Iterative Search Based Clustering for Large-Scale Software Defined Networks

Talha Ibn Aziz*, Shadman Protik*, Md Sakhawat Hossen*, Salimur Choudhury† and Muhammad Mahbub Alam*
*Department of Computer Science and Engineering, Islamic University of Technology, Dhaka, Bangladesh
†Department of Computer Science, Lakehead University, Thunder Bay, Ontario, Canada Email:
*talhaibnaziz@iut-dhaka.edu, *shadmanprotik@iut-dhaka.edu, *sakhawat@iut-dhaka.edu,
†salimur.choudhury@lakeheadu.ca, *mma@iut-dhaka.edu

***Abstract*—The controller placement problem (CPP), in software defined networks (SDNs), deals with placing an optimal number of controllers. The placement of controllers aims to maximize the network throughput and minimize the overall latency. In order to calculate the latencies among the switches and controllers, various distance metrics are used. Hop count is one such metric, which refers to the total number of hops or adjacent node jumps required to reach from source to destination. However, other properties of networks, for example– bandwidth, delay, traffic, etc., need to be addressed as well. To solve this problem, we propose a novel controller placement algorithm, namely, Iterative Search Based Clustering (ISBC). ISBC can work with weighted representations of a network, where the weights can be fitted to be any variable like hop count, latency, congestion or traffic, etc. Simulation results show that our proposed algorithm outperform existing state-of-the-art SDN clustering algorithms in polynomial time complexity.**

***Index Terms*—CPP, SDN, latencies, clustering**

I. Introduction

Software Defined Networks (SDNs) decouple the traditional protocol stack into the ‘control plane’ and the ‘data plane’. The control plane consists of “controllers” which take the routing decisions and relay this information to the data plane. The data plane is the collection of switches which forward the data according to the routing decisions provided by the control plane. The initial design of SDN [1] included only a single controller in the control plane. However, for moderate-sized networks, a bottleneck is created due to heavy traffic concentrated at the controller. Furthermore, problems like scalability [2], [3] and reliability surface as network size increases. In order to deal with these problems, most researches deploy multiple controllers [4], [5], [6], which results in the emergence of the controller placement problem (CPP). A solution to the CPP is to select an optimal number of controllers, placing them in the best possible locations, and assigning them switches in an optimal way [7], [8]. In addition, there are more than one constraints that need to be handled– minimizing the latency among switches and controllers, maximizing reliability and resilience, and minimizing deployment cost and energy consumption, which is why it is NP-Hard.

In recent years, several solutions have been proposed to address the CPP [9], [10]– some solutions minimize latency [11] or control traffic [12], some maximize reliability [13], [14] and resilience [15], while some optimize a composite metric [16], [17]. One such solution, Density Based Controller Placement (DBCP) [18], clusters the network to minimize latency and maximize reliability using hop count as the distance metric. However, hop count is not adequate to represent the overall condition of a network. Therefore, our proposed method clusters a network to optimize a variable which can be set to any parameter. The contribution of our work can be summarized as follows:

- Our proposed algorithm work with weighted links between switches, thus allowing us to create clusters based on network traffic, bandwidth, transmission rate, etc., which are essential in determining the condition of a network.
- Our proposed algorithms cluster SDN networks in polynomial time complexity.
- We define an improvement ratio to determine an optimal point of cessation when increasing the number of controllers.

The rest of the paper is organized as follows– We represent the CPP as a graph theory problem along with some notable works on CPP in Section II. We explain our proposed algorithm in Section III. Simulation results and Performance Evaluation are presented in Section IV and we conclude in Section V.

II. Problem Formulation and Related Works

In this paper, we consider the network as a bi-directional graph $G = (S, L)$ consisting of nodes and edges. Here the set of nodes S represent the switches and the set of edges L represent the links between the switches. The edges can be either weighted or unweighted based on the requirements. Our objective is to cluster the graph G into multiple sub-networks S_i such that, each sub-network is a set of switches $S_i = \{s_{i,1}, s_{i,2}, ...s_{i,j}\}$. There cannot be any common switch between two sub-networks and all of the switches of the network must fall into a sub-network, where each sub-network will have one and only one controller.

Let us assume that the network will be clustered into k partitions. The sub-networks can be presented as,

$$S_{net} \leftarrow \{S_1, S_2, ..., S_k\} \tag{1}$$

where, S_{net} is the entire network and $S_1, S_2, ..., S_k$ are k disjoint sets of switches or sub-networks.

III. Proposed Algorithm

Our proposed algorithm solves the controller placement problem in three phases. In the first phase, we cluster the network into k sub-networks and in the second phase we select a controller for each of those sub-networks. In our third and final phase, we change the clustering in each iteration until no further improvement is possible based on an evaluation function. Therefore, we name our algorithm Iterative Search Based Clustering (ISBC).

Let us assume that the network is a graph $G = (S, L)$, where S is the set of switches and L is the list of edges connecting each pairs of switches. We sort S according to descending degree value $deg(s)$ of a node s, denoted by,

$$deg(s) = \sum_{j:l_{i,j} \in L} l_{i,j} \tag{2}$$

where, $l_{i,j} = 1$ if there exists a link between switchs i and j. The first k switches with maximum $deg(s)$ are selected as the cluster heads. As a result the switches which have high connectivity are likely to be selected as the cluster heads. The remaining switches are assigned to the clusters of the nearest cluster heads. After the initial clusters have been formed, we select the controllers for each cluster using three latency variables. The average latency for a switch v is $avgLat(v)$ and denoted by,

$$avgLat(v)_{v:v \in S_i} = \frac{1}{|S_i| - 1} \sum_{s \in S_i} d(v, s) \tag{3}$$

where, $d(v, s)$ is the edge weight of the link between switches v and s. Therefore, $avgLat(v)$ is the average of the distances of v from all other nodes s in the same cluster S_i. In the worst case scenario, this latency is the maximum of the distances of v from all other s and is denoted by,

$$maxLat(v)_{v:v \in S_i} = \max_{s \in S_i} d(v, s) \tag{4}$$

The variables $avgLat(v)$ and $maxLat(v)$ are measures of the distances among the switches of the same cluster. However, a controller also communicates with other controllers when setting up a new flow for a data packet. To accommodate this latency into the controller selection process we have to calculate the inter-controller latencies, although the controllers have not yet been selected. Accordingly, we use another variable $icLat(v)$, which is the average distance of switch v in cluster S_i from the switches outside of S_i, and can be expressed as follows,

$$icLat(v)_{v:v \in S_i} = \frac{1}{|S - S_i|} \sum_{s \in (S - S_i)} d(v, s) \tag{5}$$

The switch v of each cluster with the minimum value of $lat(v)$, which is the sum of all three controller selection selection variables, is set as the controller of that cluster. After selection of the controllers, the current clustering is evaluated using a performance metric $m_{latency}$, which we will discuss in detail in section IV-B.

The final phase of our algorithm changes the configuration of the clustered network and evaluates the change in each iteration. In each change, a node is randomly inserted into another cluster. If the $new\ m_{latency}$ is less than the old $m_{latency}$, then the network configuration is changed again. This step is iterated until no further improvement is possible. The pseudocode for ISBC is given in algorithm 1.

TABLE I: Randomized Networks used as Input

Scenario	Nodes	Edges	Edge/Node
1	40	54	1.35
2	50	66.7	1.334
3	60	81.4	1.35667
4	70	99.3	1.41857
5	80	111.2	1.39
6	90	125.5	1.39444
7	100	138.9	1.389

Algorithm 1 : Iterative Search Based Clustering (ISBC)

```
1:  procedure ISBC(k,S,L)
2:  Sort $S$ according to descending $deg(s)$
3:  Select first $k$ switches from sorted $S$ as cluster heads
4:  Assign remaining switches to nearest cluster heads
5:  $Clusters.Controllers$ = Controller-Selection(k,S,L)
6:  Calculate $m_{latency}$
7:  $Clusters$ = Calibrate($Clusters$,k,S,L)
8:  Return $Clusters$
9:  procedure Controller-Selection(k,S,L)
10:     for each node $s_i$ in cluster $C_{j:j=1..k}$ do
11:         Calculate $avgLat(s_i)$
12:         Calculate $maxLat(s_i)$
13:         Calculate $icLat(s_i)$
14:         $lat(s_i) = avgLat(s_i) + maxLat(s_i) + icLat(s_i)$
        Select $k$ $Controllers$ with minimum $lat(s_i)$
15: Return $Controllers$
16: procedure Calibrate($Clusters$,k,S,L)
17:     while improvement do
18:         Insert a switch randomly to another cluster
19:         $new\ Controllers$ = Controller-Selection(k,S,L)
20:         Calculate $new\ m_{latency}$
21:         if $new\ m_{latency} < m_{latency}$ then
22:             $m_{latency} = new\ m_{latency}$
23:             $Clusters.Controllers = new\ Controllers$
24: Return $Clusters$
```

IV. Performance Evaluation

A. Simulation Environment

We perform the experiments using C++ (High-level language). We use randomly generated networks with different numbers of switches starting from 40 to 100, at regular intervals of 10. There are 10 different datasets for each interval, giving a total of 70 different networks. The link to switch ratio is kept from 1.3 to 1.45 to keep the networks considerably sparse (similar to current worldwide networks). The networks do not contain any self-loops or multiple links between two switches but may have cycles. The seven scenarios are given in the table I along with the average values of their total number of switches, links, and link to switch ratio.

B. Performance Metric

In software defined networks (

C. Result Comparison

V. Conclusion

References

[1] K. Greene, “Tr10: Software-defined networking,” *Technology Review (MIT)*, 2009.

[2] A. Dixit, F. Hao, S. Mukherjee, T. Lakshman, and R. Kompella, “Towards an elastic distributed sdn controller,” in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 7–12.

[3] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, “On scalability of software-defined networking,” *IEEE Communications Magazine*, vol. 51, no. 2, pp. 136–141, 2013.

[4] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, “Hedera: Dynamic flow scheduling for data center networks.” in *Nsdi*, vol. 10, 2010, pp. 19–19.

[5] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu *et al.*, “B4: Experience with a globally-deployed software defined wan,” in *ACM SIGCOMM Computer Communication Review*, vol. 43, no. 4. ACM, 2013, pp. 3–14.

[6] M. Berman, J. S. Chase, L. Landweber, A. Nakao, M. Ott, D. Raychaudhuri, R. Ricci, and I. Seskar, “Geni: A federated testbed for innovative network experiments,” *Computer Networks*, vol. 61, pp. 5–23, 2014.

[7] R. Ahmed and R. Boutaba, “Design considerations for managing wide area software defined networks,” *IEEE Communications Magazine*, vol. 52, no. 7, pp. 116–123, 2014.

[8] S. Lange, S. Gebert, J. Spoerhase, P. Rygielski, T. Zinner, S. Kounev, and P. Tran-Gia, “Specialized heuristics for the controller placement problem in large scale sdn networks,” in *Teletraffic Congress (ITC 27), 2015 27th International*. IEEE, 2015, pp. 210–218.

[9] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, “A survey on software defined networking with multiple controllers,” *Journal of Network and Computer Applications*, 2017.

[10] A. K. Singh and S. Srivastava, “A survey and classification of controller placement problem in sdn,” *International Journal of Network Management*, p. e2018, 2018.

[11] 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.

[12] L. Yao, P. Hong, W. Zhang, J. Li, and D. Ni, “Controller placement and flow based dynamic management problem towards sdn,” in *Communication Workshop (ICCW), 2015 IEEE International Conference on*. IEEE, 2015, pp. 363–368.

[13] Y. Hu, W. Wendong, X. Gong, X. Que, and C. Shiduan, “Reliability-aware controller placement for software-defined networks,” in *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on*. IEEE, 2013, pp. 672–675.

[14] T. Erlebach, A. Hall, L. Moonen, A. Panconesi, F. Spieksma, and D. Vukadinović, “Robustness of the internet at the topology and routing level,” in *Dependable Systems: Software, Computing, Networks*. Springer, 2006, pp. 260–274.

[15] Y. Zhang, N. Beheshti, and M. Tatipamula, “On resilience of split-architecture networks,” in *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE*. IEEE, 2011, pp. 1–6.

[16] 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.

[17] A. Sallahi and M. St-Hilaire, “Optimal model for the controller placement problem in software defined networks,” *IEEE communications letters*, vol. 19, no. 1, pp. 30–33, 2015.

[18] J. Liao, H. Sun, J. Wang, Q. Qi, K. Li, and T. Li, “Density cluster based approach for controller placement problem in large-scale software defined networkings,” *Computer Networks*, vol. 112, pp. 24–35, 2017.