Switch Migration in SD-IoT Networks for Urban Sensing

Yabo Li¹, Lifen Jiang¹, *, Jinqi Zhu¹ and Ke Cao¹
¹College of Computer and Information Engineering, Tianjin Normal University, Tianjin 300387, China
*Corresponding author e-mail: wxxjlf@sina.com

Abstract. Large scale software-defined IoT (SD-IoT) networking need to deploy multiple controllers. How to ensure load balancing among multiple controllers is important to network performance. This paper proposes a switch migration strategy (MMM) which aims to minimize the global migration cost in SD-IoT networks. MMM not only considers the migration cost when making a migration decision, but also takes both the load of the overloaded domain and other domains after migration into consideration when selecting a switch for migration. Simulation experiments prove the effectiveness of the proposed scheme.

1. Introduction
Software Defined Networking (SDN) is a novel paradigm that decouples the network control and forwarding functions. It especially fits for applications with dynamically changing network traffic. SDN is used in Internet of Things (IoT) services and applications, termed as software-defined IoT (SD-IoT) networking [1]. With the advent of 5G networks, there will be a large number of IoT devices connected to the network in the city, making massive data in the network. Moreover, due to the wide range of the urban area, deploying multiple controllers in SD-IoT networks can ensure the reliability and scalability of services [2]. However, load imbalance is inevitable among multiple controllers [3]. To ensure load balancing, switches controlled by overloaded controllers need to be migrated to other domains with lower utilization periodically. By dynamically modifying the mapping between the switch and the controller, imbalance in the network is avoided.

As one of the most popular issues in SDN, dynamic migration of switches has attracted a lot of attentions. However, most of existing migration strategies have not considered the cost of migration and thus the migration costs are high in these works [4]. Though some prior researches pay attention to the migration cost, they have high time overhead and require multiple migrations to achieve load balancing [5]. To the best of our knowledge, few studies have considered the impact of inter-domain data flows generated by migration on the load of intermediate domains. In fact, the migration process may increase loads of intermediate domains.

This paper proposes a switch migration strategy (MMM) with minimal global migration cost in SD-IoT networks. MMM not only considers the migration cost when making a migration decision, but also takes both the load of the overloaded domain and other domains after migration into consideration when selecting a switch for migration. Simulation experiments show that compared with the existing relevant strategies, MMM always has the minimum total network load and migration cost. At the same time, it can better ensure the load balancing among controllers.

2. Network model
The SD-IoT network structure for city-aware applications is comprised of three layers: the IoT layer,
the **Software-defined network layer**, and the **cloud server layer**. The entire network model is shown in Figure 1:

**IoT layer:** The IoT layer includes various sensing devices (such as smartphones, vehicles, shared bicycles, sensors, etc.) which generate sensing data.

**Software-defined network layer:** The software-defined network layer is responsible for transmitting the sensed data to the remote server. This layer is divided into two parts: the control plane and the data plane.

**Cloud server layer:** The cloud server layer contains a large number of cloud data servers. These servers collect sensed data and process data.

![Figure 1. The architecture of SD-IoT network.](image)

SD-IoT network deploys two layers of controllers at the SDN layer, which is the top layer with the global controller and the middle layer with controllers. The data plane is divided into multiple domains. Each controller in the middle layer manages one domain. There are one or more switches in each domain. The global controller at the top level is responsible for monitoring the global network information. It also decides whether to perform switch migration operations or not based on the load balancing status of the controllers in the middle layer. Flows in the network can be divided into two types, as the Intra-domain flow and the inter-domain flow. Intra-domain data flow is the flow with the source and destination are in the same control domain; Inter-domain data flow is the type of flow which source and destination are not in the same control domain.

### 3. Strategy Implementation

#### 3.1 Data stream transmission

In urban sensing applications, each sensing event has a location. When the sensing device senses an event, it generates data to describe the event. The sensing device transmits the sensing data to a gateway (such as gateway $g$), and then the gateway transmits the data to the remote data server through the software-defined network layer. Let the set of all gateways be $G$, the set of all events be $E$, and the set of data servers be $D$. As shown in Figure 2 (a), it is assumed that four events $e_1, e_2, e_3$ and $e_4$ occurred in a certain area of the city and four gateways are deployed in this area, namely $g_1, g_2, g_3$ and $g_4$. Let the load of transmitting event $e_i$ to the gateway $g_i$ be $L(e_i, g_i)$. We assume that the system obtains the load of each sensing event from the historical sensing task statistics near each location. In addition, we assumed that each sensing event can be transmitted to $h$ different gateways at most. For event $e_i$, the set of all gateways it uploads to is $S_i$, and it can be seen that $S_i$ is a subset of $G$. For example, as shown in Figure 2(b), for the above four events, $S_1 = \{g_1, g_2\}, S_2 = \{g_1, g_2, g_3\}, S_3 = \{g_2, g_3\}, S_4 = \{g_2, g_3, g_4\}$.

Since the flows of the SDN layer originates from the gateway, we consider that each gateway ($g \in G$) sends all the collected sensing data to a controller $c$ ($c \in C$). Therefore, the gateway-to-controller function is defined as $F: g \rightarrow c$, that is, each gateway is divided into the domain where a controller is located. Let $p_{i,j}$ indicates the route from the gateway $g_i$ to the data server $d_j$, and $L(p_{i,j})$ indicates the
number of requested data flows through the route. A route at the SDN layer is an alternating sequence of switches and links. The set of all routes in the network is $P$.

(a) Multiple gateways and multiple events in urban areas
(b) Gateway-to-server mapping

Figure 2. The diagrammatic sketch for transmit events to gateways.

3.2 MMM migration algorithm

3.2.1. Calculating migration costs. The load of each controller is composed of two main parts. When the switch receives a newly generated data flow or an inter-domain data flow in the network, it will send a Packet-in message [5] to the controller. The controller then calculates a route and sends routing entries. Thus a rule load is generated. Second, after the controller delivers routing entries, the controller push the flow rules to the flow tables for all switches. Thus, we have:

$$L(c_i) = f_{packet} \cdot \sum_{s_k \in F} \sum_{c_j \in C} r_{s_k} h_{s_k}(s_k \in c_i) + f_{packet} \cdot \sum_{c_j \in C} h_{s_k} r_{s_k}(s_k \in c_i)$$

(1)

Where $L(c_i)$ is the total load of controller $c_i$ in the network, $f_{packet}$ is the average Packet-in packet size, $r_{s_k}$ is the flow request rate of switch $s_k$, $h_{s_k}$ is the number of hops between switch $s_k$ and controller $c_i$, $l(s_k \in c_j)$ indicates whether the switch $s_k$ belongs to the control domain where $c_j$ is located. If it is, the value is 1, otherwise it is 0. $r_{s_k}$ is the speed at which the controller rotates the switch $s_k$.

The load of switch $s_k$ is defined as the total number of request data flows routed through switch $s_k$, as follows:

$$L(s_k) = \sum_{p_{i,j} \in P} L(p_{i,j}) l(s_k \in P_{i,j})$$

(2)

Where $p_{i,j}$ indicates the route from the gateway $g_i$ to the data server $d_j$, and $L(p_{i,j})$ denote the number of requested data flows through the route. $l(s_k \in p_{i,j})$ indicates whether the path $p_{i,j}$ passes $s_k$, if it is, the value is 1; otherwise it is 0.

When the switch $s_k$ is migrated from the controller $c_i$ to the controller $c_j$, the controller load change of the $c_i$ domain caused by the migration is:

$$L'_{f_{packet}} = \sum_{s_k \in U_{i,j}} \sum_{s_k \in U_{i,j}} L(s_k) f_{packet} h_{s_k}$$

(3)

Where $U_{i,j}$ indicates the set of all switches in the domain of controller $c_i$, and $h_{s_k}$ is the number of hops between the switch $s_k$ before and after the migration. $L(s_k)$ represents the number of data flows passing through $s_k$ through the current route.

In addition, the migration cost $L_{s_k}^{c_i \rightarrow c_j}$ is divided into two parts. The first part is the migration request cost $L_{s_k}^{c_i \rightarrow c_j}$ that the switch $s_k$ sends to the controller $c_j$, similar to formula (1) is calculated as follows:

$$L_{s_k}^{c_i \rightarrow c_j} = \sum_{s_k \in U_{i,j}} L(s_k) f_{packet} h_{s_k} + \sum_{s_k \in U_{i,j}} f_{packet} h_{s_k} r_{s_k}$$

(4)
Where $h_{kj}$ is the number of hops from switch $s_k$ to controller $c_j$. The second part is the additional cost $L_{sk}^c$ to all other domains (the source domain and destination domain are not included) when the switch $s_k$ is migrated from domain $c_i$ to domain $c_j$, as follows:

$$L_{sk}^c = \sum_{c_j \in \{C \setminus \{c_i, c_j\}\}} L(s_k) \cdot f_{packet} \cdot h_{kk} + \sum_{c_p \in \{C\}} p_k \cdot h_{kp} - r_k \cdot h_{kp}$$

(5)

3.2.2. Implementation of migration algorithm. To ensure the efficiency of switch migration, the cost of the switch migration request $L_{sk}^c$ should be as small as possible, and the cost of inter-domain flow changes $L_{f}^{c_i \rightarrow c_j}$ should be as large as possible, and keep the second part of the cost (that is, the migration cost to other domains) as small as possible. Thus, we have:

$$L_{sk}^c \propto \frac{L_{f}^{c_i \rightarrow c_j}}{L_{packet}}$$

(6)

To perform switch migration efficiently, the switch migration cost problem can be formulated as the following optimization problem as follows:

$$\min\sum_{s_k \in \{C\}} L_{sk}^c$$

s.t. $|L(c_i) - L(c_j)| \leq \eta$, $1 \leq i, j \leq k$ and $i \neq j$,

$L(c_i) \leq L_{max}$ \quad \forall i \leq n$

Where $\eta$ is the load imbalance parameter and $L_{max}$ is the maximum load that a single controller can handle.

We use the knapsack algorithm to solve the above problems. We regard each controller domain as a small knapsack with a maximum capacity of $L_{max}$, and consider the entire topology network as a large knapsack containing $n$ ($n$ is the number of control domains) small knapsacks. Consider the migration task of each switch $s_k$ as an item whose weight is its own load $L(s_k)$ and its cost is $L_{sk}^c + L_{f}^{c_i \rightarrow c_j}$. And aggregate all migration tasks into a large group. In this way, we convert the optimization problem of formula (7) into a knapsack problem. The description of the knapsack problem: select an item from a large group and place it in a small knapsack, under the constraints of the capacity of the large and small knapsacks, to achieve the purpose of making the total weight larger and the total cost smaller in the large knapsack. There are many algorithms to solve the backpack problem, such as greedy algorithm, bat algorithm and heuristic algorithm.

4. Experimental simulation

4.1 Simulation environment and performance evaluation indicators

The simulation experiment environment includes 3 servers with the same experimental configuration (InterlCore i7 3.5GHz 4GBRAM), 2 servers running Opendaylight [8] controller, and 1 server running Mininet [9]. In order to make the experiments more real and reliable, all experiments were based on authoritative network topology, namely Internet2 OS3E [10] and TopologyZoo [11].

To verify the performance of MMM, we compare it with three other strategies: (1) SC strategy is the switch and the controller are connected fixedly; (2) DCM strategy migrates the switch to the nearest controller every time; (3) EOSDM [7] strategy based on performance optimization. It sets trigger factors to determine load imbalance, and establishes a migration efficiency model to consider both the load balancing rate and the migration cost to determine the migration switch and the migration controller. We use the following three indicators to analyze the performance of each strategy:

- Migration cost: the cost of communication costs incurred by the switch per unit time during the migration process.
- Controller throughput: The amount of data processed by the controller per unit time after the switch is migrated.
- Load balancing rate: the load balancing status of the network after the switch completes the migration. The variance is used to calculate the degree of load balancing [7]. The smaller the value, the more balanced the load. Normalize the load balancing rate during the simulation experiment, as follows:

\[ x_{scale} = \frac{x - x_{min}}{x_{max} - x_{min}} \]  

(8)

Where \( x \) is the load variance, \( x_{min} \) is 0, and \( x_{max} \) is the maximum controller load.

4.2 Migration cost and controller throughput under different strategies

In this group of experiments, we compared the different migration costs of MMM, DCM, and EOSDM during switch migration when the controller in a certain domain of the network is overloaded, and the controller throughput after the switch migration. The experimental results are shown in Figure 3 and Figure 4. Since the SC strategy is a fixed connection between the controller and the switch, there is no migration cost. From the two Figures, we can see that the migration cost of DCM is lower, but the controller throughput is also lower. That’s because DCM will lead to excessive load of the neighboring controllers. Although the migration cost of EOSDM is slightly higher than that of DCM, it is obviously better than DCM in terms of the throughput of the controller. The MMM strategy has less migration cost than EOSDM, and it has the highest throughput among the three strategies. Because the MMM strategy not only considers the migration cost, but also considers the additional cost of data flow during cross-domain migration, which makes the additional cost is the smallest.

4.3 Load balancing rate under different strategies

In this set of experiments, the two network topologies are selected in TopologyZoo to compare the load balancing rate of the four strategies with the number of overloaded controllers. The two topological information is as follows: BT: 36 nodes, 76 links; Columbus: 70 nodes, 85 links. We set 10 controllers in the network, and the number of overload controllers does not exceed 80% of the total number, that is, 8 overload controllers. The experimental results are shown in Figure 5 (a) and Figure 5 (b), we can see that the load balancing rate decreases with the number of overloaded controllers increases. As the number of overloaded controllers gradually increases, the load balancing rate of SC decreases rapidly. When the neighboring controllers are overloaded, the load balancing rate of DCM will decrease and EOSDM can maintain the load balancing rate better. Since MMM considers the impact on all domains during migration, not only the source domain and destination domain, it can ensure efficient migration and its performance has always been better than the other three strategies.
5. Conclusion

In this paper, we investigate the problem of switch migration for city-aware software-defined IoT networks and propose a dynamic switch migration strategy MMM. MMM can perform dynamic switch migration according to the dynamic changes of data traffic in the network, ensure load balancing among controllers and minimize migration cost during the migration. In simulation experiments, we evaluate the performance of MMM by comparing the four strategies at the cost of migration and controller throughput. We further analyze the impact of the number of overloaded controllers on the load balancing rate under different network topology sizes. The experimental results prove the effectiveness of the proposed scheme.

Acknowledgments
The study was supported by Natural Science Foundation of China (Grant No. 61702370), Natural Science Foundation of China (Grant No. 18JCYBJC85900,18JCQNJC70200, 17JCYBJJC16400).

References
[1] C Dong, Y Q Qi and M Liu 2018 One-Request Scheme for M2P Data Transmissions in Software Defined IoT Networks J. IEEE access 13090-13100
[2] J Liu, Y Li, M Chen and W Dong 2015 Software-defined internet of things for smart urban sensing J. IEEE Communications Magazine vol. 53, no. 9, pp. 55–63
[3] Bari M, Roy A R, Chowdhury S R, et al. 2013 Dynamic controller provisioning in software defined networks A. Proceedings soft International Conference on Network and Service Management C.IEEE 18-25.
[4] J K Yu; Y Wang; K K Pei, et al. 2016 A Load Balancing Mechanism for multiple SDN Controllers based on Load Informing Strategy J. IEEE
[5] T Hu, P Yi, et al. 2018 A distributed decision mechanism for controller load balancing based on switch migration in SDN J. China Communications
[6] McKeown N, Anderson T, Balakrishnan H, et al. 2008 OpenFlow: Enabling innovation in campus networks J. ACM sigcomm Computer Communication Review 38(2):69-74
[7] L Yao, T Hu, P Yi, et al. 2019 Switch dynamic migration strategy base on efficiency optimization in SDN J. Journal of Electronics 47(7):1482-1489
[8] OpenDaylight[OL].http://www.opendaylight.org/,2020.
[9] Mininet[OL].http://mininet.org/,2020.
[10] Internet2 Open Science, Scholarship and Services Exchange[OL].http://www.internet2.edu/.
[11] Knight S, Nguyen H X, Falkner N, et al. 2011 The Internet topologyzoo J. IEEE Journal on Select ed Areas in communications 29(9):1765-1775.