This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE Globecom 2011 proceedings.

On Resilience of Split-Architecture Networks

Ying Zhang, Neda Beheshti, Mallik Tatipamula
Ericsson Research

***Abstract*—The split architecture network assumes a logically centralized controller, which is physically separated from a large set of data plane forwarding switches. When the control plane becomes decoupled from the data plane, the requirement to the failure resilience and recovery mechanisms changes. In this work we investigate one of the most important practical issues in split architecture deployment, the placement of controllers in a given network. We first demonstrate that the location of controllers have high impact on the network resilience using a real network topology. Motivated by such observation, we propose a min-cut based controller placement algorithm and compare it with greedy based approach. Our simulation results show significant reliability improvements with an intelligent placement strategy. Our work is the first attempt on the resilience properties of a split architecture network.**

***Index Terms*—Split architecture; OpenFlow; Network resilience; Placement**

I. Introduction

Unlike the traditional network architecture, which integrates both forwarding (data) and control planes on the same box, split architecture decouples these two and runs the control plane on servers which might be in different physical locations from the forwarding elements (switches). The split architecture allows making the forwarding platform simple and bringing network's intelligence into a number of controllers that oversee the switches.

Tight coupling of forwarding and control planes in the traditional architecture usually results in overly complicated control plane and complex network management. Due to high complexity, equipment vendors and network operators are reluctant to employ changes and the network itself is fragile and hard to manage. This is known to create a large burden and high bearer to new protocols and technology developments. Despite the rapid improvement on line speeds, port densities, and performance, the network control plane mechanisms have advanced at a much slower pace.

In the split architecture, controllers collect information from switches, and compute and distribute the appropriate forwarding decisions to switches. Controllers and switches use a protocol to communicate and exchange information. An example of such protocol is OpenFlow [1], which provides an open and standard method for a switch to communicate with a controller, and has drawn significant interests from both academic and industry [1], [2], [3]. The architecture of an OpenFlow switch is shown in Figure 1. Switches are modeled as a flow table in which there are three columns: rules, actions, and counters. The rules column defines the flow. Rules are matched against the headers of incoming packets. If a rule matches, the corresponding actions are applied to the packet and the counters in the counter column are updated. The de-coupled control platform eases the task of modifying the network control logic and provides a programmatic interface upon which developers can build a wide variety of new protocols and management applications. In this model, the data and control planes can evolve and scale independently, while the cost of the data plane elements is reduced.

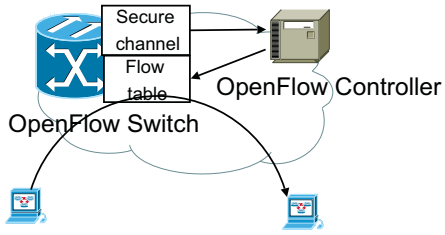

Fig. 1. OpenFlow Architecture

In evaluating a network design, the network resilience is an important factor, as a failure of a few milliseconds may easily result in terabyte data losses on high-speed links. In traditional networks, where both control and data packets are transmitted on the same link, the control and data information are equally affected when a failure happens. The existing work on the network resilience analysis have therefore assumed an in-band control model, meaning that the control plane and data plane have the same resilience properties. However, this model is not applicable to split-architecture networks. On one hand, the control packets in split-architecture networks can be transmitted on different paths from the data packet (or even on a separate network). Therefore, the reliability of the control plane in these networks is no longer linked with that of the forwarding plane. On the other hand, disconnection between controller and the forwarding planes in the split architecture could disable the forwarding plane: When a switch is disconnected from its control plane, it cannot receive any instructions on how to forward new flows, and becomes practically offline.

In this paper, we take the first attempt to study the resilience issues in the split architecture network. The resilience problem in such network is formulated and different types of failures are considered. To the best of our knowledge, this is the first proposal on the formulation of resilience problems in the new network architecture with controlling and forwarding plane decoupled. We further analyze the impact on the network resilience induced by such decoupling using simulation on a real network topology. Interestingly, we observe significant differences in loss of connectivity in the network when placing

978-1-4244-9268-8/11/$26.00 ©2011 IEEE

the controller in different locations.

Motivating by such observation, we next study the strategies for controller placement. More specifically, we focus on the controller placement problem given the distribution of forwarding plane switches. We consider that control platform consists of a set of commodity servers connecting to one or more of the switches. Therefore, the control plane and data plane are in the same network domain. The question is, among all n switches, how to select a subset for connecting k controllers where the total network resilience is maximized. In this work, we propose a min-cut based graph partitioning algorithm for controller placement. The algorithm aims at maximizing the resilience between controller and the switches. For comparison, we further develop a greedy based algorithm. Our simulation results show significant benefit of our placement algorithm, in comparison with random and greedy schemes.

The rest of the paper is organized as follows: We first introduce the related work of OpenFlow and existing work on resilience in Section II. Then we formulate the resilience analysis problem in Sec III and present the analysis in Sec IV. The controller placement strategies are introduced in Sec V. Simulation results on both synthesis and real topology are given in Sec VI. We finally conclude and discuss future work in Sec VII.

II. Related Work

There has been a significant interest on the reliability analysis on the Internet [4], [5], [6], [7], [8]. These work computes the number of node or link disjoint paths between any pair of ASes, *i.e.,* path diversity, as one of the important reliability metrics of the Internet. Teixeira *et al.* [9] studied the path diversity problem both inside an AS (Sprint network) and across multiple ASes based on the CAIDA topology. Erlebach *et al.* [10] proposed a min-cut based algorithm to compute the maximum disjoint paths between a pair of ASes, as the quantitative metric of the Internet route diversity. Another metric is the routing dynamics caused by any failures [11]

The emerging OpenFlow [1] platform enables switches to forward traffic in the data plane based on rules installed by a separate controller. Most of the existing work focus on the design and implementation to achieve high performance for switches using commodity hardware [12], [13], [14] and building controller to support a large number of networks and a diverse set of protocols [15], [16]. Building upon these OpenFlow tools, there have been work on improving the performance of the data plane by more intelligently utilizing the hardware resources [17], [18]. Other work focus on new applications for OpenFlow technology including virtualization [19], load-balancing [20] and convergence across technologies [21]. However, there is no work on any reliability and robustness analysis on the split architecture/OpenFlow network, which is one of most important factor for a networking infrastructure. Our work takes the first step about how to place the controller to maximize the resilience.

III. Problem formulation

Before diving into the formal definition of network resilience, we illustrate the concept of resilience in the split architecture through an example.

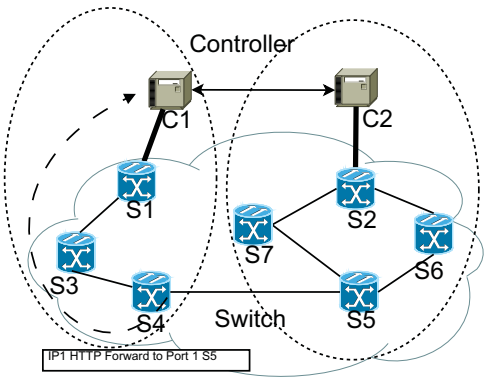

Fig. 2. Split-architecture network example

Figure 2 shows a network, which consists of 7 switches and 2 controllers. We assume a static binding between controllers and switches, *e.g.,* C_1 is the assigned controller to S_1, S_3, and S_4. These three switches are only controlled by C_1 even though they are also reachable by C_2. We also assume shortest-path routing between each switch and its controller.

In this network, if the link between S_4 and S_5 fails, connections between any of switches S_1, S_3, S_4 to any of switches S_2, S_5, S_6, S_7 would be interrupted.If the link between S_1 and controller C_1 fails, then until a backup path is built and used, S_1 will lose its connection to its controller. Assuming that in this case the switch invalidates all its entries, then S_1 cannot reach any other switch in the network, until it reconnects to its controller. This is like S_1 itself is failed for a period of time.

A. Problem formulation

The physical network topology $G(V, C, E = \{E_v, E_c, E_{vc}\})$ contains a set of switches V, a set of controllers C, and a set of bi-directional links E. Set E consists of three disjoint subsets: links between switches $E_v = \{e_{ij}, i, j \in V\}$, links between controllers $E_c = \{e_{ij}, i, j \in C\}$, and links between one controller and one switch $E_{vc} = \{e_{ij}, i \in Cj \in V\}$. The relationship among E_c, E_v, and E_{vc} depends on different network design choices. Some networks may use the same network infrastructure for interconnects both between switches and between controllers. In such networks, there are no separate links between controllers, *i.e.,* the controller uses network forwarding elements to reach each other or $E_c = \emptyset$.

Let R be a set of routes (paths) used by the network for connections between any pair of switches, controllers, and between controller and switches. In case of multiple routes between two nodes, the shortest one is picked, without considering any protection mechanisms. Routes are computed by the controllers, with the knowledge of the full network topology.

Upon failure, the controller re-computes the routes and installs new entries on the switches. During the recovery period, all packets passing the failed link or node will be dropped. In this work, we assume the controller-to-switch assignment is fixed, *i.e.,* each switch has (only) one controller assigned to it. Therefore, whenever a switch loses its connection to its controller, the switch can no longer receive any new routing instructions and drops all packets. We assume that the route between any switch and its controller uses existing connections between switches, meaning that $R_{vc} = \{e_{ij}\}, e_{ij} \in E_v \cup E_{vc}$. We assume uniform traffic distribution in the network, meaning that each route carries one unit of traffic.

Our failure model classifies failures into three categories:

Link failure: A link failure indicates that traffic traversing the link can no longer be transferred over the link. The failure can be either of a link between two switches or of a link between one controller and the switch it connects to. We assume network links fail independently.

Switch failure: A switch failure indicates that the corresponding node is unable to originate, respond, or forward any packet. Switch failures can be caused by software bugs, hardware failures, misconfigurations, *etc.*. Again, we assume network nodes fail independently.

Special case: connectivity loss between switch and controller: A switch may lose connectivity to its controller due to failures on the intermediate links or nodes along the path. In this work, we assume that whenever a switch cannot reach its controller, the switch will discard all the packets on the forwarding plane, even though the path on the forwarding plane is still valid. Therefore, this can be considered as a special case of switch failure.

IV. Resilience Analysis of Split Architecture

Assuming failure probability P_{e_i} for link e_i and failure probability P_{v_i} for node v_i in the network, the chance that communications from node a to node b fails is

$$1 - \prod_{e_i, v_j \in R_{ab}} (1 - P_{e_i})(1 - P_{v_j}),$$

where R_{ab} is the shortest path from node a to node b. The product term in the above equation is the probability that all nodes and links on the path from node a to node b work finely. In the split-architecture network, as mentioned in Section III, node failures also include cases where a node works properly but loses its connection to the controller (*e.g.,* when another link on the shortest path from the node to its controller breaks). The best controller location for minimizing the average disconnection chance in the split-architecture network, is the one that minimizes the above equation, where the average is over all pairs of nodes that communicate with each other.

To get a sense of how link and node failures affect connections in a split-architecture network, we make some calculations in an example network, with the topology of Internet2 [22] network, as shown in Figure 3. Here, we will calculate the number of connections that each link failure causes. We will also find the best controller locations by calculating the average number of interrupted connections that is caused by any given number of link failures.

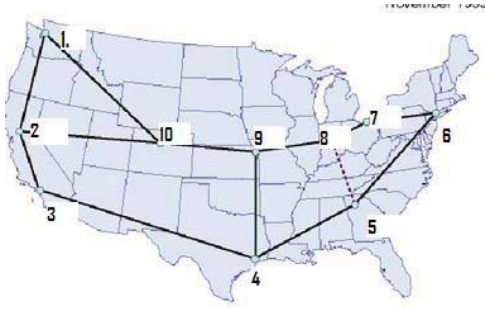

Fig. 3. Internet2 network topology

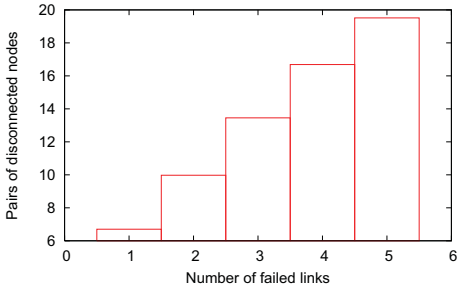

Fig. 4. Affected node pairs in traditional network

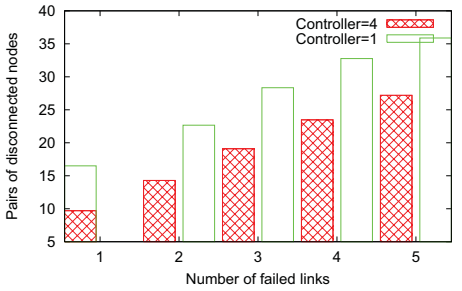

Fig. 5. Affected node pairs in split architecture (one controller)

Figure 4 shows the basic resilience of the network. As expected, the number of disconnected shortest path increases as more links fail. Considering split architecture is deployed in the same network topology, we first need to determine where the controller is placed. We enumerate all the possible controller locations and show two examples in Figure 5. Interestingly, we can observe significant difference caused by the location of the controller. For placing one controller in this topology, location 4 is a much better choice than location 1. Compared to Figure 4, the disconnected pairs are much more in the split architecture scenario. When a link fail, it may disconnect the path between two switches, or the path between controller and the switch. Figure 6 shows such break-down. More pairs are disconnected because of the loss of connectivity to the controller. Increasing the number of controller in the network can also significantly improve the resilience, as shown in Figure 7.

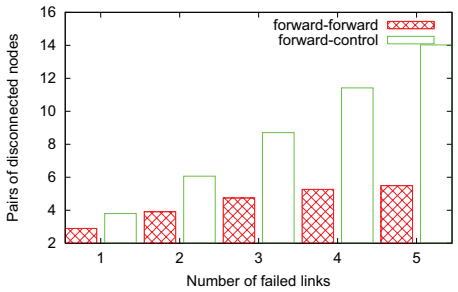

Fig. 6. Breakdown of disconnected node pairs (controller=4)

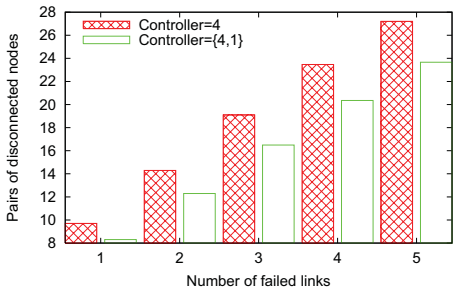

Fig. 7. Affected node pairs in Openflow network (two controller)

V. Controller Placement

An important issue in designing a split-architecture network is where to place the controller in order to maximize network resilience. A good placement best utilizes existing network connectivity among the switches. One naive solution is to connect all controllers to all switches, forming a mesh. This will significantly increase the deployment cost and wiring complexity. Moreover, it is not scalable as the network size grows to a large number of switches spreading across multiple geographic locations.

In this section, we introduce our proposal on the controller placement puzzle to maximize the network resilience subject to some constraints. We formulate the placement problem as an optimization problem of maximizing the reliability or minimizing the failure probability. For this purpose, we use the following notation. Let $G = (V, E)$ denote the basic network, with node set V representing network's switches, and edge set E representing network's links (which are assumed to be bidirectional). The objective is to pick a subset M ($|M| = k$) of the nodes, among all candidates $N = |V|$, and co-locate controllers with switches in these nodes so that the total failure likelihood is minimized. Once these k nodes are selected, a solution to assign switches to controllers, *Map(V)=C*, is also needed to achieve maximum resilience.

In a first approach, we can enumerate all the $|V| \times |V|$ pairs of switches and identify the routes between them. Then we select the top k ones with maximum number of paths (or average shortest path length) to reach all other switches. However, this method may result in selecting k locations with similar path diversity to reach the rest of the nodes. For instance, many paths could share one critical link. In the following, we propose to solve the problem as a graph partitioning or clustering problem. A clustering of a graph is a partition $C = C_0, C_1, \ldots, C_k$ of V, *i.e.,*, $\forall i, j : C_i \cap C_j = \emptyset$, and $C_0 \cup C_1 \cup \ldots \cup C_k = V$. A cost function f assigns a real number to any given clustering of G. The goal is to find a clustering that minimizes a given cost function. For instance, the cost function could be the sum of the distance between each node and its centroid, or it could be the negative sum of edge weights between clusters. Typically there are two ways to partition a graph: *agglomerative approach* which initializes each element to belong to its own cluster and proceeds to merge clusters until a certain terminating condition is met; and *partitive clustering* which starts with a single cluster containing all elements and proceeds by splitting clusters.

A large body of algorithms has been proposed for server placement including clustering, k-means, greedy, and min-cut. By analyzing the requirements, we find that a modified version of min-cut is a good fit to our problem. There are two competing criteria that define a good partitioning, *i.e.,* high intra-cluster homogeneity and low inter-cluster connectivity. Intuitively, if graph edges represent relationships between vertices, then we want many edges within clusters and few edges between clusters. However, if we define the cost function to be the number of inter-cluster edges, then the problem of minimizing it is trivial, *i.e.,* picking the clustering that contains a single cluster. Thus, in addition to the minimum cut requirement, we require that the partition needs to be as balanced as possible. The balance property can also help providing balanced load and best resource utilization on the controller. For instance, if one best location is used to serve all the switches, it may easily create bandwidth bottlenecks. In the following, we use *cuts*, *inter-cluster edges* and inter-site traffic interchangeably.

We define the center of each partition: $centroid(C) = \frac{1}{|C|}\sum_{v \in C} v$. To maximize the resilience, we start with two cost functions:

$$f = \sum_{C \in P} \sum_{u \in C} length(u, centroid(C)) \tag{1}$$

$$g = -\frac{1}{|V|} \sum_{u \in V} \frac{1}{h} \sum_{j=1} \delta(u, nei_{uj}) \tag{2}$$

where P is a giving partition solution, $length(u, v)$ is the weight of edge (u, v), which is defined to be the switch to centroid shortest path length in our problem. nei_{uj} means u's j^{th} neighboring switch. j is used to enumerate all of $u's$ neighbors. $\delta(u, v)$ is a binary valuable to denote if u and v are in the same partition. Note that g measures inter-cluster connectivity to be minimized, while f measures intra-cluster difference which should also be minimized. Together, they do not favor trivial clusterings where all nodes are assigned to the same cluster, or each node is assigned to its own cluster.

The key idea to combine these two objectives is to first identify the partitions with minimum cuts across boundaries. Then we assign the controller location to the centroid, which has the shortest paths to all switches in the same cluster. Algorithm 1 shows the details of finding the partitions with minimum cuts.

Algorithm 1 Mincut based Controller Placement Algorithm

procedure*Find-MinCut(G=(V,E),k)*

1: Initialize cluster partition $P = P_1, P_2$
2: For any pair of unmarked $(v_1 \in P_1, v_2 \in P_2)$, compute $g(v_1, v_2) = g(v_1) + g(v_2) - 2\omega(v_1, v_2)$
3: Find (v_1, v_2) with maximum $g(v_1, v_2)$, swap and marked.
4: *IF* $\forall v_1, v_2, g(v_1, v_2) < 0$ or $iter = MAX_ITER$ or all nodes are marked
5: Stop
6: *ELSE* Goto Step 2.
7: *IF* $k > 1$
8: Find-MinCut($G_1 = (P_1, E_1)$,k-1)
9: *IF* $k > 2$
10: Find-MinCut($G_2 = (P_2, E_2)$,k-2)

Algorithm 2 Greedy based Controller Placement Algorithm

procedure*Greedy(G=(V,E),k)*

1: $j = 0$
2: **while** $j < k$ **do**
3: **for** all $v \in V$, $v \notin C$ **do**
4: Calculate number of routes to another node i: $|\{r_j\}| = n(v, i)$ and average length len(v,i)
5: $div(v, i) = Max_l \frac{\sum \psi(r_j, l)}{n(v,i)}$
6: $RE(v) = \sum_{i \in V, i \neq v} \frac{n(v,i)}{div(v,i) \times len(v,i)}$
7: **end for**
8: Select Minimum c_j=RE(v). $C = C \cup \{c_j\}$. $j++$.
9: Assign closest $\frac{|V|}{k}$ nodes to c_j.
10: **end while**

Given an initial bisection, we try to find a sequence of node pair exchanges that leads to an improvement of the cut size. Let P_1, P_2 be the bisection of graph G=(V,E), *i.e.,* $P_1 \cup P_2 = V$ and $P_1 \cap P_2 = \emptyset$. For each $v \in N$ we define $int(v) = \sum_{(v,u) \in E \& P(v) = P(u)} \omega(v, u)$ and $ext(v) = \sum_{(v,u) \in E \& P(v) \neq P(u)} \omega(v, u)$. We define the moving gain of v to a different partition to be $ext(v) - int(v)$. In each iteration, we find the best pair of nodes $v1 \in P_1$ and $v_2 \in P_2$ to exchange to maximize the gain. This process run recursively until no further gain can be obtained by changing any pairs.

Once we identify the partition P from Algorithm 1, within each partition P_i, we calculate the centroid of the switches in this group. The set of centroids $C = c_i, \cdots, c_k$ is calculated such that $c_i = \frac{1}{|P_i|} \sum_{x_j \in P_i} x_j$, where $P_i := argmin \|v - c_i\|^2$. Then each switch is naturally assigned to the centroid of its partition.

For evaluation purpose, we compare Algorithm 1 with a greedy based algorithm, shown in Algorithm 2. For all pair of nodes, we calculate the number of distinct paths, the average path lengths and the fraction of overlapping link between them. $\psi(r_j, l) = 1$ if link l is in the route r_j. We choose the one node with maximum value of RE which finds a balance between maximizing number of paths, minimizing path length, as well as minimizing overlapping links. In each iteration, we greedily select the node with best RE value and assign $\frac{|V|}{k}$ switches to it. This is to balance the load over all controllers. We will provide the comparison in Sec VI.

VI. Simulation Results

In this section, we use simulation to verify our analytical models in both synthetic and realistic network topologies. We quantify the impact of the new OpenFlow architecture on network resilience. In the simulation, we first study the connectivity on three different general networks, *i.e.,* ring, star, and fat tree. For each topology, we simulate multiple failure probability and network sizes. The goal is to investigate how the resilience metric changes as the network condition changes. For each set up, we calculate the average loss of connectivity likelihood among all pairs of nodes in the network. Since there is no existing work on the resilience analysis on OpenFlow network, we implemented our own simulation tool in perl. We simulate the resilience under different failure likelihood in a large range. In reality, the failure probability varies depending on different types of root causes [23]. Therefore, we examine a wide range to cover all scenarios.

A. Connectivity resilience between controllers and switches

The connectivity to the controller is extremely important for the OpenFlow network resilience. We define the resilience of controller to data plane as the average likelihood of loss of connectivity between the controller and any of the OpenFlow switches. Figure 8 and Figure 9 show the controller connectivity changes as the failure probability of each link and the size of the network. Here we vary the probability of all links, including E_v and E_{vc}. Among the regular topologies, the star topology performs best due to the smaller path length. The ring topology is least resilient as it has longest average path length. It shows that the path length between the controller and the switch is a significant factor for all three networks. While in the star and tree network the loss of connectivity likelihood increases linearly as the link failure probability, it increases more slowly for ring topology. On the other hand, the average resilience metric does not change significantly for star and tree networks, when the number of switches in the network increases.

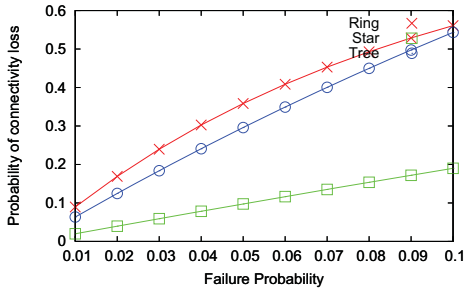

Fig. 8. Controller Connectivity Loss w. diff failure probability

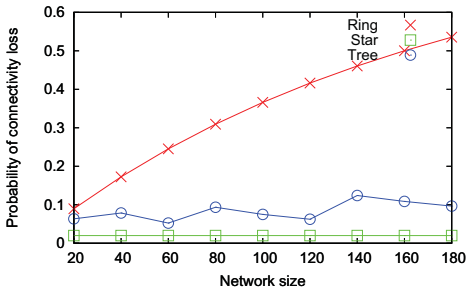

Fig. 9. Controller Connectivity Loss w. diff network size

B. Resilience under different placement algorithm

Another important factor that has large impact on resilience is the placement of the controllers. To evaluate Algorithm 1, we evaluate the placement of 5 controllers in a network of 100 switches using the fat-tree topology.

We also use simulation to verify the benefit of our controller placement algorithm in Figure 10. It shows a clear benefit of deploying 3 controllers in greedy and min-cut schemes compared to the random scheme. We can observe up to 10% improvements with min-cut based algorithm, when the failure probability is large, *i.e.,* $p = 0.1$. Even if $p = 0.01$, the improvement of min-cut is 4% compared to the random placement and 2% compared to greedy based placement. In summary, we evaluate our placemen algorithm in comparison with greedy and random scheme. Our min-cut based algorithm shows a 5% improvement. The improvement is not so significant as the Abilene topology is quite simple with only 10 nodes. It will have more benefits with a larger-scale network.

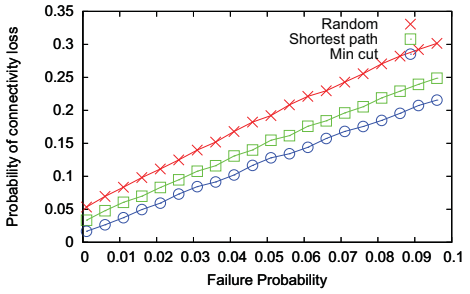

Fig. 10. Comparison on resilience with diff. placement strategies

VII. Conclusion and Future work

In this paper, we investigate the resilience of the newly proposed network architecture where the controlling and forwarding planes are separated. From the analysis using Abilene network's topology, we found the location of the problem can significant affect the network resilience metrics. Motivated by this observation, we propose a min-cut based placement algorithm to minimize the likelihood of loss of connectivity between the controller and the switch. Our simulation results show that it outperforms random and greedy schemes with significant reliability improvements. In this work, we make two simplified assumptions: each switch only connects to one controller using one path. The controller placement method will change with multiple controllers per switch or multipath routing. We plan to investigate the solutions with relaxed assumptions in the future work.

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.*, vol. 38, pp. 69–74, March 2008.

[2] M. Brunner, D. Dudkowski, C. Mingardi, and G. Nunzi, "Probabilistic decentralized network management," in *Proceedings of the 11th IFIP/IEEE international conference on Symposium on Integrated Network Management*, IM'09, pp. 25–32, 2009.

[3] O. M. M. Othman and K. Okamura, "Design and implementation of application based routing using openflow," in *Proceedings of the 5th International Conference on Future Internet Technologies*, CFI '10, pp. 60–67, 2010.

[4] R. Albert, H. Jeong, and A.-L. Barabasi, "Error and attack tolerance of complex networks," *Nature*, 2000.

[5] R. Cohen, K. Erez1, D. ben Avraham, and S. Havlin, "Resilience of the Internet to Random Breakdowns," *Phys. Rev. Lett.*, 2000.

[6] G. Iannaccone, C.-n. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, "Analysis of link failures in an ip backbone," in *Proceedings of the 2nd ACM SIGCOMM Workshop on Internet measurment*, pp. 237–242, 2002.

[7] M. Lad, X. Zhao, B. Zhang, D. Massey, and L. Zhang, "Analysis of bgp update surge during slammer worm attack.," in *IWDC'03*, pp. 66–79, 2003.

[8] L. Wang, X. Zhao, D. Pei, R. Bush, D. Massey, A. Mankin, S. F. Wu, and L. Zhang, "Observation and analysis of bgp behavior under stress.," in *Internet Measurement Workshop'02*, pp. 183–195, 2002.

[9] R. Teixeira, K. Marzullo, S. Savage, and G. M. Voelker, "Characterizing and measuring path diversity of internet topologies," in *Proc. ACM SIGMETRICS*, September 2003.

[10] T. Erlebach, A. Hall, L. Moonen, A. Panconesi, F. Spieksma, and D. Vukadinovic, "Robustness of the Internet at the Topology and Routing Level," *Lecture Notes in Computer Science*, vol. 4028, 2006.

[11] X. Zhao, B. Zhang, A. Terzis, D. Massey, and L. Zhang, "The Impact of Link Failure Location on Routing Dynamics: A Formal Analysis,," in *Proceedings of ACM SIGCOMM Asia Workshop*, 2005.

[12] J. Naous, D. Erickson, G. A. Covington, G. Appenzeller, and N. Mckeown, "Implementing an openflow switch on the netfpga platform," in *Architecture for Networking and Communications Systems*, pp. 1–9, 2008.

[13] A. Greenhalgh, F. Huici, M. Hoerdt, P. Papadimitriou, M. Handley, and L. Mathy, "Flow processing and the rise of commodity network hardware," *Computer Communication Review*, vol. 39, pp. 20–26, 2009.

[14] B. Pfaff, J. Pettit, T. Koponen, K. Amidon, M. Casado, and S. Shenkerz, "Extending networking into the virtualization layer,"

[15] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "Nox: towards an operating system for networks.," *Computer Communication Review*.

[16] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and S. Shenker, "Onix: a distributed control platform for large-scale production networks," in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, OSDI'10, (Berkeley, CA, USA), pp. 1–6, USENIX Association, 2010.

[17] M. Yu, J. Rexford, M. J. Freedman, and J. Wang, "Scalable flow-based networking with difane," in *ACM SIGCOMM Conference*, 2010.

[18] M. B. Anwer, M. Motiwala, M. M. B. Tariq, and N. Feamster, "Switchblade: a platform for rapid deployment of network protocols on programmable hardware," in *ACM SIGCOMM Conference*, 2010.

[19] R. Sherwood, G. Gibb, K.-K. Yap, G. Appenzeller, M. Casado, N. McKeown, and G. Parulkar, "Can the production network be the testbed?," in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, OSDI'10, 2010.

[20] R. Wang, D. Butnariu, and J. Rexford, "Openflow-based server load balancing gone wild," in *Proceedings of the 11th USENIX conference on Hot topics in management of internet, cloud, and enterprise networks and services*, Hot-ICE'11, 2011.

[21] S. Das, G. Parulkar, and N. McKeown, "Packet and circuit network convergence with openflow," 2010.

[22] http://www.internet2.edu/network/.

[23] A. Markopoulou, G. Iannaccone, S. Bhattacharyya, C.-N. Chuah, Y. Ganjali, and C. Diot, "Characterization of failures in an operational ip backbone network," *IEEE/ACM Trans. Netw.*, vol. 16, pp. 749–762, August 2008.