Advances in Engineering Research (AER), volume 124
2nd International Symposium on Advances in Electrical, Electronics and Computer Engineering
(ISAEECE 2017)

An Improved Switch Migration Approach to Controller Load Balancing in SDN

Tao Hu[1, a], Jianhui Zhang[2, b], Liye Wang[3, c], Dan Qiao[4, d]

[1,2,3]National Digital Switching System Engineering & Technological R&D Center, Zhengzhou, China

[4]China Satellite Maritime Tracking and Control Department, Jiangyin, China

[a]hutaondsc@163.com, [b]1161508468@qq.com, [c]151292849@qq.com

Keywords: Software Defined Networking(SDN); Controller; Load Balancing; Switch Migration

Abstract. The distributed SDN network has improved the scalability and reliability of controller. However, such deployment architecture introduces a new challenge to the load balancing of controllers when uneven loads distribution among controllers because of the static configuration between switch and controller. To address this problem, we improve the existing switch migration approach and propose a Staged Switch Migration Strategy (SSMS) to balance controller loads in multi-domain network. SSMS includes two stages. In stage 1, integrated several controller overheads, we select the migration objects included switch immigration domain and switch emigration domain based on genetic algorithm. In stage 2, the sequential migrating of switches are implemented through setting subdomain migration degree and switch validity. Simulation shows that SSMS can achieve the effective controller load balancing with better performance in distributed SDN network.

Introduction

Software Defined Networking (SDN) based on OpenFlow has emerged as a new paradigm which decouples control plane from data plane [1]. With the increasing of OpenFlow-enabled equipments, the traditional SDN architecture relied on the centralized controller (NOX[2], Floodlight [3]) has the limitations of scalability and reliability.

To improve extendibility and avoid the single point of failure, several researches propose a logically centralized, but physically distributed control plane with multiple controllers which work cooperatively to divide the network into some areas with separate controllers, like HyperFlow[4], Onix[5]. However, such distributed architecture produces a new challenge for controller load balancing. Due to the connections between switches and controllers are static, this will result in the unbalanced distribution of controller loads when the traffic fluctuates frequently in the network.

There are several researches have been conducted on controller load balancing. [6] firstly proposes to build switch migration model, which considers the average and maximum delay of controller in the network. [7] introduces ElastiCon with double overload thresholds to decide the shift of controller load. [8] pays attention to the change of traffic and sets traffic monitoring module, it adjusts the controller loads according to the real time flow rates. In [9], it designs a dormant multi-controller model for SDN. Part of controllers are allowed to enter dormant state under light traffic condition for saving cost. [10] sets the main performance measurements which should be taken into account to provide a “good” migration in terms of Quality of Service. In [11], the Load Informing Strategy is proposed to make load balancing decision locally as rapidly as possible. However, there are rigid migration objects and migrating conficts in the process of migration on the above researches.

This paper studies controller load balancing further building upon the work of switch migration. We improve the existing migration model and propose a Staged Switch Migration Strategy (SSMS) to balance controller loads in multi-controller network.

The main contributions are as followings.

- We make the first attempt to explore the migrating model based on multi-domain network and set the parameters included Packet-in request, the interaction traffic, communication overheads between controllers and flow table installation cost. All factors affect the controller load balancing.

- We design a SSMS to implement controller load balancing included two stages. In stage 1, we get the optimal migration domain solved by genetic algorithm. In stage 2, the multiple switches are migrated in parallel through setting subdomain migration degree and switch validity.
- The simulation demonstrates the nice load balancing performance of SSMS in real topology.

Model and Formulation

Model Description. The distributed SDN network is partitioned into several subdomains connected with physical links. The whole network topology is presented as $G=(V,E)$, where V and E is the set of nodes and links, respectively. Control plane consists of M controllers, denoted as $C=\{c_1,...,c_M\}$. The number of switches is N in data plane, and switch set is $S=\{s_1,...,s_N\}$, $|V|=N+M$. The whole network is divided into M subdomains, so $G=\{G_1,...,G_M\}$. The relationship between switch s_i and controller c_j is a binary variable x_{ij}, where $x_{ij}=1$ represents s_i is connected to c_j.

In multi-domain SDN network, there is dynamic traffic with temporal and spatial characteristics, which is prone to produce overload controller. To solve this problem, we optimize the existing switch migration approach and propose a Staged Switch Migration Strategy (SSMS). The key components of SSMS are shown in Fig. 1, which consists of four parts. (*i*) Load collection and measurement module is responsible for obtaining the load informations and link states. (*ii*) Evaluation and decision module is used for judging whether the controller load exceeds the overload threshold. (*iii*) Storage module is designed for saving informations included link, topology and traffic. (*iv*) Migration module is the core part of designment, which is responsible for coordinating and completing switch migration.

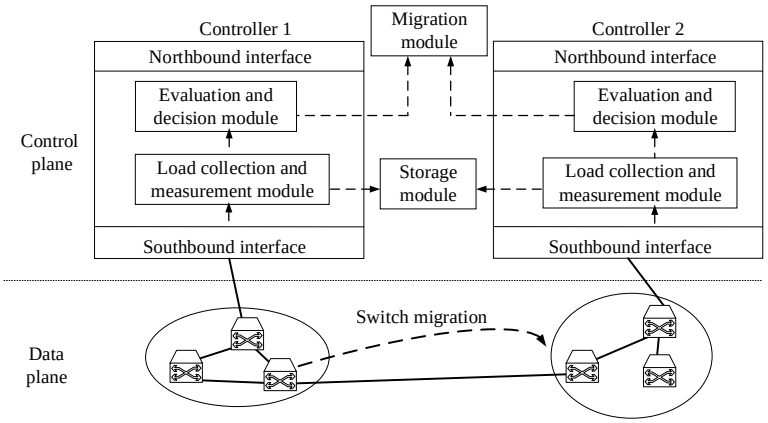

Fig.1 The distributed controller load balancing architecture

Parameter Setting. In SDN network, controller load includes four parts: Packet-in request, the interaction traffic, communication overheads between controllers and flow table installation cost.

Packet-in request sended from switch to controller is the major load for controller. Firstly, we introduce the concept of switch request degree.

Definition 1. Switch request degree. The number of Packet-in requests from switch s_i to controller is defined as switch request degree r_{s_i}. In subdomain G_j, n_j is the number of nodes, R_{G_j} is the sum of switch request degree in Eq. (1), $R_{avg}^{G_j}$ represents the average switch request degree in Eq. (2).

$$R_{G_j}=\sum_{i=1}^{n_j} r_{s_i}\cdot x_{ij} \tag{1}$$

$$R_{avg}^{G_j}=\left(\sum_{i=1}^{n_j} r_{s_i}\cdot x_{ij}\right)\Big/\eta_j \tag{2}$$

The controller communicates with switches periodically in the corresponding subdomain. T_{c_j} is the interaction traffic between controller c_j and switches in G_j in Eq. (3),

$$T_{c_j} = \sum_{i=1}^{n_j} d_{ij} \cdot \psi \cdot x_{ij} \quad (3)$$

where d_{ij} is the hop between device *i* and *j*, the longer distance between two devices, the larger T_{c_j}. ψ is the average interaction traffic for switches in the subdomain.

The packet may pass different subdomains, so controllers need to interact the state informations with each other. The communication overheads between c_j and c_k is P_{jk} in Eq. (4), ζ is the average cost of state interaction.

$$P_{jk} = \sum_{i=1}^{n_j} \sum_{s=1}^{n_k} \zeta \cdot d_{jk} \cdot x_{ij} \quad (4)$$

When a new flow arrives at switch, the corresponding controller must install the new flow table into switch. So, we set the flow table installation cost is I_{c_j} in Eq. (5), f_p is the size of Packet-in.

$$I_{c_j} = \sum_{i=1}^{n_j} f_p \cdot d_{ij} \cdot x_{ij} \quad (5)$$

The controller load is the weighted sum of aboved four costs included $R_{avg}^{G_j}$, T_{c_j}, P_{jk} and I_{c_j}. Therefore, we define the load of controller c_j is L_j in subdomain G_j, as shown in Eq. (6), where τ_1, τ_2, τ_3 and τ_4 are the weights of corresponding costs, respectively.

$$L_j = \tau_1 \cdot R_{avg}^{G_j} + \tau_2 \cdot T_{c_j} + \tau_3 \cdot P_{jk} + \tau_4 \cdot I_{c_j} \quad (6)$$

Implementation

Staged Switch Migration Strategy (SSMS) is partitioned into two stages to implement controller load balancing. The details are described as follows.

Stage 1. In this stage, we design SSMS-1 algorithm to select the optimal migration objects. The main process of SSMS-1 is as follows. Firstly, load collection and measurement module detects the controller load in each subdomain. If controllers are determined as overload controllers by comparing to the average load L_A, SSMS-1 sets function L as fitness function, and elects the specific subdomains as feasible domains to implement genetic manipulation[12]. Finally, algorithm outputs immigration domain G_l and emigration domain G_h. Algorithm pseudocode is shown in Table 1.

Table 1 SSMS-1 algorithm

Algorithm 1 SSMS-1
Input: SDN subdomains $\{G_1, G_2, ..., G_M\}$
Controller load L
Output: Switch immigration domain G_l
Switch emigration domain G_h
1: Compute $\{R_{avg}^{G_1}, ..., R_{avg}^{G_M}\}$ and select $R_{avg}^{G_h} = MAX\{R_{avg}^{G_1}, ..., R_{avg}^{G_M}\}$
2: Select the subdomain with $R_{avg}^{G_h}$ as G_h
3: Adjacent subdomain of G_h is set as $A = \{G_{h-1}, G_{h-2}, ..., G_{h-k}\}$
4: **whlie** ($A \neq \phi$)
5: Select $G_{h-\alpha}(R_{avg}^{G_{h-\alpha}} \leq R_{avg}^{G_j})$, $G_{h-\beta}(R_{avg}^{G_\beta} > R_{avg}^{G_j})$
6: Crossover $G_{h-\alpha}^{'} \leftarrow G_{h-\alpha}$, $G_{h-\beta}^{'} \leftarrow G_{h-\beta}$

ATLANTIS PRESS

7: Mutation $G^{*}_{h-\alpha} \leftarrow G^{'}_{h-\alpha}$, $G^{*}_{h-\beta} \leftarrow G^{'}_{h-\beta}$

8: Select subdomain with $L_{\min}$ as G_l

9: **endwhile**

Stage 2. Based on the output of SSMS-1, we design SSMS-2 to migrate switches in stage 2. Firstly, we define the concept of subdomain migration degree and switch validity**.**

Definition 2. Subdomain migration degree. The number of switches migrated from G_p to G_q is defined as subdomain migration degree $M(G_p, G_q)$ in Eq. 7.

$$M(G_p, G_q) = \left| (R^{G_p}_{avg} - R^{G_q}_{avg}) \Big/ \sum_{j=1}^{M} R^{G_j}_{avg} \right| \tag{7}$$

Definition 3. Switch validity, which represents the periods of valid state of switch during migration, is defined as $V(s)$. The switch validity includes two parameters year $Y(s_i)$ and life $L(s_i)$. $Y(s_i)$ is the current period of migrating switch s_i. After s_i goes through one migration process, then $Y(s_i)++$. $L(s_i)$ is the total number of round-robins in migration process, $V(s_i) = L(s_i) - Y(s_i)$. if $V(s_i) = 0$, we give up migrating s_i.

The process of SSMS-2 is as follows. Firstly, we compute switch request degree of migration domain and the number of migrationg switches is determined with migration degree. Then, we count the switch validity for migrating switch in the process of migration and select the switch that meets the constraint of validity to implement dynamic migration. Finally, we construct the new SDN topology according to migration results. SSMS-2 algorithm pseudo-code is shown in Table 2.

Table 2 SSMS-2 algorithm

Algorithm 2 SSMS-2

Input: Switch immigration domain G_l

Switch emigration domain G_h

Output: New SDN topology

1: Compute switch request degree r_{s_h} $(h \in \{1,...,n_h\})$ for s_h in G_h

2: Get the set of switch request degree Λ in descending order

3: Compute $M(G_h, G_l) = \eta$

4: Select the front switches in Λ and get $\Gamma = \{s_{h-1}, s_{h-2}, ..., s_{h-\eta}\}$

5: Switch validity $V(s_i)$ $i \in \{h-1, h-2, ..., h-\eta\}$ in Γ

6: Migrate switch s_i, $G_h = G_h \setminus \{s_i\}, G_l = G_l \bigcup \{s_i\}$

7: **if** $(L_l < L_A) \bigcup (V(s_i) = 0)$

8: **Return** 4

9: **else** $L_l \geq L_A$

10: Migration is over

11: **endif**

Experiment and Evaluation

Simulation Setting. Experiment adopts Opendaylight controller[13] and programmes the corresponding algorithm module based on Java. Network topology derives from Internet 2 [14], meanwhile, the configuration of device is Intel Core i7 CPU2.67Hz RAM 8GB.

To achieve the better simulation effects, experiment parameters are set as follows. (*i*) Controllers have the same performance, controller capacity is 10M and the average flow rate is 120KB/s; (*ii*) We

don't consider device failure during switch migration; (*iii*) We set the proportion of four weights is 0.4:0.2:0.2:0.2 to highlight the principal cost (Packet-in request).

Verification and Analysis. Based on Internet 2 topology, the entire network is divided into four subdomains in Fig. 2, and controllers are deployed on the appropriate locations like [8]. When the traffic of some switches fluctuates greatly in the network, the overload controllers appear in subdomains. We compare SSMS with Nearest Migration Strategy (NMS) and Lowest Capacity Strategy (LCS) to verify the performance of controller load balancing. NMS migrates switch to the nearest controller, and LCS selects the controller with the lowest capacity as migrating object.

The control traffic overheads of different strategies are shown in Fig. 3. Before implementing any strategies, c_3 is in overloaded state. NMS selects Subdomain 2 as immigration domain to reduce controller load of c_3, but it only achieves the local balancing. LCS selects Subdomain 1 as immigration domain because of the highest residual capacity. While multiple switches are migrated into Subdomain 1, c_1 is easy to become a new overloaded controller. SSMS optimizes the selection of migrating objects integrated four costs and implements switch validity to avoid conflict, which could ensure the balanced distribution of control traffic overheads.

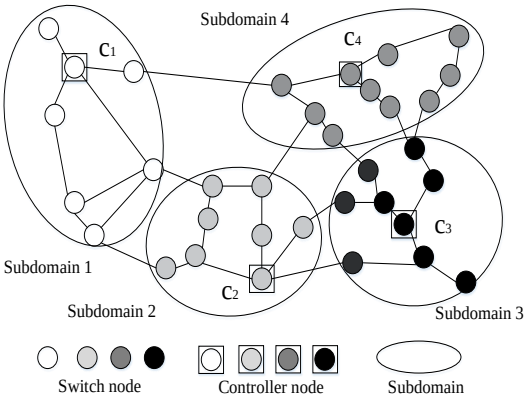

Fig. 2 Multi-domain SDN network

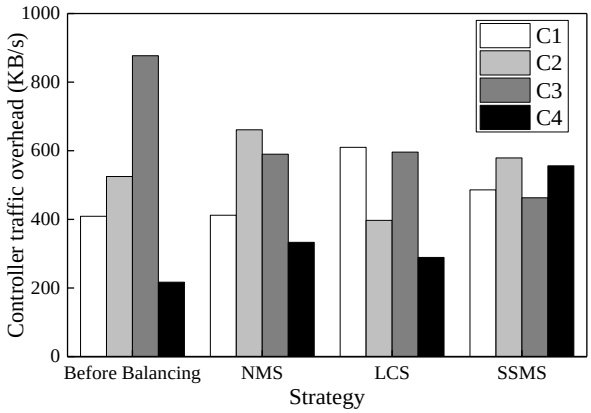

Fig. 3 Control traffic overhead in different strategy

We study the response delay between switch and controller shown in Fig. 4. When switch request rate is less than 200KB/s, there is no overload controller in the network. As the increasement of request rate, the response delay of NMS and LCS rise sharply compared with SSMS. This is because NMS and LCS only consider the single measure to migrate switch, when the traffic changes dynamically, both two strategies are prone to fall into local optimizing. SSMS selects the optimal objects from the whole network, meanwhile, switch validity ensures the sequential migrating to reduce response delay. Compared to NMS and LCS, the response delay declines 35% in SSMS.When switch request rate exceeds 480KB/s, all controllers are in overloaded state, switch migration loses efficacy for all strategies and new controller must be added into network to maintain processing performance at the moment.

Through repeating experiments, we compare the controller load balancing rate of three strategies in Fig. 5. When switch request rate is in low level, all strategies have the high controller load balancing rate. The increasing switch request rate will cause more controllers becoming overload controller. Because the neighbor subdomains of emigration domain may be in overloaded state, so the load balancing rate of NMS declines sharply. Though both LCS and SSMS search the migration domain in overall network, but LCS lacks to quantify the number of switches. SSMS sets subdomain migration degree to restrain the number of migrating switches, and has the better balance effect than LCS. When the switch request rate reaches 480KB/s, all controllers are in overloaded state, and controller load balancing rate is close to 0, meeting the prior experimental result.

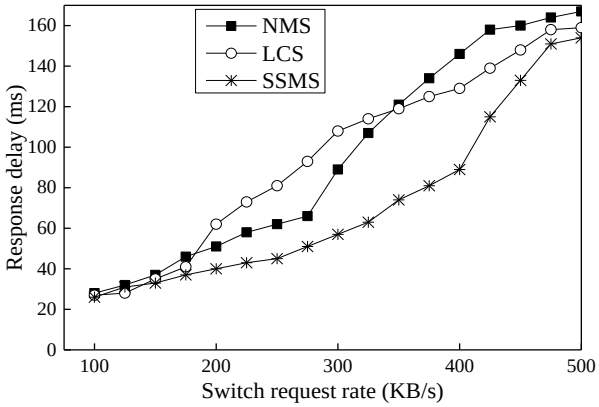

Fig. 4 The comparison of response delay

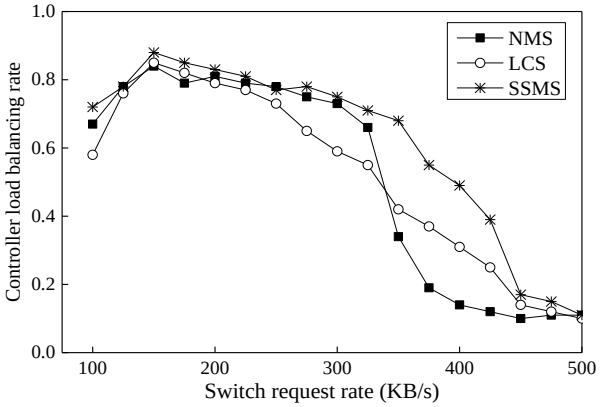

Fig. 5 The comparison of load balancing rate

Conclusion

In this paper, we make further efforts to switch migration in software defined networking. We propose a Staged Switch Migration Strategy (SSMS) included two stages to achieve controller load balancing dynamically. In SSMS, the migrating domain is determined through multi-cost synthesis based on genetic algorithm. Meanwhile, both subdomain migration degree and switch validity ensure the sequential switch migration to avoid the migrating conflict. Numerical results demonstrate that our mechanism could improve the performance of controller load balancing in distributed SDN network.

Acknowledgements

This work was financially supported by National Natural Science Foundation of China (61521003).

References

[1] N. McKeown et al., “OpenFlow: Enabling innovation in campus networks,” ACM SIGCOMM Comput. Commun. Rev., vol. 38, no. 2, pp. 69-74, Apr. (2008).

[2] NOX, [Online]. http://www.noxrepo.org

[3] Floodlight, [Online]. http://www.projectfloodlight.org/floodlight

[4] A. Tootoocian and Y. Ganjali, “HyperFlow: A distributed control plane for OpenFlow,” in Proc. ACM INM/WREN, (2010), pp. 1-6.

[5] T. Koponen et al., “Onix: A distributed control platform for large-scale production networks,” in Proc. OSDI, (2010), pp. 1-6.

[6] B Heller, R Sherwood, N McKeown. “The controller placement problem,” Proceeding of the First Workshop on Hot Topics in Software Defined Networks. Helsinki,Finland: IEEE, (2012). pp. 7-12.

[7] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, R. Kompella, “Towards an Elastic Distributed SDN Controller,” In Proc. 1st Workshop on Hot Topics in Software Defined Networking, Hong Kong, (2013), pp. 7-12

[8] BARIM F,ROYA R,CHOWDHURYS R,et al. “Dynamic controller provisioning in software defined networks,”. CNSM 2013:Proceedings of the 9th International Conference on Network and Service Management. Piscataway:IEEE, (2013), pp. 18 -25.

[9] FU Y,JUN B,WU J,et al. “A dormant multi-controller model for software defined networking ,” China Communications, (2014), pp. 45 -55.

[10] N. Perrot and T. Reynaud, “Optimal placement of controllers in a resilient SDN architecture,” 2016 12th International Conference on the Design of Reliable Communication Networks (DRCN), Paris, (2016), pp. 145-151.

[11] 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), pp. 1-4.

[12] C. Wang and X. Wang, “An island partitioning method based on cloud adaptive genetic algorithm,” 2016 International Conference on Smart Grid and Clean Energy Technologies (ICSGCE), Chengdu, China, (2016), pp. 140-144.

[13] Opendaylight, [Online]. https://www.opendaylight.org/

[14] Internet2, [Online]. http://www.internet2.edu/