Network Partitioning Problem to Reduce Shared Information in OpenFlow Networks with Multiple Controllers

Hidenobu Aoki, Junichi Nagano, and Norihiko Shinomiya
Graduate School of Engineering
Soka University
Tokyo, Japan
Email: shinomi@ieee.org

***Abstract*—This paper proposes the layered control plane of OpenFlow networks with multiple controllers. Our method logically partitions an OpenFlow network and assigns controllers to the partitioned networks as their administrative domains. In addition, this paper focuses on the relationship between the network partitioning and the amount of global network information shared among controllers. Then, this paper handles the issue as a mathematical problem based on graph clustering and analyzes effective network partitioning methods in reducing the amount of global network information.**

Keywords*—*Software-Defined Networking; OpenFlow; multiple controllers; layered control plane; graph clustering.

I. INTRODUCTION

Software-Defined Networking (SDN) has been emerging as a new networking paradigm. The fundamental idea of SDN is to achieve programmable networking by separating the control and the data planes in an individual network device, such as a switch and router [1]. As one of the standard protocols between those separated planes, OpenFlow has been developed and widely used. In an OpenFlow network, a controller is in charge of generating data forwarding rules. In contrast, OpenFlow switches distributed in the data plane are responsible only for forwarding data according to the rules. This centralized architecture where a controller manages OpenFlow switches enables network operators to dynamically configure switches and to flexibly manage their networks [2].

Although it was assumed that a single controller dominates an entire network at the beginning, the concerns of scalability and reliability has been raised [3]. As the size of the network grows, the amount of data traffic, such as flow requests to a controller would increase [4] and [5]. Furthermore, the network operation with a single controller could take a risk of whole network breakdown if a failure occurs on the controller. As a result, to deploy multiple controllers has been considered.

In an OpenFlow network managed by multiple controllers, it could be scalable approach to logically divide the network into sub-networks as administrative domains of controllers so as to handle flow requests faster and reduce computational load on each controller. Then, a collaborative framework that enables controllers to effectively communicate each other has been required and drawn attention as a SDN-related research topic [6]. This paper proposes the layered architecture of control plane and classifies the roles of controllers. In particular, this paper focuses on how to decide the administrative domains of controllers, which has not discussed in any relevant work. The organization of this paper is as follows: In Section II, the related work of distributed controllers is addressed. Section III presents the layered control plane with its definitions and functions. In addition, the issue between the network partitioning and the amount of global network information shared among controllers is described. Section IV presents the graph definitions of the layered control plane and formulates the problem. As solutions of the network partitioning, Section V describes clustering algorithms. Section VI shows the simulation results. Finally, we conclude this paper with future work in Section VII.

II. RELATED WORK

Regarding to the deployment of multiple controllers, how to disseminate network state information over multiple controllers is highlighted as an important issue [4]. HyperFlow [7] realizes the synchronization of the network information among multiple controllers by utilizing a distributed file system called WheelFS WheelFS employs publish-subscriber patterns and contains all network information so that controllers can access to sufficient information for the local control of switches. In addition, ONOS [8] maintains a global network view with the abstraction of data plane network and shares topology information across ONOS servers by adopting distributed Titan graph database and Cassandra key-value store.

In order to reduce the load on controllers in sharing network information, a concept of layered control plane has been proposed. Onix [9] logically partitions a network, and controllers are assigned to partitioned networks as their control domains. Then, each partitioned network is contracted as a logical node and used as a unit for sharing network information among controllers. This enables a controller to communicate with other controllers without knowing specific network topology and states of other partitioned networks. In this way, the reduction of the amount of network information possessed by a single controller can be achieved.

Moreover, Kandoo [10] provides a layered control method for OpenFlow networks consisting of the root controller and some local controllers. The root controller manages all local controllers and is responsible only for the events which requires information over the whole network. On the other hand,

 ISBN: 978-1-61208-398-8

the local controller deals with the local events like requesting flow setups and collecting the statistics for a network with governing a group of switches and links between them. This layered control defines the scope of operations for processing different requests efficiently, which could offload the burden of the root controller.

As stated above, notable ideas of the layered control methods have been proposed in those related work. Nevertheless, they have not mentioned how to logically partition a network to decide administrative domains of controllers against switches although it could affect the amount of network information shared among controllers. In order to consider the issue, this paper focuses on the network partitioning and examines its solutions based on the concept of graph clustering.

III. Layered control plane

This section presents the architecture of the layered control plane and describes the issue between the network partitioning and the amount of global network information shared among controllers.

A. Definitions

In an OpenFlow network with multiple controllers, the network can be logically partitioned as administrative domains of controllers. In each domain, a controller is responsible for the following two roles: (1) management of switches in own domain and (2) federation of a whole network by communicating with other controllers. In accordance with those roles, the control plane can be layered in two tiers: the local tier and the federation tier in charge of (1) and (2), respectively.

B. Network Topology in Local and Federation Tiers

In the local tier, a local control function, called a local controller, describes the network topology of each administrative domain as a local graph. Moreover, the local graph is contracted to a single node which is used as a unit of communication with other controllers. In the federation tier, a global control function, referred to a federator, gathers the contracted nodes from all controllers and unifies them with edges between local graphs to form a federation graph describing global network topology. Because of this topology contraction, it is assumed to reduce the amount of global network information shared among controllers through the distributed database. Figure 1 illustrates an example of the layered control plane. In Figure 1, there are two domains described as local graphs 1 and 2. In the federation tier, on the other hand, the federation graph has two contracted nodes, and three edges correspond to the edges between the local graphs.

C. Network Topology Acquisition

In an OpenFlow network, the network topology is obtained by a combination of LLDP (Link Layer Discovery Protocol) packets, packet_in and packet_out messages as follows:

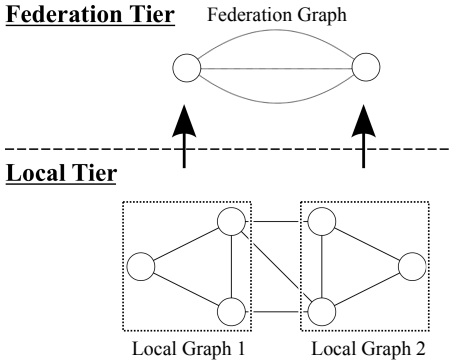


Fig. 1. An example of topology contraction in layered control plane.

1) Receiving a new packet, a switch forwards it to the controller as a packet_in message if it does not match any flow entries installed in the switch.
2) When the controller receives the packet_in message, the controller installs a flow entry to the switch if there exists the destination of the packet in its administrative domain; otherwise, in order to find the destination, the controller sends packet_out messages to direct switches to forward LLDP packets from its ports.
3) The switches send LLDP packets from their ports to adjacency switches. If the packet is received by a switch in the same domain, the packet_in message is sent to the controller; otherwise, it is forwarded to another controller.
4) When a controller receives the packet_in message, it inspects the message and detects the connectivity of switches.
5) If a controller detects the connectivity between switches within its domain, it updates own local database holding local network information; otherwise, as the controller detects an inter-domain link, it updates the distributed database keeping global network information.

Through LLDP flooding, controllers discover the destination of packets and install the corresponding flow entries in each switch in its administrative domain. Note that since LLDP packets will flood the entire network until the destination is found, this overhead will increase as the size of network becomes larger. Moreover, on the occasion of the inter-domain link detection, controllers need to update the distributed database, which may degrade their throughput. Furthermore, if there are many inter-domain links, controllers may need to access frequently to the distributed database to obtain and update the global network topology and compute routing paths. Therefore, we assume that the number of inter-domain links could affect the network performances and the load on controllers.

D. Network Partitioning in Local Tier

Considering the topology contraction in the layered control plane, it would be noteworthy that the number of inter-domain links, that is, the number of edges in a federation graph depends on how to partition a network; in other words,

 ISBN: 978-1-61208-398-8

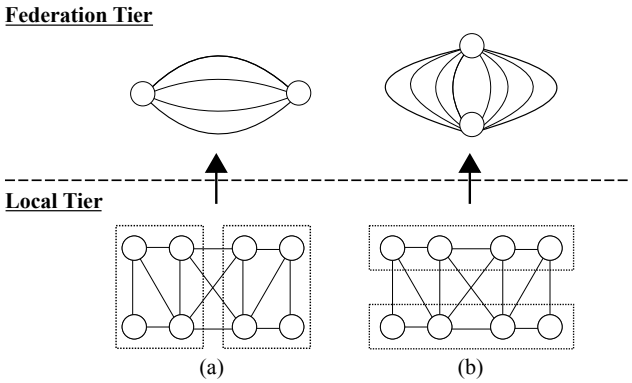


Fig. 2. An example of network partitioning in different ways.

how to decide administrative domains of controllers affects the amount of global topology information shared among controllers. Figure 2 shows an example of network partitioning in different ways. As seen in Figure 2, there are four edges in federation graph (a) while federation graph (b) has eight edges. This is because of different ways of network partitioning between (a) and (b). Therefore, it can be expected to further reduce the amount of global topology information by focusing on the network partitioning. This paper defines the problem to decide administrative domains of controllers as Network Partitioning Problem (NPP).

IV. Model Definitions and Problem Formulation

This section presents graph definitions treated in the layered control plane and formulates NPP.

A. Graph Definitions

Network topology can be described as a graph $G = (V, E)$: a set of vertices V represents network devices, such as routers and switches, and a set of edges E denotes links that connect those devices. In the local tier, local graphs are defined as

$$G_1^l = (V_1^l, E_1^l), \ldots, G_i^l = (V_i^l, E_i^l). \quad (1)$$

Note that a node can not belong to multiple local graphs. In the federation tier, on the other hand, a federation graph is denoted as

$$G^f = (V^f, E^f) \quad (2)$$

where G^f contains all local graphs as contracted nodes in V^f and edges between local graphs in E^f.

B. Problem Formulation as Graph Clustering

In the field of graph theory, graph clustering is defined as a task of grouping nodes in a graph into subsets called clusters in some predefined sense [11]. This paper applies the concept of graph clustering for network partitioning. Let a local graph G_i^l in (1) be a cluster, and a set of local graphs $\mathbf{G^l}$ is defined as a clustering: $\mathbf{G^l} = \{G_1^l, ..., G_i^l\}$. In general, it is regarded as a desirable clustering where there are many edges within each cluster called intra-cluster edges and relatively few edges between clusters referred to inter-cluster edges [12]. Considering the network topology treated in the layered control plane, intra-cluster edges correspond to edges in E_i^l, and inter-cluster edges are equivalent to edges in E^f. Consequently, we could say that the general criterion for the desirable clustering is applicable for the objective of NPP, that is, to partition a network such that the number of edges in a federation graph is reduced. Therefore, the objective function of NPP is defined as finding a clustering $\mathbf{G^l}$ such that

$$\text{Minimize } |E^f| \quad (3)$$

where an upper bound of the number of nodes in a cluster q is satisfied.

V. Clustering Algorithms for Network Partitioning

In this section, three clustering algorithms based on different measures are presented as solutions of NPP.

A. Minimum Cut Clustering

In graph theory, a minimum cut is defined as a set of the smallest number of edges which divide a graph into two disjoint subgraphs [13]. Based on the concept, we constructed Minimum cut clustering which separates a graph by minimum cut and regards the yielded subgraphs as clusters. It recursively conducts the separation process until the number of nodes in each cluster does not exceed the upper bound of the number of nodes in a cluster q as described in Algorithm 1.

Algorithm 1 Minimum Cut Clustering.

Require: $G = (V, E)$ and q: constraint of # of nodes
main
Clustering $\mathbf{G^l} \leftarrow \phi$
MinimumCutClustering(G, q)
Return $\mathbf{G^l}$
end main
function MinimumCutClustering(G, q)
(G_i^l, G_j^l) : *generate clusters based on minimum cut*
if # *of nodes in* $G_i^l \leq q$ **then**
  Add G_i^l to $\mathbf{G^l}$
else
  MinimumCutClustering(G_i^l, q)
end if
if # *of nodes in* $G_j^l \leq q$ **then**
  Add G_j^l to $\mathbf{G^l}$
else
  MinimumCutClustering(G_j^l, q)
end if
Return $\mathbf{G^l}$
end function

B. Conductance Clustering

As one of clustering indices, conductance has been defined, which compares the number of inter-cluster edges and that of intra-cluster edges yielded by a clustering [13]. By denoting a

set of all edges that have their origin in G_i^l and their destination in G_j^l as $E(G_i^l, G_j^l)$, the conductance of a cluster is defined as

$$\Phi(G_i^l) = \frac{|E(G_i^l, \mathbf{G^l} \setminus G_i^l)|}{\min(\sum_{v \in G_i^l} deg(v), \sum_{v \in \mathbf{G^l} \setminus G_i^l} deg(v))}. \quad (4)$$

Since finding a clustering with minimum conductance is known as NP-hard [12], we created Conductance clustering that chooses nodes one by one based on the conductance value as shown in Algorithm 2. The algorithm begins with a random node and assigns it to a cluster. Then, one of neighbor nodes of the node, which the cluster obtains the best conductance value, is chosen and assigned to the cluster. As this process, it expands the cluster by recursively choosing a neighbor node of the nodes in the cluster. If the number of nodes in the cluster reaches to the upper bound of the number of nodes in a cluster q, then it starts again to create a new cluster with a random node which has not belonged to any clusters.

Algorithm 2 Conductance Clustering.

Require: $G = (V, E)$ and q: constraint of # of nodes
Clustering $\mathbf{G^l} \leftarrow \phi$
$V' \leftarrow$ a list of nodes in a graph G
while $V' \neq \phi$ **do**
  $G_i^l \leftarrow \phi$
  Choose a node v_r randomly from V'
  Add v_r to G_i^l and remove v_r from V'
  while # *of nodes in* $G_i^l \leq q$ **do**
    for every neighbor node v_n of $\forall\ v \in G_i^l$ **do**
      if v_n is in V' **then**
        G_c^l = G_i^l
        Add v_n to G_c^l
        Calculate $\Phi(G_c^l)$
      end if
    end for
    Choose G_c^l with minimum conductance $\Phi(G_c^l)$
    Add v_n in the G_c^l to G_i^l
    Remove v_n from V'
  end while
  Add G_i^l to $\mathbf{G^l}$
end while
Return $\mathbf{G^l}$

C. Distance-k Cliques Clustering

In addition to minimum cut and conductance, distance is also a general clustering measure. We apply one of distance-based clustering algorithms called Distance-k cliques clustering [14]. Distance-k cliques clustering measures the strength of a relationship between two nodes in a graph in terms of the shortest path length between two nodes and generates clusters such that every pair of nodes is connected by a path of length at most k. As shown in Algorithm 3, Distance-k cliques clustering algorithm obtains an initial clustering at first. In an initial clustering, clusters are generated by choosing a node with the highest degree and its neighbors. Based on the initial clustering, the next step is to combine the clusters while both constraints of the number of nodes in a cluster q and the diameter of a cluster k are satisfied. Note that in order to fit the simulation setting of this paper, we added the constraint of the number of nodes in a cluster q which is not considered in the original algorithm defined in [14].

Algorithm 3 Distance-k Cliques Clustering.

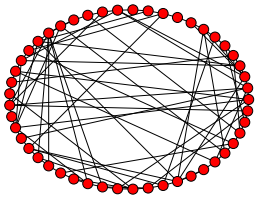
Require: $G = (V, E)$ and q: constraint of # of nodes
STEP1 : Obtain an initial clustering
Clustering $\mathbf{G^l} \leftarrow \phi$
$V' \leftarrow$ a list of nodes in a graph G
while $V' \neq \phi$ **do**
  $G_i^l \leftarrow \phi$
  Find the highest degree node v_{max} in V'
  while # *of nodes in a cluster* $\leq q$ **do**
    Add v_{max} to G_i^l and remove v_{max} from V'
    Add neighbor nodes of v_{max} to G_i^l and remove those nodes from V'
  end while
  Add G_i^l to $\mathbf{G^l}$
end while
STEP2 : Combine the clusters of the initial clustering
while True **do**
  Find u_{max}, a node connected with nodes in other clusters where the total # of nodes in adjacency clusters G_{adj}^l is the largest in $\mathbf{G^l}$.
  if # *of nodes in* $G_i^l(u_{max}) = q$ **then**
    Break
  end if
  for $\forall\ G_{adj}^l$ of $G_i^l(u_{max})$ **do**
    if # *of nodes in* $G_i^l(u_{max}) + G_{adj}^l < q$ **then**
      if $diameter(G_i^l(u_{max}) + G_{adj}^l) \leq k$ **then**
        Add nodes in G_{adj}^l to $G_i^l(u_{max})$
      end if
    end if
  end for
  Update $\mathbf{G^l}$ with $G_i^l(u_{max})$
end while
Return $\mathbf{G^l}$

VI. Simulation and Results

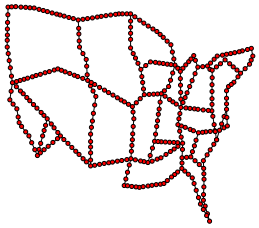
This section shows the simulation settings and results. We have developed a simulator in Python and NetworkX to evaluate three clustering algorithms in terms of minimizing $|E^f|$. In our simulation, those clustering algorithms are executed on a random graph called Newman Watts Strogatz (NWS) and two types of real network model, America and Japan models as shown in Figure 3. Note that a NWS graph is formed by connecting random pairs of nodes with a certain probability after creating a ring over n nodes [15]. The reason why we choose the random graph as well as real network models is to change the size of graph flexibly and examine the results. Therefore, we conduct two kinds of simulation for NWS graph: 1) varying the number of nodes and edges

 ISBN: 978-1-61208-398-8

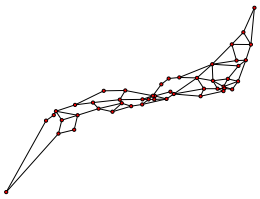
as described in Table I. 2) changing the value of the upper bound of the number of nodes in a cluster q. On the other hand, we test only 2) for the real network models since the size of the models is fixed as in Table I. Note that the distance constraint k of Distance-k cliques clustering is set to 10, and all simulations are executed 20 times.



(a) NWS [15].



(b) America Model [16].



(c) Japan Model [17].

Fig. 3. Examples of Network Model.

TABLE I. THE SIZE OF GRAPHS FOR SIMULATIONS.

	NWS			
The # of nodes	49	100	225	400
The # of edges	98	200	450	800
	America		Japan	
The # of nodes	365		48	
The # of edges	772		82	

A. Changing the Size of NWS Graph

Figure 4 shows $|E^f|$ on different number of nodes in the graphs where q is fixed to 15. The result indicates Conductance clustering demonstrates with the least $|E^f|$ on different size of the graph. In addition, as the size of the graph obtains larger, the difference of $|E^f|$ becomes considerable.

B. Varying the upper bound of nodes in a cluster

Figures 5 to 7 describe $|E^f|$ on different value of q from 10 to 25 while the number of nodes in a graph is fixed. Note that the number of nodes in a NWS graph is set to 225.

The negative slopes in those results indicate that as the value of q obtains greater, which means the larger number of nodes can be included in a cluster, it would yield the larger number of intra-cluster edges and the less number of inter-cluster edges. However, the results of Distance-k cliques clustering show horizontal slopes in Figures 5, 6, and a part of 7. This would be because of the constraint of distance k. Even if the number of nodes in a cluster does not reach to its limitation q, the distance constraint would be more influential because the algorithm does not allow a path length of any pairs of nodes to exceed k.

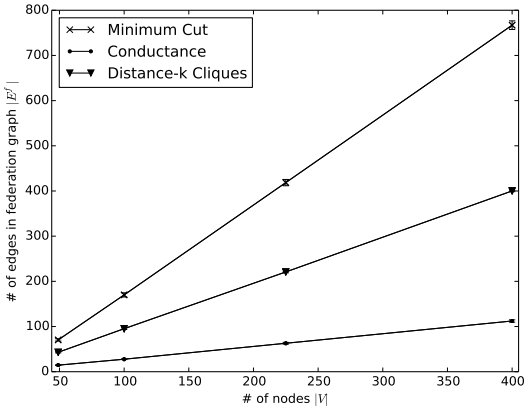


Fig. 4. $|E^f|$ on different size of NWS graph.

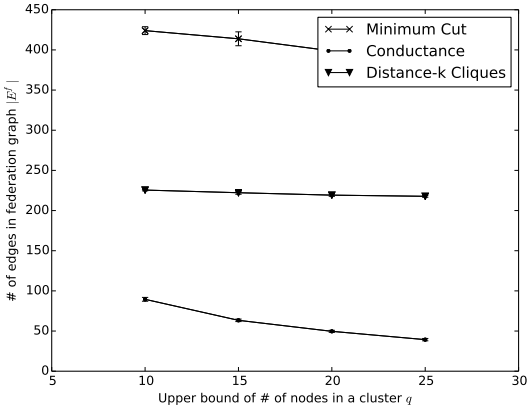


Fig. 5. $|E^f|$ on different value of q (NWS)

Furthermore, we can see that even when the value of q is varied, Conductance clustering reduces $|E^f|$ at most on any types of graph in our simulation (Figures 5 to 7). From the all results, we could say that conductance would be an important measure for effective network partitioning in reducing $|E^f|$, which indicates the reduction of the amount of shared topology information among controllers in OpenFlow networks. This is because Conductance clustering selects a node to assign a cluster based on the conductance value in (4) considering not only the number of cutting edges but also the density of yielded clusters. Therefore, it could have a tendency to yield a clustering with less $|E^f|$. On the other hand, Minimum cut clustering does not consider the quality of a clustering. It focuses on separating a graph based on the minimum number of cutting edges, which may results in separating only a small portion of a graph. As a result, the recursive graph separation by minimum cut could increase the number of clusters containing relatively small number of nodes, which ends up with a large number of inter-cluster edges as shown in our simulation results.

 ISBN: 978-1-61208-398-8

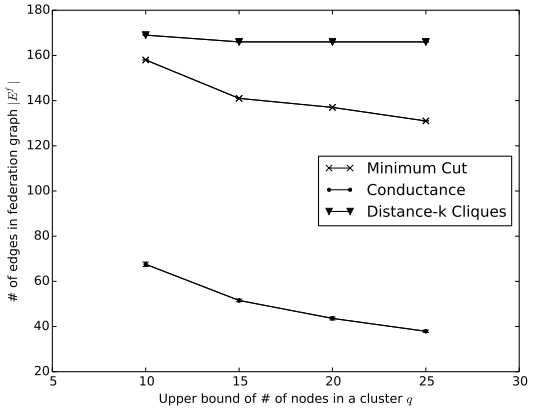


Fig. 6. $|E^f|$ on different value of q (America)

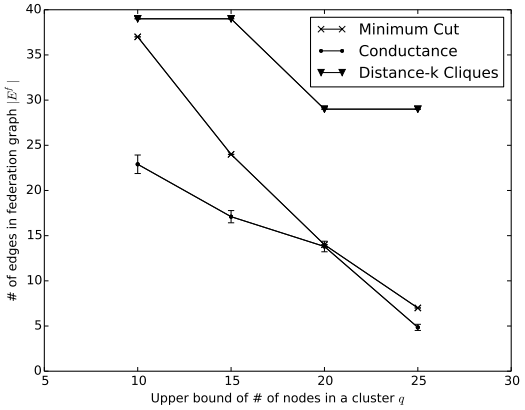


Fig. 7. $|E^f|$ on different value of q (Japan)

VII. Conclusion and Future Work

This paper proposed the layered control plane of OpenFlow networks and classified its functions in detail. In addition, the issue between the network partitioning and the amount of global network information in the layered control plane is addressed and converted to a mathematical problem as NPP. As a solution of NPP, this paper provided the network partitioning methods based on graph clustering and examined them on different network models. Our simulation results indicate that Conductance clustering performs the best in reducing the amount of network topology information shared among controller. Since our simulation is limited to theoretical approach, our future work will be an implementation of the layered control plane and examination of how reducing global network information by network partitioning can affect the load on controllers and the network performances under real network scenarios.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number 26330120.

References

[1] S. Sezer and et al., "Are we ready for SDN? implementation challenges for software-defined networks," Communications Magazine, IEEE, vol. 51, no. 7, July 2013, pp. 36–43.

[2] K. Suzuki and et al., "A survey on openflow technologies," IEICE Transactions on Communications, vol. 97, no. 2, 2014, pp. 375–386.

[3] S. Kuklinski and P. Chemouil, "Network management challenges in software-defined networks," IEICE Transactions on Communications, vol. 97, no. 1, 2014, pp. 2–9.

[4] S. Yeganeh, A. Tootoonchian, and Y. Ganjali, "On scalability of software-defined networking," Communications Magazine, IEEE, vol. 51, no. 2, February 2013, pp. 136–141.

[5] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood, "On controller performance in software-defined networks," in Proceedings of the 2nd USENIX conference on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services. USENIX Association, 2012, pp. 10–10.

[6] D. Marconett and S. Yoo, "Flowbroker: A software-defined network controller architecture for multi-domain brokering and reputation," Journal of Network and Systems Management, 2014, pp. 1–32.

[7] 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, ser. INM/WREN'10, 2010, pp. 3–3.

[8] P. B and et al., "Onos: towards an open, distributed sdn os," in Proceedings of the third workshop on Hot topics in software defined networking. ACM, 2014, pp. 1–6.

[9] T. Koponen and et al., "Onix: A distributed control platform for large-scale production networks." in OSDI, vol. 10, 2010, pp. 1–6.

[10] S. Hassas Yeganeh and Y. Ganjali, "Kandoo: a framework for efficient and scalable offloading of control applications," in Proceedings of the first workshop on Hot topics in software defined networks. ACM, 2012, pp. 19–24.

[11] R. Kannan, S. Vempala, and A. Vetta, "On clusterings: Good, bad and spectral," Journal of the ACM (JACM), vol. 51, no. 3, 2004, pp. 497–515.

[12] S. E. Schaeffer, "Graph clustering," Computer Science Review, vol. 1, no. 1, 2007, pp. 27–64.

[13] U. Brandes and T. Erlebach, "Network analysis." Springer Berlin Heidelberg, 2005.

[14] J. Edachery, A. Sen, and F. J. Brandenburg, "Graph clustering using distance-k cliques," in Graph drawing. Springer, 1999, pp. 98–106.

[15] M. E. Newman, D. J. Watts, and S. H. Strogatz, "Random graph models of social networks," Proceedings of the National Academy of Sciences, vol. 99, no. suppl 1, 2002, pp. 2566–2572.

[16] N. Shinomiya, T. Hoshida, Y. Akiyama, H. Nakashima, and T. Terahara, "Hybrid link/path-based design for translucent photonic network dimensioning," Journal of Lightwave Technology, vol. 25, no. 10, 2007, pp. 2931–2941.

[17] T. Sakano and et al., "A study on a photonic network model based on the regional characteristics of japan (in japanese," IEICE Technical Report, PN2013-01, Tech. Rep., 2013.

 ISBN: 978-1-61208-398-8