DCF: Dynamic Cluster Flow Architecture for SDN Control Plane

Hadar Sufiev*,‡ Yoram Haddad*,
*Jerusalem College of Technology
Jerusalem, Israel,
Email: {hadarch,haddad}@g.jct.ac.il
‡The Open University of Israel
Israel

***Abstract*—Software Defined Networking is considered as the new telecom revolution. Within SDN the control plane acts as the brain of the network and should be designed in the most efficient manner. In this paper we propose a novel architecture to deploy the controllers in a SDN based network. We distribute the controllers into clusters which are managed dynamically by a super controller. We develop a load balancing algorithm to avoid overloaded controllers and prove that our solution outperform the existing ones in term of complexity.**

***Index Terms*—Controller; Architecture; SDN; Wireless**

I. Introduction

In Software Defined Networking (SDN) the data plane is separated from the control plane[1]. The control plane is made of controllers which decide how the switches should handle the traffic (specifically the flows). Even though only one controller may handle the traffic for a small network [2], this is not realistic when we deal with large network at the internet scale [3]. Therefore in the very last years several architectures have been proposed, considering multiple controllers. For instance [4], [5] proposed a hierarchical control plane where a Super Controller (SC) performs load balancing among the controllers. The problem with this approach is that the SC has to match each switch to a specific controller which raises scalability issues. In addition, this approach is based on periodical load balancing and does not consider the case when a controller is being overloaded in the middle of a period. In [6] a method called "HybridFlow" is proposed. It consists in distributing the controllers into clusters when inside each cluster an overloaded controller can transfer part of its load to another controller in the same cluster. This approach doesn't consider the possibility to change dynamically the clusters which in fact can lead to an overloaded cluster. In this paper we propose a novel approach called Dynamic Cluster Flow (DCF) that tackles the aforementioned drawbacks of the existing methods. In our model we break the dependency between the SC and regular controller(RC), and split the work between them in a more efficient way. The advantage of this model is that we can improve the run time of the load balancing operation and as well as the performance of the load balancing reducing the difference between controller's loads. This is critical when the data plane consists in Base Station and Access Points, where it is mandatory to deploy controllers locally to enable timely processing of the feedback such as Channel Quality Indicator (CQI). Since the possibility to move the location of each RC is not always realistic and moreover it requires a very high amount of data processing we opted in our work for an approach where RC location is static and only the switch to controller assignment is dynamic similarly to [5]. The remaining of the paper is organized as follows. Section II presents the DCF model with its algorithms. In Section III we analyze the performance of our algorithms. Finally, we conclude the paper in Section IV.

II. Dynamic Cluster Flow Architecture

In DCF we propose an architecture based on one SC and multiple clusters of RCs. We assume a fixed cluster size. In Fig 1 we illustrate our DCF architecture and show two possible clusters arrangements(first one in solid line and second one in dotted line). Load balancing is performed at two different levels: High level operations in SC and low level operations in RC. We should mention that to enable SC and RCs to operate without any interdependencies 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.

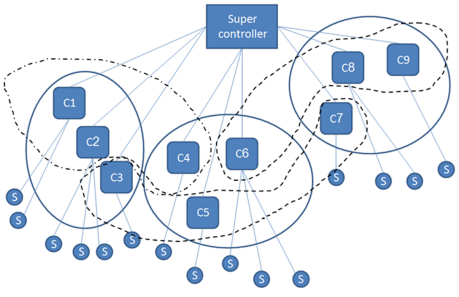

Fig. 1. The dynamic cluster architecture: two possible clusters arrangements

A. High level Periodical Operation

On a periodical basis the SC performs some rearrangement and eventually reorganizes the clusters as soon as an overload event has been identified. For this purpose, the SC has to run an algorithm that balances the load between the clusters. We

called the algorithm to be run by the SC "partition algorithm". In this algorithm we ought to split the controllers according to their loads into clusters. Each cluster has a total load which is the sum of the load over all the RCs inside the cluster. To get a relatively well distributed load over all the clusters we seek a "min-max" i.e. we try to minimize the load in the most loaded cluster. The constraint on the groups is to be in the same size. If we use the simplest heuristics to solve this problem, we sort the controllers by load. Each controller, starting with the biggest one, is matched to the group with the minimum cost function Cg, if the group is not full. Where Cg represents the Cost function which can be calculated by summing the loads of all the controllers in the group plus the load of the controller to be added. Figure 2 presents the pseudo-code for the partition algorithm where we consider 3 controllers per cluster.

- Data: set of average flow-requests number of all N controllers.
- G- {1,2, … , N/3} list of empty groups

1. Sort N in descending order of average flow-requests number
2. **foreach** descending ordered controller I **do**
3. //Allocate into a group
4. For each open group g in G **do**
    1. Cg:=cost of adding controller to group s

    end
    1. Allocate controller to group t = argmin Ct
    2. If size of t is N/3 , close t

    end

Fig. 2. DCF partition algorithm

B. High level operation upon request

As we mentioned before, when a cluster is saturated, the SC should perform some rearrangement and eventually reorganize the clusters. As soon as a SC gets a request from a RC, it collects controllers' loads from the all RCs to be able to update the CV of the requesting controller. The idea is to make minimum change i.e. update the minimum number of CVs to accommodate this overloaded situation until next periodic update. For this purpose the SC can simply replace the overloaded controller with another controller in another cluster. However this replacement needs to satisfy two constraints: first, the selected RC from another cluster to replace the overloaded RC should be enough far from its own threshold to be able to accommodate the overload generated, since all other RCs in the original overloaded cluster will naturally be overloaded themselves as well. Second, the "hosting cluster", which is the one receiving the overloaded RC should be also enough below its threshold to avoid being overloaded by the arrival of this new overloaded RC.

III. Performance Evaluation

A. Periodical load balance operation Run time

In this section we assess the performance of our architecture and algorithms in comparison to Balance Flow and HybridFlow mentioned earlier regarding the running time of the algorithms. In the following we analyze the running time complexity of the algorithm executed periodically. In Balance, the sorting operation of the m switches takes $m * logm$ time, and the matching of each switch to a controller takes $m * n$ time. In addition updating the switches with the new controller take m time. In Hybrid, for $n/3$ clusters ,in the worst case, half of the clusters are overloaded. For each overloaded cluster we have to check part of the switches, and for each checked switch we need to read a matrix with size n. Regarding the worst case on the number of switches to move to other clusters, it can reach half of all switches, so its takes $m * n$ time. In DCF, sorting the n controllers takes $n * logn$ time, matching each controller to its group takes $n * n$ time, and updating the controllers with the new cluster vector takes n time. In Table I we summarize the comparison analysis which clearly shows that the DCF method is the best where $n << m$.

B. Local load balance within periodical time running time

In Balance, the real time scenario is not mentioned. In Hybrid, SC sends n controllers' loads to the RC, to let it know which controllers can get more load. RC checks each of its switches against the n loads in $O(m * n)$ time. In DCF, SC rearranges the clusters in $O(n^2)$, sends a vector with $o(n)$ size, and the RC just updates its CV. The comparison in table I we summarize the comparison analysis which clearly shows that DCF method is the better.

Method	Balance Flow	Hybrid Flow	DCF
Periodical Run Time	$O(max(m * n, m * logm))$	$O(m * n)$	$O(n^2)$
Local Run Time	NA	$O(m * n)$	$O(n^2)$

TABLE I
PERIODICAL AND LOCAL RUN TIME COMPARISON

IV. Conclusion

In this paper we proposed a multi controller load balancing approach for SDN called DynamicClusterFlow (DCF). This method simplifies the load balancing operation and breaks the dependency between the SC and RCs during the periodical load balancing. We showed how our architecture and algorithms can improve the running time, while keeping the benefits of decreasing the number of requests to the SC, and reducing the amount of control messages required.

References

[1] C. Chaudet and Y. Haddad, "Wireless software defined networks: Challenges and opportunities," in *Microwaves, Communications, Antennas and Electronics Systems (COMCAS), 2013 IEEE International Conference on*, Oct 2013, pp. 1–5.

[2] 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.

[3] 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.

[4] 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.

[5] T. Wang, F. Liu, J. Guo, and H. Xu, "Dynamic sdn controller assignment in data center networks: Stable matching with transfers," in *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, April 2016, pp. 1–9.

[6] 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.