Master Controller Election Mechanism Based on Controller Cluster in Software Defined Optical Networks

Jie Mi[1(✉)], Xiaosong Yu[1], Yajie Li[1], Yongli Zhao[1], Jie Zhang[1], Chuan Liu[2], and Gang Zhang[2]

[1] State Key Laboratory of Information Photonics and Optical Communications, Beijing University of Posts and Telecommunications, Beijing, China
{mijie, xiaosongyu, yajieli, yonglizhao, lgr24}@bupt.edu.cn
[2] Global Energy Interconnection Research Institute, Beijing, China

Abstract. In large-scale software defined optical networks (SDON) with tens of thousands of network elements, multiple controllers have to be deployed simultaneously, because single controller cannot bear too many service requests. Then, survivability of controller becomes an important issue for SDON. Controller cluster deployed with master and slave controllers is considered as an effective solution for this issue. A SDON controller cluster architecture is given in the paper, based on which a master controller election mechanism (MCEM) is proposed. Simulation results show that MCEM can get better performance in terms of operation time, traffic loads and fault tolerance compared with Paxos algorithm.

Keywords: SDON · Survivability · Controller election · Master/slaves mode

1 Introduction

With the growth of cloud computing, mobile Internet, Internet of things, as well as the sharp increase of the network traffic, network complexity and the continuous emergence of new business, the broadband business represented by video and massive data aggregation model represented by large data centers all drive the development of communication network. However, legacy optical network with poor flexibility and intelligence in network operation which can not adapt to the present network environment and needs revolution [1]. In traditional networking, the routing control and packets forwarding services are on the same hardware, this unified network structure makes communication mechanism becomes sophisticated. In order to change the embarrassment of such relatively closed and cumbersome network architecture, the concept of intelligent and open software defined network (SDN) is quickly introduced. In software defied networking, the most significant improvement is that the control function is separated from the hardware, it allows control plane to perform more flexible control function over the whole software defined network, and the data plane is merely forwarding packets conducted by control plane. Since legacy optical network

Q. Chen et al. (Eds.): ChinaCom 2016, Part I, LNICST 209, pp. 148–155, 2018.
DOI: 10.1007/978-3-319-66625-9_15

with rigid, non-programming transport capacity blocks the service innovation, vendors and service providers combines SDN with latest optical transport technologies to provide more responsive and flexible optical networks which produces software defined optical networks (SDON). In SDON, it allows network operators to program the optical layer to a set of shared, common resources, that can be used on demand.

In SDON, especially the large-scale SDON architecture, the control layer is the kernel part of the whole system, which is very important for achieving a reliable network. However, a single-controller SDON system has a limited processing capacity to forward routing decisions for each new flow in the data path, therefore, a single-controller structure would easily become a bottleneck for network extending. Some researchers consider that a controller is unable to handle plenty of new flows with OpenFlow protocol in 10 Gbps speed networks is a big problem [2]. At present, the speed of most optical transport networks reaches over 100 Gbps, so introduction of multiple controllers to form a controller cluster is a reasonable method to solve this problem for SDON. HyperFlow is a first logically centralized but physically distributed control plane for OpenFlow, it is based on Nox and implements event-based services [3]. FlowVisor is another similar control plane comprises multiple controllers which slices network resources and delegates the control of each slice to a single controller [4]. Kandoo is a hierarchy-based control plane contains two types of controllers: root controller and local controllers. Root controller is responsible for non-local services and global network management such as harmonize local distributed information, and local controllers make routing decision for their own network devices [5]. Master and slave controllers is a distributed control system with high reliability, scalability and good performance.

In this paper, we study on the master controller survivability issue in master and slave controllers. As a distributed multiple-controllers structure, the master controller election is a complex issue and requires corresponding distributed consensus algorithm to complete the election procedure. It is proposed a master controller election algorithm named Master Election and a customized Master Controller Election Mechanism (MCEM). The preparation work before starting an algorithm is detailed introduced, then we describe the inputs, main body of the algorithm and final outputs. In the end, we conduct a stand-alone simulation, in which compares Paxos algorithm with proposed Master Election algorithm in terms of operation time, traffic loads and fault tolerance to verify that Master Election algorithm can solve the problem of master controller election in SDON Master/Slaves system, meanwhile, guarantees high efficiency, reliability and fault tolerance for election mechanism.

2 Master/Slaves Mode and Controller Election

In Master and slaves structure, shown as below in Fig. 1, JGroups is applied for communications between the controllers, the node management and data cache in cluster are achieved by IGMP and Hazelcast technologies respectively. The master controller in cluster is responsible for maintenance and update controller-switch device mapping function in the global network scope. [6] In the operation process of master and slaves system, there are three abnormal scenarios: the system initialization, the

slave controller failure and the master controller failure. To solve these problems, we propose corresponding solutions. When a slave controller is invalid, the failed node should be removed, then the master node distributes the traffic load to the rest slaves to balance network load. When the system initiates or the master node is disabled, it should start a master controller election mechanism in the system to find a new master controller to guarantee the system works properly, in addition, the entire failover process is transparent to all switches in the bottom data layer.

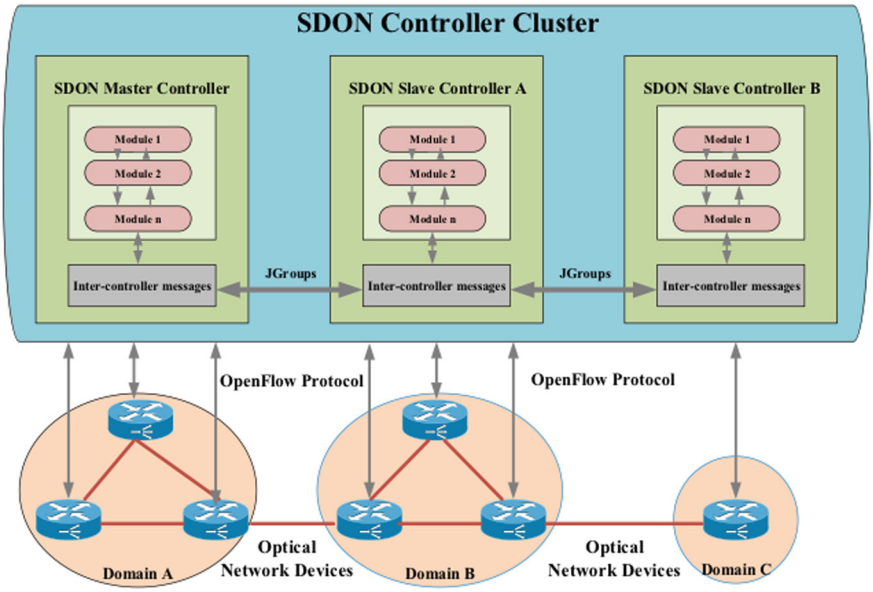

Fig. 1. A master/slaves structure.

Controller election is a complex issue in Master and slaves structure because it is a distributed structure which need introduce a message consistent distributed algorithm into the issue. One of the most classical consistent distributed algorithms is Paxos algorithm, which is the foundation of other distributed algorithms [7]. However, Paxos has three weaknesses for the master controller election. First, livelock problem means there are multiple Proposers initiate proposals to the same Acceptor would result in vicious circle of system resource occupation [8]. Second, each controller in Paxos (a Proposer) recommends itself to acceptors without reviewed, which contributes to heavy traffic loads. Last, Learners must learn the value of the determined proposal in the end, therefore, it would cause large bandwidth resources wasting [9]. We propose a master controller election algorithm named Master Election to solve the controller election issue in master/slaves mode of SDON.

3 Design and Implementation of MCEM

MCEM is a distributed master controller election mechanism which chooses a master controller from qualified slave controllers. Master Election algorithm is the core technology in this mechanism, it contains two main procedures and two roles for each of controller. Before implementing Master Election algorithm, it should carry some judgment procedures for preparation works. The whole process of MCEM is shown in Fig. 2 below.

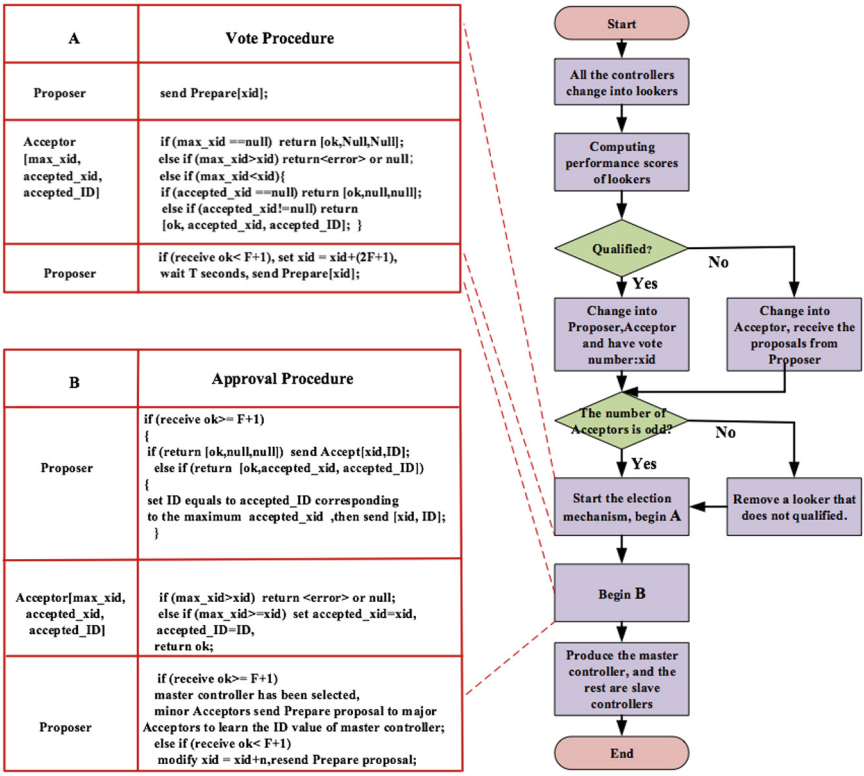

Fig. 2. Master Election mechanism process. (*max_xid is the maximum xid received by an Accetor, accepted_xid is the approved xid, accepted_ID is the approved controller ID*)

3.1 Election Qualifications for MCEM

In this algorithm, there are three types of roles of controllers: Master, Slave and Candidate. When the master and slaves system initiates, all the controllers change into Candidates. When the master controller fails, the rest are Slaves. Before start election, all the Slaves should be reviewed whether they meet the MCEM standard or not. The controller which qualifies for standard becomes a Candidate, otherwise, a Slave.

MCEM sets the standard for Candidates: the performance scores of qualified controllers must be in [x, 1) (x is the threshold of the score). In this paper, we propose a performance calculating formula shown as follows.

$$P = S + \alpha D + \beta E \tag{1}$$

According to this formula, P represents the performance score of a controller. S, D, E represent stability factor, load factor and the efficiency factor respectively. S is the rate that a controller keeping normal, if a controller has higher frequency of disable, of which value of E is less. Load factor presents the load capacity of one controller, if a controller has heavy load now, the score of D is low. Efficiency factor presents the speed of controller dealing with requests. S, D, E are all relative values comparing with a base score. α,β are coefficients of D and E. The range of P is [0, 1], 0 indicates controller failure. x is threshold that only those Slaves of which value of P is greater than x qualified for election.

The number of the controllers must be an odd number (2F + 1). In Master Election algorithm, only more than a half of acceptors agree with the proposal, the proposal can be approved. If the number of controllers is an even number, a controller which does not qualified for election is removed randomly.

3.2 Master Election Algorithm of MCEM

Master Election algorithm is mainly divided into two procedures: Vote procedure and Approval procedure. Each procedure is then subdivided into some sub stages respectively. The Candidate has Proposer and Acceptor two roles, a controller which is not qualified only has one role, that is, Acceptor. The procedures of proposed algorithm is as follows.

3.2.1 Vote Procedure

Step1: Every qualified controller gets a vote number as *xid*. A *Proposer* sends *xid* to the most of *Acceptors*

Step2: An *Acceptor* has three variables (*max_xid, accepted_xid, accepted_ID*) to store maximum received xid, approved xid and approved proposed controller ID. Comparing the value of request *xid* with *max_xid*, there are three different scenarios. The detailed description is displayed in Fig. 2, A phase.

Step3: If *Proposer* does not receive ok messages from most of the *Acceptors*, *xid* adds *(2F + 1)* ((*2F + 1*) is the number of controllers), then waits *T* (random waiting time) milliseconds, repeat Step 1.

3.2.2 Approval Procedure

Step4: If *Proposer* receives *ok* messages from the most of *Acceptors,* then sends messages to all Acceptors that reply to its vote requests. Message contents are vary according to the received *ok* messages in step 2. The detailed description is shown in Fig. 2, B phase.

Step5: *Acceptors* reply messages to the *Proposer*, the process is described in Fig. 2, B phrase.

Step6: If a *Proposer* receives *ok* messages over *a half of* Accepters' number, the proposal raised by this Proposer is approved, a master controller has been elected. Otherwise, repeat Step 3.

4 Simulation Results

We use Core2 Duo CPU PC, 2.93 GHz, 4 GB memory of PC and Linux Ubuntu14 operating system in this experimental environment. This simulation experiment chooses C language as the main programming language to achieve the classic Paxos algorithm and Master Election algorithm, and uses multiple terminal windows to simulate the role of multiple controllers. There are some Proposers and 5 Acceptors. These following technical appraisals are usually chosen for performance evaluation in distributed consensus algorithms.

4.1 The Evaluation Indicators in Experiment

Running time reflects speed of consistency algorithm to achieve the agreement. In this experiment, we set $\alpha = 0.6$, $\beta = 0.5$, $x = 0.6$, that is, only those controllers of which P is in [0.6, 1) qualified for election. After Calculating, there are 2 qualified out of 3 controllers group, and 3 qualified out of 5 controllers group.

Traffic load reflects the internal system communication load, the more number of messages, the heavier communication load, and the lower work efficiency it has. We take communication message number as indicator of traffic load.

Fault tolerance means whether a controller failure or abnormal data transmission condition would cause drastic effect on the system. In this experiment, we use 3 and 5 terminals to carry out simulation experiments for 10 times. In 3 terminals system, we suspend a terminal after the 5^{th} time experiment. In 5 terminals system, we choose a machine incapable after the 3^{rd} time and the 7^{th} time operation respectively.

4.2 The Analysis of Numerical Results

Figure 3 illustrates the value of running time of 3 terminals operating Paxos and Master Election algorithm. The value of running time decreases after 5^{th} experiment, for one terminal is closed. Figure 4 illustrates the value of running time of 5 terminals. After 3^{rd} and 7^{th} experiments, the value of time decreases progressively. Comparing Figs. 3 and 4, it is concluded that Election Master algorithm works more efficient than the Paxos algorithm.

Figure 5 illustrates the number of messages communicate between 3 controllers, the number of messages reduces after 5^{th} experiment. Figure 6 illustrates the number of messages communicate between 5 controllers, we also notice that there are twice abatement messages number after 3^{rd} and 7^{th} experiments. Comparing Figs. 5 and 6, it is obvious that the Master Election algorithm operates lower traffic load than Paxos algorithm, because its number of communication messages between each controller is less than the Paxos thus results in higher system efficiency.

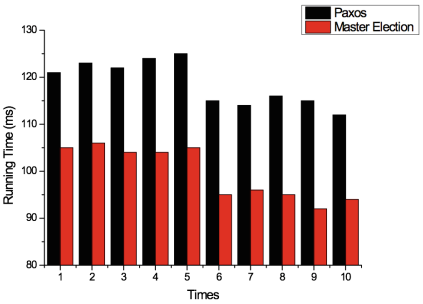

Fig. 3. Running time of 3 terminals

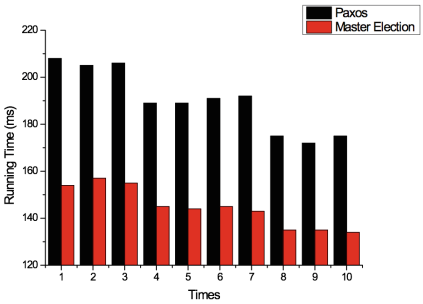

Fig. 4. Running time of 5 terminal

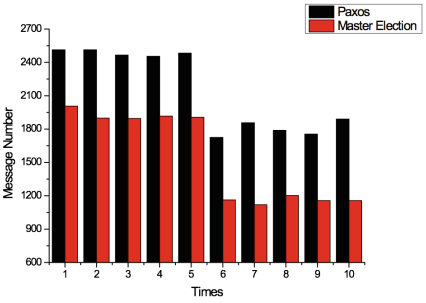

Fig. 5. Message number of 3 terminals

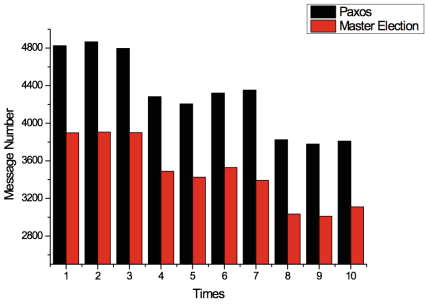

Fig. 6. Message number of 5 terminals

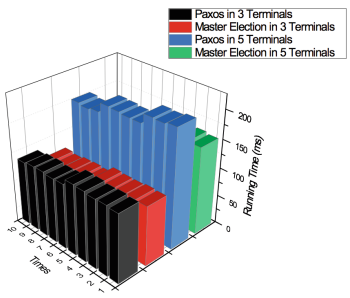

Fig. 7. Running time of 3 and 5 terminals

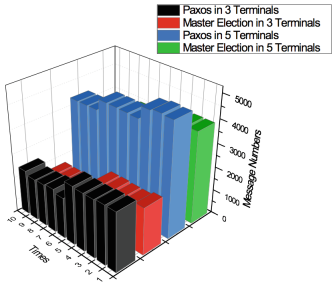

Fig. 8. Message number of 3 and 5 terminals

According to the simulation results, we conclude that Master Election mechanism meets fault tolerance requirement. In master and slave controllers system, when the number of failed controller reaches a half of sum, both Paxos algorithm and Election Master algorithm still work normally and guarantee the system stability and robustness. In 3 terminal windows simulation experiments, when 1 controller becomes disabled and reaches the a half of total number. As the simulations results show, this system also keeps stable and implements master controller election internally. In 5 terminal windows

experiments, there are two terminals closed and the system also works properly, which proves that both Master Election and Paxos algorithm implement good performance in fault tolerance. Figures 7, 8 are the two-dimensional figure of the running time, messages number of 3, 5 terminals in Paxos and Master Election algorithm.

5 Conclusion

In this experiment, we compare Master Election algorithm with Paxos and get conclusion that Master Election algorithm can solve the master controller election problem. According to the three performance indicators of distributed algorithm, we find Master Election algorithm meets fault tolerance requirement. For in a master and slave controllers system, when the number of failed controllers reaches the half of the total controllers, both Paxos algorithm and Election Master algorithm can ensure system normal operation. In addition, Master Election has shorter running time, lower traffic load than Paxos which proves that we proposed algorithm has superiority in master controller election for SDON system.

Acknowledgements. This work is supported by NSFC project (61271189, 61571058), Open Fund of State Key Laboratory of Information Photonics and Optical Communications, BUPT (IPOC2014ZZ03), and supported by Key Technology Research of Software Defined Optical Network Oriented to Multi-domain Interaction of Power Communication Project.

References

1. Mingming, C., Guochu, S., Yihong, H., Zhigang, G.: Enabling software-defined optical networks based on openflow extension. In: Opto-Electronics and Communications Conference, pp. 1–3. IEEE Press, Shangha (2015)
2. Michael, J., Simon, O., Daniel, S., Rastin, P.: Modeling and performance evaluation of an OpenFlow architecture. In: 2011 23rd International Teletraffic Congress, pp. 1–7. IEEE Press, San Francisco (2011)
3. Amin T., Yashar G.: HyperFlow: a distributed control plane for OpenFlow. In: Proceedings Internet Networking Management Conference Resolution Enterprise Network, p. 3. USENIX Assoc., CA, USA (2010)
4. Rob, S., Glen, G., Kok-Kiong, Y.: FlowVisor: a network virtualization layer. In: OpenFlow Switch Consortium, Stanford, CA, USA (2009)
5. Soheil Hassas, Y., Yashar G.: Kandoo: a framework for efficient and scalable offloading of control applications. In: Proceedings of the First Workshop on Hot Topics in Software Defined Networks, New York, pp. 19–24 (2012)
6. Yazici, V., Sunay, M.O., Ercan, A.O.: Controlling a software-defined network via distributed controllers. In: Proceedings of the 2012 NEM Summit, Turkey, pp. 16–20 (2012)
7. Leslie, L.: Paxos Made Simple (2001)
8. WenCheng, S., JianPing, L.: Research on consistency of distributed system based on Paxos algorithm. In: 2012 International Conference on Wavelet Active Media Technology and Information Processing, pp. 257–259. IEEE Press, Chengdu (2012)
9. Barbieri, R.R., Vieira, G.M.: Hardened Paxos through consistency validation. In: 2015 Brazilian Symposium on Computing Systems Engineering, pp. 13–18. IEEE Press, Foz do Iguacu (2015)