Peer-to-Peer Netw. Appl. (2019) 12:440–451
DOI 10.1007/s12083-017-0605-1

SCPLBS: a smart cooperative platform for load balancing and security on SDN distributed controllers

Hong Zhong[1] · **Jianqiao Sheng**[1] · **Yan Xu**[1] · **Jie Cui**[1]

Received: 15 July 2017 / Accepted: 25 August 2017 / Published online: 19 September 2017

Abstract Software-Defined Networking (SDN) is a network architecture which has received much attention in recent years. It represents the future of network industry. As the Internet continues to exceed expectations of rapid development, a single centralized controller can be extended to distributed multiple controllers architecture. However, the distributed multiple controllers architecture is facing more and more serious challenges in the aspects of scalability, stability and security. In order to solve these problems, we propose a smart cooperative platform for load balancing and security on SDN distributed controllers, named SCPLBS. The collaborative platform is built on the control plane. A secure communication mechanism based on message authentication code is adopted between the cooperative platform and the controllers. Collaborative platform uses a data collection algorithm adapting to data fluctuation to collect the controllers' status and load information. Collaborative platform takes strategy to achieve the distributed controllers load balancing and failure recovery. In this paper, we use the Floodlight controller and develop the cooperative platform based on restlet framework. We test the effectiveness of the proposed scheme. The experimental results show that this scheme can well achieve the load balancing and failure recovery of the distributed controllers on the basis of the secure communication between the cooperative platform and the controllers.

This article is part of the Topical Collection: *Special Issue on Software Defined Networking: Trends, Challenges and Prospective Smart Solutions*
Guest Editors: Ahmed E. Kamal, Liangxiu Han, Sohail Jabbar, and Liu Lu

✉ Jie Cui
cuijie@mail.ustc.edu.cn

[1] School of Computer Science and Technology, Anhui University, Hefei 230039, China

Keywords Software-defined networking · Cooperative platform · Load balancing · Distributed multiple controllers security · Data collection

1 Introduction

The Internet is developing at an unprecedented rate and it has been an important information infrastructure. The Internet brings great convenience to people's life and communication. However, the traditional network structure is complex and difficult to manage, leading to its limited development. It is difficult to adapt to the current cloud computing, cloud storage, virtualization and other network development applications [1]. SDN is a network architecture based on the idea of software programmable. It can enhance the automation control and management of the network flexibly through the centralized control plane and the distributed forwarding plane [2]. The controller is an important part of SDN and it is the bridge connecting the underlying switching equipment and the upper application. With the increasing scale of network and the increasing number of network interconnection devices [3], for extensive switches flow requests and control plane expansion [4], the single centralized controller has been unable to meet the performance requirements of large-scale networks [5, 6]. This facilitates the generation of the distributed multiple controllers architecture. The distributed controllers can cooperate to realize the expansion of the network and the management of the network devices more conveniently [7].

The distributed controllers architecture has been studied and implemented by researchers, such as HyperFlow [8] and Onix [9]. They focus their attention on how to implement a distributed control plane and how to provide a global network topology for the application plane. Although it improves the scalability and stability of the distributed controllers to a

certain extent, it does not consider the load balancing and failure recovery among the distributed controllers [10]. The overloaded controller should migrate the high-load switch smoothly to the low-load controller, and it is necessary to have the mechanism of failure recovery when a controller break down. Hu et al. [11] designed a "Super Controller" to control the flow and achieve the load balancing among the distributed controllers. Zhou et al. [12] designed a dynamic and adaptive algorithm (DALB). The algorithm is running as a module of controller and the distributed controllers coordinate with each other to maintain load balancing. Hai et al. [13] used predefined load threshold, besides a load balancing plane is added between the control plane and the data plane. Selvi et al. [14] designed a controller load-balancing scheme for hierarchical networks(COLBAS). The controllers periodically release their load and coordinate with each other to realize the load balancing. They focus their attention on how to implement load balancing of distributed controllers, but they do not take into account the security issues of the distributed controllers architecture [15]. Malicious controllers may access the network, obtain network topology, steal and modify network data [16]. This may result in network paralysis. In addition, the designs of these papers realize the load balancing among the controllers by changing the controller's role over the switch. They do not consider that the routing strategy of each controller is different and direct migration of the switch may cause the current business interruption. Also the algorithms in these designs are not accurate enough to collect the controllers' load. Periodical collection of controllers load may also lead to a waste of resources. And timely and effective controllers failure recovery strategy is an important aspect to be taken into account.

This paper proposes a smart cooperative platform for load balancing and security on SDN distributed controllers, named SCPLBS. We mainly focus on the disadvantages of the existing multiple controllers load balancing. Our scheme attempt to improve the multiple controllers load balancing in the security and accuracy. The collaborative platform is built on the control plane. A secure communication mechanism based on message authentication code is adopted between the cooperative platform and the controllers. Collaborative platform uses a data collection algorithm adapting to data fluctuation to collect the controllers' status and load information. Collaborative platform takes strategy to achieve the distributed controllers load balancing and failure recovery. This paper mainly includes the following three contributions:

- Secure communication mechanism based on message authentication code is used between SCPLBS and controllers. This can prevent the attacker from accessing the controllers to steal and modify the network data and improve the security of the mutual communication between the cooperative platform and the controllers.
- Collaborative platform uses a data collection algorithm adapting to data fluctuation to collect the controllers' status and load information. According to the fluctuation of data collected in the data collection interval, the cooperative platform can adaptively adjust the time interval of data collection.
- SCPLBS stores the routing policy of each controller. According to the controllers load information collected, SCPLBS maintains the load balancing of the distributed controllers by migrating the control permission of the switch. At the same time, SCPLBS can avoid the network paralysis caused by failure of single controller.

2 Related works

2.1 The SDN architecture

Software-Defined Networking(SDN) is a new type of software programmable network architecture, which is based on the separation of data and control [17]. SDN uses a centralized control plane and a distributed forwarding plane, the two planes are separated from each other [18]. SDN interface protocol open up the SDN programmability. These protocols achieve the connection and communication among various parts. The northbound interface protocol provides the interaction between the developer and the control plane. The southbound interface protocol accomplishes the interaction between the control plane and the data plane. The northbound interface protocol also accomplishes part of the management and configuration function. OpenFlow [19] is the standard that normalizes communication between the control plane and the forwarding plane under the SDN architecture. OpenFlow has become one of the mainstream southbound interface protocol. SDN controller is an important part of SDN. On the one hand, controller uses the southbound interface protocol to connect the underlying network switching equipment for centralized management and state monitoring. Also the controller generates forwarding decisions to process and schedule traffic in the data plane. On the other hand, the controller uses the northbound interface to open multiple levels of programmability for the upper application.

2.2 OpenFlow and multiple controllers

SDN southbound interface (OpenFlow protocol) provides a lot of message types. From the beginning of the OpenFlow1.2 protocol, a switch can be connected to the multiple controllers, which makes load balancing and failure recovery among the distributed controllers be possible. When a switch is connected to multiple controllers, the controller in MASTER role can fully manage the switch and receive all the

asynchronous messages sent from the switch(such as Packet-in message,Flow-Removed message). However, the controller in SLAVE role has read-only permissions to switch and does not receive asynchronous messages sent from the switch. When a controller changes its role to MASTER, the switch change all other controllers which role is MASTER to have the role SLAVE. This makes the switch determine the only controller.

When the southbound interface uses the OpenFlow protocol, the controller's primary load comes from the Packet-in message. As shown in Fig. 1, when the data packet received by the switch does not have a match in the flow table, the switch will encapsulate the Packet-in message and forward the packet to the controller. The controller analyzes the Packet-in message and sends the Packet-out or Flow-mod message to the switch. By this way, the data packet gets the forwarding path. When a controller receives a lot of Packet-in messages, the load of the controller will increase. When the load exceeds a certain threshold, the network will have problems, such as delay increasing, packets not reaching, etc.

2.3 Message authentication code

Message authentication [20] is a mechanism for verifying message integrity. Message authentication can ensure that the data received is exactly the same as when sent (not modify, insert, delete, reproduction). And message authentication can also ensure the sender's identity is real and valid. Before transmitting the data, the sender first uses the hash function negotiated by the two sides of the communication. Then the sender obtains the message authentication code under the shared session keys. And the message and the message authentication code are sent together. After receiving the message, the receiver uses the hash function to calculate the hash value corresponding to the message and compares the value with the message authentication code. If the two are equal, the message is authenticated.

3 Design scheme

3.1 System model

In this paper, the system model is shown in Fig. 2. SCPLBS is built on the control plane. SCPLBS contains Message Authentication module, Collect Data module, Load Balancing and Failure Recovery module. SCPLBS also contains a database for storing the load information and the routing strategy of the controllers. A secure communication mechanism based on message authentication code is adopted between SCPLBS and the controllers. SCPLBS uses a data collection algorithm adapting to data fluctuation to collect the controllers' status and load information. SCPLBS takes strategy to achieve the distributed controllers load balancing and failure recovery.

3.2 Message authentication

This design uses the identity authentication based on message authentication code to realize the communication security between the distributed controllers and SCPLBS. Message

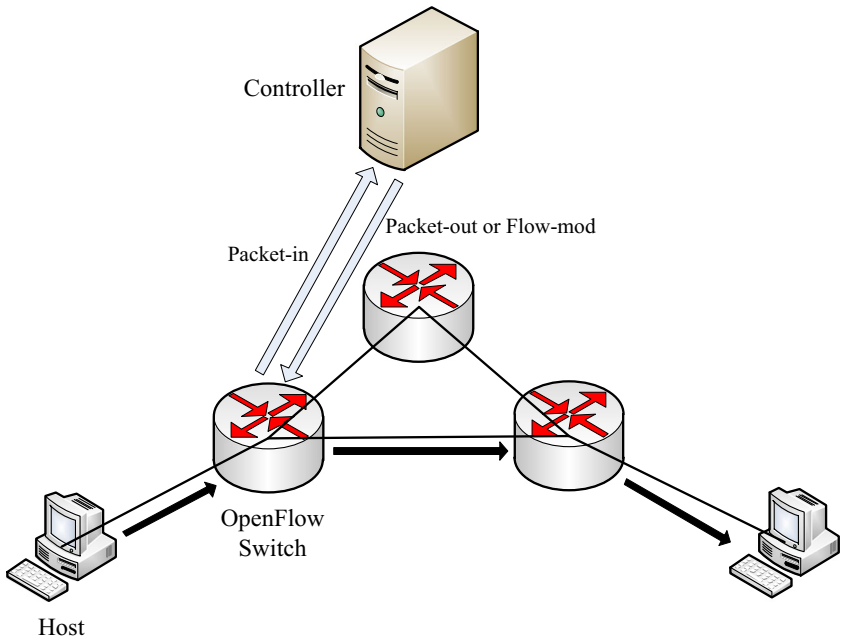


Fig. 1 Message handling

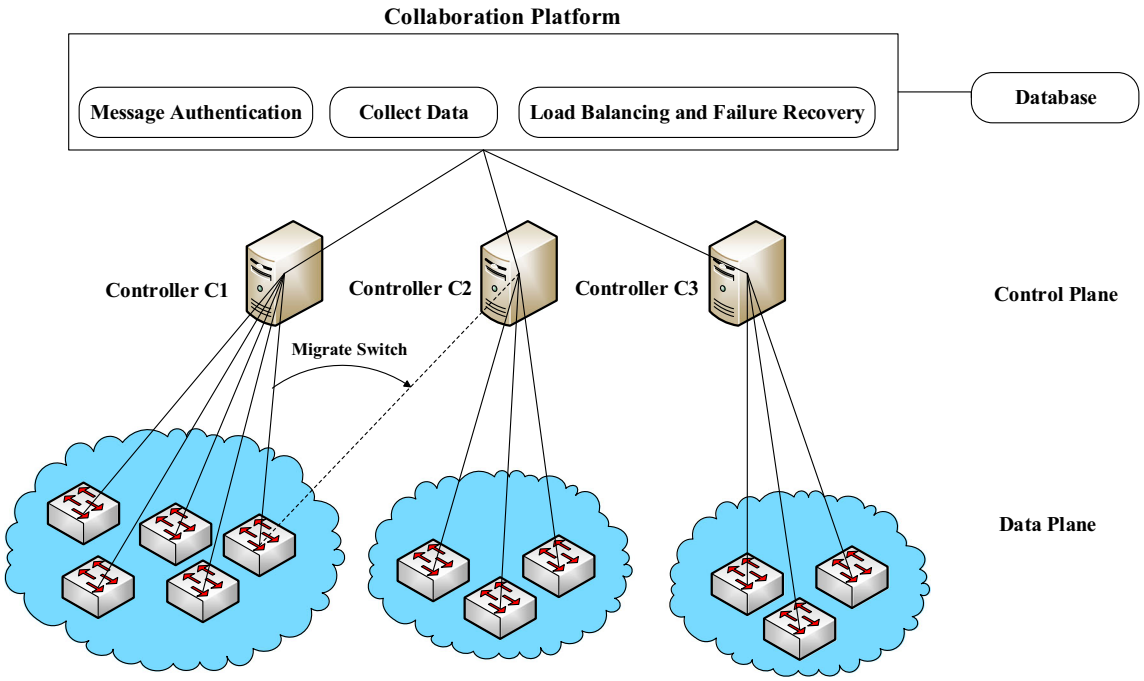


Fig. 2 System model

authentication code is a kind of authentication technology. This technology uses a key to generate a fixed length of short data blocks. And the data block will attach to the message. The design of the message authentication is shown in Fig. 3.

Step 1: Before sending the message, each controller and SCPLBS negotiate the common hash function and key value.
Step 2: When the controller sends a message to the SCPLBS, the controller uses the hash function and the key to calculate the hash value (MAC) of the message. Then the controller sends the message along with the MAC to SCPLBS.
Step 3: When the SCPLBS receives the message and MAC, SCPLBS uses the negotiated hash function and key to calculate the hash value corresponding to the message. Then SCPLBS compares this local calculated hash value with the received MAC. If the two are equal, the message is authenticated.

SCPLBS can believe that the message has not been modified. Because if the attacker changed the message, but he could not change the corresponding MAC. So the MAC calculated by SCPLBS will not equal to received MAC. SCPLBS can believe that the message is from the true sender controller, because all other parties do not know the key and fake party could not produce the correct MAC. If the message contains a serial number, SCPLBS can believe that the message sequence is correct. Because the attacker could not successfully modify the serial number. When the SCPLBS sends the message to the controller, SCPLBS uses the same message authentication mechanism. This can ensure the communication security between SCPLBS and the controller.

3.3 Data collection algorithm

SCPLBS uses the data collection algorithm adapting to data fluctuation to collect the controllers' status and load information. Some of the notations used in the data collection algorithm are provided in Table 1.

As shown in the Algorithm 1, SCPLBS collects the average rate of Packet-in messages received by each controller in the data collection interval as the current load of the controllers. In order to measure the load of each switch to the controller,

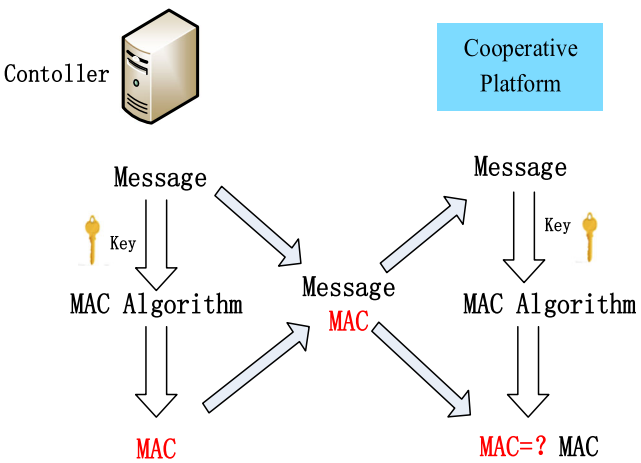


Fig. 3 Message authentication

SCPLBS also collects the number of Packet-in messages produced by switches connected to the controller in the data collection interval. SCPLBS stores the load information in the database. SCPLBS updates the data in the database according to the time interval of data collection. SCPLBS updates the data every time when the load is collected.

As shown in Fig. 4, SCPLBS adaptively adjusts the time interval of data collection according to the load change of each controller in a time interval. When the load change of each controller is gentle, we can increase the time interval of data collection. When the load change of each controller is drastic, we can reduce the time interval of data collection. This can be more accurate to collect each controller's real-time load. And this can also avoid frequent collection of data and the waste of resources.

As shown in the Algorithm 1, when the time interval of data collection is (T, T + t), SCPLBS collects num_i in t interval time. SCPLBS calculates $CP_{ti} = num_i/t$ as the average rate of Packet-in messages received by each controller in t interval time. And SCPLBS uses (CP_{t1}, CP_{t2},…,CP_{tn}) as the average load of each controller in the t interval. Assume that the load of each controller in the previous data collection interval is (CP_{T1},CP_{T2},…,CP_{Tn}),SCPLBS calculates the $F = (|CP_{t1}-CP_{T1}| + |CP_{t2}-CP_{T2}| + \ldots. + |CP_{tn}-CP_{Tn}|)/n$ as the average value of controller load fluctuation. When F is large, the time interval of data collection needs to be reduced. When F is small, the time interval of data collection needs to be increased. We calculate the F/w to measure the size of the data fluctuations. The w represents the arrival rate of Packet-in messages and the controller uses w to control flows. The time interval of data collection algorithm adapting to data fluctuation is adjusted according to the following rules:

(1) When $0 < F/w \leq 0.2$,set the time interval of load collection to the maximum data collection interval of 20s.
(2) When $0.2 < F/w \leq 0.6$,set the time interval of load collection to 15 s.
(3) When $0.6 < F/w \leq 1$,set the time interval of load collection to 10s.
(4) When $F/w > 1$, set the time interval of load collection to the minimum data collection interval of 5 s.

Algorithm 1

Input:
$\{CP_{t1},CP_{t2,}\ldots,CP_{tn}\}$
$\{CP_{T1},CP_{T2},\ldots,CP_{Tn}\}$
Output:
Dt

1: $\{ CP_{T1},CP_{T2},\ldots\ldots,CP_{Tn} \}$ =LoadCollect(T)
2: $\{ CP_{t1},CP_{t2,}\ldots..,CP_{tn} \}$= LoadCollect(t)
3: $F=(|CP_{t1}-CP_{T1}|+|CP_{t2}-CP_{T2}|+\ldots.+ |CP_{tn}-CP_{Tn}|)/n$
4: switch ( )
{
case 0< F/w≤0.2: return 20;
case 0.2< F/w≤0.6 : return 15;
case 0.6< F/w≤1 : return 10;
case F/w >1: return 5;
}

3.4 Load balancing and failure recovery

SCPLBS periodically reads the load information of each controller from the database. SCPLBS achieves the distributed controllers load balancing and failure recovery by migrating the control permission of the switch.

The process of SCPLBS achieving load balancing is shown in Algorithm 2. And some of the notations used in the load balancing algorithm are provided in Table 2.

SCPLBS calculates the ratio m between the average load of each controller and the maximum load of each controller. The m is obviously less than 1. When m is in the floating range we

set, it indicates that the load of each controller is relatively balanced. When m is not in the floating range we set, SCPLBS needs to execute load balancing algorithm. The specific steps are shown as follows:

Step 1: SCPLBS set the load balancing factor LB and the load floating factor FL, the load balancing floating interval is set to (LB-FL, LB + FL). The initial LB and FL is set to 0.8 and 0.2. When the load balancing accuracy requirements are relatively high, we can adjust the value of the parameters.
Step 2: SCPLBS periodically reads each controller load information from the database. SCPLBS calculates the average value of each controller load as AVG_CP. SCPLBS selects the MAX_CP and MIN_CP.Then SCPLBS calculates the m = AVG_CP/MAX_CP;
Step 3: When the m is not in the (LB-FL, LB + FL), SCPLBS executes the load balancing strategy in Step 4.
Step 4: When SCPLBS needs to execute load balancing strategy, SCPLBS elect out the switch producing the most Packet-in message from the maximum load controller. That is to find the maximum value in $\{SP_1,SP_2,\ldots,SP_j,\ldots\}$ connected to the C_MAX. By modifying the role of the controller, SCPLBS migrates the maximum load switch to the minimum load controller C_MIN.

Algorithm 2

Input:
$\{CP_1,CP_2,\ldots\ldots CP_n\}$
LB
FL
C_MAX: Controller with maximum load
C_MIN: Controller with minimum load
Output:
0: no need to execute load balancing algorithm
1: execute load balancing algorithm

1: $\{CP_1,CP_2,\ldots\ldots CP_n\}$ =ReadDatabase ( )
2: AVG_CP=Avg($\{CP_1,CP_2,\ldots\ldots CP_n\}$)
3: MAX_CP=Max($\{CP_1,CP_2,\ldots\ldots CP_n\}$)
4: MIN_CP=Min($\{CP_1,CP_2,\ldots\ldots CP_n\}$)
5: **If** m=AVG_CP/MAX_CP in interval [LB-FL,LB+FL] **then**
6: return 0
7: **else**
8: Migrate the maximum load switch of C_MAX to C_MIN **then**
9: return 1
10: **end if**

Here we need to pay attention to a problem. After executing the load balancing algorithm, it is necessary to update the load data collection interval Dt to the minimum data collection interval. This ensures that relatively accurate load is stored in SCPLBS database. This can also ensure that the next round of SCPLBS determining whether the current the load of controllers is balancing more accurate.

SCPLBS continuously monitors the status of each controller in the whole network by sending heartbeat packets to the controllers. When SCPLBS finds a controller failure, it will timely execute the failure recovery strategy. In this scheme, the failure recovery strategy is to migrate the switches of the failure controller to the normal operational controllers in hash mode. The hash function is calculated based on the ID number of the switch and the number of controllers. Hash function can be set to modulo operation. The switch will set the other normal operational controllers as its own Master controller. For example, assuming that there are controller C1, controller C2, controller C3 currently. Controller C1 manages 2 switches. When controller C1 fails, SCPLBS finds the heartbeat packet connection disconnecting

Table 1 Some of the notation used in the data collection algorithm

Notation	Definition
n	Total number of controllers
T	Initial time point for load collection
t	Time interval of data collection
num_i	The number of Packet-in messages received by each controller(i = 1,2…,n)
CP_{ti}	The average rate of Packet-in messages received by each controller in t interval time (i = 1,2…,n)
CP_{Ti}	The average rate of Packet-in messages received by each controller in the previous data collection interval (i = 1,2…,n)
F	Average data fluctuation
w	The Packet-in messages arrival rate used by the controller to control the flows
Dt	Data collection time interval

from controller C1. According to the 2 switches ID number S1 and S2 managed by controller C1,and the number of normal work of the controller is 2,SCPLBS calculates hash(S1) = 1mod2 = 1 and hash(S2) = 2mod2 = 0. SCPLBS can migrate switch S1 and switch S2 to the remaining controller C2 and controller C3. In this way, SCPLBS can realize failure recovery of the controllers. When the failure controller is repaired, it can continue to access the network.

The database of SCPLBS not only stores the load of the controllers and the switches, but also stores the routing strategy of each controller. When SCPLBS executes load balancing and failure recovery strategy, there will be some switches migrated to other controllers. In this scheme, the region or business of the switch is marked at the beginning. When a switch is migrated to the new controller, the new controller will read the routing policy of the initial controller of the switch from the database. And the new controller executes route forwarding based on this routing policy. This can reduce the business interruption and the flow table conflict caused by the migrated switch.

4 Experiment result and performance analysis

4.1 Experiment deployment

We uses open source controller Floodlight [21] and Mininet [22] to deploy our experiment. Floodlight is an open source SDN controller based on Java language and it supports OpenFlow protocol. The overall architecture of the Floodlight consists of the core controller functions and its application. The application and the controller can interact via Java interface or REST API. Mininet is a network simulation tool which can be used to build large scale SDN prototype system on the limited resources. The system consists of virtual terminal node (End-Host) and OpenFlow switch. This makes it possible to simulate the real network and provide development verification for a variety of ideas. In our scheme, we use the Floodlight controller and develop the cooperative platform based on restlet [23] framework. SCPLBS mainly contains Message Authentication module, Collect Data module, Load Balancing and Failure Recovery module, a database system. We use MySQL as the database. This scheme is based on the Floodlight controller for the experiment and test. We will set the parameter w in data collection algorithm to 1000 [12]. Experiment network topology is shown in Fig. 5.This experiment creates controller C1, controller C2, controller C3 and 12 switches. Each switch connects two hosts. The Floodlight controller and SCPLBS operate at Windows 7, i5 CPU and 8G RAM. Mininet operate at Ubuntu14.04, i5 CPU and 4G RAM.

4.2 The experiment and analysis of load balancing and failure recovery strategy

Controller C1, controller C2, and controller C3 are connected to SCPLBS at the beginning. Controller C1 controls 8

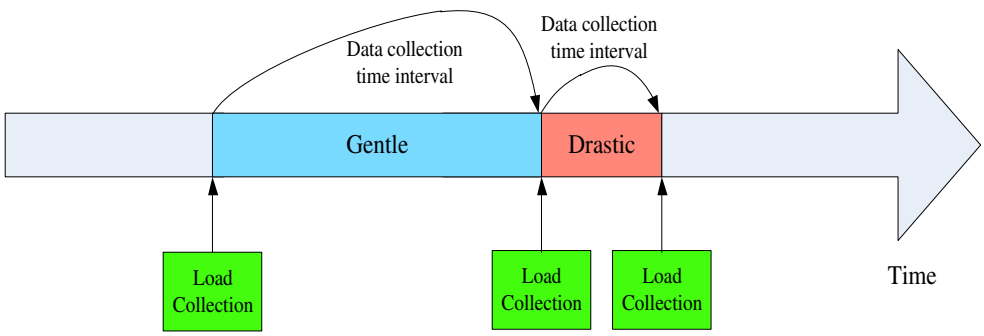


Fig. 4 Load collection

Table 2 Some of the notation used in the load balancing algorithm

Notation	Definition
n	Total number of controllers
CP_i	The load of all controllers(i = 1,2…,n)
LB	Load balancing threshold
FL	Floating parameter
AVG_CP	The average value of each controller load
MAX_CP	The maximum value of each controller load
MIN_CP	The minimum value of each controller load
m	The ratio m between the AVG_CP and MAX_CP
SP_j	The number of Packet-in messages produced by switches connected to the controller

switches in the role of Master. Controller C2 and controller C3 control 2 switches in the role of Master. We use these 12 switches to simulate a domain network. We let the host do Ping operation between each other in Mininet persistently. This leads to lots of ICMP protocol traffic in the network, so that the switches produce Packet-in messages and send Packet-in messages to the controller. In this experiment, SCPLBS reads the load information of each controller from the database every 10s, as to obtain average rate of Packet-in messages received by the three controllers. As shown in Fig. 6, at the beginning, controller C1 is responsible for controlling the 8 switches and there is no flow table in the switches. The controller C1 receives relatively many Packet-in messages. So the load of controller C1 is relatively high and the load of controller C2, controller C3 is relatively low. With the implementation of SCPLBS load balancing strategy, the controller C1 gradually migrates the switches to controller C2 and controller C3. The load of controller C1 decreases. The load of controller C2 and controller C3 increases. And with the passage of time, the load of controller C1, controller C2, controller C3 is gradually balanced. Fig. 6 shows that the data collection interval will also become longer after 50s, due to the load change of each controller is gentle.

In the experiment, we measure the controllers' memory utilization before and after load balancing. As shown in Fig. 7, the load of controllers is not balanced in the initial state, the memory utilization of the controller C1 is relatively high. With the implementation of load balancing algorithm, when the load of the controllers is balanced, the memory utilization of the controllers is relatively similar. This can also avoid a controller maintaining a high load of work for a long time.

When the load of controller C1, controller C2, controller C3 is relatively balanced, we further measure the effect of SCPLBS on controller CPU utilization. We let the host do Ping operation between each other in Mininet randomly. This is to inject random traffic to the switches. We analyze the CPU utilization of each controller. After 100 s, we disconnect the controllers and SCPLBS. We continue to do Ping operation samely and analyze the CPU utilization of each controller. Figs. 8, 9 and 10 show the CPU utilization comparison of the controller C1,controller C2,controller C3 before and after disconnecting from SCPLBS. It can be seen that when the controllers are connected to SCPLBS, the controllers have less fluctuation on the CPU utilization. The experimental result shows that when the controller is connected to the SCPLBS we designed, the controllers can maintain a relatively stable CPU utilization under the load balancing strategy.

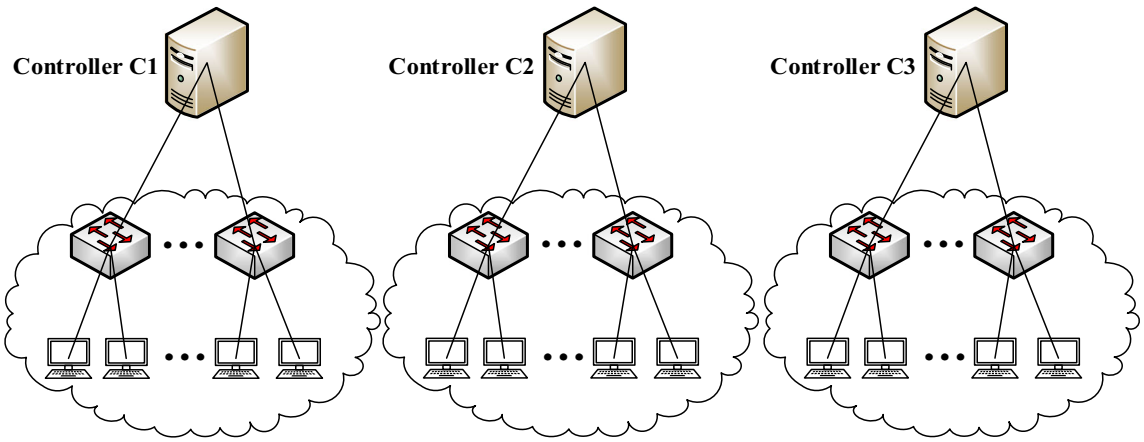


Fig. 5 Experiment network topology

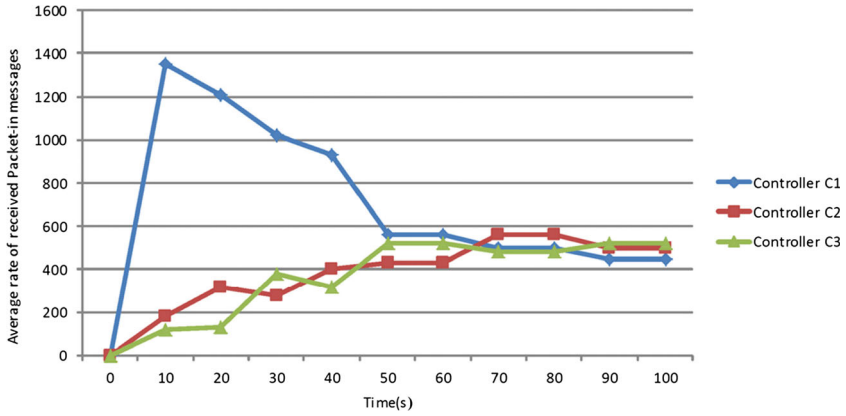


Fig. 6 Controller load variation

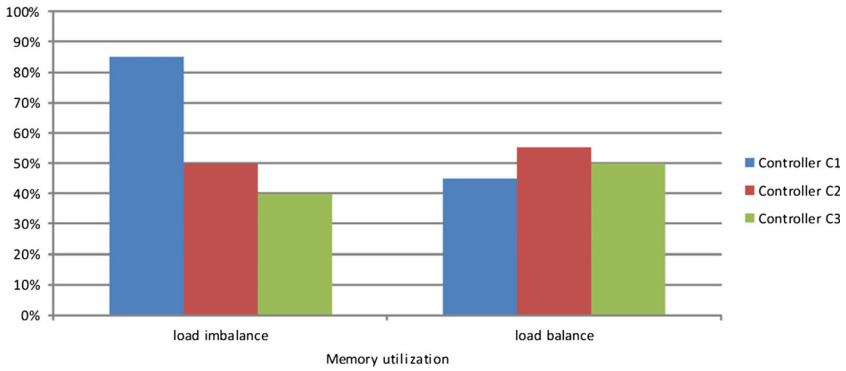


Fig. 7 Controller memory utilization

When the controller C1, controller C2, controller C3 are in normal operation, we choose to turn off the controller C3. This is an analog that controller C3 has failure. At this point, we continue do pingall operations in Mininet. In the experiment, we can see that each host can still communicate with each other, and there are still new flow tables in the switches connected to the C3 controller that has failure before. The experiments show that the controller C1 and controller C2 can take over the switches of controller C3 that before the failure occurs. In this way, we can avoid the network failure caused by single controller failure. This verifies that SCPLBS can basically achieve the failure recovery of the controllers.

4.3 The analysis of security policy

Compared with the existing distributed controller architecture, the security authentication module is added in this experiment. As shown in Fig. 11, the key that the controllers and SCPLBS negotiating is set to 0xFFFF in the experiment. The hash function is set that mapping the message string to an integer number. Here we use the hashCode method in Java and get the hashCode value corresponding to the message. The hashCode value XOR with the key 0xFFFF to produce the message authentication code. For example, in this design,

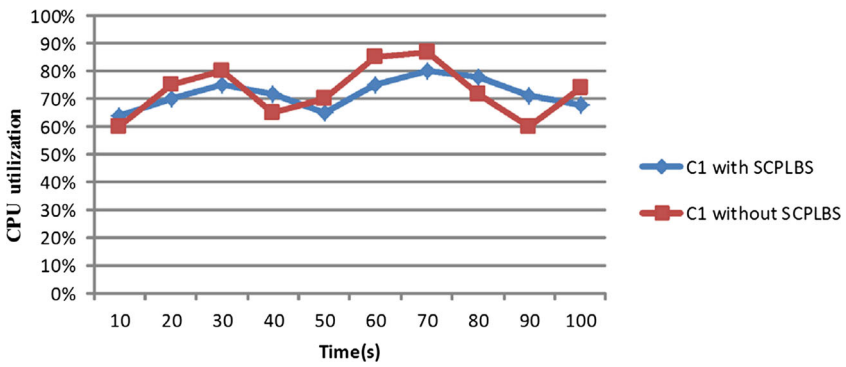


Fig. 8 Controller C1 CPU Utilization

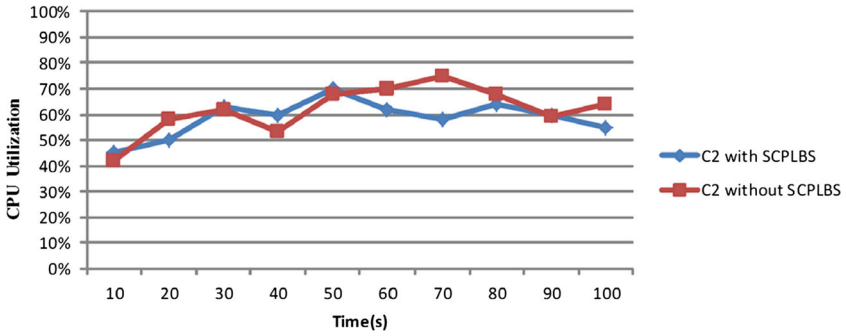


Fig. 9 Controller C2 CPU Utilization

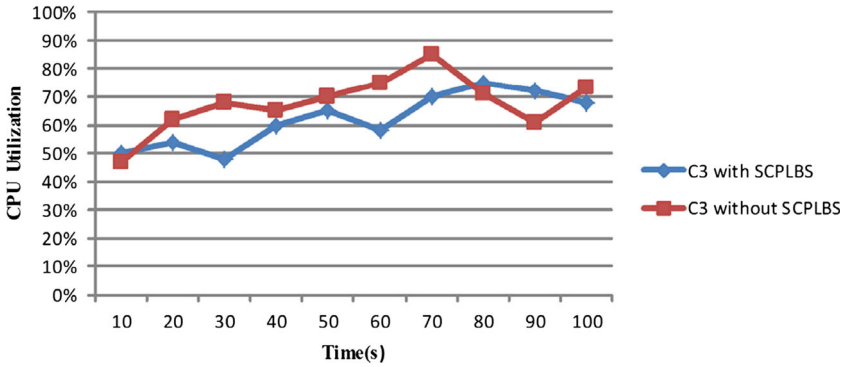


Fig. 10 Controller C3 CPU Utilization

when the SCPLBS requests to connect the controller, the "requestconnect" message sent to the controller by SCPLBS will be encrypted as "requestconnect577b". When the controller receives the message "requestconnect577b", the controller calculates the MAC by using the hash function and the message "requestconnect". The controller gets "577b" and it is as same as the received message authentication code. This means that the success of message authentication. As shown in Fig. 11, SCPLBS can be seen as Bob, the controller can be seen as Alice. If the attacker Tom changed the message, but he could not change the corresponding MAC, so the MAC calculated by the receiver Alice will not equal to the MAC receiving. The recipient Alice can believe that the message is from Bob, because Tom does not know the key and cannot produce a message with the correct MAC. In this way, the communication security between SCPLBS and control plane can be realized.

5 Conclusions and future work

This paper proposes a smart cooperative platform for load balancing and security on SDN distributed controllers, named SCPLBS. Our scheme is applicable to SDN distributed controllers architecture for a single domain network. We use the Floodlight controller to develop the cooperative platform based on restlet framework. SCPLBS contains message authentication module, data collection module, load balancing

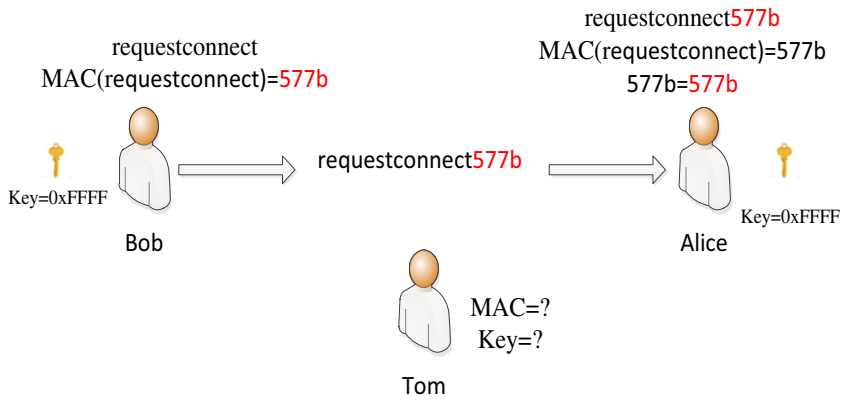


Fig. 11 Message authentication example

and failure recovery module. The experiment and analysis show that the platform can realize the load balancing and failure recovery of the distributed controller based on the secure communication. Our scheme still has some limitations. In the process of changing the controller's role over the switch, it may lead to a brief delay of the key business. In the future, we will improve the security encryption mechanism in this scheme to provide more effective trust management. And we will attempt to solve the problem of key business delay by setting the weight to the switch. Furthermore, we will do more in-depth study on the issues about security and scalability of multi-domain distributed controllers.

Acknowledgments The work is supported by the National Natural Science Foundation of China (No. 61572001, No.61502008), the Research Fund for the Doctoral Program of Higher Education(No. 20133401110004), and the Doctoral Research Start-up Funds Project of Anhui University. The authors are very grateful to the anonymous referees for their detailed comments and suggestions regarding this paper.

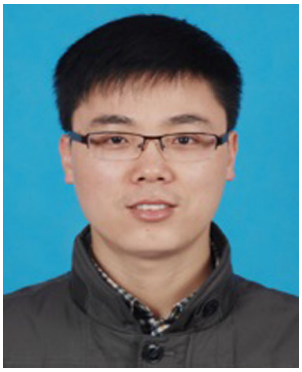
References

1. Wood T, Ramakrishnan KK, Hwang J et al (2015) Toward a software-based network: integrating software defined networking and network function virtualization[J]. IEEE Netw 29(3):36–41
2. Tuncer D, Charalambides M, Clayman S et al (2015) Adaptive resource management and control in software defined networks[J]. IEEE Trans Netw Serv Manag 12(1):18–33
3. Agarwal S, Kodialam M, Lakshman TV (2013) Traffic engineering in software defined networks[J]. Proceedings - IEEE INFOCOM 12(11):2211–2219
4. Yassine A, Rahimi H, Shirmohammadi S (2015) Software defined network traffic measurement: current trends and challenges[J]. IEEE Instrum Meas Mag 18(2):42–50
5. Kim H, Feamster N (2013) Improving network management with software defined networking[J]. IEEE Commun Mag 51(2):114–119
6. Voellmy A, Wang J (2012) Scalable software defined network controllers[J]. Acm Sigcomm Comput Commun Rev 42(4):289–290
7. Yu Y, Lin Y, Zhang J et al (2014) Field demonstration of datacenter resource migration via multi-domain software defined transport networks with multi-controller collaboration[C]// optical fiber communications conference and exhibition. IEEE 2014:1–3
8. Tootoonchian A, Ganjali Y (2010) HyperFlow: a distributed control plane for OpenFlow[C]// internet network management conference on research on enterprise NETWORKING. USENIX Association 2010:3–3
9. Koponen T, Casado M, Gude N et al (2010) Onix: a distributed control platform for large-scale production networks[C]//OSDI 10:1-6
10. Dan M, Lei L, Yoo SJB Optical FlowBroker: load-balancing in software-defined multi-domain optical networks[C]// optical fiber communication conference 2014:W2A.44
11. Hu Y, Wang W, Gong X et al (2012) BalanceFlow: controller load balancing for OpenFlow networks[C]// IEEE, International Conference on Cloud Computing and Intelligence Systems 780–785
12. Zhou Y, Zhu M, Xiao L et al (2014) A load balancing strategy of SDN controller based on distributed decision[C]// IEEE, international conference on trust, security and privacy in computing and communications. IEEE Comput Soc 851–856
13. Hai NT, Kim DS (2016) Efficient load balancing for multi-controller in SDN-based mission-critical networks[C]// IEEE, International conference on industrial informatics. IEEE
14. Selvi H, Gür G, Alagöz F (2016) Cooperative load balancing for hierarchical SDN controllers[C]// IEEE, international conference on high PERFORMANCE switching and routing. IEEE 100–105
15. Porras P, Cheung S, Fong M et al (2015) Securing the software defined network control layer[C]// network and distributed system security symposium
16. Chua RL, Pearce AK, Palmer M (2015) Authentication for software defined networks:, US9038151[P]
17. Feamster N, Rexford J, Zegura E (2014) The road to SDN: an intellectual history of programmable networks[J]. Acm Sigcomm Comput Commun Rev 44(2):87–98
18. Dave T (2014) OpenFlow: enabling innovation in campus networks[J]. Acm Sigcomm Comput Commun Rev 38(2):69–74
19. Mckeown N, Anderson T, Balakrishnan H et al (2008) OpenFlow: enabling innovation in campus networks[J]. Acm Sigcomm Comput Commun Rev 38(2):69–74
20. Priyadharshini MMD, Ananth C (2015) A secure hash message authentication code to avoid certificate revocation list checking in vehicular Adhoc networks[J]. Int J Appl Eng Res 10(2):1250–1254
21. Floodlight. http://www.projectfloodlight.org/floodlight[M]
22. Mininet. http://mininet.org/[M]
23. Louvel J, Templier T, Boileau T (2013) Restlet in action : developing RESTful web APIs in java[J]. Meap Began

Hong Zhong is a Professor (from 2009) and the Dean of the School of Computer Science and Technology, Anhui University, China. She received PhD degree in University of Science and Technology of China in 2005. She has published over 100 papers. Her current research interests include applied cryptography, IoT security, vehicular ad hoc network, and software-defined networking (SDN).

Jianqiao Sheng is now a research student in the School of Computer Science and Technology, Anhui University. His research interests include Software-Defined Networking.

Yan Xu is a Lecture in the School of Computer Science and Technology, Anhui University, China. She received PhD degree in University of Science and Technology of China in 2015. Her research interests cover network and information security.



Jie Cui received his Ph. D. degree in Computer Science and Technology from University of Science and Technology, China in 2012. He is currently an Associate Professor with the School of Computer Science and Technology, Anhui University, China. He has published over 50 papers. His current research interests include applied cryptography, IoT security, vehicular ad hoc network, and software-defined networking (SDN).