2016 ICSEE International Conference on the Science of Electrical Engineering

A Dynamic Load Balancing Architecture for SDN

Hadar Sufiev
The Open University of Israel
and Jerusalem College of Technology
hadarch@jct.ac.il
Jerusalem, Israel

Yoram Haddad
Jerusalem College of Technology
haddad@jct.ac.il
Jerusalem, Israel

***Abstract*—Software-defined networking (SDN) can realize the separation between control and data planes. Relying on a single controller in future networks imposes a potential scalability problem. To tackle this problem, it has been proposed to use multiple controllers to manage the large wide-area network, where load balancing among the multiple controllers becomes the major challenge. This paper proposes a novel multi-controllers architecture for SDN. In this architecture a SuperController communicate with RegularControllers split into clusters. Thanks to our architecture which enables dynamic load balancing we improve the run time of the periodic operation and break the usual dependency between SuperController and RegularController existing in previous works.**

***Index Terms*—Load balancing; Scalability; Software-defined networking.**

I. Introduction

Software Defined Network (SDN) paradigm shift consists in separating the control plane from the data plane [1]. For this purpose the traffic is labeled as flow where for each flow that enters a switch, a dedicated flow table dictates what action should be performed on it [2]. In case there is no flow entry with a matching rule for the entering flow, the switch should send a request to a SDN controller which will decide what should be the action to be applied to this new flow. According to [3] one controller is enough for small networks, whereas large networks require multiple controllers to be able to process all the requests in a reasonable time [4]. One approach to achieve this goal is to use multiple same instantiations of a single controller [5], where each instantiation of the SDN controller does basically the same work as the others, each switch being linked to one SDN controller. Handling multiple controllers gave rise to some important question namely: where to place the controllers and how to match each control to a switch .These questions are not only relevant during the network deployment based on statics information [6] but should also been considered regularly due to the network dynamic nature [7].In [8] a novel approach is proposed to deal with the aforementioned issue. The authors proposed a method called ”BalanceFlow” where the load balancing between the SDN controllers is performed by a SuperController (SC). The super controller collects data (namely number of flows that entered each switch supported by the controller) from the so-called regular SDN controllers on a periodical basis. Then the SC executes the load balancing algorithm and matches each switch to a specific controller. Switches are updated about the new address of their supporting regular controller. During the load balancing by SC, regular controller can continue to process the switches' flows. This approaches exhibits several drawbacks: first, the SC might be quickly overloaded by the number of matching possibilities to process which can lead to serious scalability issues. Moreover, a controller that reached its maximum capacity (in term of number of flows that can processed) in the middle of a period will not be able to see its load reduced until next iteration of the load balancing algorithm at the end of the period. For these reasons, in [9] the authors suggested a new method called ”HybridFlow” which consist in splitting the controllers into clusters, such that in each cluster the controllers can help each other and perform load balancing within the same cluster. Once all the controllers in a cluster are completely saturated, then a request is issued toward the SC to lower the number of switches to be supported in this cluster. This kind of ”local” load balancing considerably reduce the burden on the SC without impacting the overall balancing of the load balancing. However the architecture of ”Hybrid Flow” is problematic for several reasons: first, the necessity for the regular controllers to wait for the SC decision leads to meaningful delay Second, in this approach there is a disproportional work separation, causing the RC to do most of the load balancing task. Third, in this approach, load balancing decision is taken by RC based on a global view that is only partial which can lead in some cases to not optimal decision. In this work we proposes a novel multi-controllers architecture for SDN. In this architecture a SuperController communicate with RegularControllers split into clusters.

The rest of this paper is organized as follows. Section II presents our model with its algorithm. Section III describes the advantages of the DCF architecture over existing ones. Finally, we conclude the paper in Section IV.

II. Dynamic Cluster Architecture

In our proposed architecture we consider a SC and several clusters of RCs where in each cluster we require the same number of RCs. In Fig 1 we show two different possibilities of splitting the RCs into clusters (first one in solid line and second one in dotted line) As in [9], we consider the two thresholds approach where we distinguished between two situations requiring load balancing. The first one is during the periodical checking of the imbalance between clusters initiated by the SC, based on the cluster's maximum load threshold. The second one can happen at any time, when a RC reaches its threshold. In this case, an overloaded RC can transfer part of its loads to another less loaded RC within the same cluster if possible, otherwise, the loaded RC issues a request to SC to know which RC in other clusters could take over the overload. Load balancing is performed at two different levels. High level operations in SC and low level operations in RC. Before delving into more details regarding these two levels we should mention that to enable SC and RCs to operate without any interdependency we define ClusterVector (CV) as a vector that contains addresses of the controllers in the same cluster. Thus each RC can ask directly another RC, via its address known from its CV, about how loaded it is. Each controller has its own CV.

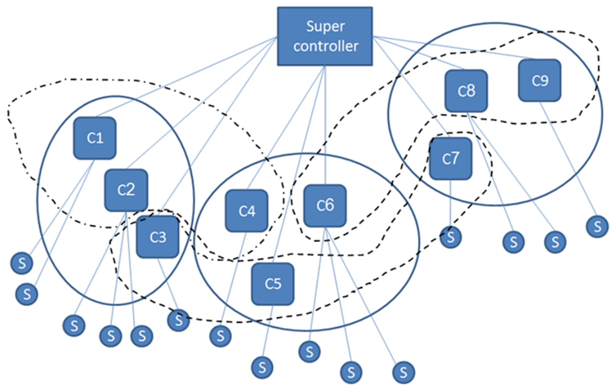

Fig. 1. The dynamic cluster architecture: two possible clusters arrangements

A. Low level operations

We consider that load balancing occurs at the low level when it is initiated by a RC. When a RC is overloaded there are two possibilities. The first one is to be relieved by another RC within the same cluster as proposed in [9]. The second one consist in finding a potential other RC outside the current cluster. In the next section we detail how the RC can be informed by the SC about potential RCs outside the cluster (see Fig 3.

B. High level operation

On a periodical basis, the SC collects global data RCs loads and detects clusters' imbalance. Then the SC runs the partition algorithm (detailed further in this section) and rearranges the clusters of RCs according to their loads. Finally the SC sends the new CVs to the controllers (see Fig. 2). This high level operation takes place within the periodical time, as well as when an overloaded RC cannot be relieved with the help of other RCs in its cluster because the cluster is already saturated. In this situation, the overloaded RC sends a request to the SC which collects controller's loads from RCs. After updating the CVs concerned by the changes, the SC sends them to the relevant RCs. This processing of an overloaded event should be treated by the SC in a timely manner to avoid congestion on switches that cannot get the rules from its RCs regarding new flows (see Fig. 3).

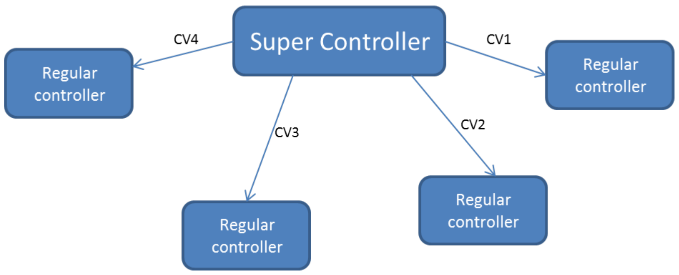

Fig. 2. Periodical operation in our dynamic cluster architecture

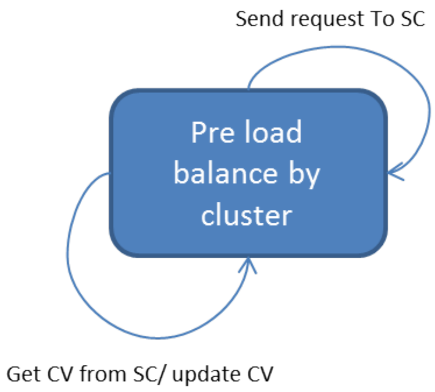

Fig. 3. Regular Controller state in our architecture

III. Architectures analysis: advantages and drawbacks

In this section we would like to analyze our solution against existing solution namely BalanceFlow [8] and HybridFlow[9].

A. Breaking dependency

One of the major drawback of HybridFlow method is the dependency between SC and RCs which can lead to important delays in the load balancing performance. In HybridFlow SC acts as a provider of RCs loads. This requires from the SC to maintain a matrix containing the load for each cluster. As shown in Fig. 4, in HybridFlow the SC cannot provide the load matrix to all the RCs at the same time because SC has to wait for the updated load matrix from a given RC before being able to send the updated version to the next RCs. Moreover, this waiting time at SC cannot leads to a situation where the updated information for next RCs is no longer relevant since network state has changed meanwhile. Needless to precise that long delays also cause the network to remain in an imbalance state which is not desirable. As we explain in section II, our model, propose the running of the load balancing at different levels where SC operation does not rely on the results of RC operation. Finally, it worth to mention that our architecture and model enables flexible algorithms development (i.e. one can develop and modify the algorithms executed at the SC and the one at the RCs at the same time since there are independent).

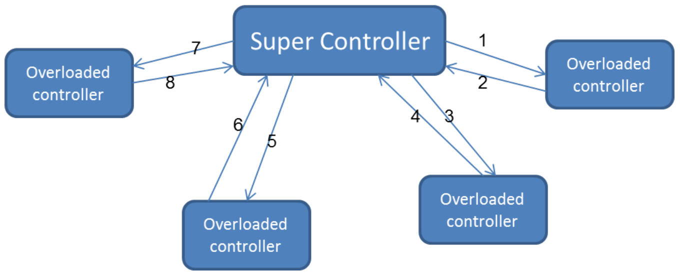

Fig. 4. periodically load operation in Hybrid Method

IV. Conclusion

In this paper we proposed a multi controller load balancing approach for SDN called DynamicCluster. When an overloaded cluster is detected the SC runs a partition algorithm that rearrange the RCs into clusters, and updates the cluster vectors of RCs. This method simplifies the load balancing operation and breaks the dependency between the SC and RCs during the periodical load balancing.

Acknowledgment

Mrs Hadar Sufiev would like to thank Mr Dror Mughaz from JCT and Bar Ilan Univ. for the helpful discussions.

References

[1] M. Yang, Y. Li, D. Jin, L. Zeng, X. Wu, and A. V. Vasilakos, "Software-defined and virtualized future mobile and wireless networks: A survey," *Mobile Networks and Applications*, vol. 20, no. 1, pp. 4–18, 2015.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "Openflow: enabling innovation in campus networks," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, 2008.

[3] B. Heller, R. Sherwood, and N. McKeown, "The controller placement problem," in *Proceedings of the first workshop on Hot topics in software defined networks*. ACM, 2012, pp. 7–12.

[4] J. Hu, C. Lin, X. Li, and J. Huang, "Scalability of control planes for software defined networks: Modeling and evaluation," in *2014 IEEE 22nd International Symposium of Quality of Service (IWQoS)*. IEEE, 2014, pp. 147–152.

[5] A. Tootoonchian and Y. Ganjali, "Hyperflow: A distributed control plane for openflow," in *Proceedings of the 2010 internet network management conference on Research on enterprise networking*, 2010, pp. 3–3.

[6] S. Lange, S. Gebert, T. Zinner, P. Tran-Gia, D. Hock, M. Jarschel, and M. Hoffmann, "Heuristic approaches to the controller placement problem in large scale sdn networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 1, pp. 4–17, 2015.

[7] S. Auroux, M. Draxler, A. Morelli, and V. Mancuso, "Dynamic network reconfiguration in wireless densenets with the crowd sdn architecture," in *Networks and Communications (EuCNC), 2015 European Conference on*, June 2015, pp. 144–148.

[8] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "Balanceflow: controller load balancing for openflow networks," in *2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems*, vol. 2. IEEE, 2012, pp. 780–785.

[9] H. Yao, C. Qiu, C. Zhao, and L. Shi, "A multicontroller load balancing approach in software-defined wireless networks," *International Journal of Distributed Sensor Networks*, vol. 2015, p. 10, 2015.