An Experimental Study on Dynamic Network Reconfiguration in a Virtualized Network Environment Using Autonomic Management

Xuan Liu, Parikshit Juluri, Deep Medhi
Networking Research Lab, University of Missouri–Kansas City, USA
{xuan.liu, pjuluri, dmedhi}@umkc.edu

***Abstract*— In a *virtualized network*, *network virtualization* is achieved by creating multiple instances of virtual routers over existing physical routers. Critical aspects of *network virtualization* are resource allocation and resource re-allocation, especially in the case of router failures or re-allocation of routers due for maintenance. In this paper, we present a *dynamic network reconfiguration scheme* that can provision a new *virtual network* and recover it from router failures using *autonomic management*. In order to provide a highly available virtual network, our approach considers replacing failed routers with the most suitable standby routers. We have setup an experimental virtualized network on the GpENI Virtual Network Infrastructure (GpENI-VINI) testbed in which virtual routers are geographically distributed. Our autonomic management function invokes the replacement module in the event of a router failure. Our study considers not only end-to-end metrics (latency, bandwidth, loss-rate) but also the routing table convergence time to understand the transient behavior in this autonomic management environment.**

***Index Terms*—Dynamic Reconfiguration, Autonomic Management, Virtual Routers, Virtualized Networks.**

I. Introduction

Autonomic Computing, first proposed by IBM in 2001, has been widely considered as an rising technology to reduce the complexity and cost of management in the modern computing environment. [1] defines four properties of autonomic management: self-configuration, self-optimization, self-healing and self-protection, and these properties provide a paradigm for an autonomic management system design.

A. Background

Modern technologies and interconnected heterogeneous domains have made present network architecture more complex and costly for administration from the human-side, and a more self-configurable and self-adaptive network architecture is desired to improve network reliability and reduce management complexity. A recent survey [2] gives a comprehensive summary on current researches on autonomic network architecture design. [3] proposes an autonomic management architecture focusing on self-configuration and self-optimization, among the four properties of autonomic computing. FOCALE [4] addresses the challenges in autonomic network management, such as context awareness and resource/service provisioning. In [5] the authors demonstrate a programmable platform for autonomic management.

An effective autonomic network management scheme is an important feature in the next generation network. One of the typical cases is the *virtualized networks*. Network virtualization allows tremendous flexibility in the core network in terms of adaptive configuration and provisioning of services on demand. Network virtualization enables us to create multiple *virtualized networks* over the existing physical networks through multiple instances of virtual routers on physical routers. An important aspect to note about resource allocation to *virtualized networks* for different customers is that the resource requirement may change over time. In this case, *virtualized networks* must be reconfigurable to make an optimal allocation of resources to *virtualized networks* while keeping in mind that new requests for resources from existing or new customers can also arrive. In a fast and growing virtualized network environment, it becomes imperative to introduce autonomic management so that dynamic network reconfiguration is possible.

B. Related Work

There have been some studies on the dynamic reconfiguration of networks to overcome router failures and link failures. These works mainly concentrated on optimizing the routing algorithms to provide better support for dynamic reconfiguration. The algorithms considered were studied by either using mathematical models, simulations [6]–[8], or by testing them on networks using local lab machines [9]. In order to implement such approaches on a wide-area core network, hardware and software would need to be modified on every router. Secondly, with simulation or mathematical models, it is necessary to make certain assumptions that may not be applicable in real networks. While a testbed can be set up in a research lab in a confined space, it is not capable of emulating the real world network dynamics of a geographically distributed wide-area network. In our case, we study the effect of router replacement in a virtualized network testbed employing software based routers. To the best of our knowledge, there's very little work in this direction in wide-area network testbeds such as [10], [11].

In this paper, we focus on self-configuration and self-healing properties of autonomic network management, in order to recover the router failures in the network. We propose a *dynamic network reconfiguration scheme* in a realistic wide-area virtualized network testbed, where a set of standby virtual routers are used to accommodate changes in the network through autonomic management. In particular, we consider failure as an instance to address the issue of standby routers being configured on the fly and how this impacts the overall networked system.

This paper is organized as follows. Section II presents the motivations of this work. Section III presents the dynamic network reconfiguration scheme to recover router failures in a *virtualized network*. Section IV presents the experimental environment and the scenarios and metrics we have considered. We present our results in Section V and finish by providing the conclusion and future work in VI.

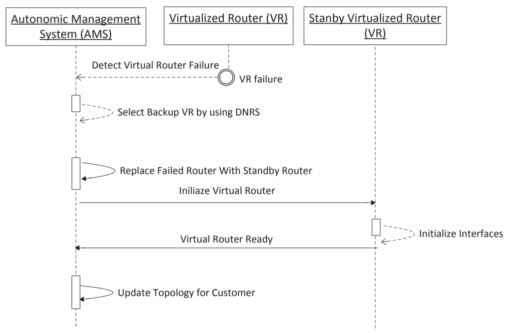

Autonomic Management System (AMS)
Virtualized Router (VR)
Stanby Virtualized Router (VR)
Detect Virtual Router Failure
VR failure
Select Backup VR by using DNRS
Replace Failed Router With Standby Router
Iniliaze Virtual Router
Initialize Interfaces
Virtual Router Ready
Update Topology for Customer

Fig. 1. Sequence Diagram to Replace a Failed Router in a Virtualized Network

II. Motivation

Autonomic network management aims to monitor, maintain and control a network with minimal human intervention, especially in a virtual network environment. Over the past few years, we have seen that the size and the amount of data carried by IP networks has been increasing rapidly. Apart from increasing sizes, virtualized networks are more dynamic and more heterogenous with greater demands on reliability. Demands and requirements of heterogenous traffic are also greatly varied. With the advent of virtual routers, it is possible to provide different architectures to serve different types of traffic. By using an *autonomous management system* (AMS), virtual network resources (i.e. virtual routers, virtual links, bandwidth, etc.) are provisioned and managed in a centralized manner. Since self-configuration and self-healing are two important features of autonomic management to ensure the reliability, we aim to address these two aspects in the virtual network environment.

In this work, we deploy the *AMS* in the *virtualized networks* to achieve dynamic network reconfiguration, in order to recover virtual router failures without involving manual operations. Our goals span to four aspects: (1) Conduct an experimental study on a wide-area *virtualized network* testbed to evaluate the *dynamic network reconfiguration scheme*; (2) Understand that network dynamics as realistic packet loss or bandwidth availability are part of the environment. (3) Evaluate the impact on the routing convergence time under current routing protocol (i.e., OSPF) during the reconfiguration; (4) Test autonomic management in a realistic environment and in turn, allows us to learn the challenges of applying *AMS* in such an environment.

III. Dynamic Network Reconfiguration Scheme

Fig. 1 presents the process of dynamic reconfiguration to overcome a router failure in the network. In this paper, we focus on evaluating the network performance during a reconfiguration process, so detecting failures is not out of this paper's scope. During this replacement process, any active sessions that are running on the current topology could be affected. If there is no alternative path for the data, then existing sessions would be interrupted until the recovery takes place. On the other hand, if there is an alternate path for the data, then we can expect that the communication would continue over that path until recovery is complete.

The *reconfiguration scheme* plays an important role when the network needs to be updated, or recovered seamlessly, and the whole process is transparent to the end hosts. Hence, an optimal selection strategy will largely reduce the impact on the end-to-end network performance due to the network reconfiguration.

Our reconfiguration scheme is based on selecting the best candidate among *standby routers* in order to replace failed routers. In general, we may consider a few options such as access latency, router's capabilities, router's geolocation, etc. In our study, we consider the third strategy based on the following reasons:

- *Standby routers* are not connected with any other active routers; so the access delay information cannot be used as a priori information when making a selection decision. Even though *standby routers* can be accessed from the *AMS*, we can only measure the instantaneous access delay from the *AMS*, and we cannot presume that the access performance is as good as that from other routers.
- Our experiment testbed is a virtualized network; the access delay on virtual links is not the same as it is on a physical link. Consider, two virtual routers are directly connected through an EGRE tunnel; the ping test between the virtual interfaces showed the round trip delay to be 0.0017ms, whereas between their two physical interfaces it was around 200ms. Therefore, although the virtual router's physical interfaces were active, the access delay to them cannot be used as a reference.
- Since we are running software-based virtual routers, instead of commercial routers, we do not have enough information on the router's capability as a reference to make a selection.

As a result, our first strategy is selecting a router that is closest to the failed router and the second one is to randomly select a candidate from the *standby routers* set. This is reasonable because the closest *standby router* should have a similar physical connection on the substrate, so it will least likely affect the original network performance. Algorithm. 1 selects a *standby router* based on the router's geographical information. The *Standby Router* is selected so that it is geographically the closest to the failed router in the network.

In order to do so, whenever a router failure is detected in the network, the *AMS* looks up the failed router's geolocation, and selects the closest router from the standby list to replace the failed router in the topology. Algorithm. 2 presents how the links are to be updated once a standby node is selected.

```
Algorithm 1: Selection based on Geolocation
    Input: $R\_id_{fail}$, $Set_{standby}$, $R_{info}$
    Output: $R\_id_{selected}$
    //Search the lat-long information of the failed router
1   ( $lat_{fail}$, $long_{fail}$) ← lookup ($R_{info}$, $Rid_{fail}$)
2   $geo_{fail}$ ← ( $lat_{fail}$, $long_{fail}$)
    //Initialize the minimal distance as infinity
3   $d_{min} \leftarrow \infty$
    //Initialize the selected standby router's ID
4   $R\_id_{selected} \leftarrow 0$
5   for $R\_id_{standby}$ in $Set_{standby}$ do
        //Search lat-long information of standby routers
6       ( $lat_{standby}$, $long_{standby}$) ← lookup ( $R_{info}$, $R\_id_{standby}$)
7       $geo_{standby}$ ← ( $lat_{standby}$, $long_{standby}$)
        //Get the distance between the failed and the standby router
8       $d$ ← distance( $geo_{standby}$, $geo_{fail}$)
9       if $d < d_{min}$ then
10          $R\_id_{selected} \leftarrow R\_id_{standby}$
```

```
Algorithm 2: Virtual Topology Update Algorithm
    Input: $N_f$, $N_s$, A graph $G$ with a set of nodes $V$, and a set of links $E$
    Output: An updated topology $G'$ with new node set $V'$ and new link set $E'$
1   for $u \in V$ do
2       if $u == N_f$ then
3           for each $v \in adj(u)$ do
4               linkdown (u,v) ;
5               linkup ($N_s$, v) ;
6               $E' \leftarrow (E - \{(u,v)\}) \bigcup \{(N_s,v)\}$ ;
7               $V' \leftarrow (V - \{u\}) \bigcup \{N_s\}$
```

IV. Running The Experiment

We have deployed our experiment on the GpENI Virtual Network Infrastructure (GpENI-VINI), a layer-3 programmable virtual routing testbed that is geographically distributed on the GpENI backbone network. We first give a brief background on the GpENI backbone network as well as the virtual network infrastructure on GpENI.

A. Experimental Platform: GpENI Testbed

GpENI (Great Plains Environment for Network Innovation) [12] is an international experimental testbed for future Internet research. It was originally built in collaboration between four universities [13] in the Midwest region of the United States. At present, the GpENI backbone has been expanded to Europe.

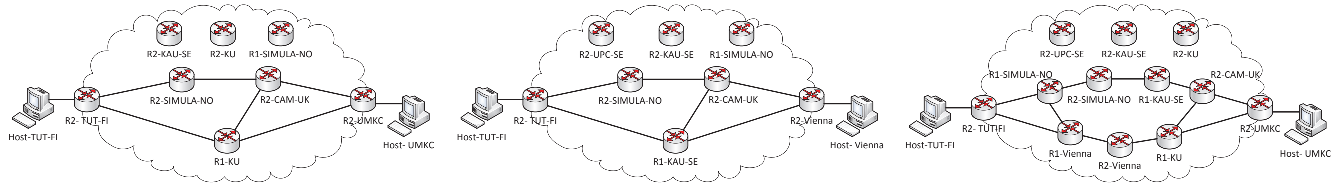
The GpENI testbed provides programmable capabilities on multiple layers of the network viz., the application layer, the network layer, and the optical layer. [14] describes how GpENI-VINI manages virtual network resources provisioning. Users can request a number of virtual routers and create a customized virtual topology using those virtual routers. Currently, there are 18 active GpENI-VINI routers that are distributed across the United States and Europe. This gives us an opportunity to evaluate the *dynamic network reconfiguration scheme* in a wide-area virtual network environment.

B. Configuring the Experimental Setup

Recall the *dynamic network reconfiguration scheme* that helps recover the network failure due to a router failure, replacing a failed router involves creating and removing links between routers. The GpENI-VINI testbed provides us a flexible and programmable layer-3 virtualization environment to deploy the *dynamic network reconfiguration scheme* [14]. The benefits are summarized from the following aspects:

- First, GpENI-VINI allows users to create an arbitrary topology in a slice.
- Second, virtual interfaces are pre-created on each virtual router. As a virtual router is added into the topology when configuring a slice [14], each virtual interface is assigned an IP address. Thus, we do not have to configure interfaces for every router in our experiment, which improves the efficiency of building the experimental environment.
- Third, default routing protocol configuration files are pre-created at every virtual router in the virtual network in a GpENI-VINI slice that has been configured with topology information. As a result, users may use this default configuration file while running routing software without any extra work.
- Fourth, a routing automation tool is provided for users to install and start routing software to all virtual routers at one time. When there are a large number of nodes in a slice, this enables high efficiency on an experiment pre-setup.

Due to the benefits we list above, we can enforce the *dynamic network reconfiguration scheme* within a GpENI-VINI slice. Instead of creating a fixed specific topology at the slice level by pre-configuration, we added all active GpENI-VINI nodes to the slice and created a full mesh topology consisting of all these nodes, so that each virtual router has a virtual interface to every other virtual router generated. On the other hand, an OSPF configuration file will be created based on this full mesh topology, where all direct links are assigned 10 as a default distance. Once the full mesh topology has been created for the slice, the *AMS* can construct any type of topology by enabling proper virtual interfaces on selected routers and disabling all other interfaces. Thus, when bringing up a new virtual router to the existing topology, *AMS* just needs to re-enable corresponding virtual interfaces on the newly



(a) Topology-I(a): US-Europe (b) Topology-I(b): Europe Only (c) Topology-II: US - Europe

Fig. 2. Five-node and Nine-node Wide Area Virtual Topologies

added virtual router and wait for the OSPF routing table to get convergence on every other virtual router in the topology. In the following subsections, we show our study on OSPF convergence time.

C. Experimental Scenarios

We considered three virtualized topologies on the GpENI-VINI testbed for our experiment. Topology-I(a) consists of five virtual routers, spreading across both Europe and the USA (see Fig. 2(a)). Fig. 2(b) (Topology-I(b)) shows the same topology as Topology-I(a), but it is composed of virtual routers located in Europe only. Note that the link speeds available and end-to-end delay in the substrate where GpENI is built are different between these two topologies. Hence, they represent two different sub-scenarios of Topology-I.

Topology-II, as shown in Fig. 2(c), is a nine-node virtual topology spanning both Europe and the USA. On the US side, we used nodes at the University of Missouri–Kansas City and the University of Kansas, and for the Europe side, we selected nodes from six institutions located in different countries (Norway, Sweden, Austria, Finland, UK, and Spain).

For all three networks, two nodes from the testbed were assigned as end hosts to inject traffic flows. We also assigned one node from the GpENI-VINI testbed as coordinator of the *AMS*. In Table I, we list the end-to-end round-trip time for all three topologies to illustrate how the topologies, while appearing to be similar, are different as imposed by bandwidth available from the GpENI substrate.

We are interested in three possible factors that might affect the network performance during the experiment: (1) the number of virtual routers in the virtual topology, (2) the geolocation of the virtual routers, (3) the *standby router* selection strategy.

TABLE I
END-TO-END ROUND TRIP TIME

Topo	min-RTT (ms)	avg-RTT (ms)	max-RTT (ms)
I(a)	170.223	171.681	182.732
I(b)	342.879	343.155	343.453
II	465.159	465.343	465.553

The end hosts exchange traffic data during a live session. In particular, we considered both a UDP-based end-to-end session and a TCP based end-to-end session. They are invoked by *iperf* [15]. We measured the instantaneous bandwidth over the end-to-end TCP session, as well as the instantaneous datagram loss rate and throughput over the end-to-end UDP session. For each TCP test with iperf running, we set MTU as 1000 bytes and the window size as 250 Kbytes. For each UDP test with iperf running, we set the default bandwidth as 2 Mbps, and buffer size as 250 Kbytes.

D. Metrics and Logs Collected

We collected several metrics and logs during the experiment as described below:

1) End to End Metrics and Logs:

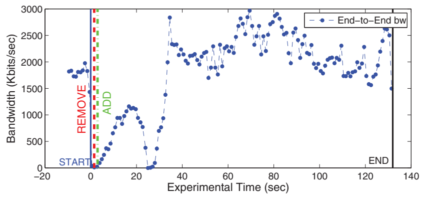
- *Instantaneous Throughput and Loss Rate*: This was obtained using iperf. By collecting the real-time UDP test logs, we got the best estimate of the throughput measured every second, particularly when the reconfiguration happened.
- *Instantaneous Bandwidth*: The Iperf TCP test helped us to estimate the actual bandwidth between end nodes that ensured packets were correctly received by the client side. Thus, by running the Iperf TCP test, we estimated how reliable the network behaved during the reconfiguration process.

2) Network Metrics and Logs:

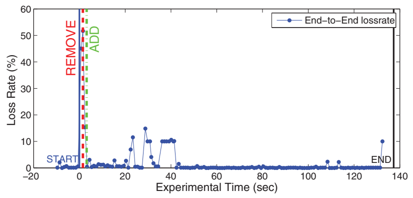
- *Recovery Time*: When a failure was detected, the *AMS* invoked the *dynamic network reconfiguration scheme* to recover the failed routers. The *dynamic network reconfiguration scheme* first disabled links from the failed router to its neighbors and then the failed router was replaced with the selected *standby router*. *Recovery time* indicates the duration between the occurrence of a failure and the time when a *standby router* is added.
- *OSPF Routing Table Convergence Time*: We continually monitored the OSPF routing table at all nodes for any changes during each run. We collected snapshots of the routing table every time it was updated. These snapshots were timestamped and logged on for each individual router. These snapshots helped us determine the time taken for the routing table to converge at each router, once the recovery had finished.
- *OSPF Logs*: We collected the OSPF neighbor table for each of the routers to log any updates to the neighbor information.

V. RESULTS

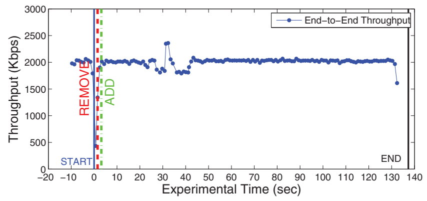
For each of the three topologies described in the previous section, we conducted ten independent runs to avoid any



(a) TCP Session Bandwidth



(b) UDP Session Loss



(c) UDP Session Throughput

Fig. 3. Geolocation-Based Strategy on Topology-I(a)

artifacts such as those due to router update cycles and from the GpENI substrate.

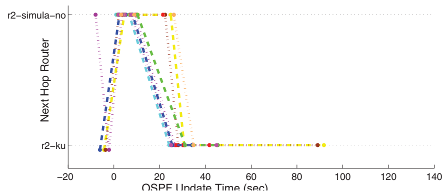


Fig. 4. Geolocation-Based Strategy on Topology-I(a): Nexthop Information from Routing Table Over 10 Runs

A. *Topology-I(a)*

We first discuss Topology-I(a). We utilized two different standby-router selection strategies: geolocation-based selection and random selection.

1) Geolocation-based Selection: Fig. 3(a) - Fig. 3(c) depict the instantaneous bandwidth, datagram loss rate and throughput, observed from topology shown in Fig. 2(a) . Note that the red dotted line indicates the time when a virtual router (R1-KU) is failed, and the green dotted line indicates the time when a standby virtual router (R2-KU) was added to replace R1-KU. The gap between these two dotted lines shows the recovery time is about 1.5 seconds, which means that the entire process of dynamic reconfiguration is fast. Recall that there is an alternative path to the destination in the topology. Hence, when the R1-KU failed (red dotted line), there was an immediate packet loss and bandwidth drop, and then the traffic was redirected to the alternative longer path, and the performance improved slowly afterward.

The second aspect we are interested in is how much time it takes to get the OSPF routing table to be consistent. Fig. 4 presents the next hop to the destination HOST-UMKC from R2-TUT-FI's perspective . In Fig. 4 we use dotted lines in different colors to indicate the OSPF update time during each single run, by tracking the next hop router to the destination. The dot clusters in the figure point out that there are two main OSPF update time periods: one is 3 - 5 seconds (R1-KU fails), and the other is 23 - 28 seconds. The rest of the dots in the figure may be treated as exceptions. Therefore, the routing convergence time for the OSPF routing table is between 20 - 23 seconds on average, which matches the duration time between two major troughs in the bandwidth pattern. This is because when the routing table converges, the traffic is again redirected to its original path with respect to the shortest path.

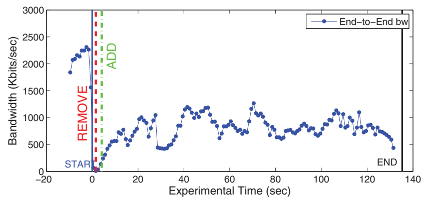
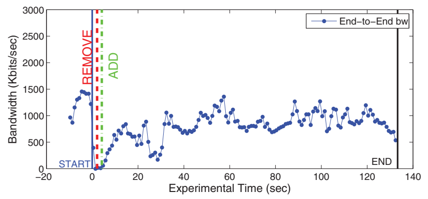
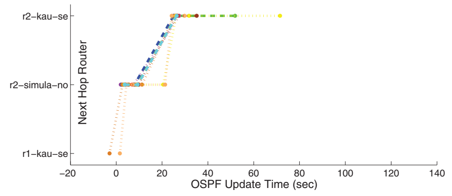


Fig. 5 Random Selection Method on Topology-I(a): TCP Session



(a) TCP Session Bandwidth



(b) Nexthop Information from Routing Table Over 10 Runs

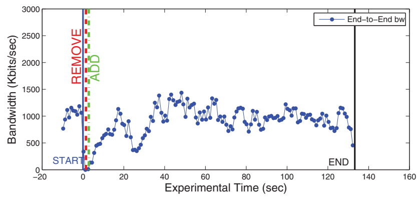
Fig. 6. Geolocation-Based Strategy on Topology-I(b)

2) Random Selection: In order to study whether the geolocation-based selection is worthwhile, we compared it to the case when randomly selecting a standby router. In this random selection case, we still failed the R1-KU physically located in the US, but randomly picked a node R2-KAU-SE, which is physically located in Sweden.

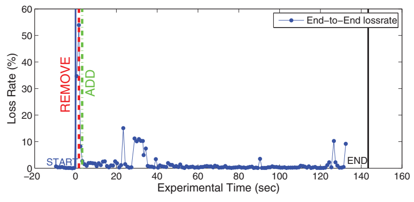
Fig. 5 shows the end-to-end instantaneous bandwidth during the reconfiguration. Compared with Fig. 3(a), we observed that the two major bandwidth drops happened at a similar time as in the geolocation-based method, indicating that the OSPF convergence time was not affected by the *standby router* selection strategy. However, by inspecting the instantaneous bandwidth value after OSPF has converged, the end-to-end performance under a geolocation-based selection is better than the random selection

B. *Topology-I(b)*

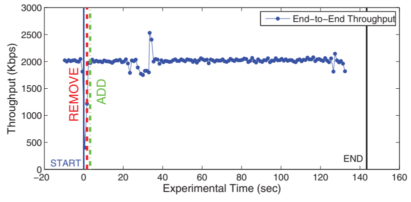
Fig. 2(b) is the same virtual topology as Topology-I(a), but with nodes from European sites only. During the reconfiguration process, R1-KAU-SE failed, and R2-KAU-SE was



(a) TCP Session Bandwidth



(b) UDP Session Loss



(c) UDP Session Throughput

Fig. 7. Geolocation-Based Method on Topology-II

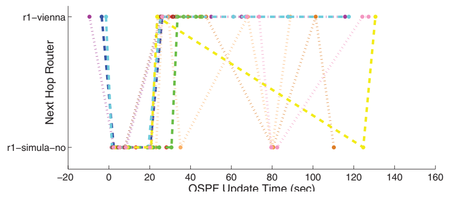


Fig. 8. Geolocation-Based Method on Topology-II: Nexthop Information from Routing Table Over 10 Runs

selected based on the geolocation strategy. Fig. 6(a) plots the instantaneous bandwidth. As we mentioned at the beginning of Section IV, the link speed on the GpENI substrate for Topology-I(a) and Topology-I(b) are different; thus, the effective bandwidth throughput observed in Fig. 6(a) is different from Fig. 3(a).

Fig. 6(b) presents the nexthop router to the destination for about 3 - 5 seconds and 23 - 25 seconds, which point to the two major bandwidth troughs in Fig. 6(a). The routing convergence time we obtained is similar to Fig. 4. Hence, we infer that the location of virtual routers in the topology did not affect the routing table convergence time.

C. *Topology-II*

We enforce the geolocation-based selection strategy in a larger network as shown in Fig. 2(c), and present the relevant results in Fig. 7. In this scenario, we failed R1-KU, and R2-KU was selected from the standby router pool. Fig. 7(a) shows the measured end-to-end bandwidth is not as large as the one we obtained for Topology-I(a). However, if we focus on the recovery time, it is still around 1.5 seconds, because the number of neighbors to the failed router (R1-KU) is still three. The bandwidth pattern is similar to Fig. 3(a), where two main troughs can be easily observed.

Fig. 8 presents the nexthop to the destination HOST-UMKC for each individual run observing from R2-TUT-FI, when the OSPF routing table updates. Because the network between the two end hosts was larger, and we added another crossing link (R1-Vienna to R1-SIMULA-NO), the OSPF routing table might change even though there is no failure in the network at a random time. This is why there are a few dots presenting R1-SIMULA-NO after 70 seconds, where R1-SIMULA-NO indicates the traffic is not routed via the shortest path. However, if we consider the clusters that represent most of the runs, there are still two major OSPF updating time periods: 2 - 7 seconds (failure occurs), and 24 - 32 seconds. The average update time of the OSPF convergence is around 22 - 25 seconds. By comparing the five-node and nine-node topologies, we find that although the network size increased, the OSPF convergence time did not increase as much as we anticipated, and the network performance was only affected when there was a traffic redirection. This was due to either a failure or OSPF routing table convergence after the network reconfiguration.

VI. Conclusion and Future Work

In this paper, we studied the impact of dynamic reconfiguration in a *virtualized network* by considering the case of replacing a failed virtual router with a standby virtual router. We have presented a geolocation-based algorithm to select a candidate from a *standby router* resource pool. The impact of the replacement scheme on the end-to-end traffic and routing metrics was studied on a wide-area network on the GpENI testbed. First, the geolocation replacing scheme less likely reduces the network performance after the replacement than a randomly replacing scheme. Second, the reconfiguration process was fast so that a standby virtual router could be easily added to the existing topology. Third, the *virtualized network* constructed by a different set of virtual routers does not affect the network recovery time nor the OSPF routing convergence time. This enables a flexible way to allocate virtual routers for the users. Finally, we note that the OSPF routing convergence time was not affected greatly by the size of the network during the reconfiguration, so our method is scalable.

There are two important lessons we learned from this experimental study:

- An autonomic network management function can help improve the overall performance of the network in case of a failure by using standby routers in a virtualized environment. However, *AMS* cannot control the inherent properties of the network. In our case, the OSPF convergence time is independent on the *AMS*.
- In an experimental study on a realistic network testbed that supports network virtualization, it is important to consider multiple independent runs to see how the network substrate (in our case GpENI) can influence results.

There are several directions we plan to pursue in the future. For example, we did not use access delay as a selection metric mainly because there is no a priori information that the *AMS* can obtain ahead of time. Furthermore, we plan to extend our approach to include multiple slices where each slice corresponds to an independent *virtualized network* allocated to different customers, and each slice is optimized for different applications.

Acknowledgement

Supported in part by National Science Foundation's GENI program (GPO Contract No. 9500009441) and National Science Foundation Grant CNS-1217736.

References

[1] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003. [Online]. Available: http://dx.doi.org/10.1109/MC.2003.1160055

[2] Z. Movahedi, M. Ayari, R. Langar, and G. Pujolle, "A survey of autonomic network architectures and evaluation criteria," *IEEE Commnications Surveys and Tutorials*, vol. 14, pp. 464–490, 2012.

[3] H. Derbel, N. Agoulmine, and M. Salaün, "Anema: Autonomic network management architecture to support self-configuration and self-optimization in ip networks," *Computer Networks*, vol. 53, no. 3, pp. 418–430, February 2009. [Online]. Available: http://dx.doi.org/10.1016/j.comnet.2008.10.022

[4] B. Jennings, S. van der Meer, S. Balasubramaniam, D. Botvich, M. O Foghlu, W. Donnelly, and J. Strassner, "Towards autonomic management of communications networks," *IEEE Communications Magazine*, vol. 45, no. 10, pp. 112–121, October 2007.

[5] M. Femminella, R. Francescangeli, G. Reali, J. Lee, and H. Schulzrinne, "An enabling platform for autonomic management of the future internet," *IEEE Network*, vol. 25, no. 6, pp. 24–32, November 2011.

[6] I. Gojmerac, T. Ziegler, F. Ricciato, and P. Reichl, "Adaptive multipath routing for dynamic traffic engineering," in *Proceedings of IEEE GLOBECOM*, 2003, pp. 3058–3062.

[7] R. Casado, A. Bermudez, F. Quiles, J. Sanchez, and J. Duato, "Performance evaluation of dynamic reconfiguration in high-speed local area networks," in *Proc. of Sixth International Symposium on High-Performance Computer Architecture, 2000. HPCA-6*, 2000, pp. 85–96.

[8] J. Garcia and J. Duato, "An algorithm for dynamic reconfiguration of a multicomputer network," in *Proceedings of the Third IEEE Symposium on Parallel and Distributed Processing, 1991.*, December 1991, pp. 848–855.

[9] M. Schroeder, A. Birrell, M. Burrows, H. Murray, R. Needham, T. Rodeheffer, E. Satterthwaite, and C. Thacker, "Autonet: a high-speed, self-configuring local area network using point-to-point links," *IEEE Journal on Selected Areas in Communications*, vol. 9, no. 8, pp. 1318–1335, October 1991.

[10] "PlanetLab." [Online]. Available: http://www.planet-lab.org/

[11] "VINI Veritas." [Online]. Available: http://www.vini-veritas.net/

[12] "GpENI – Great Plains Environment for Network Innovation." [Online]. Available: http://www.GpENI.net

[13] J. Sterbenz, D. Medhi, B. Ramamurthy, C. Scoglio, D. Hutchison, B. Plattner, T. Anjali, A. Scott, C. Buffington, G. Monaco, D. Gruenbacher, R. McMullen, J. Rohrer, J. Sherrell, P. Angu, R. Cherukuri, H. Qian, and N. Tare, "The Great Plains Environment for Network Innovation (GpENI): A programmable testbed for future Internet architecture research," in *Proc. of 6th International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities (TridentCom)*, Berlin, Germany, May 2010, pp. 428–441.

[14] R. Cherukuri, X. Liu, A. Bavier, J. Sterbenz, and D. Medhi, "Network virtualization in GpENI: Framework, implementation amp; integration experience," in *IFIP/IFIP IM Workshop on Management of Future Internet (ManFI'2011)*, May 2011, pp. 1216–1223.

[15] "iperf." [Online]. Available: http://sourceforge.net/projects/iperf/