Wireless Pers Commun (2017) 94:3549–3574
DOI 10.1007/s11277-016-3790-y

Load-Balancing Multiple Controllers Mechanism for Software-Defined Networking

Yi-Wei Ma[1] · Jiann-Liang Chen[2] · Yao-Hong Tsai[3] · Kui-He Cheng[2] · Wen-Chien Hung[4]

Published online: 27 October 2016

Abstract Software-Defined Networking (SDN) is a new form for networking that has the potential to have a major impact on Internet technology. Key aspects of SDN include the separation of data plane and the control plane. SDN architecture facilitates the management of complex networks and represents a new solution for future network applications Internet technology. As the size of networks increases, the scalability of the centralized controller becomes increasingly important. This work proposed a load-balancing mechanism for use in a multiple-controller SDN environment by implement a hierarchical control plane with both a meta-control plane and a local control plane. The meta-control plane analyzes the resources and utilization of the local control plane to optimize processing performance. This mechanism supports the load balancing of the local control plane to optimize data plane performance and eliminate the bottleneck of the centralized control in a network. This work analyzes the proposed load-balancing mechanism in a multiple-controller SDN environment. The results indicate that the meta controller based manager mechanism can monitors and effectively deal with the loading of the overloaded local controller. Based on the results thus obtained, the proposed local control plane scheduling balances the loadings of the multiple controllers and increases the network throughput by 12.7 and 9.2 % over those achieved using a centralized controller and multiple controllers without a scheduling mechanism.

Keywords Software-defined networking · Load balancing · Multiple controllers · Hierarchical control · Meta controller

✉ Yi-Wei Ma
yiweimaa@gmail.com

[1] China Institute of FTZ Supply Chain, Shanghai Maritime University, Shanghai, China

[2] Department of Electrical Engineering, National Taiwan University of Science and Technology, Taipei, Taiwan

[3] Department of Information Management, Hsuan Chuang University, Hsinchu, Taiwan

[4] Smart Network System Institute, Institute of Information Industry, Taipei, Taiwan

1 Introduction

With the rapid development of the relevant technology, an increasing number of services depend on the Internet. Traditional networks face such difficulties in the areas of flexibility, programmability, and virtualization [1–3]. Software-Defined Networking (SDN) architecture is a new networking architecture with the purpose of improving the traditional network. Key aspects of SDN include the separation of data planes and control planes. Accordingly, SDN provides the advantages of programmability, automation, and network control, enabling the operator to build highly flexible and programmable networks that readily adapt to a changing network environment.

OpenFlow uses the centralized control model and is composed of the controller, switch, and protocol [4–6]. In the centralized control model, all of the network routes are selected by the controller, so the first packet in each data flow is sent to the controller. Then, the controller computes the routing path for each data flow and sets the flow table to the corresponding switches in a manner consistent with a global network view. Here, the first packet of each data flow is typically the data flow initialization request. Additionally, the request processing capacity of a single controller is limited. For instance, a NOX controller can process approximately 30,000 requests and a Maestro controller can process approximately 600,000 requests per second. Based on the above analysis, a controller may cause congestion or overload of a controller channel. Figure 1 shows the congestion/overloading problem of interest in the SDN control plane [7]. As the scale of deployment increases, the problems of performance and scalability of centralized control become increasingly apparent [8, 9].

Based on the above analysis, to increase the network capability of the control plane, this study applies the idea of load balancing, decentralizing, data sharing and grouping to the SDN, and proposes a mechanism of scalable intra-domain control, called a multiple-controllers decentralized strategy. The first step is to balance the many data flow initialization requests, and then decentralize the controllers based on a shared resources by the

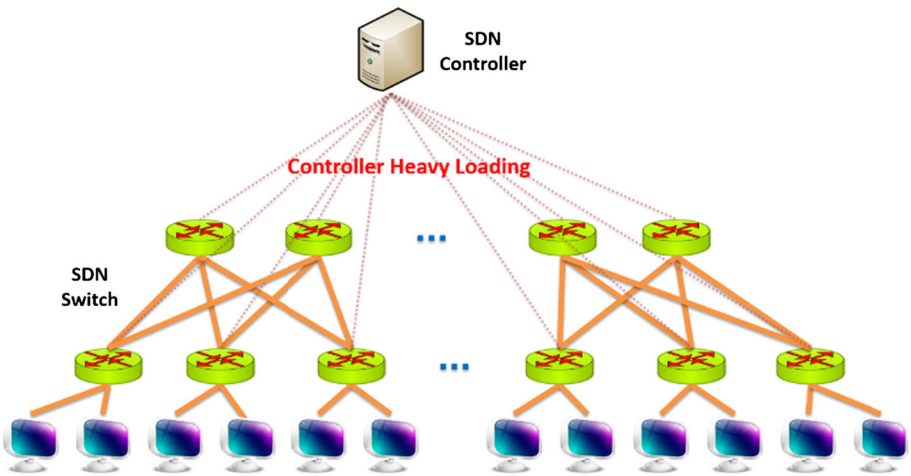

Fig. 1 The congestion/overloading problem in SDN control plane

controllers themselves in global network view. Each controller then computes and installing a routing path for every request according to its privileges. Hence, the strategy based on multiple decentralized controllers can completely solve the problem performance and scalability [10–12].

The Ref. [13] are focus on energy efficient in resource management for real time vehicular. The Ref. [14] are focus on servers load balancing on software defined network. The Ref. [15] are focus on anchor point selection for mobility management. The Ref. [16] are focus on container supply chain network optimization. This work focus on load-balancing for use in a multiple-controller SDN environment by implement a hierarchical control plane with both a meta-control plane and a local control plane.

The rest of the paper is organized as follows. Section 2 presents the background of this study. Section 3 introduces the proposed load-balancing multiple controller mechanism. Section 4 presents the performance analysis of this study. Finally, conclusions are drawn in Sect. 5.

2 Background Knowledge

2.1 SDN

Traditional network architectures cannot satisfy the requirements of today's corporations, carriers, and end users. SDN increases the operating efficiency for cloud computing data centers, motivating Internet service providers, telecommunications operators and various research centers to conduct research into related technologies, with a view to addressing issues related to existing networks using SDN technology [17, 18].

SDN is a new method that has the potential to have a major impact on network environment. One critical aspect of SDN include the separation of data planes and control planes. Consequently, SDN offers the advantages of programmability, automation, and network control, enabling an operator to construct highly scalable, flexible networks that readily adapt to a changing network requirement. The architecture involves a communications protocol for establishing forward tables in switches. The switch follows the results that are specified in the flow tables and the group table to lookup or forward packets. The relevant flow entries in flow tables can be added, deleted and updated by the controller. When a flow arrives to the switch, it is matched against flow entries in the flow table. The SDN action is triggered when the flow entries are matched.

2.2 OpenFlow

OpenFlow is an open standard that is based on an Ethernet switch, with an internal flow-table, and a standardized interface for adding and removing flow entries from the network environment. The OpenFlow Consortium proposed the concept of OpenFlow in 2008, to facilitate network virtualization and scholars implementation, for example, new protocols and applications on experiment networks without affecting them existing network. An OpenFlow switch comprises a flow table, which utilized in packet lookup and forwarding, and OpenFlow-related software, which handles the transmission of packet information between the device and the controller via secure channel.

2.3 Secure Channel

A secure channel allows commands and packets to be sent between a controller and a switch using the OpenFlow protocol, which provides an open and standard way for a controller to communicate with a switch, as well as an encryption function. The two forms of secure channel are in-band control, which sends data packets and control packets via a single plane, and out-of-band control, which sends data packets and control packets via different planes.

2.4 Controller

Although a controller consumes only one-third of the OpenFlow, it is nevertheless important. A controller controls and adjusts all of the OpenFlow switches, establishes a virtualized network and forwards the packets. Both in-band and out-of-band control require a network manager to define the policy on the controller. In this work, NOX is utilized as the controller. The NOX controller is an open-source development that it is developed using such an open platform. In this work, this platform is utilized to control the entire network. Still under heavy development, NOX is currently utilized in numerous large production networks [19].

2.5 Distributed Control in SDN

There are two main architectures of distributed control planes.

- *Hierarchical SDN Control Plane* Tasks are distributed to various controllers, depending on locality requirements, and some of these may be processed topologically close to an SDN switch. The Kandoo [20], which arranged controllers in two layers. Only local control applications are run in the bottom layer. The local controllers process most of the frequent events close to the data path and guard the higher layers, which can therefore concentrate on more global events. Figure 2 shows the structure of a hierarchical control plane.
- *Flat SDN Control Plane* The network is separated into numerous controller domains, and each controller manages a non-associate set of SDN switches. The architecture is principally motivated by managerial constraints: different geographical managers own and independently manage different networks. A flat control plane may also be utilized to reduce the latency between SDN switches and controllers, and to keep the control traffic local. An example of a flat control plane is ONIX [21, 22].

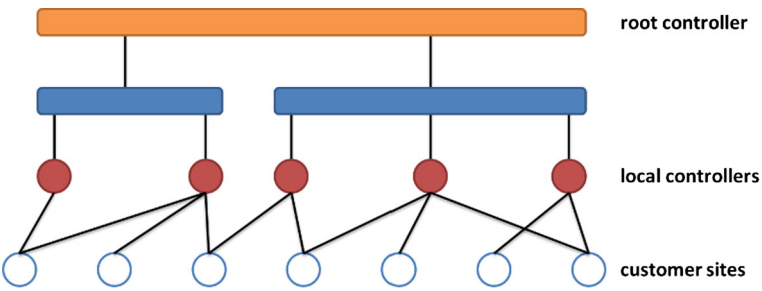

Fig. 2 Hierarchical control plane

3 Load-Balancing Multiple Controllers Mechanism

This work provides an overview of the proposed system architecture, and then describes the proposed scheduling algorithm. Then, the proposed MC-based manager mechanism is explained.

3.1 System Overview

This work proposes a Meta Controller (MC) to manage a multiple controller environment using an SDN strategy with the purpose of identifying the switch that should switch to a new controller when a controller is overloaded. Figure 3 shows the proposed architecture of the MC. In this work, the system is deployed for a large WAN or data center that is composed of SDN switches. In the architecture, controller states are distributed by placing the MC and multiple local controllers in a hierarchical structure. The system is composed of four main parts. The physical layer contains the connections of the physical network of switches and controller. The abstraction layer is composed of numerous domains, each of which includes abstracted switches, which are controlled by an active local controller. The local controller layer includes numerous Active Local Controllers (ALC) and Inactive Local Controllers (ILC). A local controller is declared active if it has at least one switch; otherwise, it is considered inactive. Inactive controllers keep listening for HELLO messages from an MC while awaiting new assignments of switches. The MC layer manages the local controller layer and load balance among local controllers when the ALC is overloaded. The MC monitors the utilization of each local controller and the network utilization to optimize the scheduling of the switches. The scheduling framework regularly evaluates the current assignment of the switch to a local controller and determines whether to

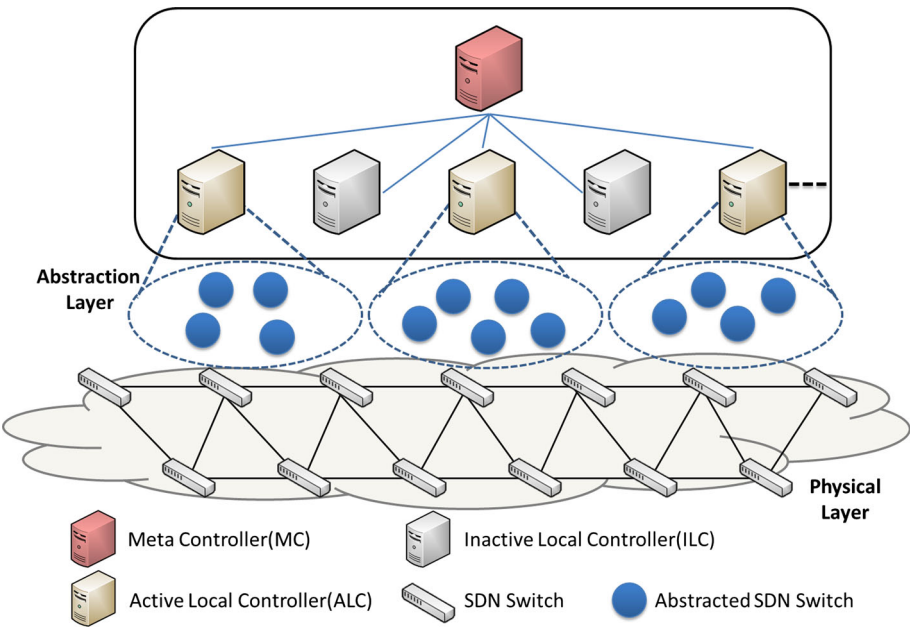

Fig. 3 System architecture

reassign it under the specified constraints. When a reassignment is conducted, the scheduling framework also relocates the switch to the local controller to which it is assigned in the network.

3.2 System Operation and Architecture

This work uses the OpenFlow protocol for communication between switches and controllers in the SDN. ALC performance is evaluated to determine whether the ALC is overloaded the MC. If one of the ALCs is overloaded and some of the switches must be assigned to another ALC or ILC, then the overloaded ALC will trigger events and notification of the Global Agent. The Global Agent that information about them is received, the MC will collect the information form every ALC that was in charge of tasks. Concurrently, the MC must newly schedule the switches and all local controllers in a manner consistent with the network environment. Thereafter, the MC will schedule loading and send messages to the Local Agent. Figure 4 shows the system architecture, which is composed of an MC-based manager, three ALCs and two ILCs.

3.3 Load Balancing Module

A load balancing module for distributing the loading of ALCs is presented herein. Figure 5 shows the functions of this load balancing module. The Meta Control Plane is composed of two blocks the Global Agent and the Resource Scheduler. The Global Agent monitors all Local Agents to collect the statuses of local controllers and computes the loads of the ALCs. The Resource Scheduler schedules the local controllers, arranges the SDN switches and plans to the local controllers. Then, the MC will be triggered, based on the result of the

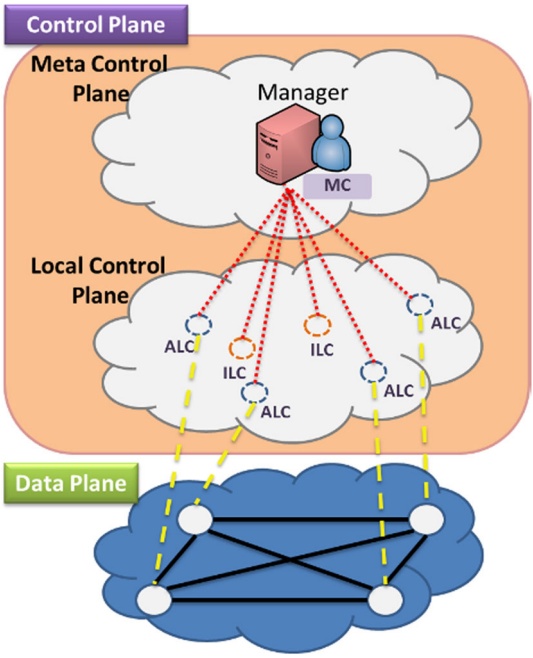

Fig. 4 Concepts of system architecture

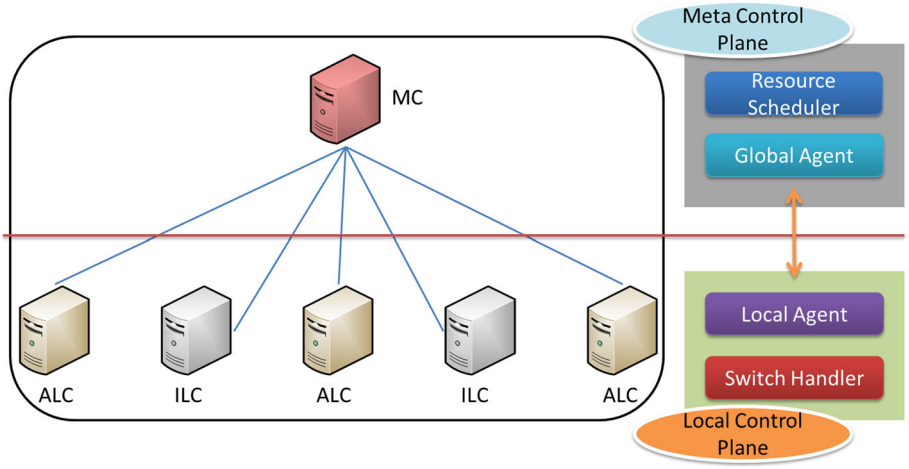

Fig. 5 Functions of load balancing module

Resource Scheduler analysis of whether the ALC is overloaded, and it delivers the relevant data to the Local Agent.

The Local Control Plane is composed of two blocks, which are the Local Agent and the Switch Handler. The Local Agent receives data concerning over-loaded ALCs from the Resource Scheduler, triggering a request for allocating information. The Switch Handler arranges the switches based on the results of scheduling.

3.3.1 Meta Control Plane

The MC plane, which is composed of Global Agent and Resource Scheduler, is presented herein. Figure 6 shows the operation of the MC plane, whose capabilities are described below.

- *Global Agent* The global agent primary collection of data is to monitor the statuses of the ALCs, which are of the CPU, the control channel traffic and the control channel latency. Then, the loads of the ALCs are estimated and ranked. If one of the ALCs is overloaded, then the resource scheduler is triggered to rearrange the controllers and switches.
- *Resource Scheduler* The resource scheduler evaluates the loads of the ALCs that are in charge of tasks and the network utilization, which is obtained from the Global Agent, to optimize the arrangement for processing the distribution from the local controllers at the lowest processing cost.

3.3.2 Local Control Plane

The local control plane has two main components, which are the Local Agent and the Switch Handler. Figure 7 shows the operation of the load balancer. The functions of the local control plane are described below.

- *Local Agent* The local agent primary collection of data is to monitor the status of each local area in terms of SDN switch traffic and latency.

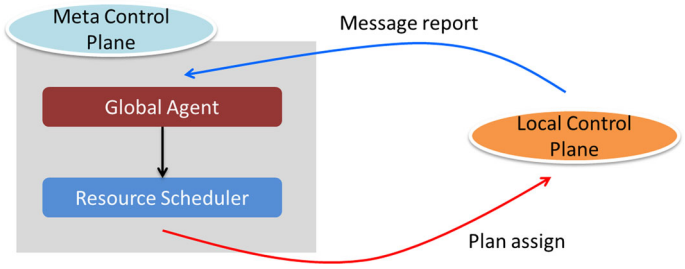

Fig. 6 Operation of the meta control plane

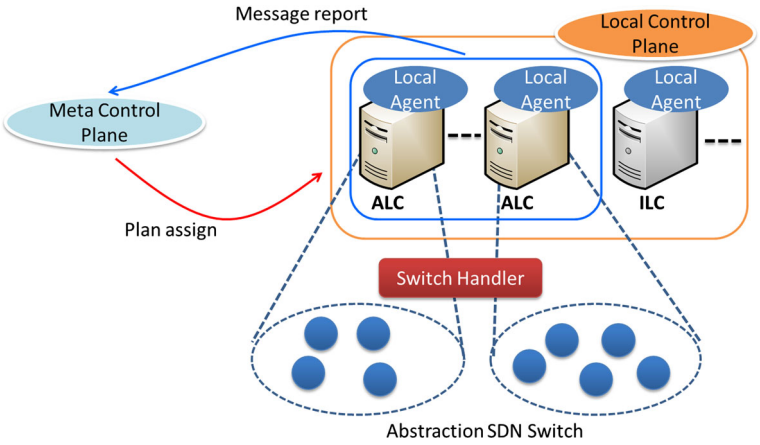

Fig. 7 Operation procedure of local control plane

- *Switch Handler* The switch handler according to the results of analysis that is obtained from the resource scheduler in the MC plane via the local agent to determine whether the switch had to reassign. If the ALC is over loaded, then the switch handler will be triggered according to plan for arranging switches.

Figure 8 shows the simple process that is conducted by the load balancing module, which is as follows.

- Step 1: The global agent monitors all of the ALCs to obtain information about the local agent and to estimate the load of the ALCs. If one of the ALCs is overloaded, then the resource scheduler is triggered to rearrange the controllers and switches.
- Step 2: When the global agent triggers the resource scheduler, the scheduler will optimally schedule the local controllers and switches based on a mathematical model at the lowest processing cost.
- Step 3: The resource scheduler sends the assignment to the local control plane according to calculation.
- Step 4: The switch handler is triggered according to plan for arranging the switches.

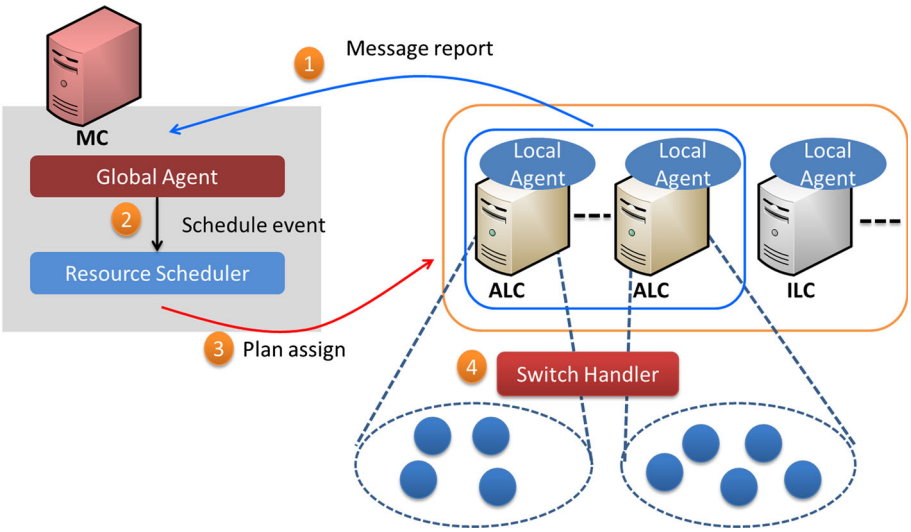

Fig. 8 Simple process of the load balancing module

3.3.3 Problem Formulation for Hierarchical Control Structure

The optimal controller scheduling problem is formulated herein as a linear programming problem. Table 1 shows the notations that is utilized in the mathematical model.

Figure 9 shows the threshold of proposed mechanism. A system utilization between zero and the "basic" value is regarded as "low". In this work, an active local controller is deactivated to increase system utilization. A utilization between basic value and the lower bound is regarded as "normal". System utilization between the lower bound and the upper bound is in the warning area. In this work, an inactive local controller can be activated to reduce the system utilization. System utilization between the upper threshold rather than bound and 100 % is regarded as "high". The scheduling module is triggered a high utilization to share loading. Notably, the manager can adjust the basic, lower and upper thresholds as required.

Specifically, the control plane is modeled as MC = {ALC, ILC}, where ALC is the set of active local controllers and ILC is the set of inactive local controllers. Let S be the set of SDN switches in the data plane. $\rho(a)(1)$ evaluates represents the load of local controller a through the CPU, traffic and latency. DL_{aj} is the latency score from the SDN switch j to local controller a in units of time. Figure 10 shows the latency score.

$$DL_{aj} = \begin{cases} L_{aj}, & if\ L_{aj} \leq ML_a; \\ 1, & if\ L_{aj} > ML_a; \end{cases}$$

Notably, the sum of α, β and γ is one. The manager can adjust the parameters according to the environment. $\tau(a,j)(2)$is the cost from the SDN switch j to the local controller a. Notably, the sum of δ and ε is one. The manager can adjust these parameters according to the environment. SLV $(a)(3)$ is the cost of the SDN switch in the local controller a. CS (4) is the set of candidate SDN switches in the local controller a. Notably, the data are sorted in descending order. $SC_{aj}(5)$ is the estimated CPU utilization of SDN switch j in local

Table 1 Notations of mathematical model

Symbol	Meanings
$\rho(a)$	The load value of local controller a
C_a	The CPU utilization of local controller a
T_{aj}	The amount of control channel traffic generated between SDN switch j to local controller a in an unit of time
MT_a	The maximum control channel traffic of local controller a
L_{aj}	The amount of control channel latency generated between SDN switch j to local controller a in an unit of time
ML_a	The maximum tolerance of control channel latency of each SDN switch in local controller a
DL_{aj}	The latency score between SDN switch j to local controller a in an unit of time
$n(a)$	The number of SDN switch in the local controller a
α	The weight of CPU utilization for $\rho(a)$
β	The weight of control channel traffic for $\rho(a)$
γ	The weight of control channel latency for $\rho(a)$
$\tau(a,j)$	The cost between SDN switch j to local controller a
δ	The weight of control channel traffic for $\tau(a,j)$
ε	The weight of control channel latency for $\tau(a,j)$
$\mathrm{SLV}(a)$	The set of is the cost of SDN switch in the local controller a
CS	The set of candidate SDN switch that sort in descending order
SC_{aj}	The estimate value of CPU utilization of SDN switch j in local controller a
$\psi(b,j)$	The estimate load value when candidate SDN switch j assign to neighbor controller b
$\theta(b)$	The estimate total of load value that assign to local controller b
C_M	The cost of the controller management
C_R	The cost of the SDN switch reassignment
P	The total cost
y_a	The local controller status
$v(j, b)$	The SDN switch reassignment status
λ_b	The basic rate
λ_{LB}	The lower bound
λ_{UB}	The upper bound

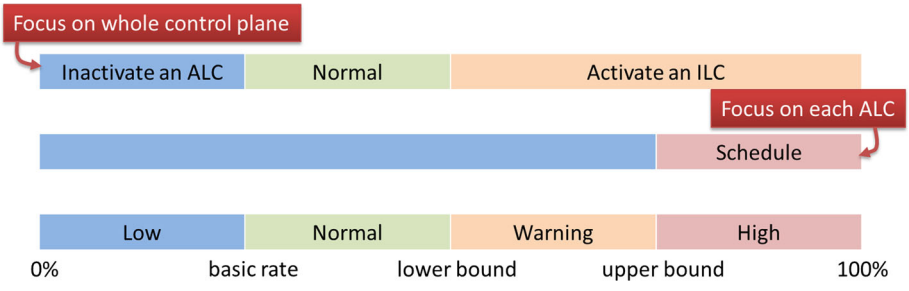

Fig. 9 Expression of threshold

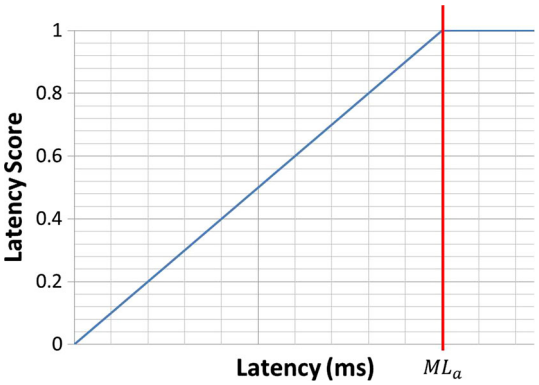

Fig. 10 Expression of the latency score

controller a. The manager can adjust parameters according to the environment. $\psi(b,j)(6)$ is the estimated load when candidate SDN switch j is assigned to its neighboring controller b. $\theta(b)$ (7) is the estimated total load that is assigned to local controller b.

$$\rho(a) = C_a \cdot \alpha + \frac{\sum_{j=1}^{n(a)} T_{aj}}{MT_a} \cdot \beta + \frac{\sum_{j=1}^{n(a)} DL_{aj}}{n(a)} \cdot \gamma \tag{1}$$

$$\tau(a,j) = \frac{T_{aj}}{MT_a} \cdot \delta + DL_{aj} \cdot \varepsilon \tag{2}$$

$$\mathrm{SLV}(a) = \{\tau(a,j) | j = 1,2...n(a)\} \tag{3}$$

$$\mathrm{CS} = \{Sort[SLV(a)]\} \tag{4}$$

$$SC_{aj} = \frac{C_a \cdot T_{aj}}{\sum_{i=1}^{n(a)} T_{ai}} \tag{5}$$

$$\psi(b,j) = SC_{aj} \cdot \alpha + \frac{T_{bj}}{MT_b} \cdot \beta + DL_{bj} \cdot \gamma \tag{6}$$

$$\theta(b) = \rho(b) + \sum_{j \in S} \psi(b,j) \tag{7}$$

Two costs that are incurred when multiple controllers are deployed in the SDN environment are defined as follows:

- **Controller management cost** (C_M) is the load on every local controller that collects statistics from its associated switches. Based on the assumption that every controller has the same capability. The cost is as follows.

$$C_M = \sum_{a \in MC} \rho(a) \cdot y_a \tag{8}$$

- **SDN Switch reassignment cost** (C_R) is the cost of assigning a switch to a new local controller. Ideally, the frequent reassignment of SDN switches should be avoided. In particular, $v(j, b) = 1$ if the SDN switch j has been assigned to local controller b; otherwise $v(j, b) = 0$.

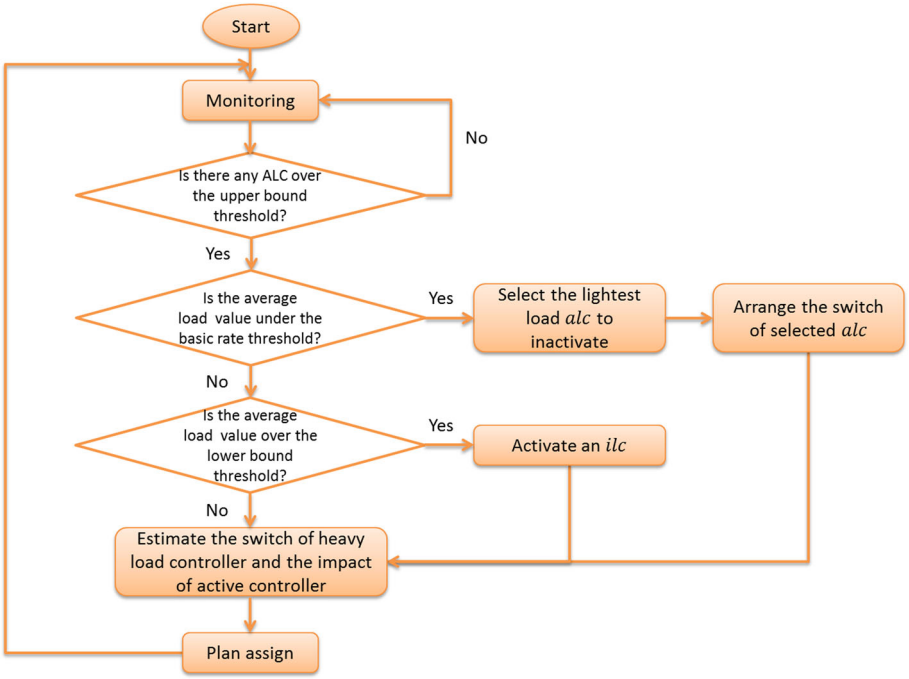

Fig. 11 Flowchart of MC-based manager mechanism

$$C_R = \sum_{j \in S} \sum_{b \in MC} \psi(b,j) \cdot v(j,b) \tag{9}$$

The objective function in the optimization problem is to minimize the sum of the two above costs and is expressed as follows.

$$Minimize : P = A \cdot C_M + B \cdot C_R \tag{10}$$

where A and B are constants, which the manager can determine to regulate the relative magnitudes of the two cost components. The following constraints must be satisfied to guarantee a feasible solution.

$$\lambda_b < \frac{\sum_{a \in MC} \rho(a)}{\sum_{a \in MC} y_a} < \lambda_{LB} \tag{11}$$

$$\forall_{b \in MC} : \theta(b) < \lambda_{UB} \tag{12}$$

$$\forall_{a \in MC} : \sum_{j \in S} T_{aj} \leq y_a \cdot MT_a \tag{13}$$

$$\forall_{j \in S, b \in MC} : v(j,b) \in \{0,1\} \tag{14}$$

$$\forall_{a \in MC} : y_a \in \{0,1\} \tag{15}$$

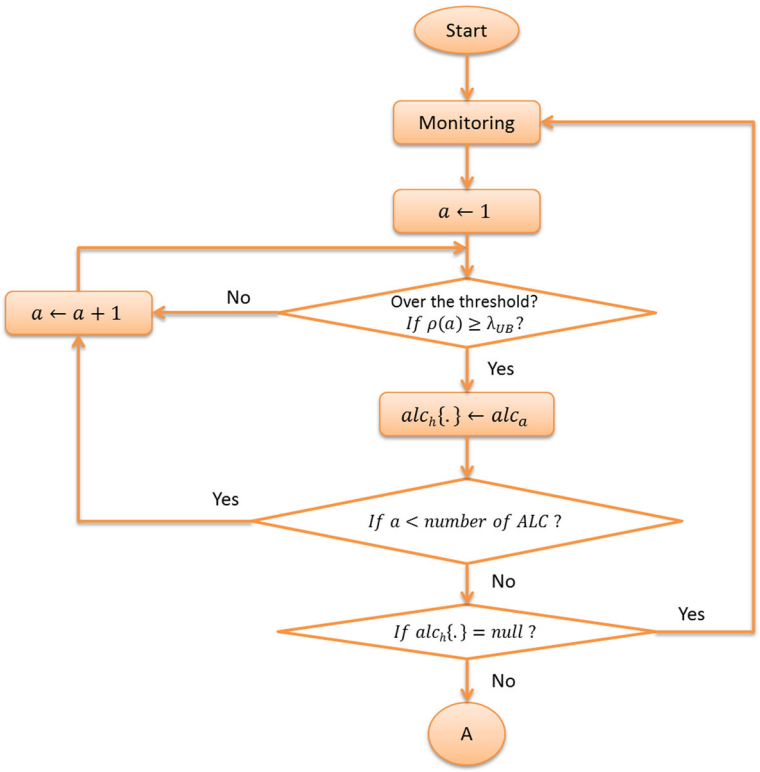

Fig. 12 Flowchart of load value evaluation mechanism

$$A + B = 1 \tag{16}$$

Inequality (11) guarantees that the mean load is between the basic and the lower thresholds in the SDN environment. The manager can adjust the thresholds λ_b and λ_{LB} as required. Inequality (12) ensures that the load of the local controller must be below the upper threshold. The manager can adjust the threshold λ_{UB} as required. Inequality (13) ensures that the total of traffic must be lower than maximum control channel traffic of local controller a. Equations (14) reveal that $v(j, b)$ are binary variables {0, 1}, which represent whether the SDN switches have moved. Equations (15) show that y_a is a binary variable {0, 1}. A value of 1 indicates an active local controller; zero indicates an inactive local controller. According to Eqs. (16) A and B are weights in the object function. The manager can adjust these parameters according to the environment.

3.3.4 Flow Chart and Algorithm of MC-Based Manager Mechanism

Figure 11 shows a flowchart of the MC-based management mechanism. The MC-based management mechanism is as follows.

- Step 1: The MC-based manager monitors the local control plane and collects information about every ALC, including about the CPU, traffic and latency.

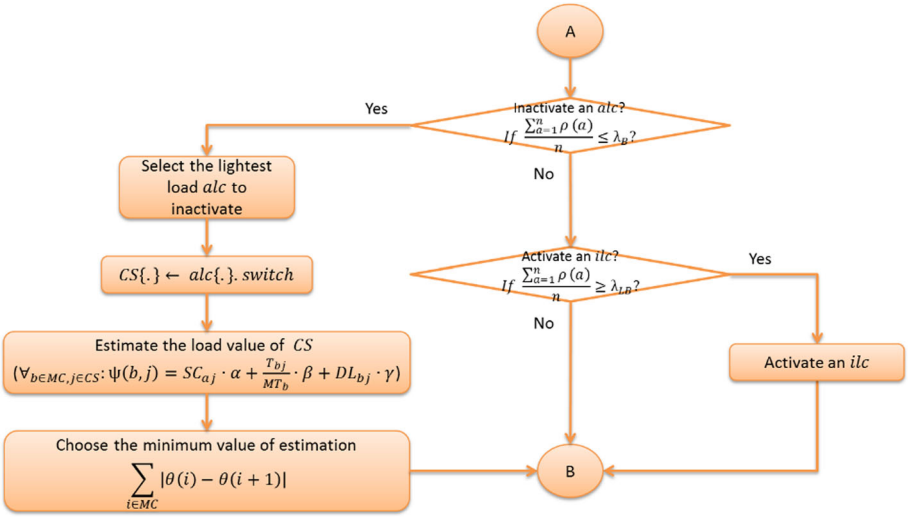

Fig. 13 Flowchart of local control plane utilization evaluation mechanism

- Step 2: The load of every ALC is evaluated. If the load of an ALC exceeds the upper threshold, then is added to the processing list and waits arranged.
- Step 3: The mean load of the local control plane is evaluated. If the system is less than the basic threshold, then the minimum load of the ALC is deactivated and the switch of the selected ALC.
- Step 4: The mean load of the local control plane is evaluated. If the average of the system exceeds the lower threshold, then the ILC is activated.
- Step 5: Estimate the switch of the heavy load controller and the impact of the active controller, using the objective function.
- Step 6: MC assigned the plan result to the appropriate controller.

Figure 12 shows the flowchart of the valuation of the load. The evaluation of load proceeds as follows.

- Step 1: The MC-based manager monitors the local control plane and collects information about every ALC including about the CPU, traffic and latency.
- Step 2: The load of every ALC, $\rho(a)$, is evaluated using Eq. (1). $\rho(a)$ If any $\rho(a)$ exceeds the upper threshold, then it is added to the set alc_h and waits arranged.
- Step 3: Flowchart A elucidates evaluation mechanism. If no exceeds the upper threshold, then the local control plane will continue to be monitored.

Figure 13 shows the flowchart of the evaluation of local control plane utilization, which proceeds as follows.

- Step 1: The mean load of the local control plane is evaluated. If the average of the system is below the basic threshold, then the minimum load of the ALC will be deactivated and the switching will be scheduled according to Eqs. (4), (5) and (6) to arrange in a manner that is appropriate.

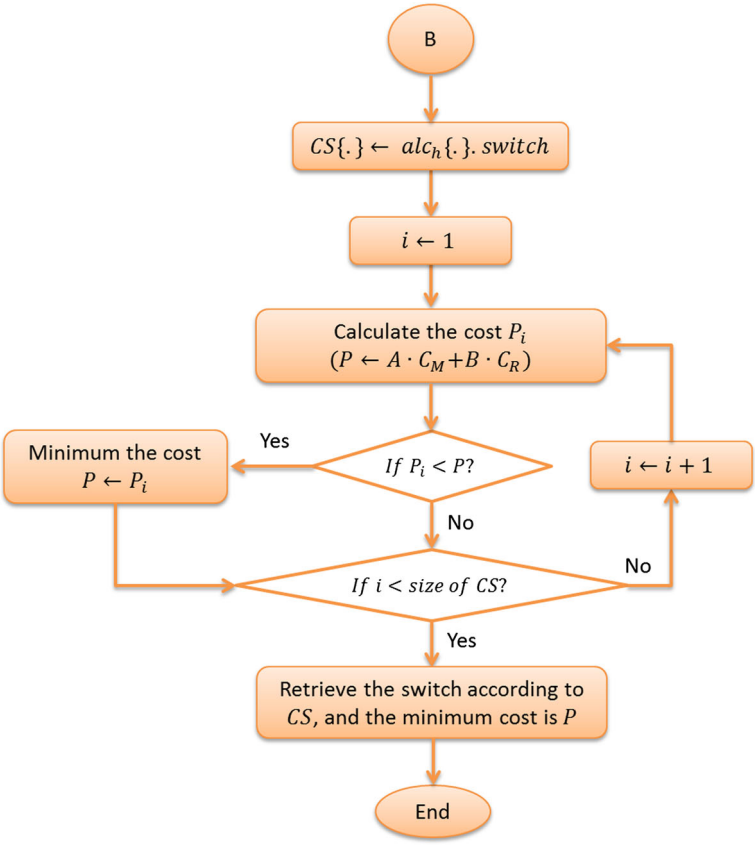

Fig. 14 Flowchart of switch arrangement mechanism

- Step 2: The mean load of the local control plane is evaluated. If the mean of the system exceeds the lower threshold, then the ILC is activated.
- Step 3: Flowchart B elucidates the evaluation of local control plane utilization.

Figure 14 shows the flowchart of switch arrangement, which proceeds as follows.

- Step 1: Add all SDN switches of alc_h to the set CS, according to Eq. (4) and sort the SDN switches in order of descending.
- Step 2: Every load is evaluated using Eq. (6) when a candidate SDN switch is assigned to a neighboring controller.
- Step 3: The cost of switch assignment is evaluated using Eq. (10) here. If other is better than the preceding one, it will replace it according to Eq. (10) and under constraint (11), (12), (13), (14), (15) and (16).
- Step 4: The minimum cost of P is retrieved and to the appropriate controller.

Figures 15, 16 and 17 shows the algorithm for load balancing. This work describes the pseudo code of local control plane scheduling that is used in the MC-based manager mechanism. The scheduling is designed to determine the most appropriate arrangement of SDN switches.

Load Balancing Algorithm

Input: *Traffic Matrix, T*
CPU Matrix, C
Latency Matrix, L
Set of active local controllers, ALC
Set of inactive local controllers, ILC
Set of switches, S
Basic rate, λ_B
Lower bound, λ_{UB}
Upper bound, λ_{LB}

Output: *New assignment, X*

1: *alc* ← *ALC*
2: alc_h ← *heavy load controller*
3: *LV* ← *controller load value*
4: *CS* ← *set of candidate SDN switches*
5: ***for*** *a* ← *1 to number of alc* ***do***
6: ***if*** $\rho(a)$ *defined by Equation (1)* $\geq \lambda_{UB}$ ***then***
7: $alc_h \leftarrow alc_a$
8: ***end if***
9: $LV \leftarrow LV + \rho(a)$
10: ***end for***

Fig. 15 Algorithm of load balancing mechanism (a)

11: ***if*** *LV / number of alc* $\leq \lambda_B$ ***then***
12: *inactivates an active controller*
13: *CS* ← *take SDN switches of alc and sort in descending order according to* $\tau(a, j)$ *defined by Equation (4)*
14: *Estimate the load value of CS according to* $\psi(b, j)$ *defined by Equation (6)*
15: *Choose the minimum estimate total of load value according to* $\theta(b)$ *defined by Equation (7) and* $\sum_{b \in MC} |\theta(b) - \theta(b+1)|$
16: ***else if*** *LV / number of alc* $\geq \lambda_{LB}$ ***then***
17: *activates an inactive controller*
18: ***end if***

Fig. 16 Algorithm of load balancing mechanism (b)

4 Performance Analysis

A test-bed with three local controllers and an MC is established as the experimental environment, which is shown in Fig. 18.

Figure 19 shows the topology of the data center. The data center network model is designed in the form of a tree and leaves. This realistic virtual SDN is created by Mininet, with 22 virtual switches and 16 virtual hosts. Switch1 and switch2 the core layer that provides routing services to other parts of the data center. Switch3 to switch6 in the aggregation layer provide connectivity among adjacent access layer switches and the core

19: $CS \leftarrow$ take SDN switches of alc_h and sort in descending order according to $\tau(a, j)$ defined by Equation (4)
20: Objective Function: $P \leftarrow A \cdot C_M + B \cdot C_R$
21: Add Constraint
22: Let $P \leftarrow \infty$
23: **for** $i \leftarrow 1$ to number of CS **do**
24: Calculate the cost P_i
25: **if** $P_i < P$ **then**
26: $P \leftarrow P_i$
27: $X \leftarrow$ Retrieve the switch according to CS
28: **end if**
29: **end for**
30: Retrieve the minimum cost of X, and the minimum cost is P

Fig. 17 Algorithm of load balancing mechanism (c)

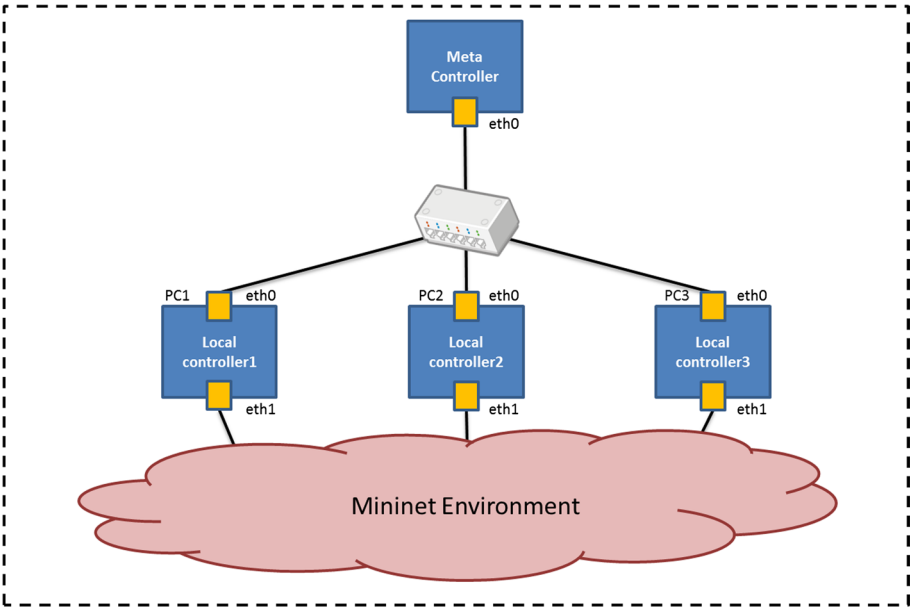

Fig. 18 Simulation environment

layer. Switch7 to switch22 in the access layer provide hosts resource that connect the components of the network to each other.

- **Case 1:** The implementation results are shown below. Figure 20 and 21 show the results of the Active Local Controller performance analysis. Figure 20 shows that at a time of almost 35 s, ALC1 was overloading and the MC-based manager mechanism was triggered to determine which ALC should take over the heavily loaded SDN switches. From 35 s to 38 s, MC a plan to the ALC with a light load for load sharing. At approximately38 s, the ALC1 performance fell from 80 % down to 54 %. Figure 21

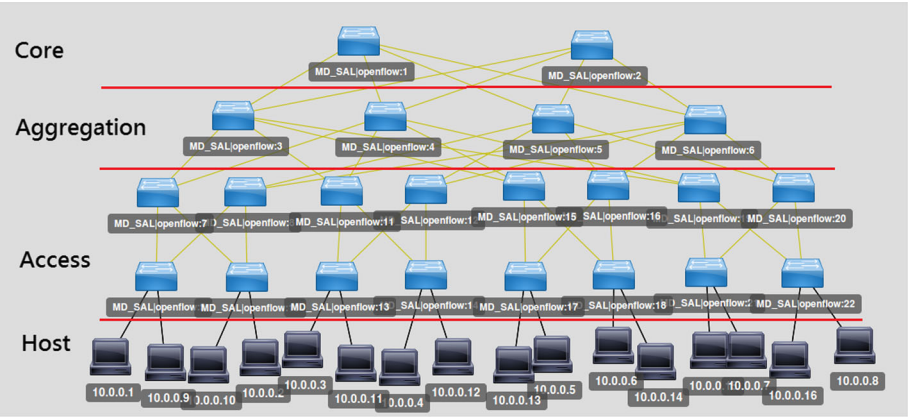

Fig. 19 Topology of data center

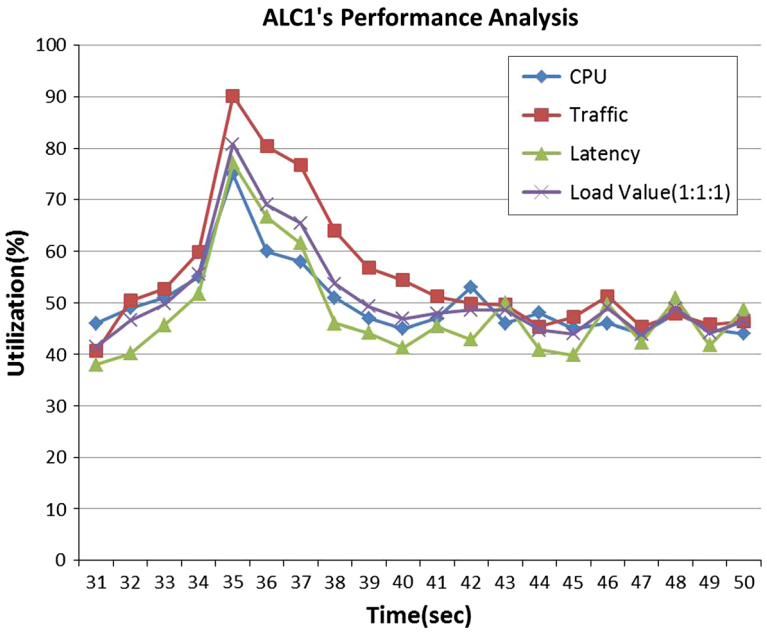

Fig. 20 ALC1's performance analysis (Case 1)

shows that at almost 36 s, ALC2 received the MC scheduling results and took over the servicing of the switches.

Figure 22 shows the analysis of the system loading. The aim of this study is to maintain the loading between the basic threshold and the lower threshold. The MC curve reveals that the average loading of the local controller was in the range specified above.

- **Case 2:** Figures 23 and 24 show the results of the analysis of the performance of the Active Local Controller. Figure 25 shows the result of the analysis of the performance of the Inactive Local Controller. Figure 23 shows that at almost 36 s, ALC1 was

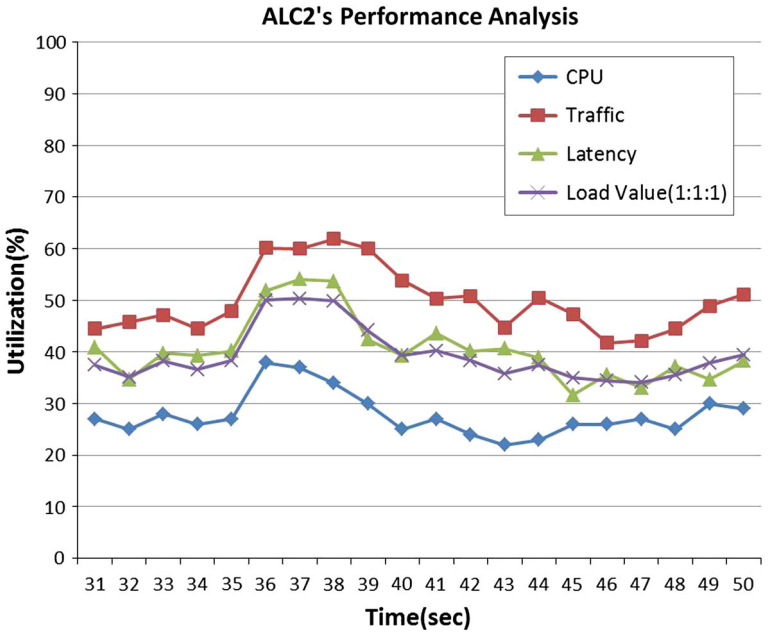

Fig. 21 ALC2's performance analysis (Case 1)

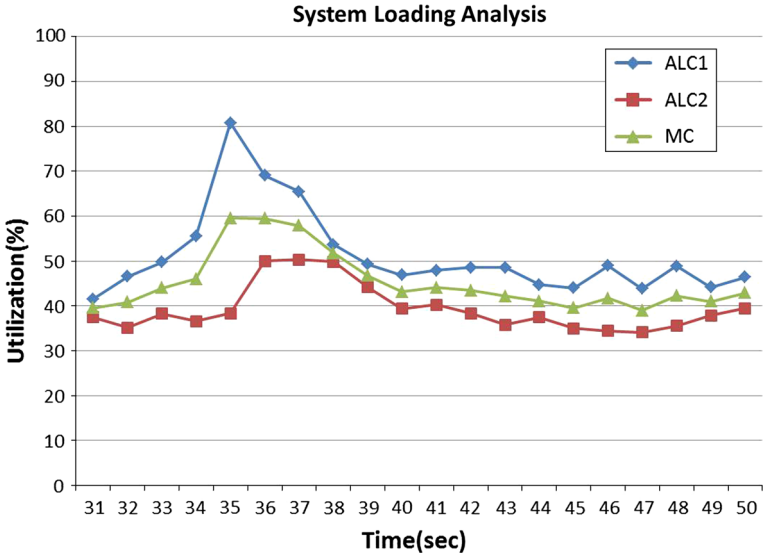

Fig. 22 System loading analysis (Case 1)

overloaded and the MC-based manager mechanism was triggered to determine which ALC should take over the heavily loaded SDN switches. The MC-based manager finds that the whole system is loaded above lower threshold and activates an ILC to become ALC3. In 36–39 s, MC sent a plan to ILC and the lightly load ALC to share the load.

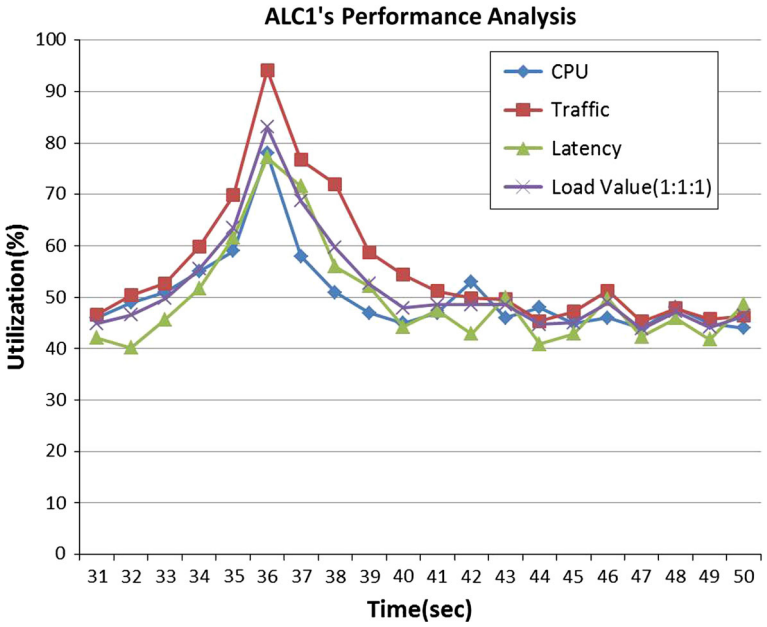

Fig. 23 ALC1's performance analysis (Case 2)

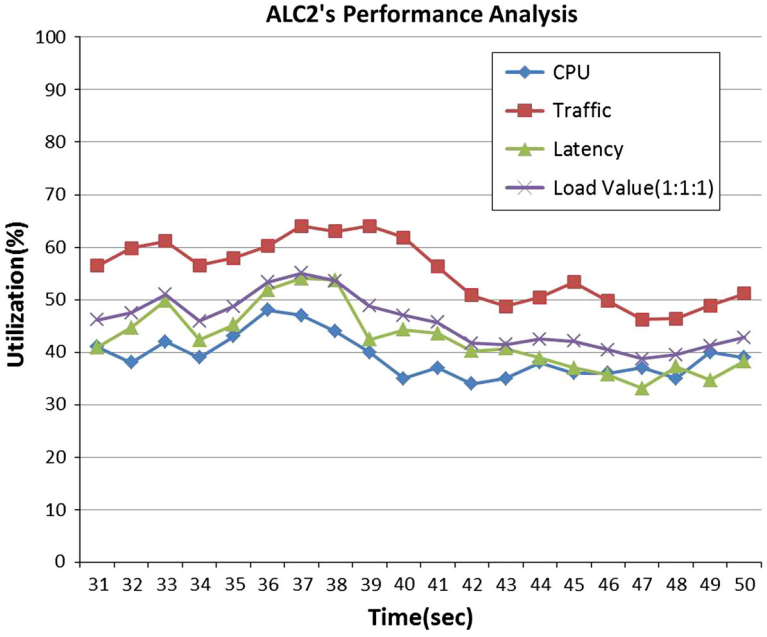

Fig. 24 ALC2's performance analysis (Case 2)

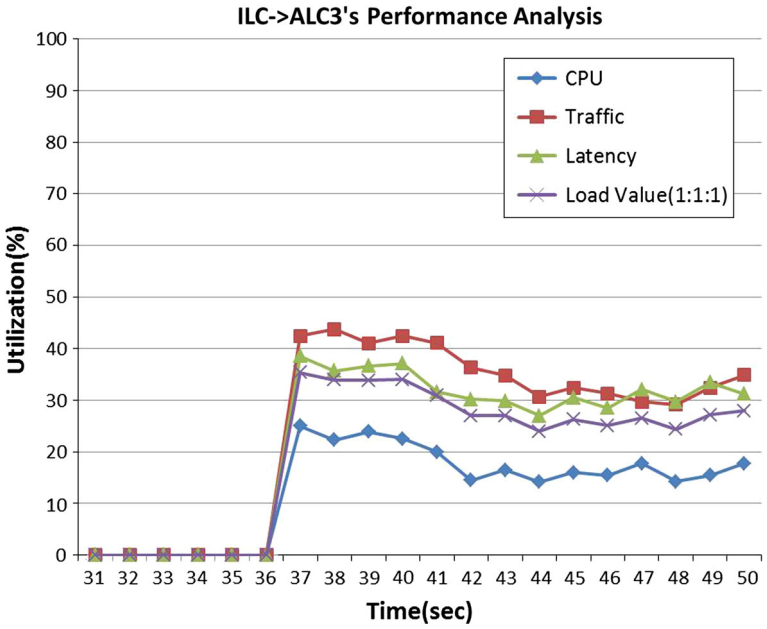

Fig. 25 ILC->ALC3's performance analysis (Case 2)

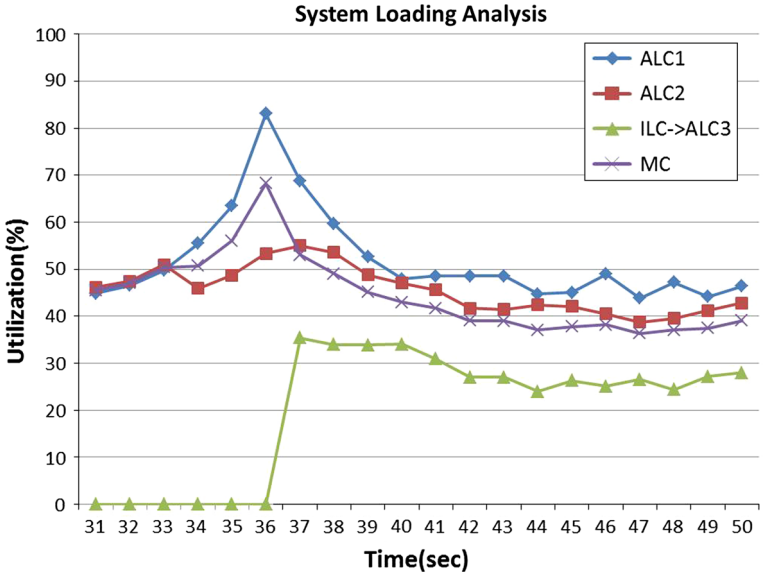

Fig. 26 System loading analysis (Case 2)

At approximately 39 s, the ALC1 performance fell from 83 to 53 %. Figures 24 and 25 show that at just before 37 s, ALC2 and ALC3 received the MC scheduling results and took over the servicing of the SDN switches.

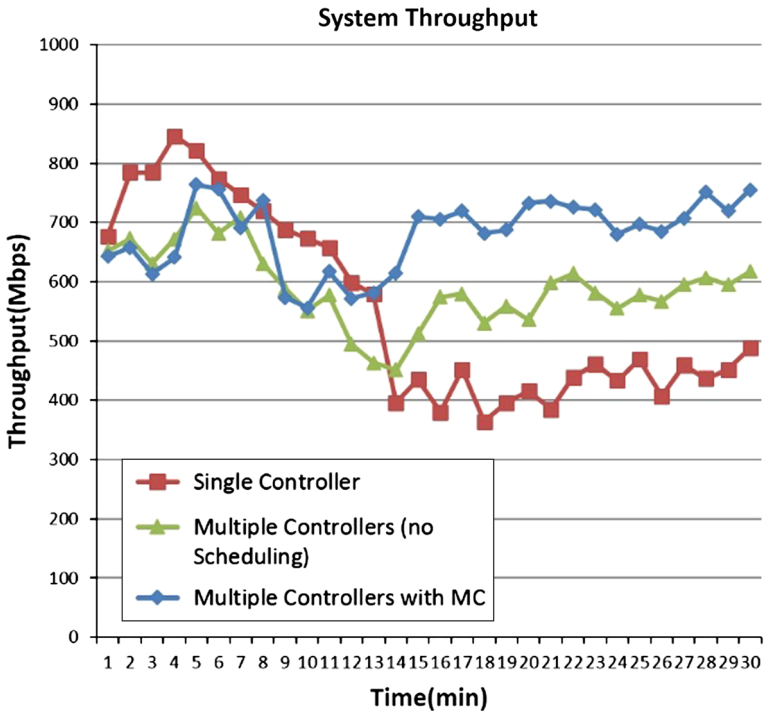

Fig. 27 Throughput analysis

Figure 26 shows the system loading analysis. In this study the goal is to maintain the loading between the basic (30 %) and lower thresholds (60 %). According to the MC curve in the figure, the average loading of the local controller is in above range.

Figure 27 shows the system throughput analysis with a single controller, multiple controllers and multiple controllers with the MC-based manager mechanism. Between 1 and 7 min, the throughput of the single controller is better than that of both multiple controllers. As the loading of the control channel increases, the single controller throughput falls. However, multiple controllers enough power to handle it. A comparison of the two cases with multiple controllers demonstrated that the multiple controllers with MC exhibited better processing performance than without MC, soothe throughput was higher. When the MC-based management mechanism was utilized, the throughput averaged 681 Mbps, or 68.1 % of the bandwidth. Without the MC-based management mechanism, the throughput averaged, or 58.9 % of the bandwidth, and the single controller yielded an average throughout of 554 Mbps, or 55.4 % of the bandwidth. These results demonstrate that the MC-based management mechanism provides a greater system throughput than normal multiple controllers and a single controller.

5 Conclusion

This work develops a load-balancing mechanism for use in a multiple controller SDN system. This work proposes an MC-based manager with multiple controllers for decentralized control in a Software-Defined Networking system. The most effective controller

for meeting the requirements of SDN switching if one of them is fully loaded in a Software-Defined Networking system is determined. This manager considers the loading of the SDN control plane. This work develops an MC-based manager mechanism with control plane decentralization to improve the performance of multiple controllers and ensure that the loading of each controller. This work proposes local control plane scheduling to determine the most appropriate allocation of the available local controller. Local control plane scheduling is based on information that is gathered from the MC-based manager; it includes controller statuses, the CPU and the network traffic. This work also proposes a means of analyzing the control channel to monitor the performance of the control plane. A hierarchical control structure is utilized to solve the problem of synchronization in a distributed controller environment. Ultimately, logically centralized control is achieved, and single controller architecture is made scalable and reliable. By analyzing the resources of the SDN controller and the utilization of the SDN system. The purpose is to construct a network with optimal processing performance. Experimental results demonstrate that the proposed MC-based management mechanism reduces the loading of the control plane. Another experiment focuses on load balancing and the utilization of the bandwidth of the SDN environment. The results of this experiment show that the proposed optimal assignment mechanism improved the load balancing of each SDN controller and increase the utilization of the bandwidth of the SDN environment by 12.7 and 9.2 % relative to that achieved using a single controller and multiple controlled without the MC-based manager.

Acknowledgments Authors thank for financial supports of the Ministry of Science and Technology, Taiwan under contract MOST 103-2221-E-011-069-MY2 and the Institute of Information Industry, Taiwan under project—SDN Test Cases Development and Implementation.

References

1. Chen, Y., Gong, X., Wang, W., & Que, X. (2012). VNMC for network virtualization in Openflow network. In *Proceedings of the IEEE international conference on cloud computing and intelligent systems* (pp. 797–801).
2. Salvadori, E., Corin, R. D., Broglio, A., & Gerola, M. (2011). Generalizing virtual network topologies in OpenFlow-based networks. In *Proceedings of the global telecommunications conference* (pp. 1–6).
3. Corin, R. D., Gerola, M., Riggio, R., Pellegrini, F. D., & Salvadori, E. (2012). VeRTIGO: Network virtualization and beyond. In *Proceedings of European workshop on software defined networking,* (pp. 24–29).
4. Shah, S. A., Faiz, J., Farooq, M., Shafi, A., & Mehdi, S. A. (2013). An architectural evaluation of SDN controllers. In *Proceedings of the IEEE international conference on communications,* (pp. 3504–3508).
5. Gorbunov, S., Ganjali, Y., Casado, M., & Sherwood, R. (2012). On controller performance in software-defined networks. In *Proceedings of the USENIX conference on hot topics in management of internet,* (pp. 10–15).
6. Othman, M. M. O., & Okamura, K. (2013). Enhancing control model to ease off centralized control of flow-based SDNs. In *Proceedings of the IEEE annual computer software and applications conference,* (pp. 467–470).
7. Phemius, K., & Bouet, M. (2013). OpenFlow: Why latency does matter. In *Proceedings of the IFIP/IEEE international symposium on integrated network management* (*IM*), (pp. 680–683).
8. Azodolmolky, S., Wieder, P., & Yahyapour, R. (2013). Performance evaluation of a scalable software-defined networking deployment. In *Proceedings of second European workshop on software defined networks* (pp. 68–74).
9. Guan, X., Choi, B. Y., & Song, S. (2013). Reliability and scalability issues in software defined network frameworks. In *Proceedings of second GENI research and educational experiment workshop*, (pp. 102–103).

10. Min, S., Kim, S., Lee, J., Kim, B., Hong, W., & Kong, J. (2012). Implementation of an OpenFlow network virtualization for multi-controller environment. In *Proceedings of international conference on advanced communication technology*, (pp. 589–592).
11. Lin, P., Bi, J., & Hu, H. (2012). ASIC: An architecture for scalable intra-domain control in OpenFlow. In *Proceedings of the international conference on future internet technologies*, (pp. 21–26).
12. Huang, W. Y., Hu, J. W., Lin, S. C., & Liu, T. L. (2012). Design and implementation of an automatic network topology discovery system for the future internet across different domains. In *Proceedings of 26th international conference on advanced information networking and applications workshops*, (pp. 903–908).
13. Shojafar, M., Cordeschi, N., & Baccarelli, E. (2016). *Energy-efficient adaptive resource management for real-time vehicular cloud services*. PrePrints: IEEE Transactions on Cloud Computing.
14. Zhou, Y., Ruan, L., Xiao, L., & Liu, R. (2014). A method for load balancing based on software defined network. *Advanced Science and Technology Letters, 45*, 43–48.
15. Bradai, A., Benslimane, A., & Singh, K. (2015). Dynamic anchor points selection for mobility management in software defined networks. *Journal of Network & Computer Applications, 57*, 1–11.
16. He, J., Huang, Y., & Chang, D. (2015). Simulation-based heuristic method for container supply chain network optimization. *Advanced Engineering Informatics, 29*(3), 339–354.
17. Wang, W., Hu, Y., Que, X., & Gong, X. (2012). Autonomicity design in OpenFlow based software defined networking. In *Proceedings of the IEEE Globecom workshops,* (pp. 818–823).
18. Fernandez, M. P. (2013). Comparing OpenFlow controller paradigms scalability: Reactive and proactive. In *Proceedings of the IEEE international conference on advanced information networking and applications* (pp. 1009–1016).
19. Oh, H., Lee, J., & Kim, C. (2011). A flow-based hybrid mechanism to improve performance in NOX and wireless OpenFlow switch networks. In *Proceedings of the IEEE vehicular technology conference* (pp. 1–4).
20. Yeganeh, S. H., & Ganjali, Y. (2012). Kandoo: A framework for efficient and scalable offloading of control applications. In *Proceedings of the first workshop on hot topics in software defined networks* (pp. 19–24).
21. Koponen, T., Casado, M., Gude, N., Stribling, J., Poutievski, L., Zhu, M., Ramanathan, R., Iwata, Y., Inoue, H., Hama, T., & Shenker, S. (2010). Onix: A distributed control platform for large-scale production networks. In *Proceedings of the USENIX conference on operating systems design and implementation* (pp. 1–6).
22. Yao, G., Bi, J., & Guo, L. (2013). On the cascading failures of multi-controllers in software defined networks. In *Proceedings of the IEEE international conference on network protocols* (pp. 1–2).

Yi-Wei Ma is a Lecturer in Shanghai Maritime University. He received the Ph.D. degree in Department of Engineering Science at National Cheng Kung University, Tainan, Taiwan in 2011. He received the M.S. degree in Computer Science and Information Engineering from National Dong Hwa University, Hualien, Taiwan in 2008. His research interests include internet of things, cloud computing, multimedia p2p streaming, digital home network, embedded system and ubiquitous computing.

Jiann-Liang Chen was born in Taiwan on December 15, 1963. He received the Ph.D. degree in Electrical Engineering from National Taiwan University, Taipei, Taiwan in 1989. Since August 2008, he has been with the Department of Electrical Engineering of National Taiwan University of Science and Technology, where he is a professor now. His current research interests are directed at cellular mobility management and personal communication systems.

Yao-Hong Tsai received the M.S. and Ph.D. degrees in information management from the National Taiwan University of Science and Technology (NTUST), Taipei, Taiwan, R.O.C., in 1994 and 1998, respectively. He was a Researcher with the Advanced Technology Center, Information and Communications Research Laboratories, Industrial Technology Research Institute (ITRI), Hsinchu, Taiwan. He is currently an Associate Professor with the Department of Information Management, Hsuan Chuang University, Hsinchu, Taiwan. His current research interests include image processing, pattern recognition, cloud computing, and internet of things.

Kui-He Cheng received the M.S. degree in Electrical Engineering of National Taiwan University of Science and Technology, Taipei, Taiwan. His research interests include digital home network, and software-defined network.

Wen-Chien Hung is the director of Broadband Network Applications Center of Institute for Information Industry (III), which is a Taiwan-government sponsored research institute. Mr. Hung is responsible for modern networking technology development and application integration, and his recent topics include LTE EPC, SDN, NFV, and vertical domain private networks. Mr. Hung is also highly involved in industrial promotion, and is one of the leaders of Taiwan SDN Alliance. Mr. Hung got his EE master degree from National Taiwan University in 1995.