PAPER • **OPEN ACCESS**

A Distributed Dynamic Adaptive and Fast Balancing SDN Controller Management

To cite this article: Yang Liu *et al* 2019 *IOP Conf. Ser.: Earth Environ. Sci.* **234** 012027

View the article online for updates and enhancements.



IOP ebooks™

Bringing you innovative digital publishing with leading voices to create your essential collection of books in STEM research.

Start exploring the collection - download the first chapter of every title for free.

This content was downloaded from IP address 106.110.68.45 on 20/09/2019 at 10:53

IOP Conf. Series: Earth and Environmental Science **234** (2019) 012027 doi:10.1088/1755-1315/234/1/012027

A Distributed Dynamic Adaptive and Fast Balancing SDN Controller Management

Yang Liu [1], Qian He[2*], Xiongying Li[2] and Shuiming Zhou[1,2]

[1]Guangxi Transportation Research and Consulting Co.,Ltd, Nanning 530007, China
[2]Guangxi Key Laboratory of Cryptograph and Information Security, Guangxi University Key Laboratory of Cloud Computing and Complex System, Guilin University of Electronic Technology, Guilin 541004, China
heqian@guet.edu.cn

Abstract. Tradition Software Defined Network (SDN) suffers from single point failure and uneven load distribution problems. Multiple SDN controllers work together to construct a distributed dynamic adaptive cluster with dynamic load balancing for large scale SDN switch stream requesting. Introducing such features as CPU, memory and network load, a Dynamic Adaptive and Fast algorithm (DAF) can select source, target and destination SDN controllers to balance load dynamically. The results of experiments show that DAF works well and improve the cloud service system throughput.

1. Introduction

Software Defined Network (SDN) becomes a promising network paradigm, separating the control plane and data plane in networks. SDN supports the control and management of the network through software programming, improving the flexibility and maneuverability in the implementation of new network technologies and protocols [1]. A SDN based architecture for enhancing the data transmission performance in cloud based applications (smart grid, smart healthcare and smart city, etc) become more and more popular [2].

Solutions of load balancing schemes of multiple controllers can be divided into two categories: centralization and distribution. ElastiCon uses a centralized load adapter responsible for collecting load information of the all controllers and adjusting the number of controllers in the controller pool based on the workload information [3]. The method measures the migration cost with a single factor to achieve load balancing between multiple controllers. In the centralized controller system, once the controller fails, it will cause paralysis of the entire network [4][5].

It becomes more and more important for the SDN controller to make balance decision locally by gathering load information of all controllers [6][7]. A load balancing strategy allows every controller makes load balancing strategies locally in [6]. The authors of [7] present a load informing strategy based on [6] to lower the frequency of load informing. They use only a factor to evaluate the load state of controllers.

Aiming at the single point failure and uneven load distribution problems of the SDN management, we design a distributed dynamic load balancing SDN controller mechanism. The distributed SDN controller cluster technology is applied to manage the smart cloud network, which avoids the paralysis of the network due to the failure of the single controller. After comprehensively evaluate the load status of controllers from four aspects of CPU, memory, switch and message rate, a distributed dynamic adaptive and fast load balancing algorithm (DAF) is proposed for cloud service.

The remainder of this paper is organized as follows: Section 2 gives the distributed SDN controller

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

system architecture. Section 3 propose the dynamic loading algorithm of SDN management. The performances are evaluated in Section 4. Finally, conclusions are drawn in Section 5.

2. Distributed SDN Controller System

2.1. System Architecture

With the increasing number SDN switches, single SDN controller cannot satisfy the massive requesting stream. Multiple SDN controllers are constructed to a distributed system to control a lot of SDN switches corporately. The distributed SDN controller system architecture is shown in Fig.1.

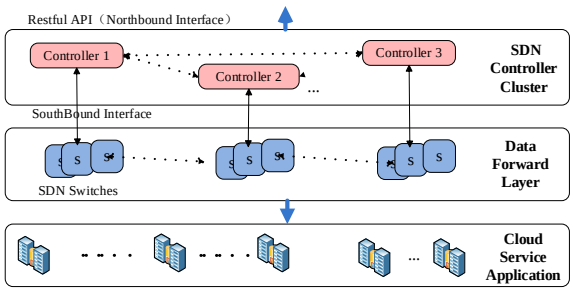

Figure 1. Distributed SDN controller system architecture

Since the SDN switches with big scale on the data forward layer may cause massive control requirements, single SDN controller cannot support massive flow table refresh requirements. Multiple SDN controllers construct a distributed SDN controller cluster which control massive SDN switches through the southbound interface. Real time network state information (RNSI) is shared in the SDN control cluster and then used to balance the load. Through the northbound interface, the SDN controller cluster can interact with such flow aware system network applications as OpenStack to improve the application user experience and whole network utility. The integration between OpenStack and SDN can provide better services for users. Based on the OpenFlow protocol, switches matching and updating its flow table flexibly by the rules controller issued. Various applications realize various functions based on SDN, and the transferred data are the changing reason for the SDN switch. It is also the scheduling objective for SDN controller to make the cloud service application working faster and fairer.

2.2. Load Balancing Module

A load balancing module is designed to carry out the load balancing strategy in the SDN controller. The DAF algorithm is implement in the load balancing module. The main components of the load balancing module are shown in Fig. 2.

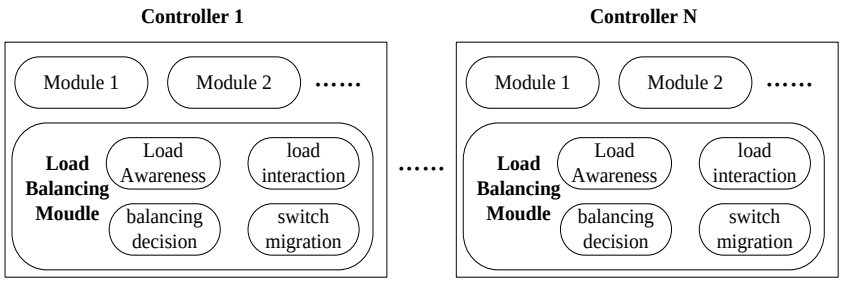

Figure 2. The components of load balancing module

Load balancing module is composed of four parts: load awareness, load interaction, load balancing decision and switch migration. The load awareness part is responsible for periodically collecting the

load information of each controller in the system. The load information interaction section is used for a controller sending its load information to other controllers. The load balancing decision section is liable for making load balancing decisions. The switch migration part is in charge of moving the selected switch to balance the load among controllers.

3. Dynmic Load Balancing Algorithm

It is important to realize a load balancing strategy to dynamically adjusts the load of controllers based on the real-time load state from SDN. When the system has an overloaded controller, it triggers the controller to execute the load balancing strategy.

3.1. Dynamic Adaptive and Fast Load Balancing Algorithm

Dynamic Adaptive and Fast Load Balancing (DAF) algorithm balances the load among controllers and improves the throughput of the cloud application system by dynamically adjusting switch-controller mapping in the whole cloud network. The detailed steps are shown in Algorithm 1. Where, L_{now} / L_{pre} is the current/previous load value, *Thr* is the load threshold, T and R_t are the testing period and round-trip time, *count* is the number of controllers, and q,x1,x2,μ1,μ2 are weight coefficients factors.

Each controller uses the load-aware part to periodically collect its own load information. A load interaction component judges whether controllers needs to send it to others to complete the load information interaction among controllers (algorithm 1 line 3). The controller triggers the switch migration action when the controller's load value exceeds its threshold. The overloaded controller use load balancing decision part to select the source controller, the target switch, and the destination controller (algorithm 1 line 5~7), then the target switch is migrated from the source controller to destination controller. Finally, after completing the migration, controllers continue to update its load information and to interact with other controllers.

Algorithm 1 **DAF(L_{now},L_{pre},Thr,T,R_t,count,q,x_1,x_2,μ_1,μ_2)**

Input: L_{now},L_{pre},*Thr*,*T*,R_t,*count*,*q*,x_1,x_2,μ_1,μ_2
Output: *LoadBalanceFlag*(**True** or **False**)
1. loadBalanceFlag = **False**
2. **while** (t = n*T*)
3. loadInforming($\mathbf{L_{now}}$, $\mathbf{L_{pre}}$, **q**, **Thr**)
4. **while** L_{now} != Ø do
5. *overLoad* ← srcControllerSelect(L_{now},*Thr*,*count*)
6. *targetSwId*,*targetR*$_s$ ← targetSwitchSelect(R_s,*overLoadId*)
7. *dstConId* ← dstConSelect(x_1, x_2,μ_1,μ_2,*targetSwId* ,*targetR*$_s$,R_t)
8. Migrate the switch with id *targetSwId* from controller with id *overLoadId* to controller with id *dstConId*
9. loadBalanceFlag = **True**
10. **end while**
11. **end while**
12. **return** loadBalanceFlag

3.2. Source, Target, and Destination Controller Algorithm

In DAF, the source, target and destination controller functions are completed in three functions: srcControllerSelect(L_{now}, *Thr*, *count*), target Switch Select(R_s, *overLoadId*), dstConSelect(x_1, x_2, μ_1, μ_2, *targetSwId*, *targetR*$_s$, R_t), respectively.

srcControllerSelect function performs separate processing according to the number of overloaded controllers. When there is only one controller in overloaded condition, this controller is selected as the source controller. We calculate the overload rate ($R_{overload}$) of each overloaded controller by Equation 1.

$$R_{overload} = (L_{now} - Thr)/Thr \quad (1)$$

And then, we sort the overloaded controllers in descending order with their overload rate and select the controller with the highest overload rate as the source controller.

After selecting the source controller, we choose which switches to be migrated in the function *targetSwitchSelect*. Through the load awareness part, we obtain the average message request rate (R_s) from each switch. The bigger the R_s value is, the switch brings more loads to source controller. In order to reduce the load of the source controller, we sort the switches controlled by the overloaded controller in descending order with R_s and select the switch with the highest R_s as the target switch.

In *dstConSelect* function, we take the load value of destination controllers and the RTT value between destination controllers and target switch into account. We choose the highest priority controller from controllers, which is not overloaded and satisfies the constraint given in Equation 2, as the destination controller. If there are several such controllers, we will choose randomly.

$$Lmig<\mu 2*(Thr-Lnow \quad (2)$$

4. Experiment Analysis

To evaluate the effectiveness of DAF algorithm, we realize DAF based on Floodlight[8]. Mininet [9] is used to simulate SDN environment and Cbench [10] is used to measure the maximum rate of Packet-In messages.

Comparing the DAF algorithm with the static algorithm and injection proposed in [7], this scheme can verify whether the DAF algorithm can effectively improve the throughput of a cloud application system. We set load threshold to 80%, monitoring period is 1s. The total number of Packet-In messages processed per second by all controller in the system will be counted as system throughput. the switch is triggered to encapsulate the packets into Packet-In messages and report the messages to the controller. From 0 to 10s, packets is send to each controller in 8000 pps (Packet-In messages per second).. Secondly, we injected 15000 pps to C_A, 6000 pps to C_B and 3000 pps to C_C at the time of 11s. We plot the results in Fig.3.

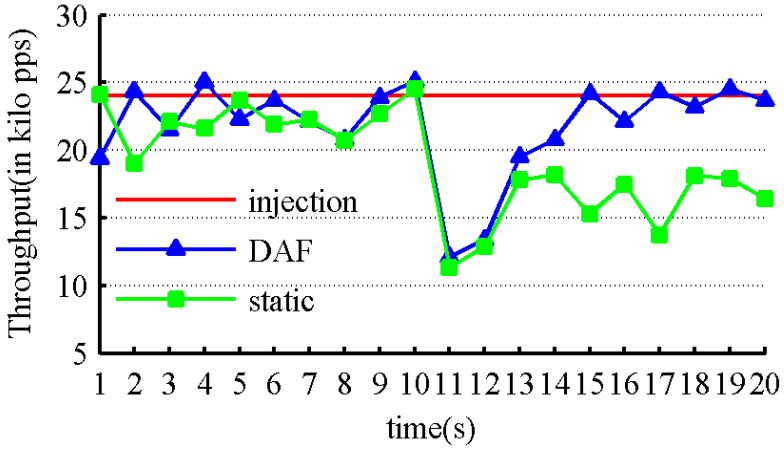

Figure 3. System throughput of SDN based application

As Fig. 3 shows, from 0 to 10s, in static algorithm, the system throughput is approximately equal to that in DAF algorithm. That is because the total number of Packet-In packets is less than the number that controllers of the cloud system can handle. However, from 11 to 20s, the system throughput in static algorithm is lower than that in DAF algorithm. Since the variance of Packet-In requests leads to load imbalance among controllers, C_A is overloaded and Packet-In packets begin loss due to buffer overflow. In DAF algorithm, overloaded C_A will trigger load balancing strategy, and the partial load of C_A will be shift to C_C dynamically without causing the C_C to be overloaded, so the total throughput of cloud system gets increased.

5. Conclusion

Aiming at single point failure and load imbalance in SDN based cloud network, we designed a distributed SDN controller cluster system with dynamic balancing capability. DAF algorithm is proposed based on RNSI. DAF algorithm can support massive SDN switches with scalability. The experimental result shows that the distributed DAF based SDN controller cluster shortens the time of balancing and improves the throughput of cloud application system. In the future, we intended to optimize the load factors and load balancing decision scheme for the distributed SDN controller cluster.

6. Acknowledgment

This work is supported in part by the National Natural Science Foundation of China (61661015, 61861013), Guangxi Innovation-Driven Development Project (AA17202024), Ministry of Education Key Lab of Cognitive Radio and Information Processing Found (CRKL160101), Guangxi Key Laboratory of cryptography and information security Found (GCIS201701) , Guangxi Collaborative Innovation Center of Cloud Computing and Big Data Found (YD16801, C77KYS02SX18) and High Level of Innovation Team of Colleges and Universities in Guangxi Outstanding Scholars Program Funding.

7. References

[1] Qingyun Zuo, Ming Chen, Guangsong Zhao, Changyou Xing, Guomin Zhang, Peicheng Jiang. Based on OpenFlow SDN technology research [J]. Journal of Software,2013, (05):1078-1097

[2] Bedhief, M. Kassar and T. Aguili, “SDN-based architecture challenging the IoT heterogeneity,”2016 3rd Smart Cloud Networks &Systems(SCNS),Dubai,2016:1-3

[3] Dixit, F. Hao, S. Mukherjee, T. V. Lakshman and R. R. Kompella, “ElastiCon; an elastic distributed SDN controller,” 2014 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), Marina del Rey, CA, 2014:17-27

[4] Levin D, Wundsam A, Heller B. Logically centralized? State distribution trade-offs in software defined networks. In: Proc. of the ACM SIGCOMM Workshop on HotSDN. 2012:1−6

[5] Mengmeng Wang, Jianwei liu, Jie Chen, Jianfei MAO. Software definition network: security model, mechanism and research progress [J]. Journal of Software,2016,(04):969-992

[6] Y. Zhou et al., “A Load Balancing Strategy of SDN Controller Based on Distributed Decision,” 2014 IEEE 13th International Conference on Trust, Security and Privacy in Computing and Communications, Beijing, 2014: 851-856

[7] Jinke Yu, Ying Wang, Keke Pei, Shujuan Zhang and Jiacong Li, “A load balancing mechanism for multiple SDN controllers based on load informing strategy,” 2016 18th Asia-Pacific Network Operations and Management Symposium (APNOMS), Kanazawa, 2016:1-4

[8] Floodlight.[EB/OL].http://www.projectfloodlight.org/

[9] “mininet”,http://mininet.org/, [Online; accessed June 02, 2018]

[10] “Cbench”, http://sourceforge.net/projects/cbench/, [Online; accessed January17,2018]