Update Version 1.1

May 6, 2019

Abstract

SDN is the modern era network which decouples the traditional network protocol stack into data plane containing switches and control plane containing controllers. The initial design of SDN included a single controller for a single network, which gave rise to various problems like scalability and bottle-neck formation for medium to large-sized networks. The solution to this predicament given by many researches come in the form of multiple controller configurations and the problem itself is known as the Controller Placement Problem (CPP). Most researches propose exhaustive and time-costly solutions to the CPP, with a few exceptions, one being Density Based Controller Placement (DBCP). Our algorithm Degree-based Balanced Clustering (DBC) outperforms DBCP in terms of FLow-setup Latency, Route-synchronization Latency, and Load Balancing. However, DBC has its own flaws and considers that switches generate constant load.

1 Degree-based balanced Clustering

Degree-based Balanced Clustering (DBC) places minimum (optimal) number of controllers in a SDN to minimize the followings:

1. **Flow setup latency:** When a switch receives a packet for which no path exists, the flow-setup procedure is initiated. The latency introduced by this procedure is the maximum delay required to set the path (and also to inform the intermediate switches) for the flow.

2. **Route synchronization latency:** When there is a change in the network, routes are changed, and all the controllers are updated (in sync) about the changes. This latency deals with controller-to-controller latency.

3. **Load of a controller:** The volume of control traffic and processing that the switches impose on the controllers.

This requires placing the controllers in a way that minimizes both the average controller-to-switch and average controller-to-controller latencies and limits the load of the controllers. DBC achieves the above with a careful selection of controllers and outperforms the state-of-the-art algorithms in terms of overall latency and load balancing. DBC considers that the switches generate constant load and all the switches have the same load. However, in reality, the loads of switches change from time to time depending on the number of new data packets that arrive at a switch in a given time.

2 Bearings

In order to improve DBC, we consider that the switches of the data plane generate variable load. The improved controller placement algorithm required setting flow-paths for new data packets while minimizing the load per controller. In regards to the above requirement, we selected a few researches which work with load balancing of SDNs.

2.1 Problem Statement

In [1] a new method is proposed to associate switches to the SDN controllers. Their approach aims to minimize latency [2] between controllers and their switches by ensuring the balancing of their loads. The problem is formulated as an optimization problem to minimize the total response time of the control plane. To solve this problem, a one-to-many matching game with minimum resource utilization has been defined, in which each controller must reach its minimum quota concerning the transmitted requests by the switches.

The stable matching algorithm used as the matching game is called Multi-Stage Deferred Acceptance [3]. Every stable matching algorithm requires precedence lists to decide the priority of each matching. In [1] the MSDA stable-matching algorithm is used to assign switches to controllers and the precedence lists are defined (in section 3).

2.2 Current Condition

We have implemented the modified MSDA algorithm which works with the defined precedence list. However, the MSDA algorithm as a very efficient and well-established stable matching algorithm. Therefore, it is very difficult to outperform the matching algorithm itself. However, the modified MSDA algorithm, which works with SDNs can be outperformed if we use a purely load balancing algorithm which used the latency and loads of switches to define the precedence lists. It is still unknown, how we can modify our algorithm DBC to incorporate load balancing as efficient as MSDA. Please let us know your feedback.

3 System Model

The algorithm works with multiple precedence lists to set the priorities when assigning switches to controllers. The parameters for prioritizing are as follows:

- $\succ_{LL}$ – Global precedence list based on generated load, sorted in descending order. Switches with higher loads get more priority.

- $\succ_{s_i}$ – Preference of the switches depending on the response time provided by the controllers and the distances between switches and controllers.

$$response = T_{res_{c_j}} \times d_{ij} \tag{1}$$

$$T_{res_{c_j}} = \frac{1}{\alpha_{c_j} - L_{c_j}(t)} \tag{2}$$

Here, $T_{res_{c_j}}$ is the response time of the controller c_j, d_{ij} is the distance from controller c_j to switch s_i, α_{c_j} is the processing capacity of j^{th} controller, $L_{c_j}(t)$ is the sum of the loads of all the switches under controller c_j at a given time t.

- $q_{c_j}^{max}$ – maximum quota, the maximum processing capacity of controller c_j, α_{c_j}.
- $q_{c_j}^{min}$ – minimum quota, which is the minimum Utilization rate U_{c_j}.

$$U_{c_j} = \frac{L_{c_j}(t)}{\alpha_{c_j}} \tag{3}$$

4 Algorithm

The modified MSDA keeps track of the set of switches with the lowest priority which will fill up the minimum quotas of the least wanted controllers. After each iteration, the minimum and maximum quotas of each controller will be updated. Accordingly, the set of switches will also be updated. The algorithm pseudocode is given in algorithm 1. In the algorithm, maximum and minimum quota for controller c, and the set of switches remaining, are represented in the k^{th} stage as q_c^k, p_c^k, and r^k, respectively. Initially, $r^0 = LL$, where LL is the Load List containing all the switches of the network in sorted order according to global preference. R is the set of switches with lowest priority and equal to the minimum quota summation of the controllers. $\mu^k(c)$ is the mapping of switches to the controller c in the k^th stage.

Algorithm 1 : Modified - MSDA

1: if r^k and R are identical, assign the switches to remaining minimum quotas, else run SDA on $r^{k-1} - R$
2: Remove matched switches from both sets. If all switches are assigned, go to step 3.
3: $q_c^{k+1} = q_c^k - |\mu^k(c)|$, $p_c^{k+1} = max(0, p_c^k - |\mu^k(c)|)$, $r^{k+1} = sum\ of\ all\ minimum\ quotas$
4: Go to $k+1$ stage

References

[1] A. Filali, A. Kobbane, M. Elmachkour, and S. Cherkaoui, "Sdn controller assignment and load balancing with minimum quota of processing capacity," in *2018 IEEE International Conference on Communications (ICC)*, May 2018, pp. 1–6.

[2] C. Yu, C. Lumezanu, A. Sharma, Q. Xu, G. Jiang, and H. V. Madhyastha, "Software-defined latency monitoring in data center networks," in *International Conference on Passive and Active Network Measurement*. Springer, 2015, pp. 360–372.

[3] D. Fragiadakis, A. Iwasaki, P. Troyan, S. Ueda, and M. Yokoo, "Strategyproof matching with minimum quotas," *ACM Trans. Econ. Comput.*, vol. 4, no. 1, pp. 6:1–6:40, Jan. 2016. [Online]. Available: http://doi.acm.org/10.1145/2841226