 2018.

[23] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, "GENI: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, pp. 5–23, mar 2014.

[24] R. Munoz, R. Vilalta, N. Yoshikane, R. Casellas, R. Martinez, 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.

[25] B. R. Al-Kaseem and H. S. Al-Raweshidyhamed, "Sd-nfv as an energy efficient approach for m2m networks using cloud-based 6lowpan testbed," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1787–1797, 2017.

[26] W. Borjigin, K. Ota, and M. Dong, "In broker we trust: A double-auction approach for resource allocation in NFV markets," *IEEE Transactions on Network and Service Management*, vol. 15, no. 4, pp. 1322–1333, dec 2018.

[27] M. Chen, Y. Qian, S. Mao, W. Tang, and X. Yang, "Software-defined mobile networks security," *Mobile Networks and Applications*, vol. 21, no. 5, pp. 729–743, 2016.

[28] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama *et al.*, "Onix: A distributed control platform for large-scale production networks." in *OSDI*, vol. 10, 2010, pp. 1–6.

[29] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, 2010, pp. 3–3.

[30] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: A framework for efficient and scalable offloading of control applications," in *Proceedings of the First Workshop on Hot Topics in Software Defined Networks*, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 19–24.

[31] A. Panda, W. Zheng, X. Hu, A. Krishnamurthy, and S. Shenker, "Scl: Simplifying distributed sdn control planes," in *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. Boston, MA: USENIX Association, 2017, pp. 329–345.

[32] S. Schmid and J. Suomela, "Exploiting locality in distributed sdn control," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13. New York, NY, USA: ACM, 2013, pp. 121–126.

[33] B. Heller, C. Scott, N. McKeown, S. Shenker, A. Wundsam, H. Zeng, S. Whitlock, V. Jeyakumar, N. Handigol, J. McCauley, K. Zarifis, and P. Kazemian, "Leveraging sdn layering to systematically troubleshoot networks," in *Proceedings of the Second ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '13. New York, NY, USA: ACM, 2013, pp. 37–42.

[34] A. A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, "Elasticon: An elastic distributed sdn controller," in *Proceedings of the Tenth ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ser. ANCS '14. New York, NY, USA: ACM, 2014, pp. 17–28.

[35] M. Canini, P. Kuznetsov, D. Levin, and S. Schmid, "A distributed and robust sdn control plane for transactional network updates," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, April 2015, pp. 190–198.

[36] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula, P. Sharma, and S. Banerjee, "Devoflow: Scaling flow management for high-performance networks," in *Proceedings of the ACM SIGCOMM 2011 Conference*, ser. SIGCOMM '11. New York, NY, USA: ACM, 2011, pp. 254–265.

[37] J. Xie, D. Guo, C. Qian, L. Liu, B. Ren, and H. Chen, "Validation of distributed SDN control plane under uncertain failures," *IEEE/ACM Transactions on Networking*, vol. 27, no. 3, pp. 1234–1247, jun 2019.

[38] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," in *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '01. New York, NY, USA: ACM, 2001, pp. 149–160.

[39] J.-Y. Le Boudec and P. Thiran, *Network calculus: a theory of deterministic queuing systems for the internet*. Springer Science & Business Media, 2001, vol. 2050.

[40] A. Burchard, J. Liebeherr, and S. D. Patek, "A min-plus calculus for end-to-end statistical service guarantees," *IEEE Transactions on Information Theory*, vol. 52, no. 9, pp. 4105–4114, Sept 2006.

[41] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 19–19.

[42] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, ser. Simutools '08. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 60:1–60:10.

[43] F. H. P. Fitzek and M. Reisslein, "Mpeg-4 and h.263 video traces for network performance evaluation," *IEEE Network*, vol. 15, no. 6, pp. 40–54, Nov 2001.

[44] D. Wu, D. I. Arkhipov, E. Asmare, Z. Qin, and J. A. McCann, "Ubiflow: Mobility management in urban-scale software defined iot," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, April 2015, pp. 208–216.

[45] K. Nahrstedt and L. Vu, "CRAWDAD data set uiuc/uim (v. 2012-01-24)," Downloaded from http://crawdad.org/uiuc/uim/, Jan. 2012.

Appendix
Proof of Theorem 4

For a given virtual network sequence V and a AP a_j, the algorithm assigns Z_j to V and obtain utility $\mathcal{U}_j(V)$. Let X_j and X_j^* denote the set of virtual networks to which AP a_j is assigned by the online assignment algorithm and the optimum, respectively. Let $U_j = u_j(X_j \cap X_j^*)$ and $R_j = r_j(X_j \cap X_j^*))$ to denote the utility and the required bandwidth of the common virtual networks between two sets. For each virtual network v_i without assigned a_j, the utility in its assignment is less than $\Phi(z_{ij})$ and $\Phi(z_{ij})$ is less than $\Phi(Z_j)$ since $\Phi(z_{ij})$ is monotone increasing of z_{ij}. Thus, $OPT_j(V) \leq U_j + \Phi(Z_j)(C_j - R_j)$ where OPT_j is the optimal assignment for AP a_j.

Since $\mathcal{U}_j(V) = U_j + u_j(X_j \ X_j^*)$, there is

$$\frac{OPT_j(V)}{\mathcal{U}_j(V)} \leq \frac{U_j + \Phi(Z_j)(C_j - R_j)}{U_j + u_j(X_j \setminus X_j^*)}. \tag{7}$$

As the value to the requirement of each virtual network v_i is no less than $\Phi(z_{ij})$, we have

$$U_j \geq \sum_{v_i \in X_j \cap X_j^*} \Phi(z_{ij}) \cdot R_{ij}, \text{ and} \tag{8}$$

$$u_j(X_j \setminus X_j^*) \geq \sum_{v_i \in X_j \setminus X_j^*} \Phi(z_{ij}) \cdot R_{ij} \tag{9}$$

where we use U_j' and U_j'' to denote $\sum_{v_i \in X_j \cap X_j^*} \Phi(z_{ij}) \cdot R_{ij}$ and $\sum_{v_i \in X_j \setminus X_j^*} \Phi(z_{ij}) \cdot R_{ij}$, respectively.

Therefore, (7) implies

$$\frac{OPT_j(V)}{\mathcal{U}_j(V)} \leq \frac{U'_j + \Phi(Z_j)\cdot(C_j - R_j)}{U'_j + u_j(X_j \setminus X^*_j)} \tag{10}$$

We plug in values of U'_j and U''_j with $U'_j \leq \Phi(Z_j)\cdot R_{ij}$ and get

$$\frac{OPT_j(V)}{\mathcal{U}_j(V)} \leq \frac{\Phi(Z_j)}{\sum_{v_i\in X_j}\Phi(z_{ij})\cdot\Delta z_{ij}} \tag{11}$$

where $\Delta z_{ij} = z_{(i+1)j} - z_{ij} = R_{ij}/C_j$ for all $v_i \in V$.

We assume that the available bandwidth of each AP is much larger than the requirement of each virtual network. Thus, we use an integration to find the approximate by

$$\begin{aligned}\sum_{v_i\in X_j}\Phi(z_{ij})\cdot\Delta z_{ij} \geq&(1-\epsilon)\cdot\int_0^{Z_j}\Phi(z_{ij})\mathrm{d}x\\ =&(1-\epsilon)\cdot(\int_0^c U^{\min} + \int_c^{Z_j}\Phi(z_{ij})\mathrm{d}x)\\ =&(1-\epsilon)\cdot(c\cdot U^{\min}+\\ \frac{U^{\min}}{\mathrm{e}}\cdot&\frac{(U^{\max}\cdot\mathrm{e}/U^{\min})^{Z_j} - U^{\max}\cdot\mathrm{e}/U^{\min})^c}{\ln(U^{\max}\cdot\mathrm{e}/U^{\min})})\\ =&(1-\epsilon)\cdot\frac{\Phi(Z_j)}{\ln(U^{\max}/U^{\min})+1}\end{aligned} \tag{12}$$

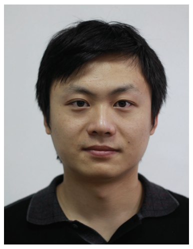
where $c = \frac{1}{1+\ln(U^{\max}/U^{\min})}$ and $\epsilon = (\max R_{ij})/C_j$.

Therefore, the competitive ratio for assigning AP a_j to virtual machine sequence V is given by

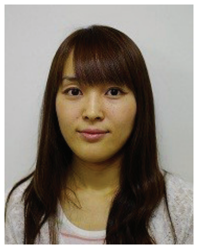
$$\frac{OPT_j(V)}{\mathcal{U}_j(V)} \leq \frac{1-\epsilon}{\ln(U^{\max}/U^{\min})+1} \leq \frac{1}{\ln(U^{\max}/U^{\min})+1}. \tag{13}$$

Since $OPT(V) = \sum_{j=1}^{|A|} OPT_j(V)$ and $\mathcal{U}(V) = \sum_{j=1}^{|A|}\mathcal{U}_j(V)$, the competitive ratio of the online assignment algorithm is

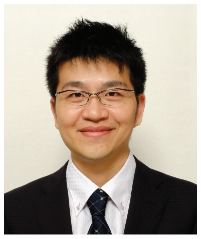
$$\frac{1}{\ln(U^{\max}/U^{\min})+1}. \tag{14}$$



He Li received the B.S., M.S. degrees in Computer Science and Engineering from Huazhong University of Science and Technology in 2007 and 2009, respectively, and Ph.D. degree in Computer Science and Engineering from The University of Aizu in 2015. He is currently an Assistant Professor with Department of Information and Electronic Engineering, Muroran Institute of Technology, Japan. In 2018, he is selected as a Ministry of Education, Culture, Sports, Science and Technology (MEXT) Excellent Young Researcher. His research interests include cloud computing and software defined networking. He has received the best paper award from IEEE VTC2016-Fall. Dr. Li serves as an Associate Editor for Human-centric Computing and Information Sciences (HCIS), as well as a Guest Associate Editor for IEICE Transactions on Information and Systems. He is the recipient of IEEE TCSC Outstanding Ph.D. Dissertation Award 2016.



Kaoru Ota was born in Aizu-Wakamatsu, Japan. She received M.S. degree in Computer Science from Oklahoma State University, USA in 2008, B.S. and Ph.D. degrees in Computer Science and Engineering from The University of Aizu, Japan in 2006, 2012, respectively. She is currently an Assistant Professor with Department of Sciences and Informatics, Muroran Institute of Technology, Japan. From March 2010 to March 2011, she was a visiting scholar at University of Waterloo, Canada. Also she was a Japan Society of the Promotion of Science (JSPS) research fellow with Graduate School of Information Sciences at Tohoku University, Japan from April 2012 to April 2013. Her research interests include Wireless Networks, Cloud Computing, and Cyber-physical Systems. Dr. Ota has received best paper awards from ICA3PP 2014, GPC 2015, IEEE DASC 2015, IEEE VTC 2016-Fall, FCST 2017, 2017 IET Communications Premium Award and IEEE ComSoc CSIM Best Conference Paper Award 2018. She is an editor of IEEE Transactions on Vehicular Technology (TVT), IEEE Internet of Things Journal, IEEE Communications Letters, IEEE Wireless Communications Letters, Peer-to-Peer Networking and Applications (Springer), Ad Hoc & Sensor Wireless Networks, International Journal of Embedded Systems (Inderscience) and Smart Technologies for Emergency Response & Disaster Management (IGI Global), as well as a guest editor of ACM Transactions on Multimedia Computing, Communications and Applications (leading), IEEE Internet of Things Journal, IEEE Communications Magazine, IEEE Network, IEEE Wireless Communications, IEEE Access, IEICE Transactions on Information and Systems, and Ad Hoc & Sensor Wireless Networks (Old City Publishing). She is the recipient of IEEE TCSC Early Career Award 2017, and The 13th IEEE ComSoc Asia-Pacific Young Researcher Award 2018.



Mianxiong Dong received B.S., M.S. and Ph.D. in Computer Science and Engineering from The University of Aizu, Japan. He is currently a Professor in the Department of Sciences and Informatics, Advisor to Executive Director, and Vice Director of Office of Institutional Research at the Muroran Institute of Technology, Japan. He was a JSPS Research Fellow with School of Computer Science and Engineering, The University of Aizu, Japan and was a visiting scholar with BBCR group at University of Waterloo, Canada supported by JSPS Excellent Young Researcher Overseas Visit Program from April 2010 to August 2011. Dr. Dong was selected as a Foreigner Research Fellow (a total of 3 recipients all over Japan) by NEC C&C Foundation in 2011. He has received best paper awards from IEEE HPCC 2008, IEEE ICESS 2008, ICA3PP 2014, GPC 2015, IEEE DASC 2015, IEEE VTC 2016-Fall, FCST 2017, 2017 IET Communications Premium Award and IEEE ComSoc CSIM Best Conference Paper Award 2018. He has been serving as the Vice Chair of IEEE Communications Society Asia/Pacific Region Information Services Committee and Meetings and Conference Committee, Leading Symposium Chair of IEEE ICC 2019, Student Travel Grants Chair of IEEE GLOBECOM 2019, and Symposium Chair of IEEE GLOBECOM 2016, 2017. He is the recipient of IEEE TCSC Early Career Award 2016, IEEE SCSTC Outstanding Young Researcher Award 2017, The 12th IEEE ComSoc Asia-Pacific Young Researcher Award 2017, Funai Research Award 2018 and NISTEP Researcher 2018 (one of only 11 people in Japan) in recognition of significant contributions in science and technology from MEXT. He is currently the Member of Board of Governors and Chair of Student Fellowship Committee of IEEE Vehicular Technology Society, and Treasurer of IEEE ComSoc Japan Joint Sections Chapter.