UDC 004.72

DOLYNNYI O.,
NIKOLSKIY S.,
KULAKOV Y.

THE METHOD OF SDN CLUSTERING FOR CONTROLLER LOAD BALANCING

The paper proposes the method of SDN clustering using connections density and controller load distribution that solves the problem of controller load balancing. Clustering efficiency criteria have been considered, including fault tolerance, controller-to-switch and intercontroller latency and network limitations. Review of the key clustering methods has been performed, and the base algorithm for modification has been chosen. Density-based controller placement algorithm is modified to solve the problem of multicontroller placement. Metric of the node boundary index is introduced to advance the efficiency of proposed algorithm. A software implementation of the developed algorithm has been created, and its performance has been tested. The algorithm's modelling results have been compared with those of the other algorithms using the metrics of distribution of service traffic propagation delay and computational speed in relation to network topology size.

Keywords: SDN clustering, network, connections density, controller placement

1. Introduction

The problem of controller placement is a critical part of implementing the Software Defined Networks (SDN) technology. The concept of SDN, which has emerged in recent years, is increasingly becoming a vital technology in the development of large-scale datacenter networks. The essence of the SDN concept is to separate the control layer from the data layer, which simplifies network management, improves network flexibility and scalability.

In large networks, the good placement of the controllers makes the best use of the existing structure of links between the network nodes. One controller is not enough to control all the nodes in a large SDN, as its performance is limited, and the signal propagation delays between the controller and all nodes in the network is very large; moreover, a single controller on the network cannot meet fault tolerance requirements. Therefore, it is logical to place many controllers at different points in the network, which together control the entire data layer. The critical question that emerges is how to cluster the network and where it will be most efficient to place the controllers. It is known as the controller placement problem.

2. Clustering efficiency factors

The problem of controller placement is defined as a multifactor complexity optimization problem [1] [2], in which the position of a controller should be decided upon with the following characteristics taken into consideration:

(1) Fault tolerance. The switches are dependent on their controllers providing them with instructions on how to forward packets. If the connection between a switch and its controller is disrupted, the switch becomes unable to forward packets and, thus, becomes essentially nonfunctional in the network [3]. Possible reactions to such failures must be taken into account when designing a network.

(2) Controller-to-switch latency. Controllers send instructions to switches, according to which the latter forward packets. Consequently, the transmission delays of those control signals influence the latency metric of the entire network. The network-wide latency cannot be minimized without minimizing local controller-to-switch latencies [4].

(3) Inter-controller communication. Each controller has a number of switches to which it sends commands. To exchange messages at a network-wide level, i.e. between remote switches managed by

different controllers, controller-to-controller communication must be implemented [5]. So the total communication delay between remote switches in the network depends on the delay of controller-to-controller information exchange.

(4) The density of controller placement. There is no predefined number of controllers that would ensure the optimal performance of an SDN network. The desired number of controllers is especially hard to decide on in large networks with advanced structure. It has to be determined using assorted methods, some of them complex and time-consuming, as they consider a large number of placement variants [6].

(5) Constraints set by controller capacity. Hardware limitations of a controller make it possible to manage only a certain number of switches without getting overloaded. Controller overload, i.e. its processing queue becoming overflown by the tasks, is a scenario that has a negative effect on the performance of the SDN as a whole [7].

3. Related works.

Some works have proposed heuristic algorithms to solve the controller placement problem and defined this as multi-objective combinatory optimization task. However, for a large-scale network such algorithms demonstrate an unreasonably high solution time [3].

On the other hand, there exists an approach in which controller placement is determined by certain defined metrics.

K-means method, developed by B.Heller in 2012, focuses on minimizing the communication latency in a network [5]. This method doesn't take into account the reliability aspect, as well as controller load when solving the clustering problem.

Another popular method is the density based controller placement (DBCP) introduced by J Liao in 2016, which uses a nodes clustering algorithm based on density distribution [6]. These nodes have higher quantity of connectivity inside the cluster and lower quantity of connectivity with nodes from other networks, thus there is one and only one controller in each subnetwork. To minimize costs for intercontoller communication, the proposed algorithm advises to place them at the closest possible distance. However, every controller's placement is chosen independently in every cluster as well as sequential order of cluster choosing for nodes, therefore controller position and cluster's load is not optimal as far as all clusters are concerned.

Pareto-Optimal Resilient Controller Placement method (POCO), introduced by D. Hock in 2013, aims to guarantee the best possible controller placement using Pareto distribution of network metrics for different nodes. This algorithm also considers such parameters as inter-controller communication and load balancing for cluster controllers, but has the drawback of low computational speed due to a large amount of calculations.

Focusing on equal load balancing goal and considering the positive and negative sides of each method, this study uses DBCP method as the base algorithm for the following modification.

4. Aim and objectives

In this study, we propose a method of network clustering, aimed at improving the efficiency and equal load balancing of an SDN network by analyzing the density distribution of links. This requirement is substantial to consider due to existence of physical restrictions for cluster size, it cannot have extremely high value due to central controller load limits.

In order to achieve the aim of our study, the following objectives are proposed:

1) to identify the factors that affect the efficiency of the clustering process;

2) to analyse the clustering methods that can be used in software-configured networks;

3) to develop a method of SDN clustering, taking into account the distribution of link density and data flows;

4) to model and analyse the effectiveness of the proposed method of SDN clustering.

5. The proposed method

5.1. General principles of clustering method construction

In this section, the basic steps for clustering method construction are observed.

Let us assume that the network topology $G(S, L)$consists of a set of routers Sand a set of bidirectional connections between themL. Note that if there is a direct connection between the nodes of the network, the length of the connection is equal to one, otherwise it is equal to zero.

Unlike the other methods, this algorithm analyzes the network topology, which is then divided into subnets. Nodes, that have the most common connectivity, have a relatively larger number of connections inside the subnet, and so have fewer connections to other subnets. Therefore, they can be considered basic nodes for separate subnets. After following the described method during the aggregation of nodes, the constructed subnet has the feature of higher fault tolerance. Delay in the subnet is also reduced due to fine-grained clustering.

In general, the clustering method consists of three main steps:

1) to analyze the network topology for the distribution of connection density between routers;

2) to distribute network routers in clusters according to the found values of density, distance from the current node to the node with a higher value of density, and the node boundary index;

3) to solve the problem of controller placement for each subnet according to a given criterion (in this case - the metrics of the average distance to other cluster nodes).

5.2. Network graph analysis of deviations in the density distribution

In the algorithm, the division of the whole network occurs due to cluster allocation. Two values are calculated for each routers$_i$: local densityρ_i and distance to a node with a higher local density value δ_i (node priority for the clustering task). These values depend only on the connections between the routers.

The local density ρ_i for each router is calculated by the formula (1), which uses the coupling factor k_{cl} and the distance between the routers d_{ij}:

$$\rho_i = \sum_j k_{cl}(d_{ij} - d_c) \tag{1}$$

The maximum distance value to a nearby node d_c is the limit of the distance that the routers assigned to the current node can have. The coupling factor k_{cl} is taken as one in the case of a positive difference between the reference distance limit and the distance between the routers, and zero otherwise. For distributed networks with more than 50 nodes, it is effective to set d_c as 30% of the network diameter.

The distance to the node with higher value of local density δ_i is calculated as the shortest distance to the node with higher value of local density:

$$\rho_i = \sum_j k_{cl}(d_{ij} - d_c) \tag{2}$$

For nodes with the highest local density in a cluster, the distance to the node with the even higher value of local density is taken as the maximum value; such nodes will be selected first as cluster centers.

As a result, we get the following pseudocode of the network graph analysis of deviations in the density distribution:

```
int analyzeDensity(G = (S, L), dc) {
   int k = 0;
   for (s: S) {
      ro[s] = getNumberOfNodesWithinDistance(s, dc, G);
   }
   for (s: S) {
```

```
        delta[s] = minDistanceToHigherDensityNode(s, ro, G);
        delta_Avg = sum(delta[0], ..., delta[S]) / S;
        if delta[s] > delta_Avg {
            k++;
        }
    }
    return k;
}
```

5.3. Algorithm of network graph clustering

To ensure that the even load on the cluster controllers is reached, it is necessary to introduce the node boundary index (distance to the cluster boundary), in addition to following the formulas 1 and 2.

Let us assume that the load of each router equals $l(s)$. Then the maximum possible controller load $L(\Theta)$ should not be less than the sum of the traffic loads of each cluster node:

$$L(\Theta) \geq \sum_{s \in S(\Theta)} l(s) \qquad 3)$$

Let us consider the case of a cluster already containing enough nodes to control, so it cannot be expanded with new nodes; so it is needed to join those nodes to other clusters or create a new cluster. Maintaining the compliance with the maximum load requirement on the cluster controller, it is necessary to distinguish between nodes that are closer to the center of the cluster and nodes that are closer to the boundaries of the cluster. The latter nodes can be considered as candidates for joining expandable clusters.

Let us denote total local density of current node as the sum of local densities for such nodes that have higher density value than current node:

$$td_i = \sum_{j}^{s_j \in UL(s_i)} \rho_j \qquad (4)$$

To find the nodes in the cluster that are on the border, we calculate a new value for each router, namely the node boundary index σ_i. The node boundary index is a value that indicates the uncertainty of the node membership in the current cluster. If the local density of the neighboring router, which is greater or equal, differs by no more than the limit value of the density difference, then this router can be considered as the border node of the cluster.

To find the node boundary index σ_i, we use the formulas from the theory of information entropy:

$$\sigma_i = \sum_{j}^{s_j \in UL(s_i)} \frac{\rho_j}{td_i} \log_{|UL(s_i)|} \frac{\rho_j}{td_i} \qquad (5)$$

In the modified algorithm, the selection of a node with a smaller value of the node boundary index has a higher priority during the process of assigning nodes to the cluster. The router is assigned to the same cluster as the nearest neighboring router with a higher local density value, unless the total load on the cluster routers will exceed the maximum possible load on the controller. If there is no cluster to which the current node can be attached, it becomes the center of the new cluster.

As a result, we get the following procedure pseudocode, that performs network graph clustering considering limitations for maximum controller load:

```
void clusteringWithLoad(Graph G = (S, L), ro, delta, controllerCapacity) {
    for (s: S) {
        delta[s] = getBorderline(s, G);
```

```
    }
    S.sortBy(delta);
    for (s: S) {
        n[s] = cluster(s);
    }
   for (s: S) {

        ul = getNeighbors(s, (ro > ro[s]));
        ul.descendingSortBy(ro);
        for (s1: ul) {
            si = union(n[s], n[s1]); //si – all connections, implicit to n[s] or n[s1]
            if (controllerCapacity >= sum(l(si))) {
                n[s1].add(s);
                break;
            }
        }
    }
}
```

6. The example of the developed clustering method

We will demonstrate the work of the developed algorithm by example.

Let the traffic flows on each router be the same and equal to 1. Then the network clustering task based on the controller load limitations and traffic flows can be reduced to the network clustering task regarding the total number of routers that can be controlled by the controller.

The network topology graph is presented in Fig. 1a. Let us assume there is a limit to the total number of routers that can be controlled by the controller, and this number is 6. We use formulas 1, 2, and 5 in our calculations.

The simulation results of the algorithm for the mentioned network graph are shown in Fig. 1b and Table 1.

34 routers are divided into 7 different clusters, where the routers of a separate cluster are marked with a separate color. It should be noted that the router 22 is allocated to a separate cluster, because all neighboring clusters are already full and cannot be expanded, so the router 22 cannot be connected to any of them.

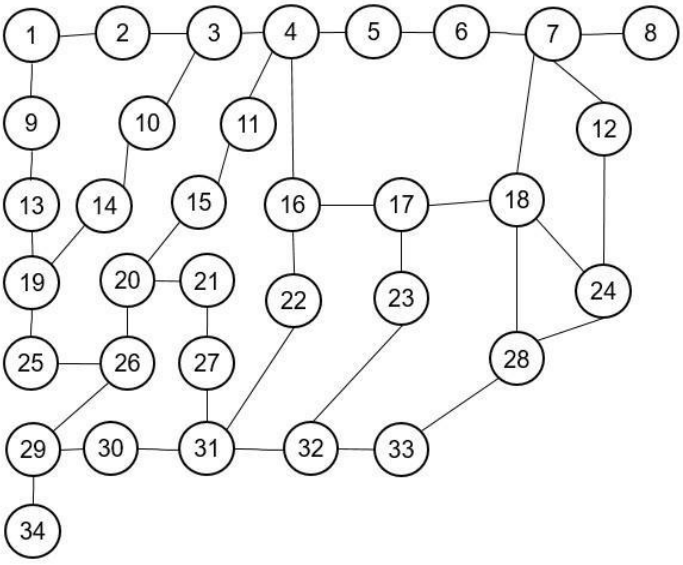

Fig. 1a. Original topology

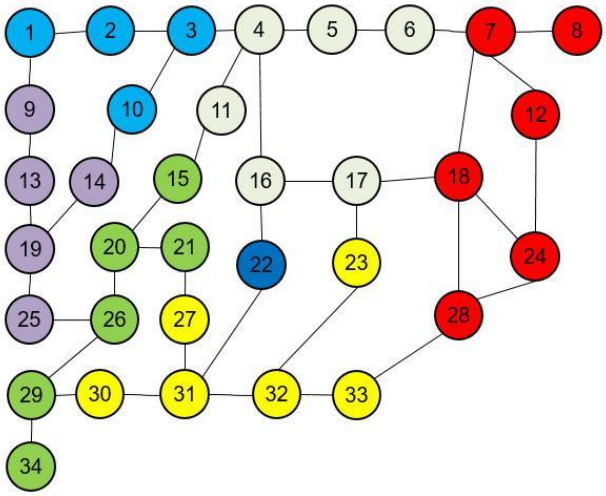

Fig. 1b. Marked topology

7. Results and discussion

The advantages of the developed clustering algorithm include the following aspects. First of all, the proposed method selects the optimal number of controllers for a particular network. Secondly, clustering based on topology link density reduces the likelihood of controllers placement in the nodes with a high risk of a failure. Finally, the problem of multiple controllers placement in the network is reduced to the problem of placing one controller in subnetwork, which reduces the time complexity of solving the problem.

Table 1
The calculation of modelling results

Node id	ρ	δ	σ	Node id	ρ	δ	σ	Node id	ρ	δ	σ
1	4	1	0.91	13	5	1	0.0	24	6	1	0.98
2	8	1	0.0	14	5	1	0.99	25	6	1	0.98
3	8	1	0.99	15	5	1	0.99	26	8	3	0.0
4	10	3	0.0	16	9	1	0.0	27	6	1	0.0
5	6	1	0.95	17	8	1	1.0	28	7	1	0.0
6	6	1	0.98	18	9	3	0.0	29	6	1	0.99
7	8	1	0.0	19	6	2	0.0	30	7	1	0.0
8	4	1	0.0	20	7	1	0.0	31	9	3	0.0
9	4	1	0.99	21	5	1	0.99	32	8	1	0.0
10	5	1	0.96	22	7	1	1.0	33	6	1	0.99
11	6	1	0.0	23	6	1	1.0	34	3	1	0.0
12	6	1	0.98								

During the testing of the obtained algorithm, several metrics were measured for the analysis of the model. A comparison of the developed algorithm with K-means and POCO algorithms was performed. The analysis used a graph generator with a step of 25 nodes, all traffic connections were taken as 1.

Fig. 2 compares the execution time of algorithms. With a relatively small number of nodes (up to 100), there is no big difference in computational speed between algorithms, but the difference begins to grow with higher order polynomial complexity between K-means algorithm and POCO and the developed algorithm.

Fig. 3 presents the results of the analysis of the delay in the propagation of service traffic depending on the number of topology nodes. The considered algorithms have similar delay indicators, differing on different values of the number of nodes, but according to the total efficiency index, the

developed method has the best metrics.

The developed method could be applicable for the large-scale networks due to quite good results of modelling metrics. However, it is necessary to note the difficulty of algorithm integration into the existing hardware infrastracture.

Further development of the algorithm could take into consideration the aspects of reliability and minimization of cluster amount. These aspects were not considered in detail in the present paper, as our aim was to guarantee the even controller load. It should be noted there is a possibility of finding other flaws in the presented method, depending on the metrics used for comparison of algorithms.

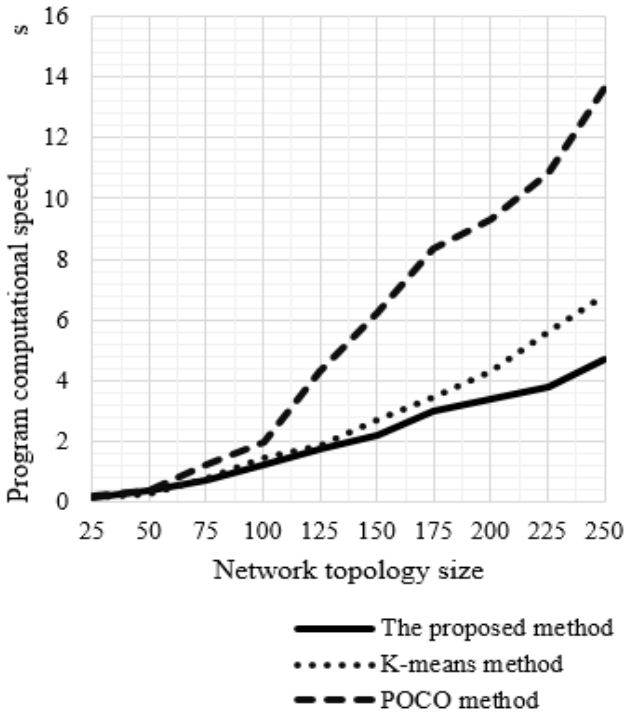

Fig. 2. Comparative graph of computational speed in relation to network topology size

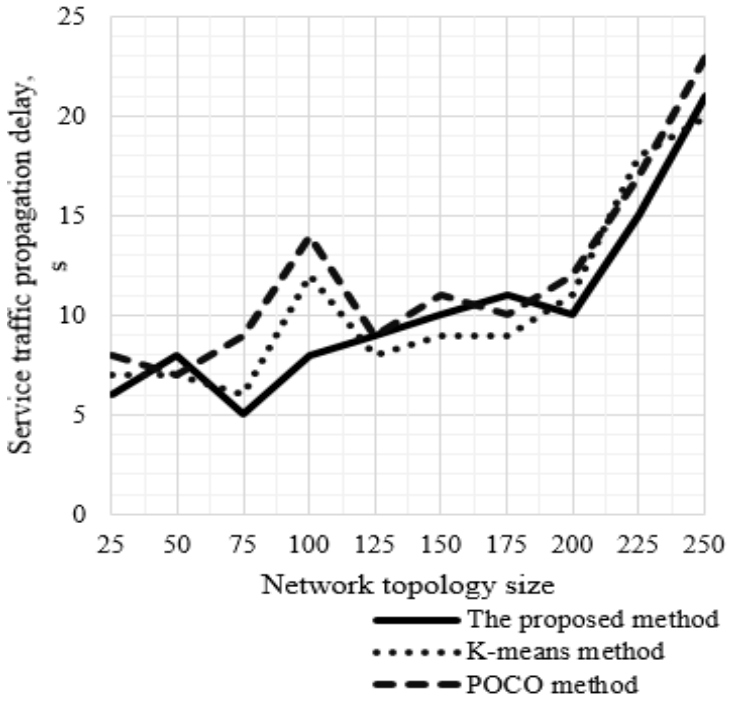

Fig. 3. Comparative graph of distribution of service traffic propagation delay in relation to network topology size

8. Conclusions

The paper considers a network clustering algorithm based on the connections density distribution and the maximum load on the controllers. The example demonstrated the procedures of the algorithm.

As a result of testing on different network topologies and analyzing the obtained results, the developed method of clustering demonstrates high efficiency in comparison with K-means and POCO algorithms regarding the existing metrics of service traffic delay and speed.

From the practical point of view, the results obtained in this paper allow increasing the efficiency of SDN networks management, especially large-scale networks, for which it is important to balance the load on the controllers.

Further development of the algorithm could take into consideration the aspects of reliability and minimization of cluster amount.

References

1. Scott-Hayward S. Are we ready for SDN? Implementation challenges for software-defined networks? / Scott-Hayward S., Chouhan P.K., Fraser B., Lake D., Finnegan J., Viljoen N., Miller M., Rao N. // Communications Magazine, IEEE . – 2013. – № 51(7) – pp. 36–43.
2. Hakiri, A. Software-defined networking: Challenges and research opportunities for future internet. // Computer Networks. – 2014. – № 75. – 453-471 pp.
3. Hu, F. A survey on software-defined network and openflow: From concept to implementation / Hu F., Hao Q., Bao K. // IEEE Communications Surveys & Tutorials. – 2014. – № 16. – 2181-2206ppc.
4. McKeown, N. OpenFlow: Enabling Innovation in Campus Networks / McKeown N., Anderson T., Balakrishnan H., Parulkar G., Peterson L., Rexford J., Shenker S., Turner J. // SIGCOMM CCR – 2008. – № 38(2).
5. Heller, B. The Controller Placement Problem / Heller B., Sherwood R., McKeown N. // HotSDN. – 2012.
6. Liao, J. Density cluster based approach for controller placement problem in large-scale software defined networkings / Jianxin Liao, Haifeng Sun, Jingyu Wang, Qi Qi, Kai Li, Tonghong Li // Computer Networks. – 2017. – № 112. – 24–35 pp.
7. Hock, D. Pareto-Optimal Resilient Controller Placement in SDN-based Core Networks / Hock D., Hartmann M., Gebert S., Jarschel P., Zinner T., Tran-Gia P. // 25th International Teletraffic Congress (ITC) – 2013.