Controller Placement for Improving Resilience of Software-defined Networks

Minzhe Guo
Dept. of Electrical Engineering and Computing Systems
University of Cincinnati
Cincinnati, OH USA
guome@mail.uc.edu

Prabir Bhattacharya
Dept. of Electrical Engineering and Computing Systems
University of Cincinnati
Cincinnati, OH USA
bhattapr@ucmail.uc.edu

***Abstract*—Software-defined Network (SDN) is a new architectural framework for networking that decouples control plane from data plane and provides a programmatic interface to network control. SDN enables innovations of networking technologies for Internet and data centers today in order to ease the management of the highly dynamic and complex infrastructure, to support the fine-grained traffic engineering, and to better address the mobility of virtual machines, services, and end-points. However, SDN itself opens many unanswered questions regarding reliability, performance, and security. Controller placement is a critical problem in SDN that may affect many aspects of SDN. This work introduces the use of interdependence network analysis to study the controller placement for network resilience, designs a new resilience metric, and proposes a solution to improve resilience.**

Keywords—Controller Placement, Network Resilience, Software-defined Network

I. INTRODUCTION

Software-defined Network (SDN) is a recent architectural framework for networking, which decouples the network control plane from the data plane at physical topology and provides a programmatic interface to network control. SDN has the potential in simplifying the network management, improving the efficiency of network utilization, and enabling network innovations. A well-known implementation of SDN is *OpenFlow* [1], which arises from an academic research project and has now gained increasing adoptions in the industry, e.g., by Google, NEC, HP, etc. SDN enables innovations of networking technologies for data centers in order to ease the management of the highly dynamic and complex infrastructure, to support the fine-grained traffic engineering, and to better address the mobility of virtual machines, services, and end-points. However, the SDN itself open many unanswered questions regarding reliability, scalability, performance, and security.

Controller placement is one of the critical problems in SDN design that studies how to select the best k nodes in a SDN for placing controllers in order to maximize an objective function. Placement problems appear in many contexts and have received extensive studies in literature. They are usually variants of the facility location problem and are NP-hard problems in general [2]. In addition, most of the placement problems study the placement for improving performance, e.g., latency minimization; few investigate the placement for reliability and security.

The research on controller placement in SDN has begun very recently. Heller et al. [2] study the placement of controllers in SDN with the objective to minimize average or worst-case latency. Zhang et al. [3] initiate the study of controller placement for resilience and propose a min-cut based algorithm for network partition and controller placement. Hu et al. [4] study the placement problem for reliability and propose a greedy algorithm for efficient placement with good reliability. Beheshti and Zhang [5] study how to place a controller in a routing tree for fast failover. All these efforts show that intelligent controller placement contributes to better SDN design and performance.

This work focuses on the controller placement for network resilience improvement in SDN and makes the following contributions: 1) analyzing the impact of controller placement on SDN resilience from the perspective of interdependent networks; 2) defining a new resilience metric based on the cascading failure analysis on the interdependence graph; and 3) proposing a partition and selection approach to controller placement for improving the resilience.

II. AN INTERDEPENDENT NETWORK APPROACH TO NETWORK RESILIENCE ANALYSIS

The core concept of SDN is the separation of control plane from data plane and thus centralizes the control logic. This centralization can benefit network resilience by reducing the reliance on short-live control messages, which are more likely to be affected by network failures [6]. However, this centralization also introduces resilience threat to SDN itself, that is the proper functioning of a SDN depends on the proper functioning of two interdependent networks: the switch-switch network (*SS*) for data forwarding, and the controller-switch network (*CS*) for network control.

Based on this observation, we extend the model of interdependence graph in [7] to analyze the impact of controller placement on SDN network resilience. The interdependence graph enables the analysis of cascading failure in multiple networks due to node dependence relationships.

2165-4999/14 $31.00 © 2014 IEEE
DOI 10.1109/ICNDC.2013.15

For better illustration of how the interdependence graphs between *SS* and *CS* can be constructed and how they can be used for analyzing the network resilience, we present the concepts and analysis using an example network, with the German Backbone Network Topology from [8], as shown in Fig. 1(a).

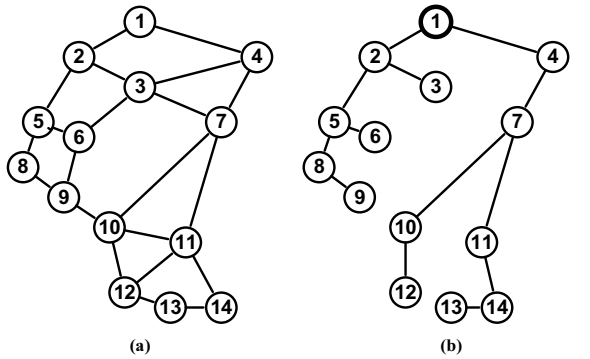

Fig. 1. Example Topology. *(a) German Backbone Network Topology [8]; (b) A shortest path routing tree for in-band controller-switch communication on the example topology when node 1 are the controllers and others are switches*.

A. SS-CS Interdependence Graph

Let $G=(V, E)$ denotes a SDN topology, in which the set of nodes V consists of the switches and controllers in SDN, and the set of edges E is comprised of the links between switches, the links between controllers and switches, and the links between controllers.

We define an interdependence graph between *SS* and *CS* of an SDN as a 5-tuple $H=(V_s, E_s, V_c, E_c, E_{cs})$:

- $V_s \subseteq V$ denotes the set of nodes in G that perform switch functions in *SS* network, such as originating, responding, and forwarding packets; for example the nodes under the label "*SS*" in Fig. 2;
- $E_s \subseteq E$ denotes the set of links that form the communication paths for V_s in *SS*; for example the edges under the label "*SS*" in Fig. 2;
- $V_c \subseteq V$ denotes the set of nodes in G that perform functions in *CS* network, such as the switches registering themselves to controllers and forwarding packets to the controllers when there are no processing rules for the packets, and the controllers generating packet processing rules and writing rules to switches; for example the nodes under the label "*CS*" in Fig. 2;
- $E_c \subseteq$ E denotes the set of links that form the communication paths for V_c in *CS network*; for example the edges under the label "*CS*" in Fig. 2;
- E_{cs} denotes the set of dependence relations between V_s and V_c. The dependence relations are directed. A dependence relation from node α in network A to node β in network B denotes that α depends on β, which means that the proper functioning of α in A depends on the proper functioning of β in B. In Fig. 2(a), node 1 in *SS* depends on node 1 in *CS*, node 3 in *SS* and node 3 in *CS* are mutual dependent, and there is no dependence relation between node 4 in *SS* and node 4 in *CS*.

Multiple dependence relations for a node can form an *AND-Dependence-Set*, an OR-*Dependence-Set*, or a hybrid set. Informally, a node with *AND-Dependence-Set* fails if any one of its dependent nodes fails; and a node with *OR-Dependence-Set* fails if all of its dependent nodes fail. In Fig. 2(b), the dependence relations between node 1 and 2 in *SS* and node 1 in *CS* form an *OR-Dependence-Set (represented using the "OR gate" symbol from Digital Logic)*, which means that the proper functioning of node 1 in *CS* depends on both the proper functioning of node 1 and node 2 in *SS*; on the other hand, the dependence relations between node 3 and 4 in *SS* and node 3 in *CS* form an *AND-Dependence-Set (represented using the "AND gate" symbol)*, which means that the proper functioning of node 3 in *CS* depends on either the proper functioning of node 3 in *SS* or the proper functioning of node 4 in *SS*.

By using the combinations of *AND-Dependence-Set* and *OR-Dependence-Set*, the interdependence graph can capture both simple interdependent relationship between the *SS* and *CS* networks, e.g., single path routing, and complex interdependence, e.g., multipath routing or service redundancy.

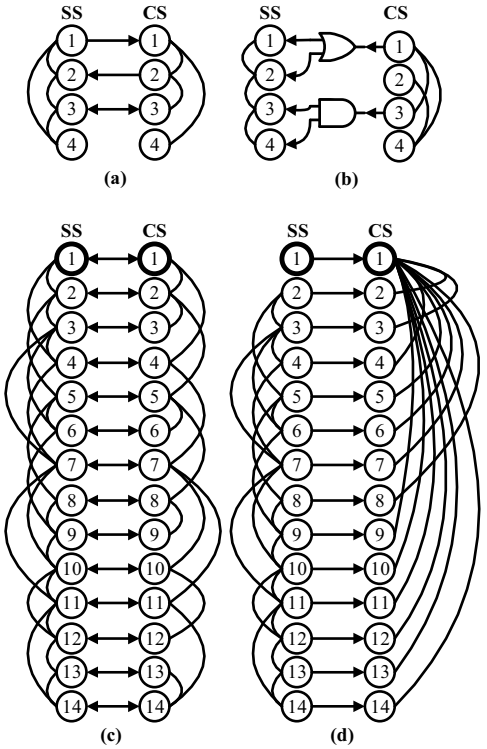

Fig. 2. Example *SS-CS* Interdependence Graphs.

Given the network topology of an SDN, the number and the locations of the controllers, an *SS-CS* interdependence graph can be constructed after specifying the controller-switch

communication strategies, such as the type of communication channels (out-of-band or in-band) and the routing protocols (e.g., single-path or multi-path routing). Fig. 2(c) presents a *SS-CS* interdependence graph for the example topology with node 1 as a controller and in-band controller-switch communication via single shortest paths (Fig. 1(b)). Nodes in *SS* and *CS* in Fig. 2(c) are mutual dependent. Fig. 2(d) presents another *SS-CS* interdependence graph for the example topology when the controller-switch channels are fully out-of-band and node 1 only plays the role of a controller in the *CS* network. The proper functioning of switches (nodes) in the *SS* network depends on their proper functioning in the *CS* network.

B. Failure Cascading on the SS-CS Interdependence Graphs

This subsection presents the use of *SS-CS* interdependence graph for cascading failure analysis. Similar to [3], we assume a static binding between controllers and switches. Switches are only controlled by a single assigned controller. When the connection between a switch and its controller fails, before a backup path is built and used, the switch fails and drop all packets. We consider the following failures:

- Link failure: a link failure indicates that all traffic passing through a link will be dropped. It can occur in a link in *SS* or in *CS*.
- Node failure: a node failure indicates that the node cannot be proper functioning in SS and/or CS networks. It can be a switch that becomes unable to originate or forward packets. It can also be a controller that fails to process packets or managing rules. Node failures can be caused by software bugs, malicious attacks, misconfigurations, hardware failures, power outage, etc.

In the following, we define the process of a failure cascading on the *SS-CS* interdependence graph.

- The process starts with a link failure or a node failure in either *SS* or *CS* network.
- The failure propagates iteratively between *SS* and *CS* networks based on their dependence relations.

  In *CS* network, a node is affected by the faillure if its dependent nodes in *SS* are affected by the failure or it is disconnected from its assigned controller; a link is affected by the failure if any node that it connects with is affected by the failure.

  In *SS* network, a node is affected by the faillure if its dependent nodes in *CS* are affected by the failure or all its links in *SS* are affected by the failure; a link is affected by the failure if any node that it connects with is affected by the failure.
- The failure cascading reaches its steady stage if no more nodes/links can be affected by the failure or all the nodes and links are affected by the failure.

Fig. 3 demonstrates a failure cascading on the example *SS-CS* interdependence graph presented in Fig. 2(c). The failure starts at node 7 in *SS*, and thus all its associated links in *SS* are affected (Fig. 3(a)). Due to the mutual dependence, the failure propagates to node 7 in *CS*, as well as its associated links in *CS* (Fig. 3(b)). Because of the failure of node 7 in *CS*, node 10, 11, 12, 13, 14 in *CS* are disconnected from the controller (node 1 in *CS*), and thus the failure propagates to those nodes and their associated links (Fig. 3(c)). Due to mutual dependence, the failure continues to affect the nodes 10, 11, 12, 13, 14 in *SS* and the associated links in *CS* (Fig. 3(d)). After this iteration, no more nodes or links will be affected, so the propagation stops, and the failure cascading reaches a steady stage.

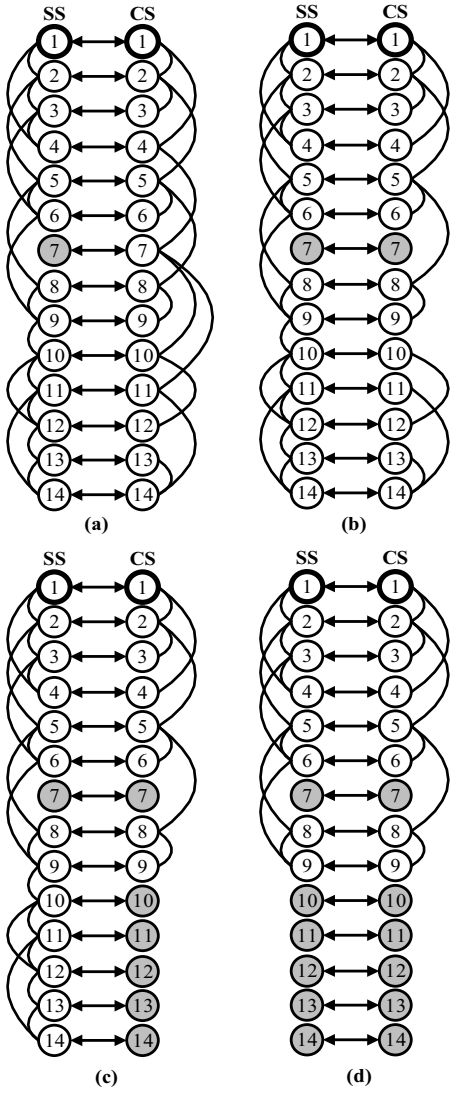

Fig. 3. An example cascading failure on *SS-CS* interdependence graph. *(Links are removed if they are affected by the failure; nodes are colored in grey if they are affected by the failure.)*

C. Impact of Controller Placement on Network Resilience

Based on the analysis above, the resilience of a network is closely related to the number of nodes survived in the steady stage of the cascading of a failure. The more the fraction of nodes survived at the steady stage, the better the network resilient to the failure. To investigate the impact of controller

placement on the network resilience, we use the same SDN topology and controller-switch communication strategies as the ones use in Fig. 3 to construct *SS-CS* interdependence graphs based on different controller placement, and then repeat the cascading failure analysis on the interdependence graphs starting from a failure at node 7 in *SS* network. Fig. 4 shows the steady stage of the cascading failure analysis. We can see that, by selecting node 10 as the controller, more nodes survived at the steady stage (Fig. 4(a)). By increasing the number of controllers to two and placing them on node 1 and node 10, the failure at node 7 does not affect other nodes. The results in Fig. 4 exemplifies the impact of controller placement on the network resilience.

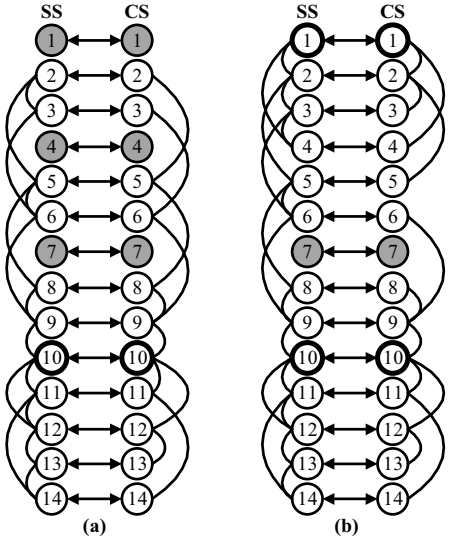

Fig. 4. The steady stage of the cascading of a failure on two *SS-CS* interdependence graphs. *(a) The steady stage of the cascading failure analysis with one controller at node 10; (b) The steady stage of the cascading failure analysis with two controllers at node 1 and node 10.*

III. Controller Placement For Improving Resilience

Given an SDN network topology $G=(V,E)$, the number of controllers k, the locations of the controllers $L \subseteq V$, and the controller-switch communication strategy ψ, an *SS-CS* interdependence graph H can be constructed and a cascading failure analysis can be performed on the graph. H, k, L, ψ, and the probability distribution of link and node failures Δ all have direct impacts on the survival nodes at the steady stage of the cascading failure analysis.

A. Resilience Metric

Given k, L, ψ and Δ, we define the resilience of H as follows:

$$\phi = 1 - \frac{1}{|H|} * \sum_{i \in H} \Delta^i * \left[\alpha \sum_{j=1}^{k} f_j^i + (1-\alpha)\mu^i \right] \qquad (1)$$

where:

- i is a node or a link in *SS* or *CS* of H for cascading failure analysis; $|H|$ denotes the total number of nodes and links for cascading failure analysis;
- Δ^i is the probability of failure at i; $\Delta^i \in [0,1]$;
- f_j^i is the fraction of nodes assigned to controller j but not in the mutually connected cluster of j at the steady stage of the cascading of a failure at i *on* H;
- μ^i denotes the fraction of nodes in the largest mutually connected cluster at the steady stage of the cascading of a failure at i *on* H;
- α controls the tradeoff between the resilience of the sub-networks controlled by individual controllers and the resilience of the network as a whole; $\alpha \in [0,1]$.

Let $V^j \subseteq V$ be the set of switches assigned to controller j, $V_s^j \subseteq V^j$ denote the nodes in the *SS* network of H that are controlled by j, and $V_c^j \subseteq V^j$ be the set of nodes in the *CS* network of H that are controlled by j. $V_s^{j,i}$ denotes the set of nodes in the *SS* that (1) are survived at the steady stage of the cascading of a failure at i and (2) are in the same connected component of *SS* at the steady stage. $V_c^{j,i}$ denotes the set of nodes in the *SS* that (1) are survived at the steady stage of the cascading of a failure at i and (2) are in the same connected component of *CS* at the steady stage with j. We define $(V_s^{j,i}, V_c^{j,i})$ as a mutual connected cluster for j if (1) no other nodes from V_s^j or V_c^j can be added and the new pair of sets still satisfies the connectivity requirements and (2) $V_s^{j,i} = V_c^{j,i}$, which means that they are correspondent to the same set of nodes in G. Buldrey et al. [8] states that only the mutual connected clusters are potentially functional. Therefore, we measure the resilience by taking into consideration the expected fraction of nodes survived in the mutual connected cluster for each controller. The largest mutual connected cluster is formed by the largest possible union of mutual connected clusters for individual controllers as long as the union satisfies the requirements of mutual connected cluster.

B. Controller Placement Algorithm for Improving Resilience

The controller placement problem in SDN is to find the best L that maximizes ϕ. As the problem is *NP-Hard*, we propose a partition and selection approach to determine the placement of controllers for improving ϕ.

The definition of network resilience in (1) is essentially a weighted average of the mean resilience of the sub-networks controlled by individual controllers and the resilience of the network as a whole. Therefore, our *Algorithm 1* for controller placement is designed as follows: (1) given G and k, the algorithm first finds communities in G using the greedy modularity optimization method described in [9] and partitions the generated hierarchical tree of nodes into k clusters, each of which will be a sub-network managed by a controller; and then (2) in each sub-network, the algorithm selects a node with the maximal closeness centrality to all the other nodes in the sub-network for placing the controller for the sub-network. The output of *Algorithm 1* is L, the locations of the k controllers.

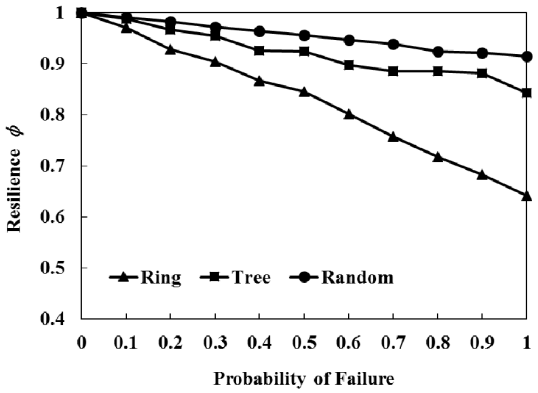

Fig. 5. Resilience with different probability of failure.

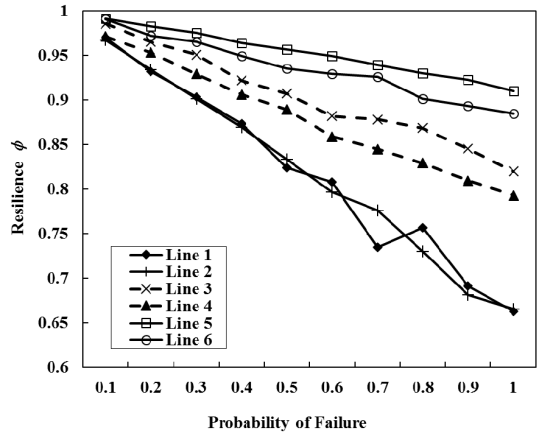

Fig. 6. Comparison of *Algorithm 1* and random placement. (*All networks consist of 50 nodes and three controllers. The controller-switch communication strategy is in-band via single shortest path. Line 1 is the result of using Algorithm 1 on ring topology, Line 2 is the result of using random placement on ring topology, Line 3 is the result of using Algorithm 1 on binary tree topology, Line 4 is the result of using random placement on binary tree topology, Line 5 is the result of using Algorithm 1 on random topology, and Line 6 is the result of using random placement on random network topology.)*

C. Numerical Analysis

In the following, we present the numerical analysis of *Algorithm1*. We used *igraph* library [10] to generate three types of network topologies for analysis, include ring topology, binary tree topology, and *Erdos-Renyi* random network topology. In the generated random networks, the probability of an edge between arbitrary two vertices was set to 0.4; therefore those random networks were mostly dense networks with small average path length. The generated ring networks had the largest average path length among the three types of topologies; while the average path length in binary tree networks was in the between. α was set to 0.5 in all the analysis and all the experiments were repeated 100 times.

Fig. 5 shows the expected network resilience changes as the failure probability of nodes. All the nodes were subject to a common probability of failure. Fifty nodes were used to generate the three types of networks; *Algorithm 1* was utilized to determine the placement of three controllers in each network. The controller-switch communication strategy is in-band via single shortest path. The results show that the expected network resilience is inverse proportional to the average path length of the network topology; and thus random networks perform best and ring networks are least resilient.

Fig. 6 shows the comparison of *Algorithm 1* and random placement. The resilience of ring networks is not affected by the placement. The average improvement of *Algorithm 1* is 1.8% compared to the random placement on random network topology and 2.4% compared to the random placement on binary tree topology. The improvement is not so significant. One reason for this can be that the resilience metric in (1) only takes the node loss into account; and thus it is less sensitive to the loss in the links and the pairwise connectivity loss.

IV. Conclusion

This work introduces the use of interdependence graph and cascading failure analysis to analyze of the impact of controller placement on network resilience, designs a new resilience metric, and proposes an approach to improve the resilience. The interdependent network analysis is a useful tool that can benefit many aspects of the architectural design of SDN. In the future, we plan to investigate the integration of pairwise connectivity loss into the resilience metric. We also plan to perform more comprehensive evaluation using *Mininet* [11] and using real-world software-defined network testbeds.

References

[1] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev*. 38: 69-74, 2008.

[2] B. Heller, R. Sherwood, and N. McKeown, "The Controller Placement Problem," *The 1st Workshop on Hot topics in Software Defined Networks (HotSDN '12)*, Helsinki, Finland, August 13-17, 2012.

[3] Y. Zhang, N. Beheshti, and M. Tatipamula, "On Resilience of Split-Architecture Networks," *GLOBECOM 2011*, Huston, TX, 2011.

[4] Y. Hu, W. Wang, X. Gong, X. Que, and S. Cheng, "On the Placement of Controllers in Software-defined Networks," *The J. of China Univ. of Posts and Telecomm*., Oct. 2012, 19 (Suppl. 2): 92-97.

[5] N. Beheshti and Y. Zhang, "Fast Failover for Control Traffic in Software-defined Networks," *GLOBECOM 2012,* Anaheim, CA, 2012.

[6] P. Gill, N. Jain, and N. Nagappan, "Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications," *SIGCOMM'11*, August 15-19, 2011, Toronto, Ontario, Canada.

[7] S. V. Buldrey, R. Parshani, G. Paul, H. E. Stanley, and S. Havlin, "Catastrophic Cascade of Failures in Interdependent Networks," *Nature* 464, 1025-1028, April 15, 2010.

[8] S. Sharma, D. Staessens, D. Colle, M. Pickavet, P. Demeester, "Fast failure recovery for in-band OpenFlow networks," *9th International Conference on the Design of Reliable Communication Networks*, Budapest, Hungary, March 4-7, 2013.

[9] A. Clauset, M. E. J. Newman, C. Moore, "Finding Community Structure in Very Large Networks, " *Physical Review E*, 70(6), Dec 2004.

[10] http://igraph.sourceforge.net/

[11] http://mininet.org.