DYNAMIC VIRTUAL NETWORK RESTORATION WITH OPTIMAL STANDBY VIRTUAL ROUTER SELECTION

A Dissertation
IN
Telecommunication & Computer Networking
and
Mathematics

Presented to the Faculty of the University
of Missouri–Kansas City in partial fulfillment of
the requirements for the degree

DOCTOR OF PHILOSOPHY

by
XUAN LIU

M.S., University of Missouri–Kansas City, USA, 2010
B.S., China University of Geosciences, Wuhan, Hubei, China, 2007

Kansas City, Missouri
2015

DYNAMIC VIRTUAL NETWORK RESTORATION WITH OPTIMAL STANDBY VIRTUAL ROUTER SELECTION

Xuan Liu, Candidate for the Doctor of Philosophy Degree

University of Missouri–Kansas City, 2015

ABSTRACT

Network virtualization technologies allow service providers to request partitioned, QoS guaranteed and fault-tolerant virtual networks provisioned by the substrate network provider (i.e., physical infrastructure provider). A virtualized networking environment (VNE) has common features such as partition, flexibility, etc., but fault-tolerance requires additional efforts to provide survivability against failures on either virtual networks or the substrate network.

Two common survivability paradigms are protection (proactive) and restoration (reactive). In the protection scheme, the substrate network provider (SNP) allocates redundant resources (e.g., nodes, paths, bandwidths, etc) to protect against potential failures in the VNE. In the restoration scheme, the SNP dynamically allocates resources to restore the networks, and it usually occurs after the failure is detected.

In this dissertation, we design a restoration scheme that can be dynamically implemented in a centralized manner by an SNP to achieve survivability against node failures in the VNE. The proposed restoration scheme is designed to be integrated with a protection scheme, where the SNP allocates spare virtual routers (VRs) as standbys for the virtual networks (VN) and they are ready to serve in the restoration scheme after a node failure has been identified. These standby virtual routers (S-VR) are reserved as a shared-backup for any single node failure, and during the restoration procedure, one of the S-VR will be selected to replace the failed VR. In this work, we present an optimal S-VR selection approach to simultaneously restore multiple VNs affected by failed VRs, where these VRs may be affected by failures within themselves or at their substrate host (i.e., power outage, hardware failures, maintenance, etc.). Furthermore, the restoration scheme is embedded into a dynamic reconfiguration scheme (DRS), so that the affected VNs can be dynamically restored by a centralized virtual network manager (VNM).

We first introduce a dynamic reconfiguration scheme (DRS) against node failures in a VNE, and then present an experimental study by implementing this DRS over a realistic VNE using GpENI testbed. For this experimental study, we ran the DRS to restore one VN with a single-VR failure, and the results showed that with a proper S-VR selection, the performance of the affected VN could be well restored. Next, we proposed an Mixed-Integer Linear Programming (MILP) model with dual–goals to optimally select S-VRs to restore all VNs affected by VR failures while load balancing. We also present a heuristic algorithm based on the model. By considering a number of factors, we present numerical studies to show how the optimal selection is affected. The results show that the proposed heuristic's performance is close to the optimization model when there were sufficient standby virtual routers for each virtual network and the substrate nodes have the capability to support multiple standby virtual routers to be in service simultaneously. Finally, we present the design of a software-defined resilient VNE with the optimal S-VR selection model, and discuss a prototype implementation on the GENI testbed.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Graduate Studies, have examined a dissertation titled "Dynamic Virtual Network Restoration with Optimal Standby Virtual Router Selection," presented by Xuan Liu, candidate for the Doctor of Philosophy degree, and certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Deep Medhi, Ph.D., Committee Chair
Department of Computer Science & Electrical Engineering

Kamel Rekab, Ph.D.
Department of Mathematics and Statistics

Baek-Young Choi, Ph.D.
Department of Computer Science & Electrical Engineering

Ken Mitchell, Ph.D.
Department of Computer Science & Electrical Engineering

Praveen Rao, Ph.D.
Department of Computer Science & Electrical Engineering

James P.G. Sterbenz, Ph.D.
Information and Telecommunication Technology Center, The University of Kansas,
Lawrence, KS 66045, USA
School of Computing and Communications, Lancaster University,
Lancaster LA1 4WA, UK
Computing Department, The Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong

CONTENTS

ILLUSTRATIONS

TABLES

ACRONYMS

ATM Asynchronous Transfer Mode

AL2S Advanced Layer-2 Service

CDN Content Delivery Network

DRS Dynamic Reconfiguration Scheme

EGRE Ethernet GRE

GENI Global Environment for Network Innovations

GpENI Great Plain Environment for Network Innovation

GRE Generic Routing Encapsulation

HTTP Hypertext Transfer Protocol

ICN Information Centric Networking

IIAS Internet in a Slice

ILP Integer Linear Programming

ISP Internet Service Provider

KVM Kernel-based Virtual Machine

LAN Local Area Network

LB Load Balancing

L2TP Layer 2 Tunneling Protocol

MILP Mixed Integer Linear Programming

MNC Minimum Network Cost

MPLS Multiprotocol Label Switching

MPM Multipath Measure

NAT Network Address Translation

OFELIA OpenFlow in Europe: Linking Infrastructure and Applications

OSPF Open Shortest Path First

OXC Optical ross-connect

PPTP Point-to-Point Tunneling Protocol

QoS Quality of Service

QoE Quality of Experience

ROADM Reconfigurable Optical Add-Drop Multiplexer

RSpec Resource Specification

RTT Round-Trip Time

SeRViTR Secure and Resilient Virtual Trust Routing

SDH Synchronous Digital Hierarchy

SDN Software-Defined Networking

SN Substrate Network

SNP Substrate Network Provider

SONET Synchronous Optical Networking

SSH Secure Shell

S-VR Standby Virtual Router

VINI Virtual Network Infrastructure

VLAN Virtual Local Area Network

VM Virtual Machine

VN Virtual Network

VNE Virtualized Networking Environment

VNM Virtual Network Manager

VNO Virtual Network Operator

VNP Virtual Network Provider

VR Virtual Router

VPN Virtual Private Network

VTRouPD Virtual Trust Routing and Provisioning Domain

WDM Wavelength-Division Multiplexing

ACKNOWLEDGEMENTS

I would like to thank my research advisor Dr. Deep Medhi for his comprehensive guidance, conscientious mentoring and great support throughout my doctoral research and master thesis. Dr. Medhi encouraged me to participate in different projects to find the topics that I was interested in, and then gave me suggestions on how to narrow down the topic and find the entry point to my dissertation research. Dr. Medhi's mentoring explicitly covers every aspect and level of research and fosters critical thinking. I also thank Dr. Medhi for supporting me to present my works at different conferences and workshops. In addition, I very much appreciate the opportunities that Dr. Medhi offered me to visit Nakao Lab at the University of Tokyo in Japan and LIP6 at University Pierre et Marie Curie (UPMC) - Paris in France. I also would like to acknowledge all the advice and help from the members of my advisory committee, Dr. Kamel Rekab, Dr. Baek-Young Choi, Dr. Ken Mitchell, Dr. Praveen Rao, and Dr. James P.G. Sterbenz.

I would like to acknowledge the two major collaborative projects, GpENI and SeRViTR, that I have participated in. From these two projects, I obtained both knowledge and hands-on experiences in the network virtualization and software-defined networking areas, which was a great stepping stone for my dissertation work. In particular, I would like to thank Dr. James P.G. Sterbenz from the University of Kansas for his leadership on the GpENI project. I also would like to thank Dr. Andy Bavier from Princeton University for his tutoring on the VINI testbed and help on troubleshooting when I deployed and upgraded the private VINI testbed on the GpENI network. For the SeRViTR project, I would like to thank the opportunity to work with the teams led by Dr. Dijiang Huang and Dr. Shingo Ata from Arizona State University and Osaka City University, respectively, where I gained experiences in both system design and OpenFlow network implementation.

I would like to thank Dr. Aki Nakao of University of Tokyo for introducing his research lab and ongoing research projects, which expanded my knowledge about new network virtualization techniques.

I have completed several research internships during my PhD study, and these experiences were important to my dissertation research. I would like to thank Dr. Ravi Ravindran and Dr. G.Q Wang for their mentoring when I did the internships at Futurewei Technologies. I would also like to thank my mentors Sarah Edwards, Dr. Niky Riga and Dr. Vicraj Thomas for their help and suggestions during my internship at BBN Technologies. Their insights and experiences about the GENI testbed was a significant support for me to design and implement the prototype of the dynamic reconfiguration scheme presented in this dissertation on the GENI testbed.

I would like to thank my lab mates at the Network Research Lab (NeTReL) and the discussion with them greatly assisted me to improve my work and skills in many ways.

Last and the most important, I'm very thankful to my grandparents, Shaofu Liu and Daofang Zhang, for all their care and love since I was a child, and I sincerely appreciate my parents, Ping Liu and Zhihong Lin, for their understanding and encouragement during the entire period of my doctoral study.

This dissertation work was supported in part by NSF Grant Nos. 0916505 and 1217736, and the Graduate Assistance Fund fellowships from UMKC Women's Council.

CHAPTER 1

INTRODUCTION

In the past two decades, virtualization technologies have been widely introduced in the computer networks and the internet has been extensively evolved to be more convenient and secure for both personal and business purposes. For instance, in the earlier time, after the virtual private network (VPN) was introduced, an enterprise private network for a company or an academic institution can be extended over the public Internet by establishing end-to-end secure tunnels, so that employees can access the business resources even though they are out side of the office. Recently, the diverse and mature virtualization technologies allow network virtualization to be a promising technology for the next-generation networking architecture.

1.1 Network Virtualization: A Historical View

The core idea of virtualizing a network is to divide the physical network into multiple logical networks, and the goal is to provide a more private networking environment with better QoS that can be dynamically updated. Network virtualization could be configured in a flat manner or a hierarchical manner (i.e., overlay networks) or in both ways. In this section, we first present the a variety of technologies used for network virtualization.

1.1.1 Virtual Local Area Network

A classical local area network (LAN) topology consists of ethernet switches that connect hosts via different ports. When a host sends a data frame to a switch, once this frame reaches a port, it will be broadcast to every device connected to this switch. This broadcasting feature has the potential to cause the traffic congestion within the LAN. The early solutions were to use routers to split a single LAN into multiple small LANs [107], but the forwarding at a router is not as fast as it is in a switch when the data rate gets faster, so the virtual local area network (VLAN) [37] was introduced, which not only splits a large LAN into small LANs, but also allows multiple end systems belonging to one or more physical LANs to be bridged into a single broadcasting logical LAN with a specific identification (i.e., VLAN tag). Fig. 1 presents a typical example of three VLANs across three physical domains, and these VLANs are differentiated by their VLAN tags (i.e., vlan1, vlan2 and vlan3).

The benefits of VLAN are manifold. First, it enables flexible network management. VLAN configuration is usually conducted by software, so it is easy to move a host from one VLAN to another VLAN by updating the VLAN identifier (VLAN ID) of the corresponding port that this host connects to. Second, we are able to virtually group devices from different physical LANs into a single VLAN, and the data forwarding is done at layer-2, rather going through a router, so that high data-rate traffic exchange can be achieved. Third, by creating multiple VLANs at a single switch, data frames will not be broadcasted to every the ports on the switch. In other words, traffic within different VLANs are partitioned and secure.

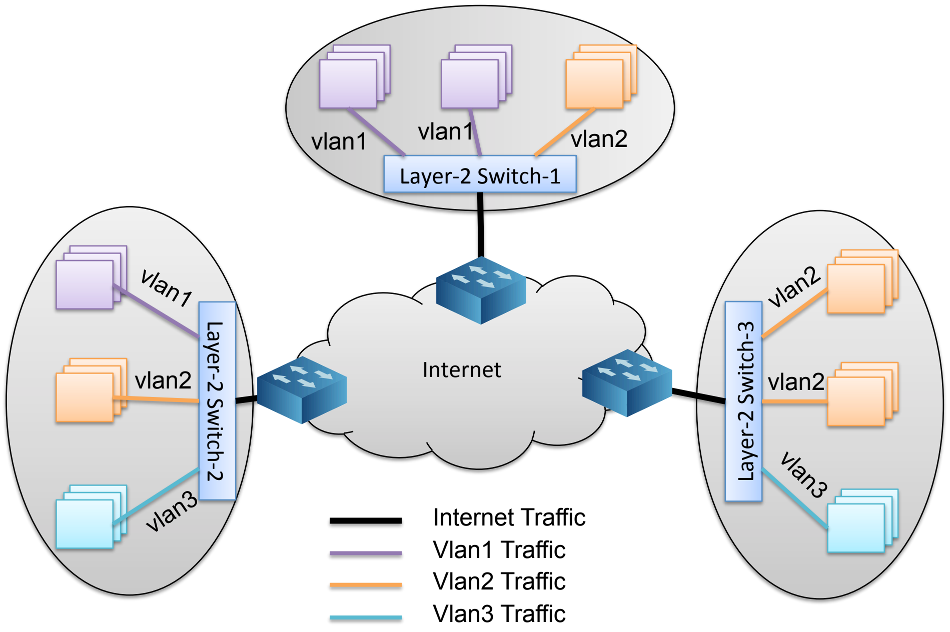


Figure 1: An Example of Virtual LANs

Creating VLANs over a network is like partitioning the network horizontally into small logical domains. Although it does not involve either node or link virtualization yet, it partitions the ports on the switches, where a logical domain with entities physically located in various domains is constructed, and traffic is segregated between logical domains.

1.1.2 Virtual Private Network

A virtual private network (VPN) [43, 47, 97] is a logical private network that shares some common resources on the public networking infrastructure, but usually requires users to access with credentials. Compared with the public network, a private network, which also refers to enterprise private network, where network entities (e.g.,

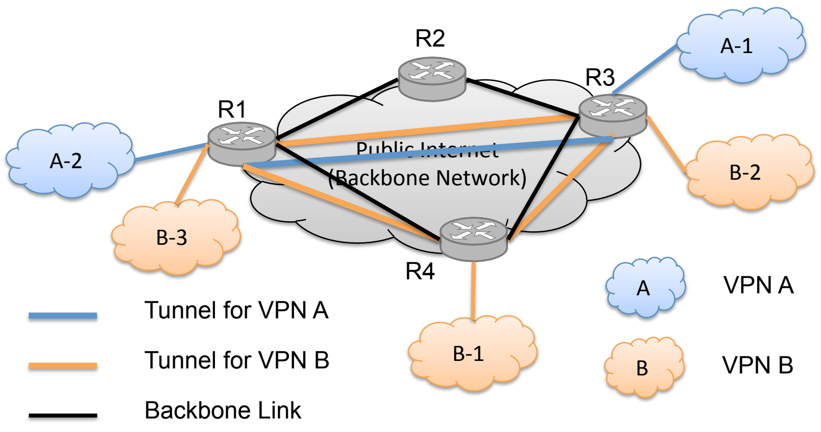


Figure 2: An Example of Two VPNs over the Public Backbone Network

routers, switches, etc.) usually are actually owned and managed by an business organization (e.g., company, university, etc.) and only people with authorized permission within the private network can access the resources. Without the VPN, if a business organization is expanded to multiple sites where each site is physically in a different location and forms a private network, although these private networks belong to the same community, users from one site could not access resources in another site, because these sites are interconnected by the public network. Thus, the VPN helps to construct a logical private network that tunnels multiple private networks owned by a single administrative domain but geographically in different locations, and the tunnels are end-to-end virtual links constructed over the common underlying public networking infrastructure.

With VPN service, an access point to a private network can be expanded to anywhere by creating encrypted tunnels over the public Internet. Fig. 2 shows an example of two VPNs constructed over the public backbone network. In this example, `VPN A` crosses two physical sites `A-1` and `A-2`, and a tunnel connecting these two sites is established between the backbone routers `R1`, and `R3`. For `VPN B`, there are three different physical sites, and the tunnels for `VPN B` are among `R1`, `R3`, and `R4`.

In general, it is the VPN service provider who provides VPN services to customers [43], and the VPN can be implemented at different layers of the TCP/IP protocol stack, along with proper tunneling protocols, the layer-1 VPN was briefly discussed in [43], and the most common types of VPNs are the layer-3 VPN and the layer-2 VPN. For the Layer-3 VPNs (i.e., IP-based VPN), the packets sent from the VPN customer sites are IP packets, and these packets then be wrapped by adding encapsulating headers (e.g., GRE [54], IPsec [64], MPLS [80, 97], PPTP [53], L2TP [108], etc.). As the encapsulated packets arrive at the VPN customer networks, the actual IP packets from the sender will be extracted and sent to the destination. On the other hand, a layer-2 VPN constructs an end-to-end tunnel for delivering encapsulated Ethernet or ATM packets. Since MPLS protocol has the advantage to carry any type of packets, it could be one of the approaches to wrap the layer-2 frames [80, 97].

Although the VPN services also achieve traffic segregation by encapsulating the packets and creating point-to-point tunnels between customer networks, the VPN traffic is still competing with other Internet traffic, so VPN is a domain-level virtualization.

1.1.3 Overlay Networks

The overlay network [35] is another form of the network virtualization, and it is a logical network over a shared underlying backbone network. The essential entities

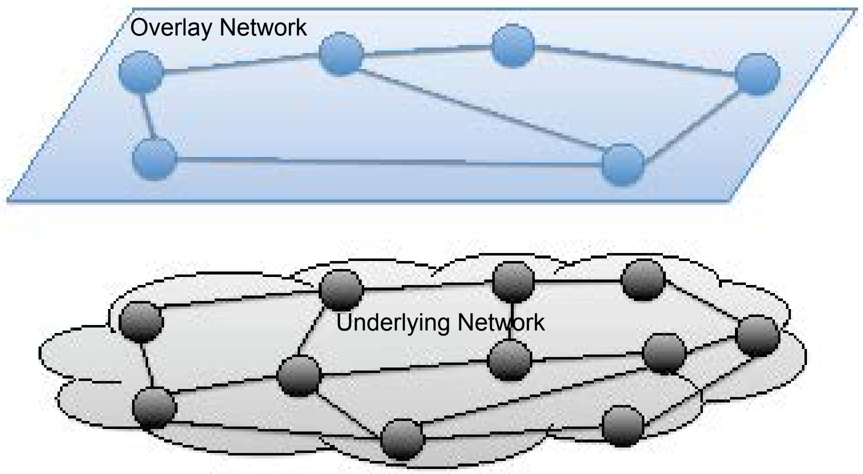


Figure 3: An Example of Overlay Network

in an overlay network are nodes interconnected by tunneling mechanisms (e.g., GRE tunnels, L2TP tunnels, etc.). The overlay network allows data to be transferred through more secure logic links (e.g., tunnels) over the transport layer or IP layer on the physical network. Fig. 3 shows a typical overlay network architecture, where the overlay topology could be different from the actual underlying physical topology, and the nodes in the overlay network are a subset of nodes from the underlying network.

The overlay network architecture enables data sharing and delivery to be more flexible and resilient, and most of the overlay networks are implemented at the application level. Peer-to-peer network [72,96] was designed for the file sharing purpose, and content delivery network (CDN) [109] allows content data (e.g., multimedia data) to be cached anywhere within the overlay network so that the latency and transport cost can be reduced. Overlay technologies also support a fault-tolerant and resilient network design [14,36,98]. According to the fundamental concept of the overlay, a VPN could be categorized as overlay network as well.

Considering the benefits of the overlay networks, the experimental testbed has been deployed for researchers and students to test and evaluate innovative ideas. The most popular world wide overlay network testbed that has been used for research since 2002 is PlanetLab [34, 87, 88], which consists of more than 1000 nodes (i.e., computers) distributed crossing over 500 sites all over the world. For the first time, Planetlab testbed used the term ”slice” to represent a container for virtual resources (i.e., virtual nodes support by LinuxVserver [38]).

As virtualization technologies evolved, network virtualization has been pushed down to lower levels (i.e., network layer or even data-link layer), and a deeply programmable virtualized networking environment can be well modeled.

1.2 Virtualized Network Environment

A virtual network inherits the generic characteristics of the overlay network, and it potentially provides a partitioned, QoS guaranteed, trustworthy environment for service providers. Thus, the network virtualization [32] is considered one of the promising technologies for Future Internet architectures, and it is also referred as virtualized networking environment (VNE). In a virtualized networking environment, the underlying physical networking infrastructure (i.e., backbone network) is defined as *substrate network* (SN), relative to the virtual networks (VNs).

As the basic elements in a VN, virtual nodes can be created in various ways. With the container-based virtualization [30], a substrate node is virtualized at the operation-system (OS) level, where each virtual node runs a replica of the operating environment that the substrate node runs and it is completely isolated from other virtual nodes. OS-level virtualization has the least overhead in terms of maintenance and resource provisioning. The nodes on PlanetLab testbed currently apply the OS-level virtualization (e.g., LXC [55]), where each *sliver* [88] is an container where applications can be independently deployed. Although container-based virtualization results less overhead on the implementation, it lacks flexibility in terms of creating heterogenous VNE (e.g., two virtual nodes from a common substrate node run different operating systems). Thus, the full virtualization (e.g., KVM [52]) and the paravirtualization (e.g., Xen [17]) techniques are employed, and both are hardware-level virtualization as discussed in [30, 90].

1.2.1 Layers of Network Virtualization

Unlike the traditional overlay network, tenants of VNs have the opportunity to not only deploy applications, but also test and deploy new protocols at a lower layer within their dedicated VNs, such as new routing protocol or even a non-IP based clean-slate networking architecture (e.g., Information-centric networking [13]). Thus, virtualization techniques allow these tenants to have operational privileges on the virtual nodes with diverse control levels. Chowdhury et al. [32] lists a number of recent network virtualization projects targeting at different layers. In the following sections, we briefly discuss current network virtualization architectures and implementations using a top-down approach.

1.2.1.1 Application Layer

Planetlab [34, 61, 87, 88] is an example for application-layer virtualization architecture, where VNs are isolated from each other over an interconnected high-speed underlying research network called *Global Research and Educational Network* (GREN) [16]. Planetlab users can develop applications within the virtual nodes (i.e., *sliver*) on physical Planetlab nodes without affecting other users' experiments. To support distributed services running on the Planetlab testbed, data planes (i.e. tunnels between virtual nodes) is provided for users to run large scale experiments. However, PlanetLab does not offer control-plane privileges, so it is a challenge for users to take control over the network events, so a lower-layer virtualized network framework is desired.

1.2.1.2 Transport layer

CoreLab [83] inherits the node management framework (i.e. PlanetLab Central management server) from PlanetLab testbed, and it has explored the flexibility of installing guest OSes to VMs on the physical nodes using KVM [52], where each VM can obtain a private IPv4 address and a particular range of port number. In an early implementation of CoreLab, accessing *slivers* from external network used destination NAT (DNAT), and the CoreLab node OS runs as the NAT server to map the incoming packets to a VM with public IP address to the VM's private IP address. However, the NAT translation reduces the throughput and it breaks the end-to-end communication. Hence, a port-space isolation approach was proposed [41] to avoid using NAT at the CoreLab nodes.

With the port-space isolation, each VM share the same public IP and MAC address with its host, and incoming packets only need to specify the host's public IP address with a specific port number that has been assigned to the VM, and the Open vSwitch [89] redirects the flow to proper VMs.

1.2.1.3 Network Layer

VINI (virtual network infrastructure) [20] supports virtualization at the network layer. By requesting resources, an experimenter is provisioned not only virtual nodes, but also layer-2 virtual links (i.e., Ethernet GRE tunnels), so that a complete virtual topology is presented to the experimenter. Moreover, users can install the Quagga [63] (i.e., a real routing software support multiple routing protocols like OSPF, BGP, etc.) to make their *slivers* as virtual routers. In this way, users has the capability to evaluate networking performance by injecting failures at the virtual links or the virtual routers.

1.2.1.4 Link Layer

The VNET project [106] used a composite of virtual private network (VPN) and virtual local area network (VLAN) technologies to construct a layer-2 overlay VNE. VMs are connected through the layer-2 tunneling protocol (L2TP). Since VNET implements a layer-2 virtualization, the upper layers (e.g., IP layer) is agnostic, so that there is an opportunity to design and test a non-IP protocol over VNET. VNET also supports flexible VM migration because the network layer configuration does not need to be changed.

1.2.1.5 Optical Layer

Nejabati et al. [84] presents the challenges and possible techniques for optical network virtualization. Similar to a virtualized IP network, a virtualized optical network comprises virtual optical nodes (e.g., optical network switches, etc) interconnected by logical optical links shared on an administrative optical network. The major challenge of implementing virtualization at the optical layer is due to the analogue nature of the optical network resources (e.g., fiber, wavelength, etc.) whose switching granularities are diverse.

Both optical node and optical link virtualization utilize the same concepts: *Partitioning (1:N)* and *Aggregation (N:1).* For virtualizing the optical nodes, partitioning means dividing 1 OXC/ROADM into N small units, where each unit is assigned a range of ports from the optical nodes; aggregation is to create a single optical super-node that aggregates N physical nodes and presents one interface to the external world. For virtualizing optical links, partitioning refers to creating sub-wavelength channels that carry a portion of data rates of the original wavelength channel. For instance, a wavelength channel of 40Gbps can be divided into two sub-wavelength channels with 30Gbps and 10Gbps, respectively. Aggregation is to create a super-wavelength channel by combining a group of wavelength changes.

1.2.2 Business Role Model and Virtual Network Management

In a VNE, the traditional Internet service provider (ISP) is divided into two business roles [32]: substrate network provider (SNP) and service provider (SP). Schaffrath et al. [99] presented a model for the VN management with four different business roles as follows:

- A substrate network provider (SNP) owns the physical network infrastructure consisting of physical nodes supporting virtualization and the high-speed backbone connectivity (e.g., fiber connection). The SNP also leases the substrate resources (e.g., computing and bandwidth or circuits) according to the demand from the customer (e.g., service provider), so that the QoS of the leased VN can be guaranteed.
- A virtual network provider (VNP) communicates with one or more SNPs to organize the virtual resources and fits them into a virtual topology that might be requested by the tenant or optimized by the VNP itself.
- A virtual network operator (VNO) mainly takes care of VN initialization such as node installation and configuration. Moreover, the VNO has the permission to run operation on the VN for not only maintenance, but also realizing VN restoration from failure through some reconfiguration mechanism.
- A service provider (SP) is the tenant of the virtual network who rents resources from the SNP and owns an dedicated VN to deliver services to its own customers.

It is possible for a company to fill more than one roles listed above [99]. As presented in Fig. 4, we assume the substrate network owner plays triple-roles of SNP along with VNP and VNO, and the latter two roles are combined into a single logical role called *virtual network manager* (VNM), which is in charge of VN initialization, maintenance and reconfiguration.

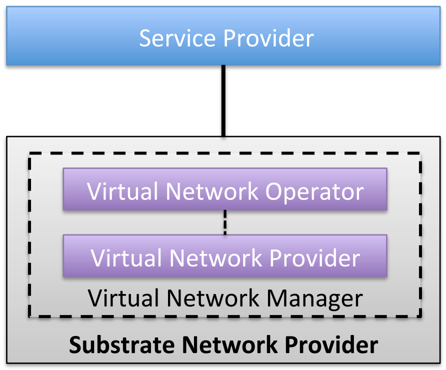


Figure 4: Virtual Network Management Business Roles

Fig. 5 shows a standard overlay virtualized network environment consisting of a virtual network plane and underlying substrate network. Two VNs are presented in the virtual network plane, and each VN has its own virtual topology consisting of virtual nodes from a different subset of substrate nodes. By renting the resources from the SN, an SP expects the VN to be well maintained and robust to failures. Therefore, an efficient virtual network management mechanism and a resilient virtual network environment is desired.

In the business role model presented in Fig. 4, the VNM conducts flexible VN management through programmable interfaces towards the virtual network plane, and such VN management includes two major tasks: VN initialization and VN reconfiguration. VN initialization usually refers to the *virtual network embedding* problem [33, 44, 56], and this is one of the main research challenges in the network virtualization. Efficiently VN embedding is to achieve optimal mapping between virtual nodes and virtual links and the substrate network, so that the SNP can accommodate as many VN requests as possible. The VN reconfiguration aims to enhance resiliency and survivability of the

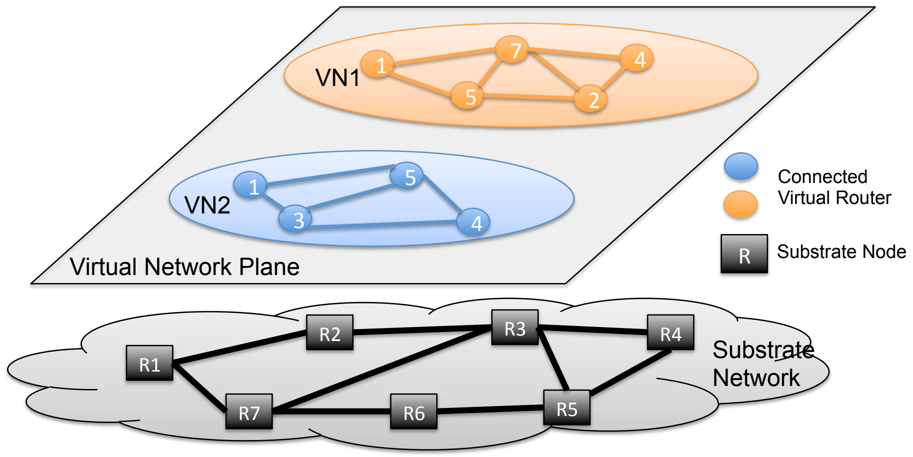


Figure 5: An Example of Virtual Network Environment

VNE under failures or other evens (e.g., network traffic engineering). Taking the advantage of the programmable interfaces towards the virtual network plane, the SNP is able to monitor the VNs, identify the abnormal behaviors or failures, and reconfigure the virtual links (VLinks) or the virtual routers (VRs) to restore the affected VNs.

1.2.3 Virtualized Networking Testbed

Consider the programmability, partition and flexibility of a virtualized networking environment, it is desired to build virtualized networking testbed to support innovative research diversity, where researchers are able to request computing resources or networking resources or both to conduct large scale experiments. There have been quite a few virtualized networking testbeds deployed globally or locally in a country or continent. Table 1 compares a number of network testbeds that support virtualization.

PlanetLab is a globally distributed networking testbed that supports application-layer virtualization. It uses the OS-based virtualization technique (i.e., LXC), so each

Table 1: Comparison table of Testbed supporting VNE

Testbed	Coverage	Major Resources	Virtualization Technique	Virtualization Layer
Planetlab [11,88]	Global	Containers	LXC	Layer-7
CoreLab [83]	Japan	VMs	KVM, LXC	Layer-4
VINI [20]	US	Containers	Trellis	Layer-3
GpENI [6,79]	US, Europe	Containers	Trellis, Linux vServer, VLAN	Layer-2,3,&7
Emulab [1,57]	Global	PCs and VMs	OpenVZ, FreeBSD Jails, Xen	Layer-2,3
OFELIA [9,39]	Europe	OFS, VCtrler	VeRTIGO	Layer 2
FITS [2,82]	Brazil	VMs, OFS	Xen, FlowVisor	Layer-2,3
GENI [3,21]	US	VMs, OFS	OpenVZ, Xen, KVM	Layer 2,3
UCLP [12,49]	Canada	Lightpaths	Partitionting/Aggregation	Optical Layer

OF: OpenFlow; OFS: OpenFlow Switch: Ctrler: Controller VMs

sliver is a container that runs the same OS as its host. PlanetLab users are able to test novel application-level services (e.g., P2P, distributed storage system, etc.) within their *slices*, but they are facing the challenges in creating arbitrary topologies and running experiments for traffic engineering. On the other hand, OS-based virtualization limits the flexibility of customizing the kernel of the OS within the *slivers*.

As we have mentioned in Section 1.2.1, there are two testbeds were deployed based on the experience of building PlanetLab: VINI [20] in the United States and CoreLab [83] in Japan, and both are running the private instance of PlanetLab Central management server called MyPLC [62] to manage the nodes on the testbed. VINI nodes run *Trellis* [22] that combines two OS-based virtualization techniques (i.e., Linux vServer and NetNS [7]) to allow each virtual network has its own network namespace. Compared with PlanetLab, the CoreLab has introduced the full virtualization technique (i.e., KVM) so that users can customize the guest OSes. Du et al. [41] have achieved port-space isolation on the CoreLab nodes using Open vSwitch [89] in their recent work.

GpENI (*Great Plain Environment for Network Innovation*) [79] is an international network testbed aiming to provide programmability across the entire protocol stack (i.e., application, transport, network, and data-link layer). The GpENI testbed was initially created over the regional optical network between the four universities in the midwest of the United States, which are the University of Kansas (KU), the University of Missouri - Kansas City (UMKC), the University of Nebraska - Lincoln (UNL) and Kansas State University (KSU). Currently, the GpENI testbed spans both the United States and Europe, and it is expanding to Asia. GpENI supports virtualization at three layers. For the application layer virtualization, it inherits the implementation of PlanetLab platform, where MyPLC is in charge of node management; for the network layer virtualization, it runs a customized instance of VINI software; and for the link layer virtualization, it utilized the VLAN technique.

Emulab [57] is a distributed PC cluster testbed. Both Xen and container-based virtualization (e.g., FreeBSD Jails [95] & OpenVZ [10]) are utilized to create virtual nodes, and the virtual links are implemented at layer-2 or layer-3. OFELIA [9] is an European OpenFlow (OF) [78] network testbed, and the main resources are OpenFlow switches and VeRTIGO [39] is a FlowVisor [103]-based network virtualization tool to slice the OpenFlow network at layer-2. FITS [2, 82] is a virtual network testbed in Brazil, and it supports virtualization in both IP networks and OpenFlow networks.

GENI [21] is a fast growing distributed large scale testbed providing virtualized networking environment in three aspects. First, computing and storage resources are virtualized using all three possible approaches (i.e., container-based, Xen and KVM). Secondly, a variety of tunneling mechanisms (i.e., GRE, EGRE) are available to create virtual links between VMs, depending on the virtualization techniques used for slicing the nodes. In particular, a tool called *stitcher* [85] is available on GENI to create isolated high speed virtual links by establishing VLAN between VMs. Compared with the data-rate limit of GRE/EGRE tunnels (i.e., 100Mbps), stitched links supports gigabits data rates, so experiments requiring large data transferring (e.g., video streaming) can be well supported. Thirdly, like OFELIA, GENI also supports virtualization in OpenFlow networks. Moreover, GENI testbed is still growing by federating with other testbed, so that GENI users can run larger and more complex experiments using both resources from both GENI and other testbeds.

User-Controlled LightPaths Project (UCLP) [12, 49] allows users to dynamic reconfigure optical networks, with the partitioning or aggregation scheme to divide or combine lightpaths.

1.3 Failure Types and Fault-Tolerance Schemes

Cetinkaya et al. [25] presented a taxonomy of challenges in the networks, and summarized events that may have impact to the network operations or cause failures in the network. Markopoulou et al. [75] collected the data regarding the failures on the Sprint backbone network and classified the failures into two top-level categories: *maintenance* (20%) and *unplanned failures* (80%). Among the unplanned failures, about 70% were identified as single-link failures, whereas 17% and 12% were identified as router-related failures and optical-related failures, respectively. In particular, the authors claims that the router-related failures may due to a router crash, a linecard failure, overload CPU, etc

[74]. For optical-related failures, since IP network can be considered an overlay network on the optical network, so an optical link failure could affect multiple overlay IP links concurrently.

Protection (proactive) and restoration (reactive) are the main fault-tolerant schemes for survivability in both optical networks [93] and IP networks [75]. In general, a protection method refers that the primary network entities are provided alternative network elements (i.e., paths, nodes, etc) in case of failures. For example (see Fig. 6), two paths (a:1-2-3-4 and b:1-6-5-4) are precomputed between node-1 and node-4, and let path b be reserved as protection path. If the link 2-3 fails, it affects the primary path a, and the traffic forwarding between node-1 and node-4 will be protected by the backup path b. Fig. 6 also showed a link protection in case of link 2-3 fails, where links 2-6, 6-5 and 3-5 are reserved for protecting the link 2-3. With the protection scheme, two types of backup have been widely discussed: dedicated and shared backup, respectively. For the dedicated backup, it is a one-to-one relation, where a critical entity the network has been assigned a backup entity for protection purpose. For the shared backup, the resources reserved for backup purpose can be used to protect more than one failure. For the restoration, the backup resources can be dynamically discovered in an on-demand manner. In other words, when a failure occurs, the restoration scheme will dynamically find a backup path or node to restore the network.

In a VNE, a fault-tolerant scheme should be able to provide protection against failures in either VNs or the underlying SN. There have been many works done toward

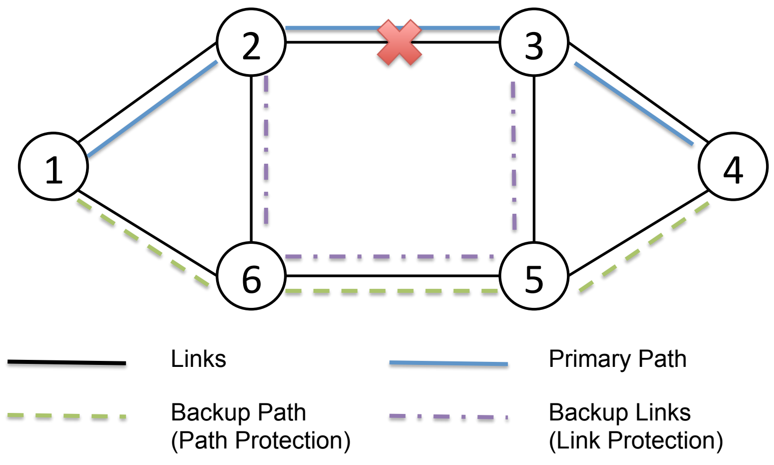


Figure 6: Path/Link Protection Scheme

survivable virtual network embedding problem that considers to allocate redundant resources while mapping VNs to the SN [56], and most of them focused on the link failure recovery. However, a node failure is more common and it has more impact in a VNE than it is in the physical network. First, a virtual node may fail due to the failure in the corresponding functional software. For example, a crash in a routing software can fail a virtual router. Second, although a physical node failure is less common than a physical link failure, it may affect multiple VNs that contain virtual nodes provisioned by this failed physical node on the SN, and the the failed virtual nodes will in turn cause multiple virtual link failures in these VNs. Therefore, it is important to design an efficient VN restoration scheme for the node failures in the VNE.

1.4 Problem Definition

The primary goal of this dissertation is to design an optimal standby virtual router (S-VR) selection model that can be adapted into a dynamic reconfiguration scheme for a software-defined resilient virtual network environment, so that virtual networks with node failures can be quickly restored with properly selected standby virtual routers.

Fig. 7 presents a system view of the overall problem definition. To design a software-defined resilient VNE, we consider three pieces of techniques: network virtualization, software-defined approach and autonomic computing [65]. For the network virtualization, we make several underlying assumptions: (1) the VNE implements the layer-3 virtualization, each virtual node in a VN is configured as a virtual router (VR) running a particular routing protocol; (2) At the VN embedding phase, an SNP is able to not only provision the basic resources based on the VN tenants' requests, but also provide additional VRs as standby virtual routers (S-VRs) for this VN to improve the resiliency; (3) We consider recovery from a failure that occurs at core VRs, *not* edge VRs of the VNs. A key feature of software-defined networking (SDN) [77] is that the control-plane functionalities are decoupled from the data-plane and they are abstracted into a centralized control system. Thus, we incorporate this idea into virtual network management, where a centralized virtual network manager (VNM) performing as both roles of VNP and VNO. From the fault-tolerance perspective, the VNM implements the dynamic reconfiguration function to automatically restore VNs from node failures using S-VRs. The concept of autonomic computing is applied into the dynamic reconfiguration process to achieve self-healing in the VNE, so that node failures are transparent to the SPs.

We consider a primary-backup mechanism for the fault-tolerance in terms of node failures recovery in the VN. Specifically, each VN is provisioned an extra group of VRs as standby virtual routers (S-VRs) and each S-VR can be used to replace any failed VR

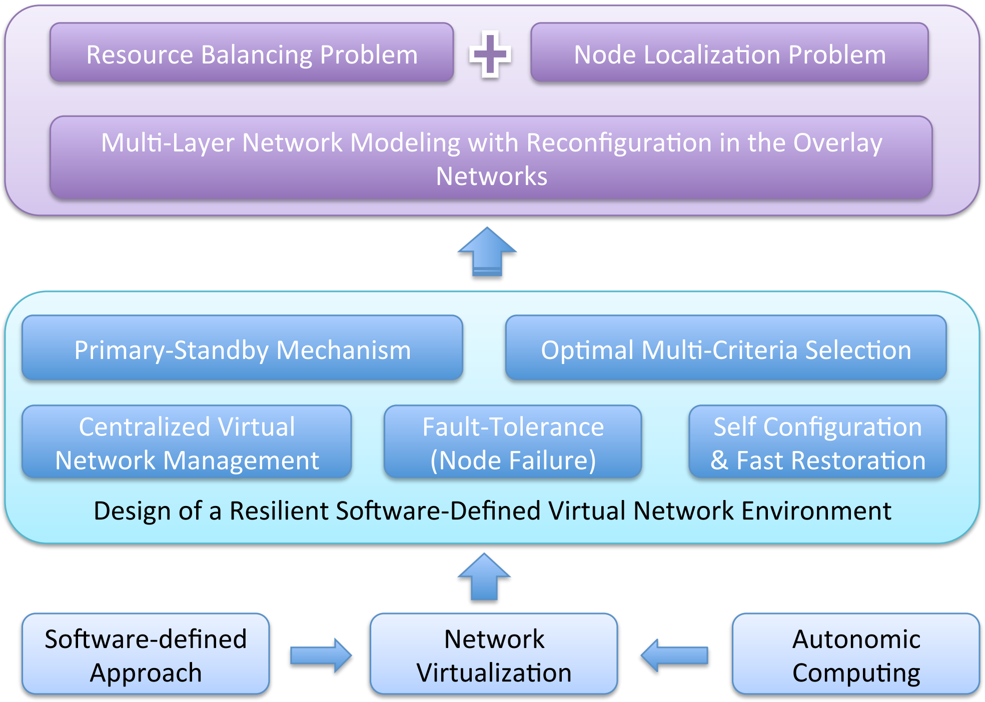

Resource Balancing Problem
Node Localization Problem
Multi-Layer Network Modeling with Reconfiguration in the Overlay Networks
Primary-Standby Mechanism
Optimal Multi-Criteria Selection
Centralized Virtual Network Management
Fault-Tolerance (Node Failure)
Self Configuration & Fast Restoration
Design of a Resilient Software-Defined Virtual Network Environment
Software-defined Approach
Network Virtualization
Autonomic Computing

Figure 7: A System View of the Problem Definition

in the existing VN. Our goal is to design an model to optimally pick an S-VR for the VN restoration under multiple selecting criteria, and this model can be applied into an overlay networking environment. More specifically, the proposed model addresses two major problems, which are the resource balance (i.e., load balance on the substrate network) and the node localization problem (i.e., which S-VR should be selected).

Therefore, in this dissertation, we address the problem of selecting optimal S-VRs when VRs in the core virtual networks are affected due to a failure. End hosts may be connected to multiple edge nodes, and the loss to end users is not the focus of this work. The scope is to recover from a failure that occurs at core VRs, *not* edge VRs of the VNs.

1.5 Contribution of the Dissertation

In this section, we briefly summarize the major contribution of this dissertation.

1.5.1 Dynamic Reconfiguration Scheme for Node Failures in a VNE

Primary-backup mechanism is commonly used for resilient network design. For example, for link failures, backup paths can be pre-computed to by detouring the traffic, and for node failures, backup routers might be available to be quickly activated. In our work, we consider to restore the VNs suffering a VR failure with a *standby virtual router* (S-VR).

Unlike the dedicated primary-backup approach, for a set of S-VRs provided for a VN, any of the S-VRs can be used to replace any failed VR in this VN. In other words, these S-VRs are identical, in terms of configuration. In order to efficiently restore the VNs with VR failures, a dynamic reconfiguration scheme is applied for this situation.

1.5.2 An Experimental Study on GpENI-VINI [67]

Before exploring the optimization model, we first designed an dynamic reconfiguration system for the virtual network manager (VNM) to achieve dynamic reconfiguration in a VN and deploy the DRS on the GpENI-VINI testbed. This is a joint work with Parikshit Juluri, who is also a PhD student in the Network Research Lab (NeTReL) at the University of Missouri - Kansas City.

The dynamic reconfiguration system has two primary tasks: (1) selecting a proper

S-VR based on the geographical distance between an S-VR and the failed VR (2) dynamically replacing the failed VR with the selected S-VR in the existing virtual topology, so that the affected VN can be restored without involving manual operations. Our goal spans to four aspects: (1) Conduct an experimental study on a wide-area virtualized networking testbed to evaluate the dynamic reconfiguration scheme; (2) Understand the DRS's impact to the network dynamics because the realistic packet loss or bandwidth availability are part of the environment. (3) Evaluate the impact on the routing convergence time under existing routing protocol (i.e., OSPF) during the reconfiguration; (4) Test autonomic management in a realistic VNE and learn the challenges of applying the dynamic reconfiguration system in such an environment.

The experimental study of deploying the dynamic reconfiguration scheme in a realistic virtualized networking testbed showed us a couple of lessons. First, dynamically restoring a VN from a VR failure is practical in a real VNE and the restoration process was fast, but the DRS cannot control the inherent properties of the network (e.g., routing table convergence time). In addition to the geographical distance, more criteria should be considered when making the selection. Second, multiple independent runs are important for implementing an experimental study on a realistic network testbed.

1.5.3 Design of a Multi-Criteria Standby Virtual Router Selection Model [68]

Based on the lessons and the experience from the experimental study, we designed an MILP model for a multi-criteria S-VR selection that can be applied into the dynamic reconfiguration scheme for any node failure in a VNE.

We first modeled the basic entities (e.g., nodes, links, topology, VNs, interfaces, etc.) within a VNE along with a number of constraints to the selection. We then defined cost functions associated with the selection process, where each cost function is represented as an aggregated weighted factors. Moreover, the final goal is to minimize both of the network cost and the maximum bandwidth utilization on the substrate nodes. According to the optimization model, we designed a heuristic algorithm as well. The proposed model is able to find optimal S-VRs to restore multiple VNs suffering from concurrent VR failures, where the VR failures could be caused by either a software failure at itself or a substrate node failure.

To evaluate the results, we first study the weight parameters defined in the cost functions and the objective function, and find out how to set the weights to each factors under a certain circumstance. Second, we designed experimental scenarios to study the impact of a variety of factors to the S-VR selection, these factors are dependent on the network dynamics and the resources provisioning. The results showed that the proposed heuristic's performance was close to the optimization model when there were sufficient standby virtual routers for each virtual network and the substrate nodes have the capability to support multiple standby virtual routers to be in service, concurrently.

1.5.4 Design of a Software-Defined Resilient VNE on GENI [66]

For this phase, by collaborating with Sarah Edwards and Niky Riga from GENI Project Office at Raytheon BBN Technologies, we designed a software-defined resilient

VNE on the GENI testbed. As we introduced in Section 1.2.3, GENI provides a large-scale distributed heterogenous VNE, and we are able to create a layer-3 virtual networks by creating tunnels and virtual routers.

Fig. 5 shows a logical view of the a VNE with a centralized VNM that implements the dynamic reconfiguration scheme. Based on this abstraction, we designed functional modules for the VNM, so that network initialization and reconfiguration can be automated on the GENI networks. In particular, we developed a tool to automate arbitrary VN initialization on the GENI testbed, and customized this tool to create VNs with a set of S-VRs. For the S-VR selection, we simplified the objective function from the optimization model, and only considered a weighted impact of the geographical distance and the VM load on the GENI nodes, and we present a prototype implementation of the VNE using GENI as the substrate network.

1.6 Additional Collaboration and Projects

I have been a collaborator on a number of projects during my PhD studies at University of Missouri - Kansas City, and they resulted in the following publications [15, 29, 42, 69, 70, 79, 94, 110, 111], and these contributions are described briefly in the following subsections.

1.6.1 GpENI Project

As one of the collaborators in the GpENI project, our (i.e., UMKC team) primary goal was to establish a layer-3 virtualization infrastructure testbed to support programmability at the network layer on the GpENI testbed. Consider the VINI project has built a programmable network-layer virtualization testbed, and there is a private VINI central management server called *MyVINI*) available for us to create an instance of VINI environment, we created a server running MyVINI at UMKC. Thus, the private VINI environment built on the GpENI network is named *GpENI-VINI*, and MyVINI server manages the GpENI-VINI nodes distributed at different GpENI sites in the US and Europe.

We have made several customizations from the ordinary MyVINI environment to support a more flexible resource provisioning and management as described in [29]. First, we successfully deployed `XORP v1.7` on GpENI-VINI *slivers*. Second, we extended the features of the original `IIAS` tool deployed for the public VINI testbed, so that the GpENI-VINI server allows users to create arbitrary virtual topologies with less restrictions on manipulating virtual links. Third, we developed a tool to automated the routing software installation process, and it allows users to efficiently install `XORP` or `Quagga` to all *slivers* in a *slice* simultaneously.

By successfully deploying these three features into the standard VINI software, we had insightful understanding about the underlying architecture of the Trellis software that runs as a core software for VINI nodes. The experience of implementing `XORP` within a *sliver* reveals that the mapping between virtual interfaces and physical interfaces shall be considered when creating the XORP configuration files of routing protocols, and this shows one of the differences between running a software router in a virtual node and a physical node. On the other hand, automating the routing software installation process for all *slivers* largely improves the efficiency of setting up an virtual IP network with active virtual routers.

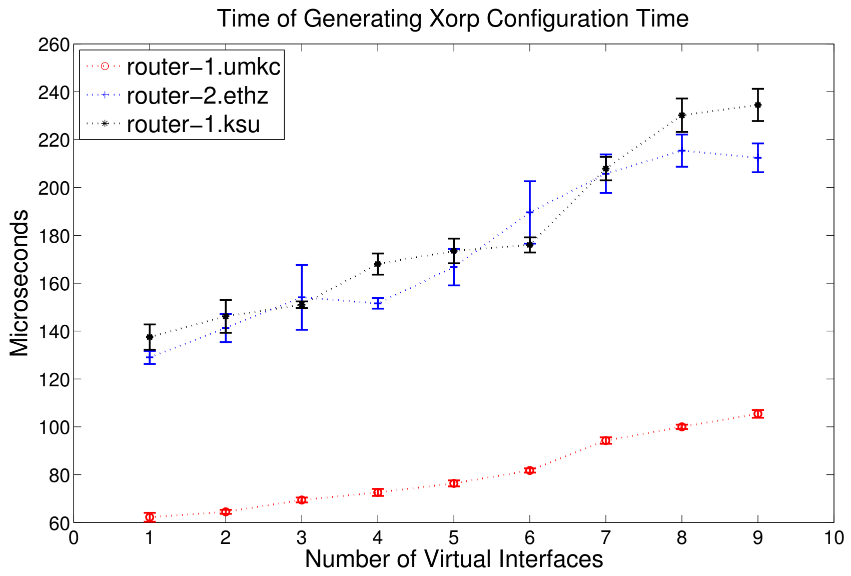


Figure 8: Time to Generate the XORP Configuration File within a Single Slice [79]

In this project, we also studied how the virtual router configuration would be affected by the substrate GpENI-VINI nodes. Since GpENI-VINI testbed uses Linux-vServer to create virtual nodes, the mechanisms of both CPU scheduling and I/O QoS use fair share and reservation [105], so users have no control on reserving resources by giving specific requests. Hence, the performance of the *sliver* depends on the hardware configuration of its substrate host. Fig. 8 presents the time to generate the default `XORP` configuration files at three different *slivers* within a single *slice*, respectively. We found the distinction between the time of generating the routing configuration files in *slivers* on different physical system. On the other hand, from a single host's perspective, hosting multiple containers does not have significant impact to the performance (see Fig. 9)

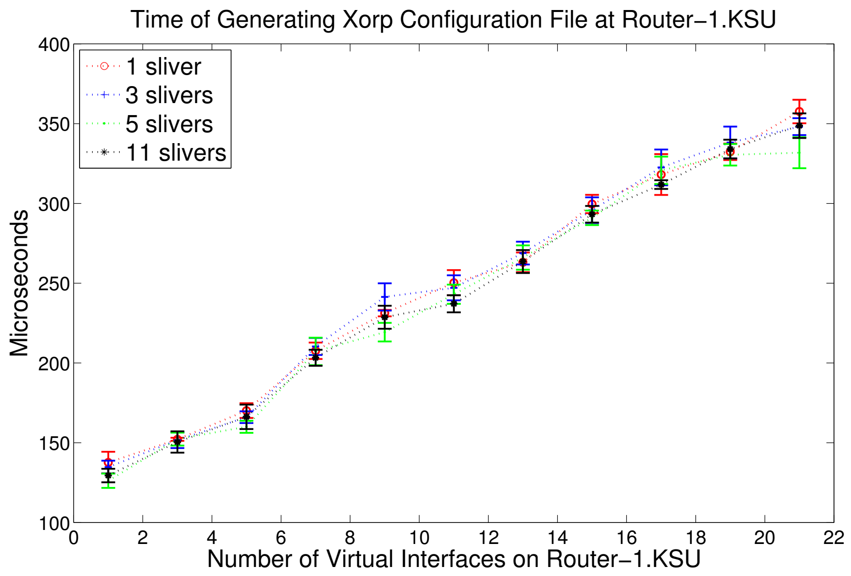


Figure 9: Time to Generate the XORP Configuration Files for Multiple Slices [79]

1.6.2 SeRViTR Project

The *Secure and Resilient Virtual Trust Routing* (SeRViTR) project has been resulted in a number of publications [15,70,110,111]. It was a joined collaboration between Osaka City University (OCU) in Japan, Arizona State University (ASU) and University of Missouri - Kansas City (UMKC) in the United States. The goal was to design a trustworthy and resilient network architecture supporting multiple virtual routing domains, and we built a testbed prototype implemented between the three universities.

Virtualized networking environment allows service providers to obtain a partitioned and QoS guaranteed virtual network. In this project, in addition to constructing virtual routing/switching domain, we further explored a fine-grained VNE design that enables user-centric virtual routing domain to be created for services with different trustworthiness. In [70], we presented a comprehensive SeRViTR framework design to realize a virtual trust routing and provisioning domain (*VTRouPD*) and a μ*VTRouPD*, where the latter one indicates a sub-domain comprising virtual routers/switches within a single VTRouPD or spanning multiple VTRouPDs, determined by a particular routing policy. For the SeRViTR framework, we designed the core components that are responsible for trust management, creating virtual routing domains and controlling flows, respectively. We validated our design using an OpenFlow-based implementation, and the virtual routing domain was distinguished by various *VLAN IDs*. In our validation testing, we considered two types of applications: SSH and HTTP, where the SSH flow was assigned a higher trust level, and the corresponding μVTRouPD for the SSH flow was created; whereas the HTTP flow was sent within a μVTRouPD with a lower trust level. Later on, we proposed a behavior-based policy management for SeVRiTR framework [110], where the trust level can be updated based on the judgement on the behavior of the flow registered at the relevant database. Furthermore, the impact to the trust level within a single VTRouPD may also affect the trust level negotiation between multiple VTRouPDs. Thus, this add-on feature to the SeRViTR framework potentially enables a trust level reconfiguration for virtual routing domains.

The second goal of this project was to build a geographical distributed SeRViTR testbed consisting of multiple VTRouPDs, where each VTRouPD belongs to an administrative domain. In [111], we presented a *Geo-Distributed Programmable Layer-2 Networking Environment (G-PLaNE)* as a substrate support for the SeRViTR testbed, which provides networking, computing and storage capabilities. For the computing resources, we use Xen to provision VMs for basic functional components (e.g., *Policy Manager*, *VTRouPD Manager*, etc) of the SeRViTR framework presented in [70]. For network resources, we used OpenFlow switches as *Flow Controllers* that redirect flows to different μVTRouPDs based on the trust level assigned by the *Policy Manager*. To interconnect the VTRouPDs constructed in the three universities (i.e., OCU, ASU and UMKC) from the US and Japan, we create a layer-2 GRE tunnel between any two universities using OpenFlow Switches. Thus, *VLAN* is supported over this layer-2 GRE tunnel.

1.6.3 Systematic Experimental Design on a Shared Testbed

By collaborating with the GENI Project Office and based on our experience on using GENI testbed, we contributed a case study in [42]. In the presented case study, we showed how to conduct experiments systematically on a publicly shared testbed. Table 1 shows that most testbeds today offer resources shared by all experimenters and they provide realistic networking environment for evaluating novel applications or protocols. To efficiently use the resources on the testbed and avoid unnecessary time-consuming debugging, we present two golden rules for systematic experimental design: (1) always starting with the smallest setup for an experiment (2) always making one change at a time when scaling up the experiment.

Another lesson we learned by running the experiments for the study case is that it is important to always save the configuration for our experiments (e.g., RSpec [21] for GENI testbed) before making any changes to the current experimental setup. This procedure also helps us to quickly recreate the identical experiments on the testbed due to various reasons. For example, if there is a planned long-term maintenance at a GENI site where we have reserved resources from, it will impact a few virtual links and nodes in our topology for a while. To bring up the same experimental environment, we can create another slice and load the same RSpec we have saved and replace the nodes under maintenance with the ones from other active sites.

1.6.4 Service-Centric Management in Information Centric Networking

The work presented in [94] was in collaboration with Ravindran et at. from *Futurewei R&D Center*. In this work, we designed a push-based multi-tier management framework for edge cloud service over the *information centric network (ICN)* [112]. ICN is another popular field in the Future Internet research, and it proposes to design a clean-slate networking architecture, where the identifier and locator of the information are decoupled and the information can be cached anywhere within the network. Relying the name-based routing proposed for the ICN architecture, we designed an overlay service access layer crossing from edge network to the core network, and simulated a conference framework as an ICN-based edge-cloud service.

1.6.5 Understand the benefit of Multipath Routing

Our recent work on [69] studied the benefit of multi-path routing by introducing a *multipath measure (MPM)*. We first presented a theoretical MPM for each of the three common traffic engineering objectives and then ran the linear programming to compute the optimal MPM for topologies in diverse types and sizes. Consider a multi-commodity scenario, for all node-pair demands under a certain traffic matrix, the numerical results showed that the exact MPM value obtained from the ILP was zero or close to zero at optimality. By comparing the objective value with the single-path routing, the multi-path routing showed little gain, especially for the topology comprising more than 25 nodes.

1.7 Outline of the Dissertation

In this dissertation, we address the problem of dynamically selecting optimal standby virtual routers for one or more virtual networks against node failures in a virtualized networking environment, with consideration of various constrains and costs in the virtual networks. In Chapter 2, we present a literature survey about the fault-tolerance scheme used in survivable network design and recent works in autonomic network management and centralized network management, respectively. We then illustrate the design of a dynamic reconfiguration scheme in a VNE in Chapter 3. In Chapter 4 we present the experimental study of deploying the dynamic S-VR selection in regard of a VR failure on the GpENI-VINI testbed. Chapter 5 presents the optimal S-VR selection model and relavent numerical results. Chapter 6 presents the prototype design and implementation on GENI testbed. Finally, we summarize our work and briefly discuss the future research goals in Chapter 7.

CHAPTER 2

LITERATURE SURVEY

A recent survey [44] classified the virtual network embedding approaches into three categories: concise/redundant, static/dynamic, centralize/distributed, where each category has two options. Our work considers one of the eight combinations of these six features, that is, $\{$*redundant, dynamic and centralized*$\}$. A virtual network embedding with redundant resource provisioning is also referred as survivable virtual network embedding [56]. Moreover, the centralized and dynamic approaches refer the management mechanisms that can be used in either the embedding or the reconfiguration procedure.

As discussed in Chapter 1, a VNE is presented as an overlay architecture, and it is conceptually similar to the traditional IP-over-optical network architecture, where the underlying network is the optical ring/mesh topology and the IP-layer links shares the fiber links. Thus, the traditional survivable design for the optical-layer could be still extendable for designing a survivable VNE. For example, the survivable virtual network embedding aims to provide spare resources to protect link/node failures in a VNE, and this approach inherits the protection scheme used in the survivable optical network design.

In this chapter, we first present a literature survey on survivable design for optical networks and virtual network embedding, and then we briefly discuss about dynamic virtual network reconfiguration and centralized management that are also popular research areas for virtual network management.

2.1 Survivable Optical Networks

In the optical networks, link failures are more common than the node failures. Sivakumar et al. [104] presents a survey on the survivability optical networks in two common topologies: the SONET/SDH ring topology and the WDM mesh topology.

For the SONET/SDH ring topology, there are three standard protection schemes [104]: (1) autonomic protection switching (APS) (2) dual homing (3) self-healing rings. The APS is used to provide link protection in either dedicated or shared backup manner. Dual-homing mainly handles the single-node failure in the ring topology, where a backup node is provided for a critical node (e.g., a major hub in the network). The self-healing ring is used to protect both link and node failures.

In a WDM mesh networks, connection between two nodes are called *lightpath*, and a lightpath usually spans a series of fiber links, or a fiber link may be shared by multiple lightpaths. Thus, a failure at a fiber link may affect multiple lightpaths that share the failed link in the optical network. Most of the works [40,73,93,118] addressed the link failures in the WDM mesh network by considering the protection or restoration scheme, and the specific survivability designs for link failures are at the path-level or the link-level. For the protection scheme, backup resources are preplanned before the failure occurs. In the dedicated-backup paradigm, the common ways are *1+1* and *1:1* [104, 118]. In the shared-backup paradigm, the backup wavelength could be shared by multiple paths that do not fail simultaneously, and the paths can be either the primary paths or the backup paths. For the restoration scheme, the backup paths to restore a link or an end-to-end path are computed dynamically after the failures. Although the restoration scheme may not guarantee that backup resources can always be determined after the failure, it efficiently saves the spare resources. Doshi et al. [40] proposed a *1+1 protection* against a single-failure scenario, where the backup path between a source and destination is preplanned with links and nodes that are disjoined with the primary path. Chan et al. [26] proposed a self-protected architecture using 1:1 protection in the WDM networks. Mohan et al. [81] proposed a hybrid method that combines the shared-backup protection and a restoration, where a primary path shares a channel with multiple backup paths.

Proactive or reactive mechanisms for node failures in a WDM network are more complex and challenge. Usually a node failure may cause multiple failures in the links that are attached to this node, and to recover these links requires multiple node disjoint backup paths to reroute the traffic traversing these links [104]. Liu et al. [71] proposed a spare capacity allocation model with successive survivable routing for node failures in a mesh topology, where each traffic flow was preplanned a node-level disjoint path so any single node failure along the path can be protected.

2.2 Survivable Virtual Network Embedding

Virtual network embedding problems have been wildly studied in the past decades, and most of them were published in the past five years. The primary goal of virtual network embedding is to accommodate each virtual network request to the substrate network, where a virtual node usually is mapped to a substrate node and a virtual link is mapped to a substrate paths (i.e., a set of substrate links). Fischer et al. [44] presents a detailed taxonomy of the virtual network embedding approaches, where about 50% of the works used the redundant approach. Thus, survivability has become an significant element for creating a resilient virtualized networking environment.

As listed in the survey [44], out of the 40 papers that considered redundancy for the virtual network embedding, 34 (about 85%) used a centralized and static approach. A centralized approach is still the most common for virtual network embedding, and the main advantage is that it has the global view of the whole VNE. With static allocation, the resources are preplanned for a virtual network request and cannot be reallocated for another virtual network request that comes later, and it lacks flexibility to adapt the changes in the VNE, such as the resource distribution on the substrate network. According to the survey, there have been only five papers combines both dynamic and redundancy in the embedding paradigms.

Survivable virtual network embedding against link failures was addressed in many works [31, 33, 51, 76, 91, 92, 101, 102, 115–117]. In particular, Shamsi et al. [101, 102] present QoS-aware resource allocation from the application's perspective. The overlay links requiring high quality were mapped to the direct paths on the substrate networks, and alternative paths were provisioned by indirect paths going through intermediate nodes on the substrate network. Zhang et al. [116, 117] also considered a QoS-based allocation, where disjoint backup paths on the substrate network were provisioned against link failures and the latency for the requested virtual network was minimized. While allocating the backup paths, the proposed model aimed to minimize the number of the substrate nodes that were selected to support backup paths. Allocating virtual links that share multi-path splited on the substrate network is another way to improve the survivability against link failures [31,33,115]. Guo et al. [51] considered link restoration by providing a number of precomputed bypass substrate paths, and the path restoration was achieved by providing backup end-to-end paths. In their work, bandwidth are considered as shared resources for the proposed on-demand or pre-allocation schemes. Rahman et al. [91, 92] presented a policy based link embedding to provide fast link restoration and path restoration.

As presented in [75], node failures took 16.5% of the total amount of failures based on the study on the Sprint network. In a VNE, virtual node failures will cause multiple virtual link failures, and substrate node failures may affect multiple virtual node failures. Thus, survivable virtual network embedding against node failures in a VNE is desirable and emerging. As presented in a recent survey about the survivable virtual network embedding studies [56], most of the works [50,60,113,114] focused on single-node failure on the substrate network. The authors in [50] discussed a two-step survivable virtual network embedding procedure to provide protection against single substrate node failure. In particular, they first enhanced the virtual network request by proactively adding a backup node with and associated links, and then applied the embedding procedure. The paper [119] proposed a two-stage virtual node mapping scheme that is able to map a virtual node to multiple nodes on the substrate network, and this idea is similar to the aggregation approach we have discussed for the optical layer virtualization. Yeow et al. [113] presents a shared-backup scheme, called *Opportunistic Redundancy Pooling mechanism* in the paper, to protect a single-node failure on the substrate network. In this work, the redundant nodes can be shared by multiple virtual networks. Yu et al. [114] enhanced the virtual topology by adding the redundant nodes with associated links, and then mapped the new virtual topology with the backup nodes and links onto the substrate network. The works we have discussed so far are all proactive approaches (i.e., before failures). Houidi et al. [59] presents a dynamic reactive restoration scheme for either virtual node failures or substrate node failures conducted by distributed agents on the SN. For virtual node failures, the agent who supports the failed virtual node will initialize an alternative virtual node to replace the failed virtual node from the same host or a different host on the SN, and the agent will also reallocate resources for virtual links associated with the new virtual node. For a substrate node failure, an agent will select a different host and migrate the affected virtual nodes and links to the new selected host.

2.3 Autonomic Network Management

Autonomic computing [65] has been a practical solution to manage complex distributed computer system, and it enables efficient self-management for computer networks, especially for a resilient network design [27]. To provide an ideal fault-tolerant network environment, besides applying a good protection or restoration scheme, fast recovery is also important to avoid the the customers' awareness of failures. In a virtualized networking environment, there are two-tier provider-customer relationship. At the first tier, regular end system users are customers of a service (e.g., video streaming, P2P, etc) provided by an SP, and at the second tier, the SP now is a tenant who rents network resources from an SNP. Thus, service customers are expecting services with good a quality of experience (QoE), and the service providers expect a good quality of service (QoS) with the VNs that the substrate network provider leases to them. Hence, an efficient survivable virtual network management framework is desired from the substrate network provider's perspective.

There are several works [23,28,99,115] using dynamic or autonomic management to create a survivable VNE with efficient resource utilization. Butt et al. [23] proposed a reactive approach to dynamically detect substrate nodes and links with bottleneck residual resources and then reallocate pre-mapped VNs onto the SN, so that a previously rejected VN request can be mapped to the SN, which in turn increases the acceptance ratio of virtual network embedding and achieve better load balancing on the SN. Yu et al. [115] proposed a flexible path-splitting mechanism for link mapping problem, and use *spliting ratio* to denote the division of the traffic on the SN. This work presents a link mapping algorithm with dynamically remapping a virtual node from the original substrate node with bottleneck link to another one that has maximum residual resource. Moreover, if there is a substrate link failure, the *splitting ratio* can be adjusted so that the affected traffic can be switched to another path. Chen et al. [28] presents a dynamic virtual node reconfiguration to balance the resource utilization on the substrate nodes.

Dynamic reconfiguration can also be applied to overcome router failures and link failures, as self-healing defined in [65]. Recent works mainly concentrated on optimizing the routing algorithms to provide better support for dynamic reconfiguration. These algorithms were studied by either using mathematical models, simulations [24,46,48], or by testing them on networks using local lab machines [100]. 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 evaluated on a wide-area network testbed.

2.4 Centralized Networking Management

Centralized control is not new from the network management point of view. For the optical networks, centralized approaches have been widely used for the restoration [40, 104], because a central controller has a global view of the whole optical network and can quickly calculate the restoration paths for the primary path, and then conduct the restoration procedure.

Software-defined networking (SDN) has been a hot research topic in recent years. In SDN, network management (control plane) and packet forwarding (data plane) are decoupled, where the control functions are deployed and implemented at a centralized server or controller to support flexible flow or routing management. For example, for the OpenFlow network [78], a controller is responsible to compute and distribute flow tables to the OpenFlow switches, and the switches only need to forward the traffic based on the flow table received from the controller. In this work, we discuss a software-defined resilient virtualized networking environment, where a centralized system provides the software-based virtual network management, including VN reconfiguration to restore connections in the VNE.

One of the disadvantage with the centralized management is referred as single-point failure. For a resilient SDN design, the primary-backup idea has been applied to the controller placement problem in recent literatures. In [45], the core component CPRecovery is designed to support communication between the primary and backup controllers. Bari et al. [18] proposed a dynamic controller provisioning mechanism. Hock et al. [58] discussed a resilient controller placement in the core SDN network, and the proposed optimal controller placement considered various failure events such as node failures including the controller failure.

CHAPTER 3

DYNAMIC RECONFIGURATION SCHEME IN A VNE

In Chapter 2, we have discussed two general approaches to achieve survivability in a VNE. The first is a proactive scheme called protection, which pre-plans spare resources for potential failures in the networks (i.e., mostly for a substrate network). The other approach called restoration is a reactive scheme, which dynamically reallocate resources (e.g., alternative paths or nodes) to restore the affected networks. For a VNE, failures are expected to be transparent to both service providers and service customers. Hence, a substrate network provider must provide efficient protection or restoration mechanisms so that any failures can be handled quickly in order to minimize the impact.

For a VNE, protection against link failures have been explored extensively, but node failures haven't been addressed much, especially with a restoration scheme. In this dissertation, we design a dynamic reconfiguration scheme for virtual network restoration against either virtual router failures or a single substrate node failure.

Unlike a regular restoration scheme where the alternative resources are discovered and allocated dynamically after the failure, we considered a restoration on basis of a protection manner. In the traditional protection scheme, alternative resources are reserved for particular primary entities in the network. For example, for critical links or paths in a network, backup paths have been precomputed in case the primary links or paths fail. In our design, we assume that the substrate network provider (SNP) is able to allocate spare

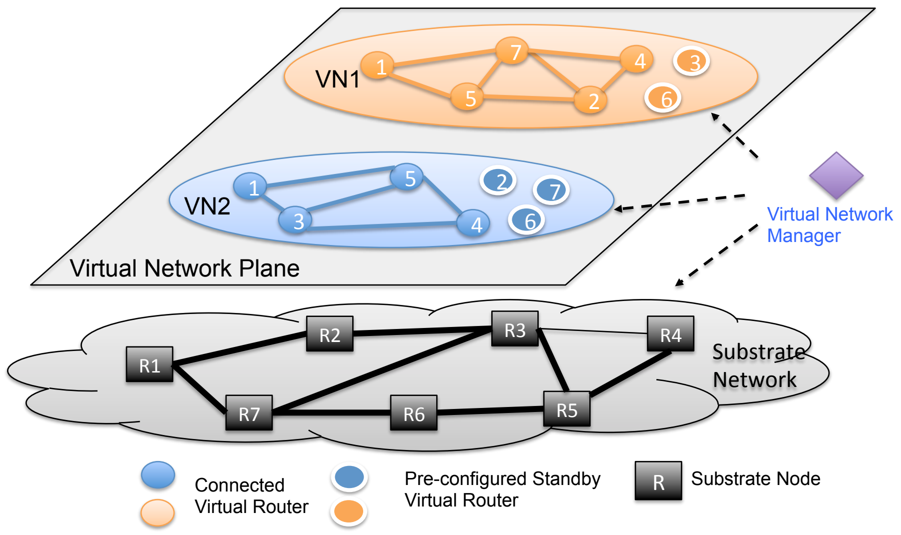


Figure 10: Dynamic Reconfiguration Scheme for Virtual Networks

virtual routers (VRs) as standby virtual routers (S-VRs) for a VN, and these S-VRs are not reserved as dedicated backups for a particular set of VRs, but for any VR failed in this VN. In other words, any of the reserved S-VR can be selected to replace any failed VR in a VN, and this shared-S-VR provisioning helps to reduce resilient cost in a dedicated S-VR provisioning. Note that the S-VR provisioning is out of the scope of this dissertation.

Our objective is to design a *dynamic reconfiguration scheme (DRS)* by optimally selecting a S-VR for each affected VN. The reconfiguration process will automatically replace the failed VR with a properly selected S-VR and the original virtual topology is sustained after the replacement. Fig. 10 presents the proposed DRS applied for the VNE shown in Fig. 5 that we illustrated earlier in Chapter 1.2

As shown in Fig. 10, the SNP employes a logical centralized *virtual network manager (VNM)* that is responsible for VN initialization, configuration, monitoring and reconfiguration in a dynamic manner. There are two VNs actively in operation on the virtual network plane, and both VNs have been provisioned a set of S-VRs, respectively. Since these S-VRs are shared-backups for any VR in the existing VN, they are functionally identical to other active VRs but in the sleep mode. When the VNM detects a VR failure for a VN, it starts the dynamic reconfiguration process, which automatically selects one of the provisioned S-VRs to replace the failed VR. The replacement procedure requires the selected S-VR to be connected to every original neighbor of the failed VR, so that the original topology could be restored. Since all the reserved S-VRs for a VN are in the sleep mode, any S-VR can be quickly activated in case of any VR failure, and the affected VN can be restored back to its original topology immediately. In this case, the restored VN does not need to make many changes to the routing tables and can minimize transient issues. For example, consider a VN that has deployed the OSPF (open shortest path first) routing protocol. OSPF can recognize a single link failure well; however, OSPF does not support a node failure function well as it has to learn from correlating the non-functioning of links associated with the failed node to eventually recognize the node failure—this may result in an extended transient window when traffic flow may suffer for the SPs. Thus, having S-VRs preconfigured can allow quick restoration from a VR failure for each VN. On the other hand, because both the traffic load and the hardware capacity are changing at the substrate nodes where these S-VRs reside, selecting an S-VR that has minimum impact to both the existing VNs and the SN and minimizing operations cost are important during the dynamic reconfiguration process, and we will discuss the optimal S-VR selection by considering different factors in Chapter 5.

We consider two conservation requirements while dynamically conducting the replacement. First, each VN should be restored to its original topology as we just discussed above. Secondly, if an S-VR is selected, the bandwidth between this S-VR and each neighbor of the failed VR should also be restored to what it was before the failure, so that the QoS of the VN will not be degraded.

In a VNE, a VR may fail because of two reasons, and we classify the VR failures into two categories based on the characteristics of the failure. The first type is referred as an *independent VR failure*, where the failure at a VR is caused by some software issues or overload within the VR and it does not affect other VRs or other VNs. The second type is referred as a *dependent VR failure*, where a VR failure is mainly caused by a failure or maintenance at the corresponding substrate node and this substrate node failure might also affect other VNs if these VNs contain VRs hosted by this failed substrate node.

The proposed DRS aims to restore VNs, in case of either independent or dependent failures. For example, if a power outage occurs at the substrate node `R5` in Fig. 10, it will immediately affect `VN1` and `VN2`, where both contain a VR hosted on `R5`, respectively. Once the centralized management system VNM identifies the failures, it will trigger the DRS and select one S-VR for`VN1` and `VN2` from each one's shared-S-VR pool, respectively. Suppose for `VN1`, the S-VR hosted on `R6` is selected, it will be activated to connect VR-1,7,2, which originally were the neighbors to the failed VR-5, and the original topology for `VN1` could be restored. Meanwhile, the VNM also select VR-2 from the shared-S-VR pool and reconfigure the topology for `VN2` so that VR-2 connects to VR-1,3,4.

For the illustration in the rest of chapters, we make following assumptions:

- Within each VN, any two VRs, including the S-VRs, are not hosted on the same substrate node.
- For each VN, at most one independent VR failure at a time.
- One substrate node fails at a time.

The first assumption reflects that it is not common to create two VRs within a VN from the same substrate node. Intuitively, for a VN, reserving two VRs from the same substrate node is for redundancy purpose, where one is activated while the other is reserved as an S-VR. Hence, in the case of a failure of the working VR, it can be automatically switched to the S-VR by the hosting substrate node, while preserving the VN topology or bandwidth. However, this kind of dedicated-backup mechanism will not work for the *dependent VR failure* scenario. In our approach, we focus on the non-trivial problem with a shared-backup mechanism, where the S-VR resides in a different substrate node from the failed VR, and it is not reserved for replacing a dedicated VR with a failure. Secondly, we assume there is at most one VR failure within a single VN at a time, so it ensure sufficient S-VR candidate for each failed VR. On the other hand, based on the first assumption, for the same VN, the chance of concurrent VR failures is reduced. Even with a substrate node failure, at most one VR within a VN could fail. In regard to the third assumption, it does not rule out the possibility that a substrate node failure may simultaneously affect multiple virtual networks.

CHAPTER 4

AN EXPERIMENTAL STUDY ON GPENI-VNI

Virtualization at the network layer is achieved by creating partitioned network namespaces at virtual nodes, where manipulating both routing tables and virtual interfaces is allowed. Critical aspects of network virtualization are resource allocation and re-allocation, especially in the case of router failures or re-allocation of routers due to maintenance. In this chapter, we present an experimental study on the DRS used in a realistic virtualized network environment — GpENI-VINI testbed [29] in which the VRs are geographically distributed across the United States and Europe. Our study considered 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.

4.1 A Detailed View of the DRS

We have introduced the generic dynamic reconfiguration scheme (DRS) in Chapter 3, and the basic idea is to automatically replace a failed VR with the most suitable S-VR. In this section, we present a detailed view of the DRS from the operation perspective.

The sequence diagram Fig. 11 presents the break-down processes of dynamic reconfiguration that overcomes a VR failure in a VN. Note that our goal is to design a DRS

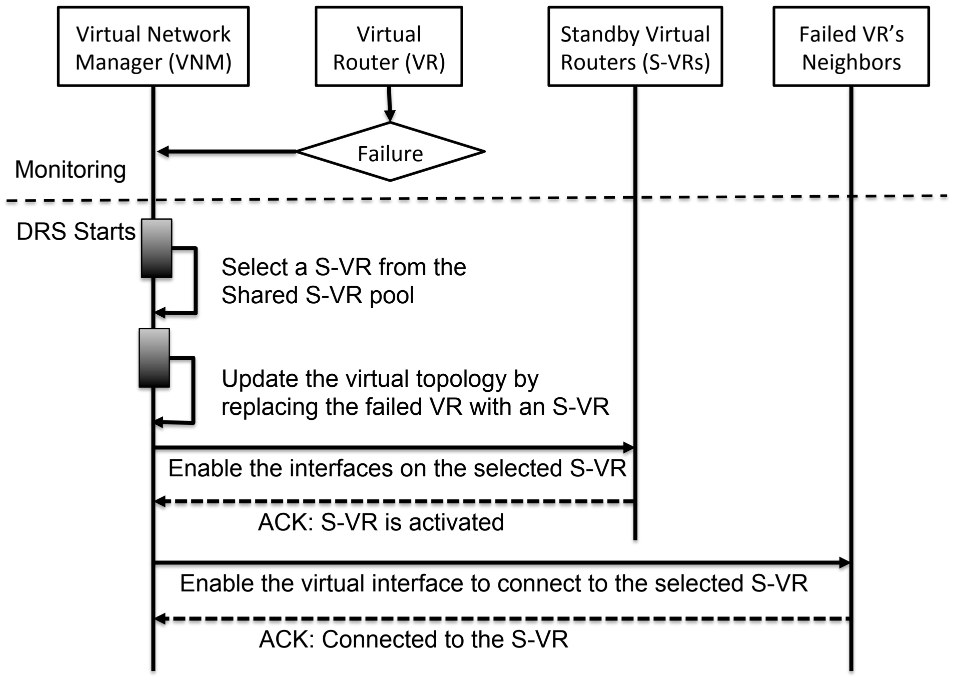

Virtual Network Manager (VNM)
Virtual Router (VR)
Standby Virtual Routers (S-VRs)
Failed VR's Neighbors
Failure
Monitoring
DRS Starts
Select a S-VR from the Shared S-VR pool
Update the virtual topology by replacing the failed VR with an S-VR
Enable the interfaces on the selected S-VR
ACK: S-VR is activated
Enable the virtual interface to connect to the selected S-VR
ACK: Connected to the S-VR

Figure 11: Sequence Diagram for the DRS

that minimizes the effect on the network performance and does not consider the detection of failures. 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, with existing dynamic routing protocol (e.g., OSPF), if there is an alternate path for the data, we can expect that the communication would continue over that path until recovery is complete.

The DRS plays an important role when the network needs to be updated, or recovered seamlessly, and the whole process is transparent to the customers and service providers as well. Hence, an optimal selection strategy will largely reduce the impact on the end-to-end network performance due to the network reconfiguration. In general, to select a proper S-VR candidate to replace failed VR, we may consider a few options such as access latency, a VR's capabilities, a VR's geographical location (i.e., latitude & longitude), etc. In our study, we consider the third strategy based on the following:

- S-VRs are not connected to any other active routers; so the latency between an S-VR and other active VRs cannot be used as a priori information for selection. Even though S-VR can be accessed from the VNM, we can only measure the instantaneous access delay from the VNM, and we cannot presume that the access performance is as good as that from other VRs.

- Our experiment testbed is a VNE with layer-3 virtualization, and our measurement shows that the latency of a virtual link between two VRs was not the same as the latency between the corresponding substrate nodes (i.e., the latency of the substrate path between these two nodes). For instance, for two VRs are directly connected through an EGRE tunnel; the ping test between the virtual interfaces showed the round trip delay to be around 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 the VRs are software routers instead of commercial routers, we do not have enough information on the router's capability as a reference to make a selection.

Therefore, we consider two strategies. The first is a geographical distance based selection, where the DRS selects an S-VR candidate geographically closest to the failed VR. This is reasonable because the closest S-VR usually has a similar physical connection

Algorithm 1: Geographical Distance based S-VR Selection for a VN

```
    Input:  $f$: the failed VR in a particular VN
            $\mathbb{S}$: set of S-VRs reserved for the VN
            $\mathbb{R}$: set of substrate nodes
    Output: $s$: the ID of the selected S-VR candidate for the VN, $s \in \mathbb{S}^j$
    // Search the geographical location of the failed VR $f$
1   ( $f_{lat}$, $f_{lon}$) ← GetLocation ($\mathbb{R}$, $f$)
    //Initialize the minimal distance as infinity
2   $d_{min} \leftarrow \infty$
    //Initialize the selected standby router's ID
3   $s \leftarrow$ NULL
4   for $i$ in $\mathbb{S}$ do
        // Search lat-long information of standby routers
5       ( $i_{lat}$, $i_{lon}$) ← GetLocation ($\mathbb{R}$, $i$)
        // Get the distance between the $f$ and the S-VR $i \in \mathbb{S}$
6       $d$ ← GetDistance( $i_{lat}$, $i_{lon}$, $f_{lat}$, $f_{lon}$)
7       if $d < d_{min}$ then
8           $s \leftarrow i$
9   return $s$
```

on the substrate, so it is least likely to affect the original network performance. For a VN that has a set of S-VRs (denoted as $\mathbb{S}$) reserved, if an active VR f fails, the VNM runs Algorithm 1 to selects a S-VR with minimum distance to f. The second way is to randomly select an S-VR candidate from the shared pool. Once the VNM has determined the S-VR for the affected VN, it runs Algorithm 2 to replace the failed VR f with the selected S-VR s to restore the VN and update the virtual topology information.

4.2 Experimental Platform: GpENI-VINI Testbed

We have deployed our experiment on GpENI-VINI, a layer-3 programmable virtual routing testbed that is geographically distributed on the GpENI testbed. We have

Algorithm 2: Virtual Topology Update Algorithm

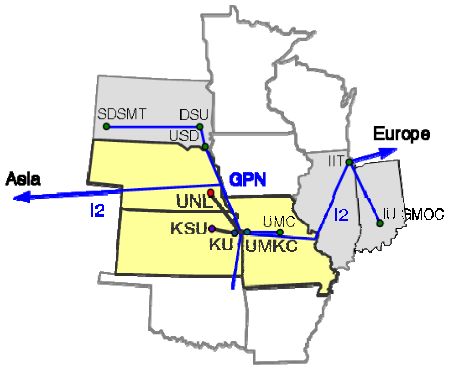
```
Input: $G = (\mathbb{V}, \mathbb{E})$: A VN ($G$) consists of the VRs ($\mathbb{V}$) and the virtual links ($\mathbb{E}$)
       $f$: the failed VR, $f \in \mathbb{V}$
       $s$: the selected S-VR candidate for the affected VN
Output: An updated topology $G'$ with new node set $V'$ and new link set $E'$
1 for $u \in \mathbb{V}$ do
2     if $u == f$ then
3         for each $v \in$ adj $(u)$ do
4             LinkDown $(u, v)$
5             LinkUp $(s, v)$
6             $\mathbb{E}' \leftarrow (\mathbb{E} - \{(u, v)\}) \bigcup \{s, v)\}$
7             $\mathbb{V}' \leftarrow (\mathbb{V} - \{u\}) \bigcup \{s\}$
8 $G' \leftarrow (\mathbb{V}', \mathbb{E}')$
9 return $G'$
```

briefly introduced GpENI testbed and GpENI-VINI in Section 1.2.3, now we present more technique details on virtualization and the virtual routing capability supported by the GpENI-VINI testbed.

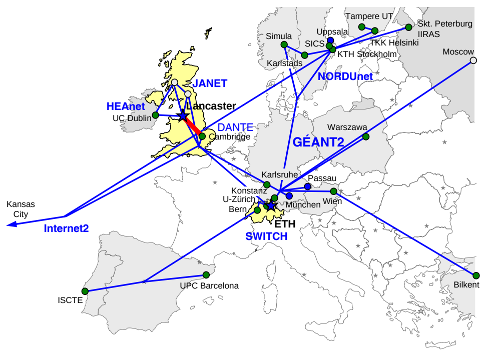
4.2.1 Overview

GpENI (Great Plains Environment for Network Innovation) [79] is an international experimental testbed for future Internet research. It was originally built in collaboration between four universities in the Midwest region of the United States (i.e., the University of Kansas, the University of Missouri–Kansas City, the Kansas State University and the University of Nebraska-Lincoln). Currently the GpENI testbed includes sites in both the United States and Europe and it's expanding to Asia. Fig. 12 displays the topology and infrastructure of GpENI testbed in these three continents.

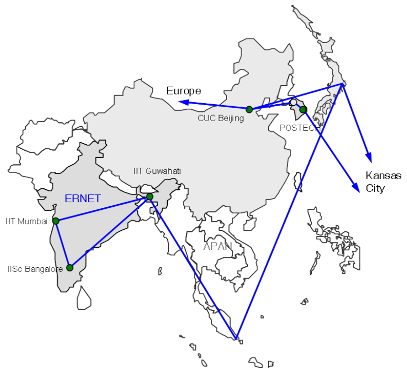
The GpENI testbed provides programmable capabilities on multiple layers of the



(a) Midwest Topology



(b) European Topology



(c) Asian Topology

Figure 12: GpENI Topology and Infrastructure in US, European and Asian Countries [6, 79]

network. In particular, we described layer-3 programmability and network resource provisioning in [29]. With GpENI-VINI testbed, users can request a number of virtual routers and create a customized virtual topology using those VRs. At the time of evaluation, there were 18 active GpENI-VINI nodes that are distributed across the United States and Europe. This gave us an opportunity to evaluate the DRS in a wide-area VNE.

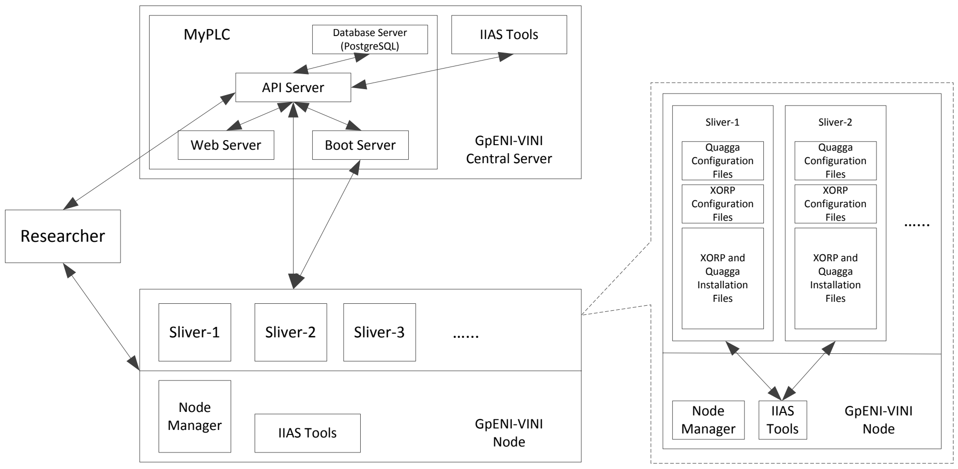


Figure 13: GpENI-VINI Architecture Overview [29, 79]

4.2.2 GpENI-VINI Architecture

GpENI-VINI is a customized instance of the public VINI testbed deployed by Princeton University by adding additional features. Fig. 13 shows the architecture of GpENI-VINI architecture. In particular, the *Internet in a Slice (IIAS)* is a client-server tool allowing users to create layer-3 virtual topology in their *slices*. We have extended the features of the `IIAS` tool on both the server side and the client side to enable flexible creation of virtual networks. GpENI-VINI nodes are distributed across the sites in US and Europe, and they are physical systems provisioning *slivers* running as software virtual routers. On the other hand, to support both host and network virtualization (i.e., Linux vServer and NetNS), each GpENI-VINI node runs an instance of ordinary `Trellis` software that also runs on the public VINI nodes. Moreover, we developed an tool to automate the routing software installation process.

The benefits of GpENI-VINI testbed is summarized as follows:

- First, GpENI-VINI allows users to create an arbitrary topology in a *slice* by just entering the virtual topology information through a web-based GUI.

- Second, virtual interfaces are pre-created on each VR. Once the user entered the virtual topology information that he wants to create, the `IIAS` [20] running at MyPLC (i.e., GpENI-VINI central server) will automatically configure private IP addresses for each *sliver* to be created from corresponding substrate. By communicating with the MyPLC server, the `Trellis` [22] software running at each node create and configure virtual interfaces within the requested *slivers* dynamically. Thus, we do not have to configure interfaces for every router in our experiment, which improves the efficiency of building the experimental environment.

- Third, once the *sliver* is up, a default configuration file for specific routing protocol has been created and a routing software (i.e., Quagga or XORP) has been installed at the *sliver*. In other words, each *sliver* automatically runs as a pre-configured software router based on the user's preference. 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 *slivers* at one time. Thus, it enables high efficiency for a large-scale experiment pre-setup.

4.3 Experimental Design

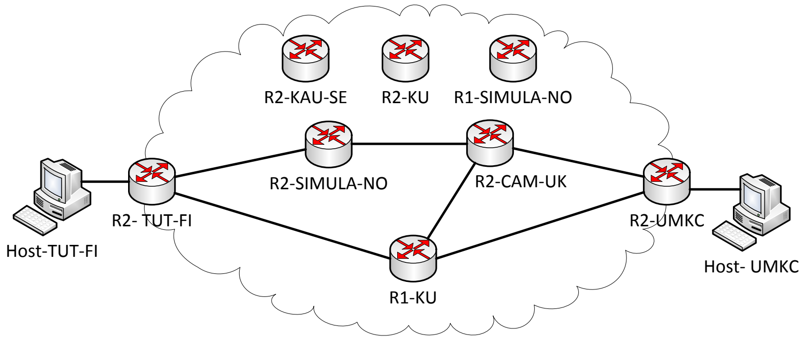
Considering the flexibility and efficiency of running a layer-3 virtual network on the GpENI-VINI testbed, we can enforce the DRS to a virtual network in a GpENI-VINI *slice*.

From the SNP perspective, we created the initial VN including both the requested virtual topology and the S-VRs in the sleep mode. In order to quickly bring up the S-VR and add it to the existing VN, we also created the virtual links between each S-VRs to the VRs in the requested VN and disable these virtual links when the corresponding S-VR was deactivated. Moreover, both regular VRs and S-VRs are all configured with the OSPF protocol.

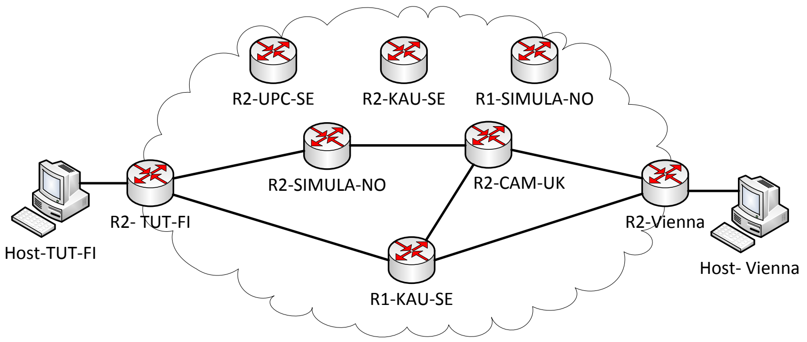
Once the initial virtual topology has been created for the slice, the VNM will disable all virtual links between the S-VRs and other VRs by turning down the their corresponding virtual interfaces, respectively. When a S-VR is selected, the VNM then turns on its virtual interfaces that are associated with the failed VR's neighbors to activate the virtual links, and wait for the OSPF routing table to get convergence on every other VR in the virtual topology. In the following subsections, we show our study on OSPF convergence time and network performance during the reconfiguration.

4.3.1 Experimental Scenarios

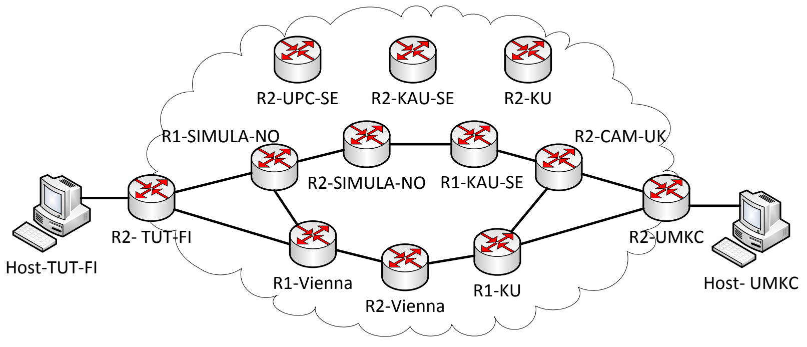
We considered three wide-area virtualized topologies of two different sizes (Fig. 14) on the GpENI-VINI testbed for our experiment, where the node information is provided in Table 2.



(a) Topology-I(a): Five VRs across US-Europe



(b) Topology-I(b): Five VRs in Europe Only



(c) Topology-II: Nine VRs across US - Europe

Figure 14: Five-node and Nine-node Wide Area Virtual Topologies

Table 2: GpENI-VINI Node Information

Node Names	Site Name	City	Country
Host-UMKC	University of Missouri - Kansas City	Kansas City, MO	US
R2-UMKC	University of Missouri - Kansas City	Kansas City, MO	US
R1-KU	The University of Kansas	Lawrence, KS	US
R2-KU	The University of Kansas	Lawrence, KS	US
R2-CAM-UK	University of Cambridge Computer Laboratory	Cambridge	UK
R1-SIMULA-NO	Simula Research Laboratory	Lysaker	Norway
R2-SIMULA-NO	Simula Research Laboratory	Lysaker	Norway
R1-KAU-SE	Karlstad University	Karlstad	Sweden
R2-KAU-SE	Karlstad University	Karlstad	Sweden
Host-TUT-FI	Tampere University of Technology	Tampere	Finland
R2-TUT-FI	Tampere University of Technology	Tampere	Finland
R2-UPC-SE	UPC CCABA Barcelona	Barcelona	Spain
Host(R1)-Vienna	University of Vienna	Vienna	Austria
R2-Vienna	University of Vienna	Vienna	Austria

Topology-I has two variations as shown in Fig. 14(a) and Fig. 14(b), respectively, depending on the geographical locations of the selected substrate nodes that provide the VRs. Topology-I(a) consists of five VRs spreading across both Europe and the United States. Fig. 14(b) shows the same topology as Topology-I(a), but it is composed of VRs speaded in the Europe countries only, so it is denoted as Topology-I(b). Because of the underlying star-like topology of the GpENI network, the virtual link speeds available to each VN and the end-to-end delay on the GpENI substrate network are different between these two topologies. Hence, they represent two different sub-scenarios of Topology-I.

Topology-II, as shown in Fig. 14(c), is a nine-node virtual topology spanning both Europe and the United States. On the US side, we used nodes located at the University of Missouri–Kansas City and the University of Kansas, and on the European side, we selected nodes from six institutions from different countries (Norway, Sweden, Austria, Finland, UK, and Spain).

For all three VNs, two *slivers* from the testbed were assigned as end hosts to generate traffic flows. We also assigned one node from the GpENI-VINI testbed running as VNM. In Table 3, we list the round-trip time measured between the two end hosts 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. An interesting observation was found for two VNs created for Topology-I. For Topology-I (a), although the geographical distance between the two end hosts (i.e., transatlantic distance) is greater than it is for Topology-I (b) (i.e., within European continent), the latency measured between the two end hosts for Topology-I(a) was less. This contradiction is because of the start-topology on the GpENI backbone network [6], where the center of the star is located at the University of Kansas (KU) in the United States. Thus, the traffic sent between two VRs in Europe was actually sent along the paths Europe-US-Europe, which increased the presented latency in the virtual links. This indicates a fact of substrate network topology's impact to the overlay VNs.

We are interested in three possible factors to the network performance during the reconfiguration: (1) the number of VRs in the virtual topology, (2) the geographical location of the VRs, (3) the S-VR selection strategy.

The end hosts exchange traffic data during a live session with the DRS involved. In particular, we considered both a UDP-based and a TCP based end-to-end session, and

Table 3: End-to-End Round Trip Time

Topology	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

they are invoked by `iperf` [8]. 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.

4.3.2 Metrics and Logs Collected

We collected two types of metrics and logs for evaluating the network performance during the dynamic reconfiguration. The end-to-end measurements were observed by running `iperf`, and the network performance were evaluated by recovery time and OSPF convergence time.

4.3.2.1 End to End Metrics and Logs

- *Instantaneous Throughput and Loss Rate*: The `iperf` UDP test allows us to obtain the instantaneous datagram loss rate, network throughput as well as latency variance (i.e., jitter) to evaluate the quality of network during the communication. By collecting the real-time UDP session logs, we got the best estimate of the throughput measured for every second, especially when the reconfiguration happened.

- *Instantaneous Bandwidth*: The `iperf` TCP test helped us to estimate the actual bandwidth between end hosts 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.

4.3.2.2 Network Metrics and Logs

- *Recovery Time*: As soon as a VR failure was identified, the *VNM* invoked the DRS to restore the VN. The DRS first disabled links from the failed VR to its neighbors and then the failed VR was replaced with the selected S-VR whose virtual interfaces were properly enabled to connect the original neighbors of the failed VR. *Recovery time* indicates the duration between the occurrence of identifying a failure and the time when a *standby router* is successfully added to the existing VN.

- *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 neighboring information from the OSPF table for each of the VR to log any updates about its neighbors.

4.4 Results Discussion

For all three topologies described in the previous section, we conducted ten independent runs to avoid any artifacts.

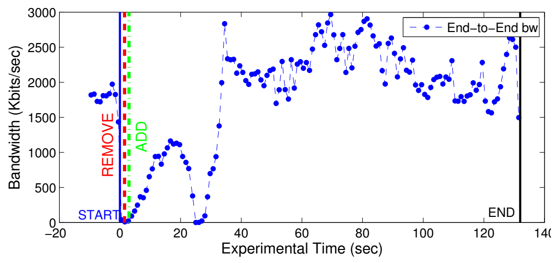
4.4.1 Topology-I(a): A Five-VR Virtual Network across US-Europe

We first discuss Topology-I(a), where we utilized two different S-VR selection strategies: *geolocation-based selection* and *random selection*.

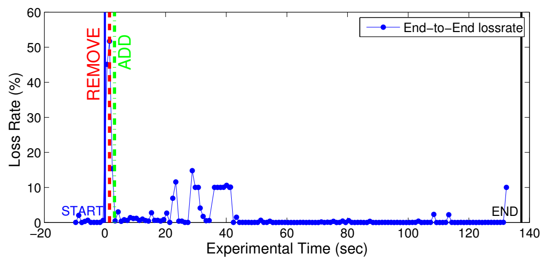
4.4.1.1 Geolocation-based Selection

As observed from the topology shown in Fig. 14(a), Fig. 15(a) - Fig. 15(c) present the instantaneous bandwidth, datagram loss rate and throughput, respectively. Note that the red dotted line indicates the time when a VR (i.e., R1-KU) was failed, and the green dotted line indicates the time when a S-VR (R2-KU) was added to replace R1-KU. The gap between these two dotted lines indicates the *recovery time*, which was about 1.5 seconds, and it means that the entire process of dynamic reconfiguration was pretty fast compared to manual operations on the virtual interfaces. Recall that there is an alternative path to the destination in the topology. Hence, when the R1-KU failed (red dotted line), although there was an immediate packet loss and bandwidth drop due to the failure, the traffic was then redirected to the alternative longer path, and the performance of the network was improved slowly afterward.

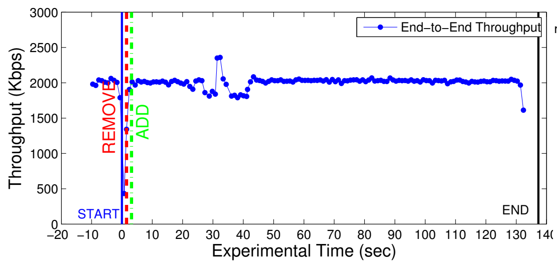
We also interested in the time it takes to get the OSPF routing table to be consistent for all VRs in the VN. Fig. 15(d) presents the next hop to the destination HOST-UMKC



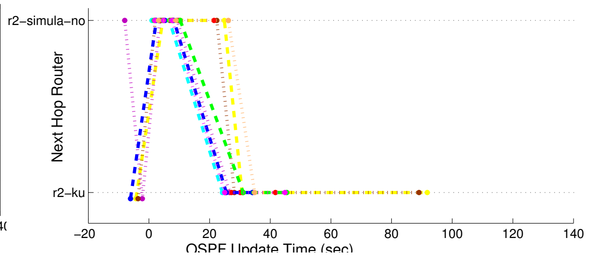
(a) TCP Session Bandwidth



(b) UDP Session Loss



(c) UDP Session Throughput



(d) Next-Hop Information Over 10 Runs

Figure 15: Geolocation-Based Strategy on Topology-I(a)

from the VR R2-TUT-FI's perspective. By parsing the OSPF routing table and the neighboring information, we tracked the next hop router information to the destination. In Fig. 15(d) we use dotted lines in different colors to indicate the OSPF update time during each single run (out of 10 runs), and the dot clusters point out two major OSPF update time periods: one was between 3 - 5 seconds (R1-KU fails), and the other was between 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 shown in Fig. 15(a). Since we use the S-VR to restore the VN back to its original topology, when the routing table converges, the traffic is again redirected to its original path with respect to the shortest path.

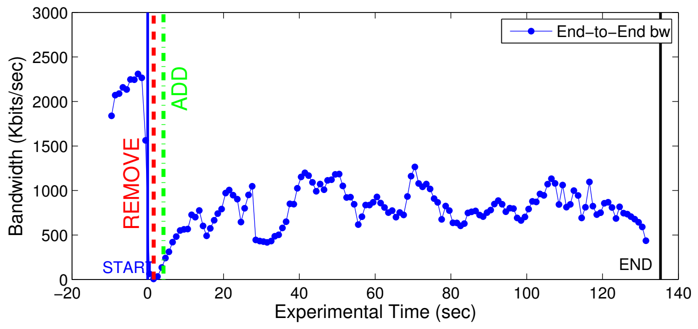


Figure 16: Random Selection Method on Topology-I(a): TCP Session

4.4.1.2 Random Selection

In order to study whether the geolocation-based selection is worthwhile, we compared it to a random S-VR selection. In this random selection case, we still failed the R1-KU physically located in the US, instead of picking a closer S-VR candidate at KU, we randomly picked a node R2-KAU-SE, which is physically located in Sweden.

Fig. 16 shows the end-to-end instantaneous bandwidth due to the reconfiguration process. Compared with Fig. 15(a), we observed that the two major bandwidth drops happened within a similar time interval as we observed for the geolocation-based method, indicating that the OSPF convergence time was not affected by the S-VR selection strategy. However, with the random selection strategy, the instantaneous bandwidth after OSPF routing table has converged was below the original value before the failure, whereas the end-to-end performance under a geolocation-based selection was recovered as before the failure. Thus, a geolocation-based S-VR selection showed less influence to the VN after the reconfiguration than the random selection did.

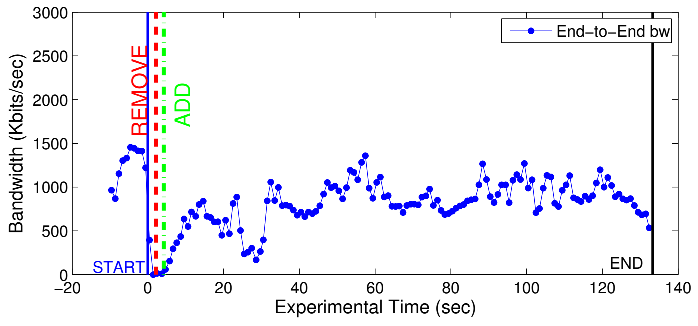
4.4.2 Topology-I(b): A Five-VR Virtual Network in Europe

Fig. 14(b) is the same virtual topology as Topology-I(a), but with nodes from European sites only. When the VR (R1-KAU-SE) failure was identified, during the reconfiguration process, the S-VR candidate R2-KAU-SE was selected because it was closest to the failed VR. Fig. 17(a) plots the instantaneous bandwidth during and after the dynamic reconfiguration process. As we mentioned at the beginning of Section 4.3, 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. 17(a) is different from Fig. 15(a).

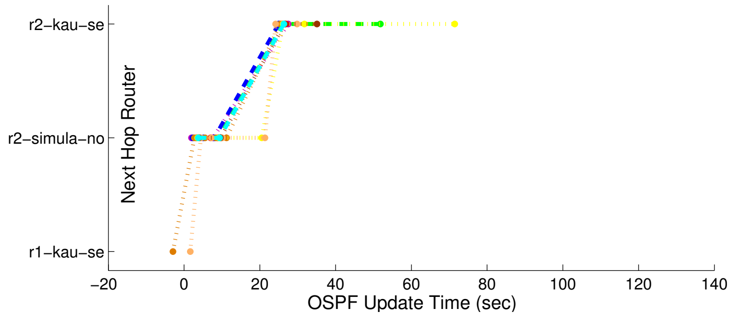
Fig. 17(b) presents the next-hop VR to the destination for about 3 - 5 seconds and 23 - 25 seconds, which point to the two major bandwidth troughs in Fig. 17(a). The routing convergence time we obtained is similar to Fig. 15(d). Hence, we infer that the location of virtual routers in the topology did not show significant impact on the routing table convergence time.

4.4.3 Topology-II

We enforce the geolocation-based selection strategy in a larger network as shown in Fig. 14(c), and present the relevant results in Fig. 18. In this scenario, we failed VR R1-KU, and VR R2-KU was selected from the standby router pool. Fig. 18(a) shows that the measured end-to-end bandwidth is not as large as the one we obtained for Topology-I(a). However, since the number of neighbors to the failed router (R1-KU) is still three, the recovery time was still around 1.5 seconds. The bandwidth pattern was similar to Fig. 15(a), where two main troughs can be easily observed.



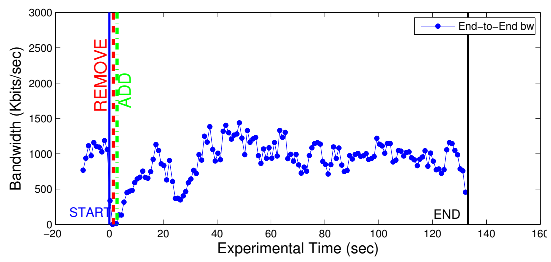
(a) TCP Session Bandwidth



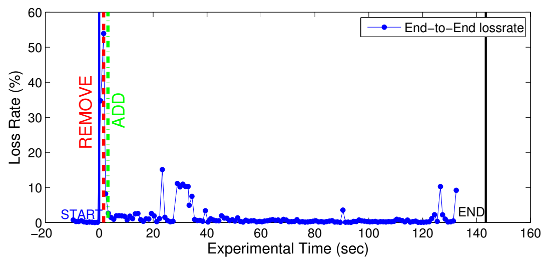
(b) Nexthop Information from Routing Table Over 10 Runs

Figure 17: Geolocation-Based Strategy on Topology-I(b)

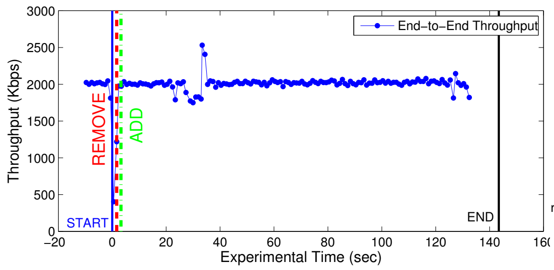
Fig. 18(d) presents the next-hop VR to the destination node HOST-UMKC for each individual run observing from the source node R2-TUT-FI, when the OSPF routing table updated. Because the network between the two end hosts was larger, and we added another crossing link (R1-Vienna to R1-SIMULA-NO), the OSPF routing table might change even though there was no failure in the network at a random time. This is why there were a few dots presenting R1-SIMULA-NO after 70 seconds, where R1-SIMULA-NO indicates the traffic was not routed via the shortest path. However, if we consider the majority dots displayed as clusters, there were two major OSPF updating time periods: 2



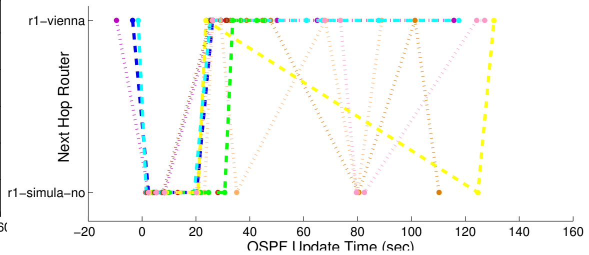
(a) TCP Session Bandwidth



(b) UDP Session Loss



(c) UDP Session Throughput



(d) Next-Hop Information Over 10 Runs

Figure 18: Geolocation-Based Method on Topology-II

- 7 seconds (i.e., the failure was identified), and 24 - 32 seconds (i.e., routing table convergenced). The average of the OSPF convergence time was 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.

4.5 Summary

In this chapter, we presented an experimental study on applying a dynamic reconfiguration scheme that automatically replacing the failed VR with a S-VR in a VN, and the S-VR was selected by considering the distance between itself to the failed VR. We have presented a geolocation-based algorithm to select a candidate from a shared S-VR pool, and the selection algorithm was implemented an dynamic reconfiguration module on the VNM.

The impact of the DRS on the end-to-end traffic and routing metrics was studied on a wide-area VN on the GpENI testbed. First, the geolocation replacing scheme less likely reduces the network performance after the replacement than a randomly replacing scheme. Secondly, the reconfiguration process was fast (about 1.5 second) and a S-VR could be easily added to the existing topology. Third, constructing a VN in the same topology but with a different set of virtual routers affect neither 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 helps to improve the overall performance of the network in case of a failure by using standby routers in a VN. The original optimal path can be quickly restored with the S-VR. However, DRS cannot control the inherent properties of the network. In our case, the OSPF convergence time is independent on the VNM.
- 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. While it is well known to

do multiple independent runs in simulation studies, the need for multiple independent runs in an experimental testbed is also important to keep in mind.

CHAPTER 5

OPTIMAL STANDBY VIRTUAL ROUTER SELECTION MODEL

From the experimental study on the implementation of dynamic reconfiguration scheme on a realistic VNE, we understood the practicalness and how the network dynamics reacted to the reconfiguration process. The GpENI testbed has limitations for us to explore more criteria for the S-VR selection. For instance, we did not have proper tools to measure or predict the latency between the S-VR candidates to other VRs in a VN; secondly, we did not have enough monitoring data about the GpENI backbone network, although this is considered one of the important factors to the selection. Therefore, we design an optimal S-VR selection model that considers all possible selection criteria, and the selected S-VR has minimum cost during the DRS and minimum impact to the substrate network performance as well.

In this chapter, we present an optimization formulation and a heuristic that can be employed by the DRS to restore multiple VNs from either independent or dependent VR failures.

5.1 Problem Formulation

Recall the DRS presented in Chapter 3, the S-VR selection model is expected to simultaneously choose an optimal S-VR for each VN that has a VR failure, and the VR failure could be either *independent failure* or *dependent failure*. In other words, by

Table 4: Substrate Network Model Entities

Symbols	Description
$\mathbb{R}$	the set of substrate nodes
$\mathbb{G}$	the set of VNs
$\mathbb{L}$	the set of substrate links
$\mathbb{Q}_{ik}$	the set of paths from i to k, $i, k \in \mathbb{R}$
$\mathbb{P}_i$	the set of physical interfaces on i, $i \in \mathbb{R}$
h_i	the limit of S-VRs that can be concurrently selected from $i \in \mathbb{R}$
$\mathbb{V}^j$	the subset of $\mathbb{R}$ providing activated VRs in VN j, $j \in \mathbb{G}$
$\mathbb{S}^j$	the subset of $\mathbb{R}$ providing S-VRs for VN j, $j \in \mathbb{G}$
$\mathbb{F}^j$	the set of failed VRs in the VN j, $j \in \mathbb{G}$
n^j	the number of failed VRs in VN j, $n^j = \lvert\mathbb{F}^j\rvert$
B_{ip}	the capacity of the interface p on $i \in \mathbb{R}$;
b_{ip}	the residual capacity of the interface p on $i \in \mathbb{R}$, $b_{ip} \leq B_{ip}$
b_l	the residual capacity of the substrate link $l \in \mathbb{L}$
T	the threshold for maximum bandwidth utilization

applying the optimal S-VR selection model, the dynamic reconfiguration module of the VNM is able to restore multiple VNs simultaneously.

5.1.1 Notations

We first briefly highlight a few key notations used in the optimization model; all are listed in Tables 4 – Table 8.

There are two main sets of components in the VNE: substrate nodes ($\mathbb{R}$) and VNs ($\mathbb{G}$). A VR that belongs to the VN j and is hosted by the substrate node i is denoted by r_i^j. For any substrate node i ($i \in \mathbb{R}$), its physical interfaces are denoted by the set $\mathbb{P}_i$, where each physical interface p ($p \in \mathbb{P}_i$) has the capacity B_{ip}, and it's residual capacity is denoted by $b_{ip}(b_{ip} \leq B_{ip})$. For any VR r_i^j, we define its virtual interfaces as a set $\mathbb{M}_i^j$.

Table 5: Virtual Network Model Entities

Symbols	Description
r_i^j	the VR hosted by the substrate node i for VN j
$\mathbb{M}_i^j$	the set of virtual interfaces on a VR r_i^j
e_{ik}^j	1 if a virtual link between r_i^j and r_k^j is up, 0 otherwise
ϵ_{ik}^j	the bandwidth of the virtual link between VR r_i^j and r_k^j .
c_{imk}^{jf}	the bandwidth required for interface m of a candidate S-VR r_i^j to connect the VR r_k^j, in case VR r_f^j fails

Table 6: Indicator & Cost Parameters

Symbols	Description
δ_i^j	1 if r_i^j is reserved as an S-VR, 0 otherwise
α_i^j	1 if r_i^j is connected in VN j, 0 otherwise
β_i^j	1 if a VR r_i^j fails, 0 otherwise
γ_{imk}^{jf}	1 if a virtual interface m on r_i^j is to connect r_k^j
$\Delta_{ql}^{(jfik)}$	link-path indicator; for a bandwidth demand between VRs r_i^j and r_k^j, $\Delta_{ql}^{(jfik)} = 1$ if a substrate path $q \in \mathbb{Q}_{ik}$ between nodes i and k uses the substrate link $l \in \mathbb{L}$, 0 otherwise.
η_{imk}^{jf}	the cost of configuring a virtual interface m of a S-VR r_i^j to connect to the VR r_k^j, in case VR r_f^j fails.
σ_{imk}^{jf}	the cost of connecting an S-VR to a VR r_k^j in a VN j via the S-VR's virtual interface m, in case VR r_f^j fails.

Table 7: Variables

Variable	Description
u_i^{jf}	binary variable; 1 if an S-VR r_i^j is selected to replace a failed VR r_f^j
v_{imk}^{jf}	binary variable; 1 if an S-VR r_i^j is connected to the failed VR r_f^j's neighbor r_k^j via virtual interface m
t	maximum bandwidth utilization on substrate nodes
$x_q^{(jfik)}$	flow variable on a substrate path $q \in \mathbb{Q}_{ik}$; under a failure of r_f^j ($f \in \mathbb{F}^j$), q is the substrate path between a S-VR v_i^j and the failed VR r_f^j's neighbor r_k^j

Table 8: Weight Parameters

Symbols	Description (all non-negative)
λ, π	Indicates the proportion of different objectives, $\lambda + \pi = 1$
w_θ	Primary weight used in (5.12): w_θ = $\langle w_{\theta 1}, w_{\theta 2}\rangle$, $w_{\theta 1} + w_{\theta 2} = 1$
w_b	Secondary weights in (5.15): w_b = $\langle w_{b1}, w_{b2}\rangle$, $w_{b1} + w_{b2} = 1$
w_a	Third level weights in (5.15): w_a = $\langle w_{a1}, w_{a2}\rangle$, $w_{a1} + w_{a2} = 1$
ρ	Threshold for the ratio of π over λ ($\frac{\pi}{\lambda}$)

For each VN j ($j \in \mathbb{G}$), we denote $\mathbb{F}^j$ as failed VRs in the VN j.

For any VN j ($j \in \mathbb{G}$) that has already been created and is in service, the substrate nodes hosting the VRs that have been connected are pre-established, is denoted by set $\mathbb{V}^j(\subset \mathbb{R})$; thus, the following parameters are set: $\alpha_i^j = 1$, $e_{i,k}^j = 1$, $\beta_i^j = 0$ for $i, k \in \mathbb{V}^j$. We also denote the substrate nodes hosting the S-VRs pre-reserved for a VN j as $\mathbb{S}^j(\subset \mathbb{R})$, so $\delta_i^j = 1$ for S-VRs, $i \in \mathbb{S}^j$. In other words, $\delta_i^j = 0$ for $i \in \mathbb{R}\backslash\mathbb{S}^j$. When a VR r_i^j fails, we set $\beta_i^j = 1$, where $i \in \mathbb{V}^j \cup \mathbb{S}^j(\subseteq \mathbb{R})$. The parameter h_i indicates the maximum number of S-VRs associated with a substrate node $i \in \mathbb{R}$ that may be concurrently activated during the reconfiguration process, and the value of h_i can be decided based on the real-time resource utilization on the substrate nodes. For example, if h_i is associated with the CPU status at a substrate node i, a lower CPU usage indicates a higher h_i value that can be set.

Residual resource on a substrate node is an important factor for the S-VR selection with respective to the load balancing on the substrate network. In this work, we define the residual capacity of an interface p at a substrate node i as b_{ip} ($p \in \mathbb{P}_i$). If we consider the substrate link l ($l \in \mathbb{L}$), b_{ip} is equivalent to the residual capacity of l whose one end is substrate node i. Assuming for an interface p on i, it's capacity is B_{ip}, then it's residual capacity can be represented in (5.1). From a substrate node i's perspective, a binary indicator μ^j_{ikp} indicates whether the traffic between two VRs r^j_i and r^j_k is going through the interface p on i.

$$b_{ip} = B_{ip} - \sum_{j \in \mathbb{G}} \sum_{k \in \mathbb{R}} \mu^j_{ikp} \, \epsilon^j_{ik} \, e^j_{i,k},$$
$$i \neq k, \forall p \in \mathbb{P}_i, \forall i \in \mathbb{R} \tag{5.1}$$

The solution to the reconfiguration problem is determined by two sets of binary variables u and v. In addition, the variable t represents the maximum bandwidth utilization of the substrate nodes after the S-VRs have been decided, while x represents the flow from the selected S-VR to the failed VR's neighbors (hereafter in this paper, we use neighbors to represent the failed VR's neighbors). Thus, our formulation lands in an MILP formulation.

5.1.2 Constraints

There are ten sets of constraints associated with the optimal S-VR selection.

- A VR r^j_i can be selected to replace a failed VR r^j_f only if it is identified as an S-VR and it is not failed.

$$u^{jf}_i \leq \delta^j_i (1 - \beta^j_i),$$
$$\forall i \in \mathbb{R}, \forall f \in \mathbb{F}^j, \forall j \in \mathbb{G}, i \neq f \tag{5.2}$$

- For any VR failure in a VN, only one identified S-VR can be selected.

$$\sum_{i \in \mathbb{R}} \delta^j_i \, u^{jf}_i = 1, \forall f \in \mathbb{F}^j, \forall j \in \mathbb{G} \tag{5.3}$$

- One reserved S-VR can replace at most one failed VR at a time:

$$\sum_{f \in \mathbb{F}^j} \delta_i^j \, u_i^{jf} \leq 1, \forall i \in \mathbb{R}, \forall j \in \mathbb{G} \tag{5.4}$$

- The allocation must satisfy the upper bound on the maximum number of S-VRs that can be selected from the same substrate node at a time, based on the substrate node resource utilization at the moment (e.g., CPU utilization).

$$\sum_{f \in \mathbb{F}^j} \sum_{j \in \mathbb{G}} \delta_i^j u_i^{jf} \leq h_i, \forall i \in \mathbb{R} \tag{5.5}$$

- A virtual interface m on an S-VR r_i^j can be enabled to connect r_k^j (a neighbor of the failed VR r_f^j), only when r_i^j is selected to replace r_f^j.

$$\begin{gathered} \gamma_{imk}^{jf} \, v_{imk}^{jf} \leq u_i^{jf}, \\ \forall f \in \mathbb{F}^j, \forall i, k \in \mathbb{R}, \forall j \in \mathbb{G}, \forall m \in \mathbb{M}_i^j, i \neq k \neq f \end{gathered} \tag{5.6}$$

- For topology integrity from before failure to after failure, conservation of links during the reconfiguration process must be satisfied.

$$\begin{gathered} \sum_{m \in \mathbb{M}_i^j} \sum_{k \in \mathbb{R}} e_{fk}^j \, \delta_i^j \, \gamma_{imk}^{jf} \, v_{imk}^{jf} = \sum_{k \in \mathbb{R}} \delta_i^j \, u_i^{jf} \, \beta_f^j \, e_{fk}^j \\ \forall j \in \mathbb{G}, \forall f \in \mathbb{F}^j, \forall i \in \mathbb{R}, i \neq k \neq f \end{gathered} \tag{5.7}$$

Here, the left side of (5.7) indicates that a candidate S-VR (r_i^j) for a failed VR (r_f^j) should be able to establish connectivities to every failed VR's neighbor (r_k^j, where $i \neq k$), via specific virtual interface m ($m \in \mathbb{M}_i^j$)—this must be equal to the right side of (5.7) that represents the total number of virtual links that should be created from r_i^j in order to replace r_f^j.

- If an S-VR r_i^j is to be selected in case the VR r_f^j fails, the aggregated bandwidth requested from its virtual interfaces to be connected should not exceed the aggregated residual interface capacity on the corresponding substrate node:

$$\sum_{m \in \mathbb{M}_i^j} \sum_{k \in \mathbb{R}} \delta_i^j \, \gamma_{imk}^{jf} \, c_{imk}^{jf} \, v_{imk}^{jf} \leq \sum_{p \in \mathbb{P}_i} \delta_i^j \, b_{ip} \, u_i^{jf},$$
$$\forall j \in \mathbb{G}, \forall f \in \mathbb{F}^j, \forall i \in \mathbb{R}, i \neq k \neq f, \tag{5.8}$$

- The capacity utilization on a substrate node does not exceed the maximum bandwidth utilization t, due to the extra bandwidth demand by the one or more selected S-VRs hosted by this substrate node. Note that μ_{ikp}^j equals to 1 if the traffic between substrate nodes i and k goes through the interface p on i, 0 otherwise.

$$\sum_{j \in \mathbb{G}} \sum_{f \in \mathbb{F}} \delta_i^j \, u_i^{jf} \, (\sum_{k \in \mathbb{R}} \sum_{m \in \mathbb{M}_i^j} \gamma_{imk}^{jf} \, c_{imk}^{jf}) + \sum_{p \in \mathbb{P}_i} \sum_{j \in \mathbb{G}} \sum_{k \in \mathbb{R}} \mu_{ikp}^j \, \epsilon_{ik}^j \, e_{ik}^j$$
$$\leq \sum_{p \in \mathbb{P}_i} B_{ip} \, t, \ \ \forall i \in \mathbb{R}, i \neq k \neq f \tag{5.9}$$

- When selecting an S-VR r_i^j, we must guarantee that the links on the path from the substrate node i hosting r_i^j to the substrate node k hosting the failed VR r_f^j's neighbor r_k^j has sufficient residual bandwidth on the substrate network, and that the bandwidth demand on the new virtual link are the same as before. This can be represented through the following demand flow and capacity constraints:

$$\sum_{q \in \mathbb{Q}_{(ik)}} x_q^{(jfik)} = \sum_{m \in \mathbb{M}_i^j} \gamma_{imk}^{jf} \, c_{imk}^{jf} \, \delta_i^j \, u_i^{jf},$$
$$\forall j \in \mathbb{G}, \forall f \in \mathbb{F}^j, \forall i, k \in \mathbb{R}, i \neq k \neq f \qquad (5.10)$$

$$\sum_{(jfik)} \sum_{q \in \mathbb{Q}_{ik}} \Delta_{ql}^{(jfik)} \, x_q^{(jfik)} \leq b_l,$$
$$\forall l \in \mathbb{L}, i \neq k \neq f \qquad (5.11)$$

5.1.3 Objective Function

Our goal is to achieve the network cost minimization and load balancing while selecting S-VRs for multiple VNs for restoration against node failures in a VNE. Thus, the proposed MILP formulation is a dual–objective problem: minimum network cost (MNC) and load balancing (LB).

We consider two different cost components associated with the MNC objective. The first is the operation cost of manipulating virtual interfaces (Type-I: η_{imk}^{jf}), i.e, configuring, activating and deactivating virtual interfaces. The second type is the connectivity cost of adding an S-VRs (Type-II: σ_{imk}^{jf}) to the existing VN in case of a VR failure, i.e., the geographical distance or the RTT between two VRs. To select a proper S-VR, we address the relative importance of the operation cost and connectivity cost. As a result, we assign a pair of weight parameters $w_\theta = \langle w_{\theta 1},\ w_{\theta 2} \rangle$ to indicate the importance of the relevant cost types, where $w_{\theta 1}$ is for Type-I cost ($\eta_{i,m,k}^{i,j}$), and $w_{\theta 2}$ is for the Type-II cost ($\sigma_{i,m,k}^{j,f}$). For the LB objective, we aim to minimize the maximum utilization of the aggregated interface capacity (i.e., t) on the substrate nodes.

Due to the dual–goal, we use two weight parameters λ and π, respectively, where λ + π = 1. The dual–problem goal can be written in a combined manner as follows:

$$\min\{\lambda\,[\sum_{j\in\mathbb{G}}\sum_{f\in\mathbb{F}^j}\sum_{i\in\mathbb{R}}\sum_{m\in\mathbb{M}_i^j}\sum_{k\in\mathbb{R}}(w_{\theta 1}\,\eta_{imk}^{jf}+w_{\theta 2}\,\sigma_{imk}^{jf})\,\delta_i^j\,\gamma_{imk}^{jf}\,v_{imk}^{jf}]+\pi\,t\},$$
$$i\neq k\neq f \tag{5.12}$$

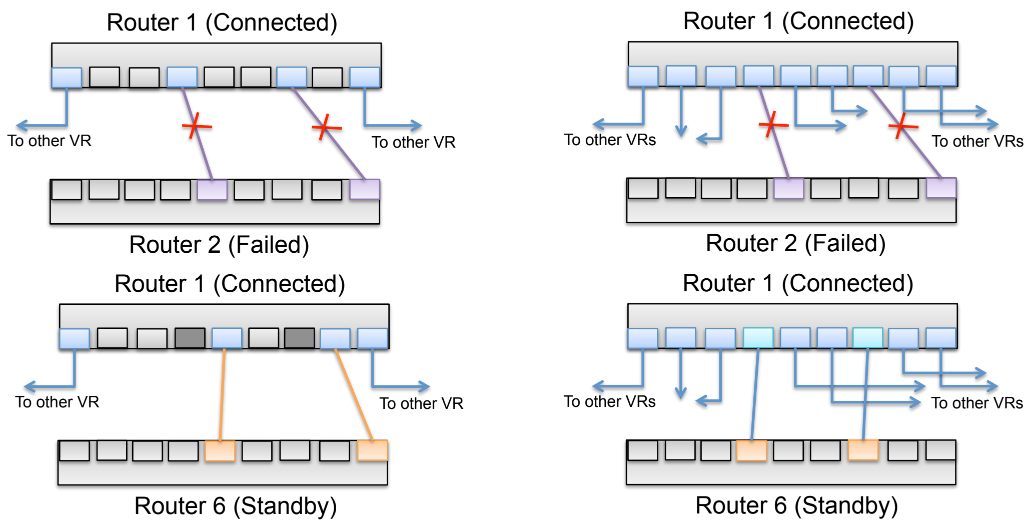
A summary of all weight parameters is presented in Table 8, and we will discuss the weight parameters associated with different cost components next.

5.1.4 Cost Components

We now elaborate on various cost components.

5.1.4.1 Type-I: Virtual Interface Operations Cost η_{imk}^{jf}

In the substrate network, when adding or removing a substrate node from the network, the basic operations are to enable or disable the interfaces if the router has already been connected. If the router has not been connected yet, then plugging cables is an extra operation. Furthermore, if an interface has not been configured, then configuration (e.g., IP address assignment) is the extra operations cost. Analogously, the operations cost on the virtual interfaces also needs to consider these three parts: creation of virtual link on a virtual interface, (similar to a plugin cable to a substrate router), virtual interface configuration, and enabling/disabling the virtual interfaces. Here, we combine the first two into one operations cost. This is because creating virtual links and assigning IP addresses can be done by the programmable operation in a VNE. For example, in [79], when a user



(a) With Extra Virtual Interfaces (b) No Free Virtual Interfaces

Figure 19: Two Scenarios on Virtual Interface Operations

specified a topology, the virtual link between two VRs was created automatically, and the corresponding virtual interfaces were assigned virtual IP addresses at the same time. The other operation cost in a VNE is due to enabling necessary virtual interfaces on the virtual interfaces. It is intuitive that not all virtual interfaces need to be enabled, and the number of active virtual interfaces on an S-VR depends on the number of VRs that it needs to be connected to. Meanwhile, the interfaces of other VRs connected to the failed VR have to be disabled as well.

To construct the Type-I cost function for the virtual interface operation, we consider two potential scenarios. First, as presented in Fig. 19(a), each VR has enough virtual interfaces with preconfigured virtual IP addresses, and all of the S-VRs' virtual interfaces are disabled. When an S-VR is selected, we just need to enable and disable its corresponding virtual interfaces. In other words, the first scenario considers only one part of the Type-I cost. Secondly, as shown in Fig. 19(b), a connected VR has a limited number of virtual interfaces and all of them are active, so if it needs to connect to the selected S-VR, the specific virtual interface must be re-initialized with a new IP address to connect the selected S-VR. Thus, this scenario needs to include both parts of the Type-I cost. Note that s^- and s^+ represent the cost of disabling a virtual interface on a neighboring VR r_i^j and enabling the virtual interface m on the S-VR r_i^j, respectively. The indicator parameter τ_i^{jf} is a binary value that indicates whether re-initialization of the virtual IP addresses for virtual interfaces is needed, and ξ is the cost of configuring virtual IP addresses. To summarize, the Type-I cost can be presented as follows:

$$\begin{aligned} \eta_{i,m,k}^{j,f} &= 2\left(s^- + s^+\right) + \tau_i^{jf}\, \xi \\ &\forall m \in \mathbb{M}_i^j, \forall k \in \mathbb{R}, i \neq k \neq f \end{aligned} \tag{5.13}$$

5.1.4.2 Type-II: Virtual Router Connectivity Cost σ_{imk}^{jf}

For the Type-II cost of connecting two VRs, we consider two factors: geographical distance (i.e., great-circle distance) and round trip time (RTT).

First, the geographical distance between two substrate nodes i and k, represented as d_{ik} ($i, k \in \mathbb{R}$), is calculated as the great-circle distance between the latitude-longitude coordinates of these two nodes. We consider two types of distance associated with an S-VR, which is the distance between an S-VR and the failed VR (d_{if}), or the failed VR's neighbor (d_{ik}). We use a pair of weight parameters $w_a = \langle w_{a1}, w_{a2} \rangle$, where $w_{a1} + w_{a2} = 1$, to indicate the portion of each type of distance's influence.

Secondly, In a VN j, a direct virtual link $e_{i,k}^j$ can be embedded onto a multi-hop path between substrate nodes i and k. Thus, the propagation delay between i and k (rtt_{ik}) is not always driven by the direct geographical distance d_{ik}. If there is a direct virtual link created between substrate nodes i and k, then $\mathbb{Q}_{i,k}$ represents the paths from i to k on the substrate network. We define a substrate link to be $l : (y, z), l \in \mathbb{L}$ on the shortest path $q \in \mathbb{Q}_{ik}$, where y and z are intermediate substrate nodes.

$$rtt_{ik} = \sum_{(yz)\in q} rtt_{yz}, q \in \mathbb{Q}_{ik} \tag{5.14}$$

$$\begin{aligned} \sigma^{jf}_{imk} &= w_{b1}\,(w_{a1}\,d_{if} + w_{a2}\,d_{ik}) + w_{b2}\,rtt_{ik}, \\ &\forall j \in \mathbb{G}, \forall m \in \mathbb{M}^{j}_{i}, i \neq k \neq f \end{aligned} \tag{5.15}$$

By considering both impact of geographical distance and RTT factors, the Type-II cost can be constructed as (5.15), where $w_a = \langle w_{a1}, w_{a2}\rangle$ emphasizes the weights between distance types and $w_b = \langle w_{b1}, w_{b2}\rangle$ emphasizes the weights for distance and RTT factors, where $w_{b1} + w_{b2} = 1$.

5.2 Heuristic Algorithm

Taking a cue from the MILP formulation, we now introduce a heuristic algorithm that selects S-VRs to restore multiple VNs from VR failures.

5.2.1 Algorithm Description

Algorithm 3 takes the following parameters as the inputs.

Algorithm 3: Heuristic Multi-Criteria S-VR Selection

```
Input: $\mathbb{G}', \mathbb{L}, \mathbb{R}, \tilde{d}, \tilde{b}, \widetilde{rtt}, w_a, w_b, w_\theta, \lambda, \pi, T, \rho$
Output: $selected$
1  $selected \leftarrow \phi$ ;
   //Sort the $\mathbb{G}'$ by the bandwidth used by $f \in \mathbb{F}^j, j \in \mathbb{G}$
2  $\hat{\mathbb{G}}' \leftarrow$ sort($\mathbb{G}$, option = "bandwidth") ;
3  for $j$ in $\hat{\mathbb{G}}$ do
       //Step 1: Selection Cost Calculation (Algorithm 4)
4      $j.svrH \leftarrow$ SCC($j, \tilde{d}, \widetilde{rtt}, w_a, w_b, w_\theta$) ;
       //Step 2: Final Selection with MNC or LB Decision
5      $j.svrH' \leftarrow$ sort($j.svrH$) ;
6      for $i$ in $j.svrH'$ do
7          $t' \leftarrow i$.UpdateUtil($j.f_{bw}$) ;
8          if $\forall k \in j.f_{nbrs}$ PathExsits ($i, k, \mathbb{L}$) then
9              if Count $(selected, i) \leq i.h$ then
10                 if $t' \leq T$ && $\pi > \rho\ \lambda$ then
11                     $i_{min}$= FindMinUtil($j$, $\mathbb{R}$, $j.f_{bw}$) ;
12                     if $i_{min} == i$ then
13                         UpdateUtil($i$, $j.f_{bw}$) ;
14                         selected.insert($i$) ;
15                         break;
16                 else if $t' \leq T$ then
17                     selected.insert($i$) ;
18                     UpdateUtil($i$, $j.f_{bw}$) ;
19                 else
20                     Exceed the utilization threshold!
21         else
22             Can not allocate Path!
23 return $selected$
```

Algorithm 4: S-VR Selection Cost Calculation (SCC)

```
    Input: $j \in \mathbb{G}', \tilde{d}, \widetilde{rtt}, w_a, w_b, w_\theta$
    Output: j.svrH
    /* The hash table j.svrH = ⟨id, cost⟩ stores the cost of selecting an S-VR in $j$ */
1   $j.svrH \leftarrow \phi$ ;
2   for $i$ in $j.svr\_list$ do
        /* Compute Type-I, Type-II and total selecting cost using (5.13), (5.15), (5.12) */
3       $i.\eta \leftarrow$ GetTypeI($i$, $j$) ;
4       $i.\sigma \leftarrow$ GetTypeII($i$, $j$, $w_a$, $w_b$, $d$, $rtt$) ;
5       $i.cost \leftarrow$ GetCost($i.\eta$, $i.\sigma$, $w_\theta$) ;
6       $j.svrH$.Insert($i$, $i.cost$) ;
7   return $j.svrH$
```

- Basic elements of the VNE: the VNs with failures $\mathbb{G}'$ ($\subseteq \mathbb{G}$), the substrate links $\mathbb{L}$, and substrate nodes $\mathbb{R}$;
- Geographical distance matrix $\tilde{d}$ of substrate nodes ($\tilde{d} = \{(i,k)|d_{ik}, i,k \in \mathbb{R}, i \neq k\}$);
- The most recent monitored residual bandwidth matrix $\tilde{b}$ on the SN ($\tilde{b} = \{(i,p)|b_{ip}, i \in \mathbb{R}, p \in \mathbb{P}_i\}$);
- The most recent monitored the RTT matrix ($\widetilde{rtt}$) of the VRs ($\widetilde{rtt} = \{(i,k)|rtt_{ik}, i,k \in \mathbb{R}, i \neq k\}$);
- Weights, w_θ, w_a, w_b, on each factor (refer to (5.15) and (5.12)) to understand the influence on the selection and the weights λ and π associated with MNC and LB objectives, respectively.
- A threshold T for the maximum residual bandwidth utilization of substrate nodes

to guarantee that the substrate nodes are not congested.

- A pre-determined ratio of π over λ, indicates the switch to determine whether to apply the MNC or LB objective.

In the proposed multi-criteria S-VR selection heuristic, we first sort the affected VNs ($\mathbb{G}'$) based on the aggregated bandwidth allocated to the failed VR within each VN, and starts to select the S-VR from the VN whose failed VR allocated with highest bandwidth.

We consider two major steps to determine the proper S-VR for each affected VN in the VNE, successively. Algorithm 4 presents the Step One: calculating the network cost of each candidate S-VR within a VN j and storing it into a hash table $j.svrH$. For Step Two, we sort $j.svrH$ obtained from Step One by the computed cost. Starting from the candidate S-VR with the least cost, we compute the potential bandwidth utilization (i.e., t') on the corresponding substrate node. The ideal S-VR should satisfy three conditions: (i) There should be at least one path existed between the S-VR and each of its potential neighbors; (ii) a substrate node i can only have at most $i.h$ (i.e., h_i as defined in Table 4) S-VRs to be activated concurrently; (iii) once the S-VR is activated, the bandwidth utilization on its host should not exceed the threshold T. Since the MILP formulation is a composite of two objectives, the heuristic also needs to consider the LB problem during the selection. We define an inequality relation between π and $\rho\ \lambda$ to determine whether the heuristic is to apply MNC or LB based selection, where ρ is a pre-assigned value indicating the decision. When $\pi > \rho\ \lambda$, the heuristic makes decision based on the LB objective, where it picks the S-VR that resides on the substrate node with the least bandwidth utilization and also has the minimum selecting cost in the rest of $j.svrH$. When $\pi \leq \rho\ \lambda$, the heuristic considers the purely minimum cost selection objective.

5.2.2 Computational Complexity

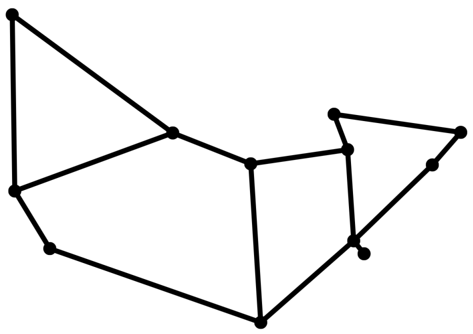
According to the model entities summarized in Table 4, a VNE has $|\mathbb{G}|$ VNs, $|\mathbb{R}|$ substrate nodes, and $|\mathbb{L}|$ substrate links. We define the average number of paths between any pair of substrate nodes as a constant $\bar{Q}$, and the average number of S-VRs for each VN is $\bar{S} = (\sum_{j\in\mathbb{G}} |\mathbb{S}^j|)/|\mathbb{G}|$.

Consider the worst case, where every VN fails, $\mathbb{G}' = \mathbb{G}$. Thus, Line 2 in Algorithm 3 has the complexity as $\mathcal{O}(|\mathbb{G}| \log |\mathbb{G}|)$. Step One has the complexity of $\mathcal{O}(\bar{S} \log \bar{S})$. In Step Two, the functions `PathExists` and `FindMinUtil` have the complexity of $\mathcal{O}(|\mathbb{R}|\ \bar{Q})$ and $\mathcal{O}(\bar{S})$, respectively. Thus, the time complexity for Step Two is $\mathcal{O}(\bar{S} \log \bar{S} + \bar{S}\ (|\mathbb{R}|\ \bar{Q} + \bar{S}))$ = $\mathcal{O}(\bar{S}^2 + \bar{S}|\mathbb{R}|)$. In summary, in the worst case, the computational complexity of the proposed heuristic algorithm is $\mathcal{O}(|\mathbb{G}| \log |\mathbb{G}| + |\mathbb{G}|\ (\bar{S} \log \bar{S} + \bar{S}^2 + \bar{S}|\mathbb{R}|)) = \mathcal{O}(|\mathbb{G}| \log |\mathbb{G}| + |\mathbb{G}|\ (\bar{S}^2 + \bar{S}|\mathbb{R}|)$.

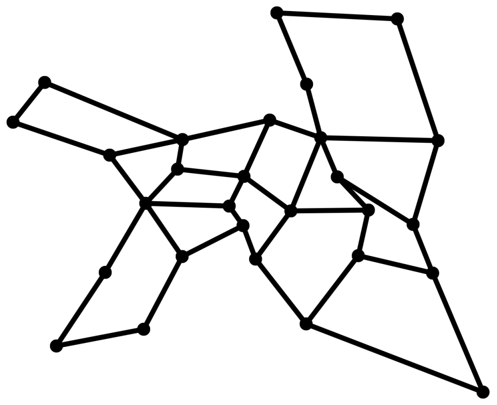
5.3 Experimental Design

We chose three topologies (i.e., *Abilene*, *Nobel-EU* and *Germany50*) obtained from the `SNDlib` (Survivable Network Design Library) [86] as substrate networks in our study, and they are shown in Fig. 20.

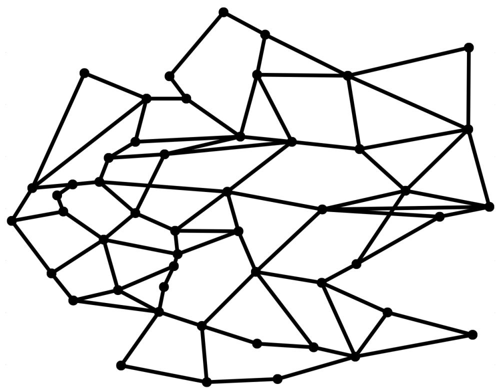
Consider the origin of a VR failure, which could be either a software failure at the VR itself, or a failure at its host (i.e., substrate node failure). For a software failure at a VR in a VN, it will not affect other VNs, so we denote this as an *independent* VR failure.



(a) *Abilene* Topology



(b) *Nobel-EU* Topology



(c) *Germany50* Topology

Figure 20: Three Topologies

For a VR failure caused by a failure at the associated substrate node, other VNs may also be affected if these VNs contain the VRs on the failed substrate node—this is referred to as a *dependent* VR failure.

Table 9 summarizes the experimental scenarios in this paper. We assume each SN provisions 10 VNs, and each VN is generated randomly. Hence, the VNs are independent from each other in terms of the S-VR availability. For each SN, the distance between substrate nodes is normalized between 0 and 1. For each physical interface of a substrate node i, we set the $B_i = 1$, so $0 \leq b_{ip} \leq 1$, and the aggregated residual capacity at i is represented as $\sum_{p \in \mathbb{P}_i} b_{ip}$ ($\forall i \in \mathbb{R}$).

We consider a number of factors to the S-VRs selection for all VNs with VR failures: (1) the number of S-VRs provided for a VN, (2) The number of VNs to be restored (i.e., VNs affected by the failed VR), (3) the restriction on the amount of S-VRs can be selected from a substrate node (refer to h_i) for restoring multiple VNs, and (4) Bandwidth allocated to virtual links of a VN. In particular, the number of VR failures can be equal up to the total number of VNs. This is possible when there is a substrate node failure that affects every VN, and we also simulate up to 10 concurrent independent VR failures. On the other hand, if the virtual links between a candidate S-VR and its potential neighbors (i.e., neighbors of the failed VR) request a high bandwidth, the corresponding substrate paths might not have enough capacity to carry traffic for these virtual links. Hence, under this circumstance, an S-VR with more aggregated residual capacity turns out to be a better candidate. In this work, we set the $T = 0.9$ for all scenarios; in other words, the maximum bandwidth utilization on any substrate node cannot exceed 90% of its aggregated interface capacity.

The MILP formulation consists of two objectives: MNC and LB (see (5.12)). For the network cost part, we defined weight parameters $w_\theta = \langle w_{\theta 1}, w_{\theta 2} \rangle$ associated with the two types of cost functions. In this study, we assumed there are sufficient virtual interfaces on each VR, and all the virtual interfaces are homogeneous, so we set $\tau_i^{jf} = 0$, and set η_{jmk}^{jf} as a constant η for Type-I cost function (5.13). Our previous study [67] showed that disabling/enabling virtual interfaces were fast. Hence, although Type-I cost

Table 9: Experimental Scenarios (10 VNs in a VNE)

Substrate Network	*Abilene*	*Nobel-EU*	*Germany50*
nodes / links	12 / 15	28 / 41	50 / 88
average node degree	2.50	2.92	3.52
VN size (# of VRs)	$3-6$	$3-16$	$3-26$
# of reserved S-VRs per VN	2 - 6	8 - 12	4 - 24
# of independent VR failures	1 - 10	1 - 10	1 - 10
# of dependent VR failures	10	10	10
h_i (max # of S-VRs on i)	1 - 10	1 - 10	1 - 10
virtual link demand (low)	$(0-1)\%$	$(0-1)\%$	$(0-0.5)\%$
virtual link demand (high)	$(0-8)\%$	$(0-4)\%$	$(0-2)\%$

(i.e. interface operation cost) depends on the number of virtual interfaces to be disabled or enabled, given the homogeneous property of the virtual interfaces, its impact is less significant than Type-II cost (i.e. connectivity cost). Hence, we set $w_\theta = \langle 0, 1 \rangle$, so that the connectivity cost is considered as the major cost. In the next section, we present the impact of the third and secondary weight parameters (w_a and w_b in (5.15)), respectively.

We modeled the MILP problem using `AMPL` (Student Version: Darwin 10.8.0 x86_32) and solved it using `CPLEX` (version 12.5.1.0) – this solution is referred to as the optimal solution, and the overall cost returned by `CPLEX` is referred as *optimal cost*. The complete `AMPL` model file can be found in Appendix B. The heuristic algorithm was implemented in `Python 2.7.8`, and the corresponding total cost calculated is referred as *heuristic cost*. We use normalized cost overhead (i.e., cost overhead) to evaluate the gain of optimal solution, and it is represented by $\frac{\text{(heuristic cost)}-\text{(optimal cost)}}{\text{(optimal cost)}}$. Since the VNs were randomly created initially, so were the failed VR and S-VRs availability for each VN, we repeated 30 independent runs for each case study, and computed the 95% confidence interval (95% CI) for the average cost overhead.

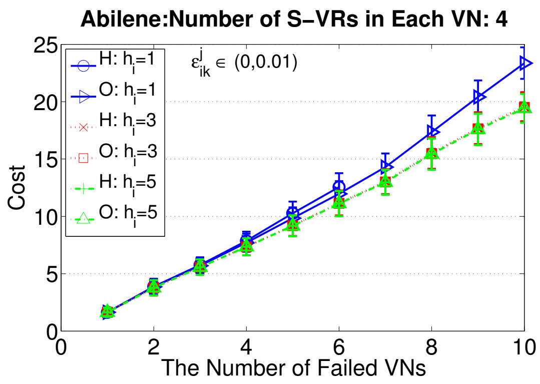
5.4 Results Discussion

In this section, we discuss the numeric results from the MILP formulation and the heuristic algorithm. We aim to answer four questions: (1) How does the the network cost grow when the number of VN failures increase? (2) What is the influence of weight parameters to the MILP solution and the heuristic solution, or the cost overhead? (3) How much is the gain of the MILP solution for different scenarios, compared to the heuristic solution? (4) What is the computational time of solving the MILP problem, compared to running the heuristic algorithm?

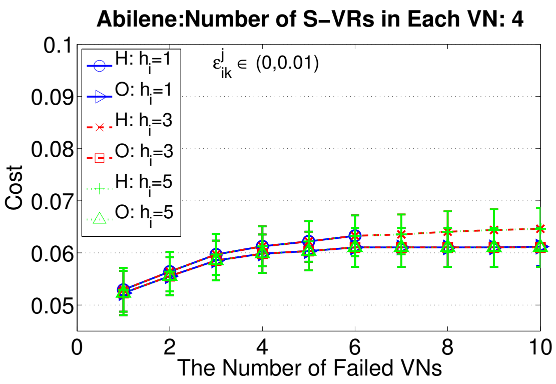
5.4.1 Growth Pattern of the Cost

We first use *Abilene* topology to discuss how optimal and heuristic cost grows as the number of failed VNs increases, respectively. Note that in the rest of the paper, we refer a VN with single-VR failure as a failed VN or a VN failure. We consider two extreme cases: MNC based selection only (i.e. $\lambda = 1, \pi = 0$) and LB based selection only (i.e., $\lambda = 0, \pi = 1$). Fig. 21 shows the growth patterns of network cost for two extreme cases, when each virtual link requested less than 1% of substrate link's capacity (i.e., low-level bandwidth request). Since each VN was assigned 4 S-VRs randomly, there might be the possibility that more than one S-VRs provisioned for different VNs are from a common substrate node. Due to the resource availability on the substrate node, we varied the value of h_i from 1 to 9 and report the results for $h_i = 1, 3, 5$, respectively.

Fig. 21(a) shows an MNC-oriented selection (i.e., $\lambda = 1, \pi = 0$). When $h_i = 1$ (i.e., no more than one VN can have the S-VR selected from a common substrate node),

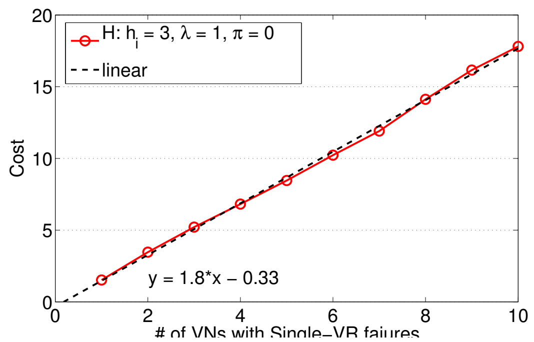


(a) $\lambda = 1, \pi = 0$

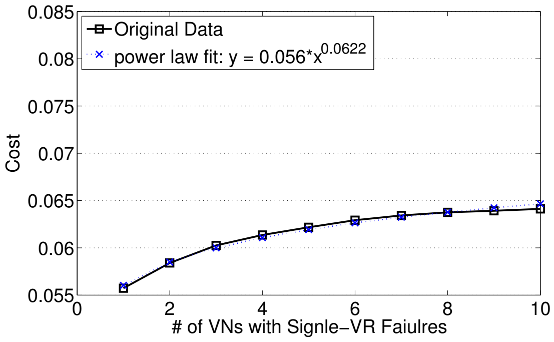


(b) $\lambda = 0, \pi = 1$

Figure 21: *Abilene*: Independent VR Failures (Cost v.s. Failures)



(a) $\lambda = 1, \pi = 0 : h_i = 3$



(b) $\lambda = 0, \pi = 1 : h_i = 3$

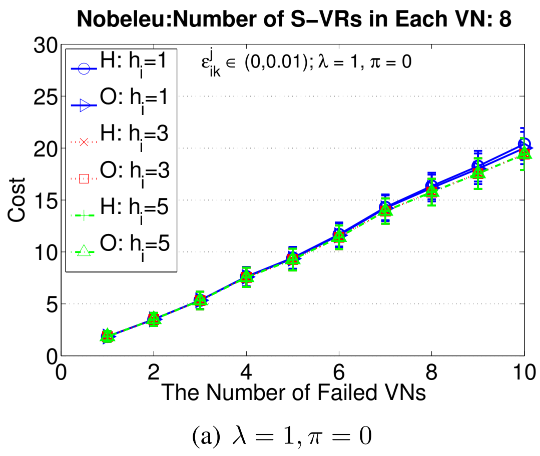
Figure 22: *Abilene*: Regression Analysis on Cost v.s. Failures

if more VNs failed concurrently, the heuristic cost was higher than the optimal cost (i.e., 3 - 6 VN failures) or the heuristic did not find feasible solutions (i.e., more than 6 VN failures). Note that feasible solution means all affected VNs can be restored with proper selected S-VRs, if we cannot find a proper S-VR to restore even one VN, then the solution to the whole problem is considered as infeasible. When we increased h_i, for the cases with the same amount of failures, both optimal cost and heuristic cost decreased, and the heuristic algorithm found feasible solution for the case when all 10 VNs suffered single-VR failure concurrently. This implies that substrate nodes with more physical resources improved the heuristic solutions. In order to understand the relation between the cost and the number of failures, we did a regression analysis for the average heuristic cost v.s. failures when $h_i = 3$ (see H:$h_i = 3$ plot in Fig. 21(a)). From Fig. 22(a), we found that the cost grew linearly as the failed VNs increased, and the fitted model for this particular case was $y = 1.8x - 0.33$. Thus, for the MNC-oriented problem, we expect the cost grows linearly as the number of failure increases for both heuristic and optimal solution.

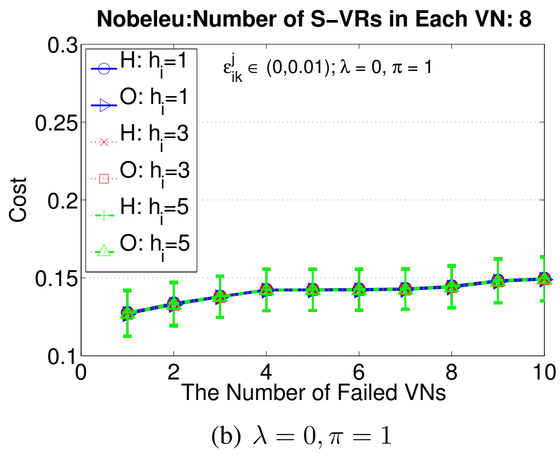
Fig. 21(b) shows an LB-oriented selection (i.e., $\lambda = 0, \pi = 1$). By increasing h_i values, the optimal cost was not reduced. In other words, when $h_i = 1$, the optimal solution largely distributed S-VRs on the substrate network to balance the load on the substrate node, if more than S-VRs are allowed to be activated on the same substrate node ($h_i > 1$), the aggregated bandwidth utilization may be greater than it is for $h_i = 1$. However, the heuristic algorithm found feasible solutions for more than 6 VN failures when h_i value was increased from 1 to 3 or 5, although the difference between heuristic cost and optimal cost increased. To understand how cost grows, we also did the regression analysis. Fig. 22(b) shows the fitness model the average heuristic cost (i.e., H: $h_i = 3$ in Fig. 21(b)), and we found that a power law model, $y = 0.056x^{0.0622}$, fit well to the original plot. Thus, for the LB-oriented problem, we expect both optimal and heuristic cost grew sub-linearly.

In addition, similar growth patterns were observed for *Nobel-EU* and *Germany50* topology, as presented in Fig. 23 and Fig. 24, respectively.

To learn how much gain obtained by applying the proposed optimization model, we next use the metric *cost overhead* to illustrate results for all scenarios we have summarized in Table 9.

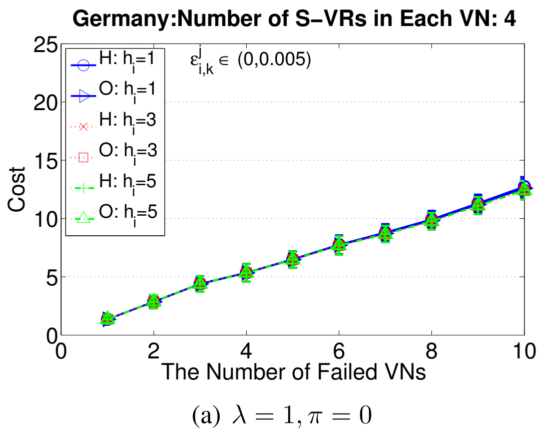


(a) $\lambda = 1, \pi = 0$

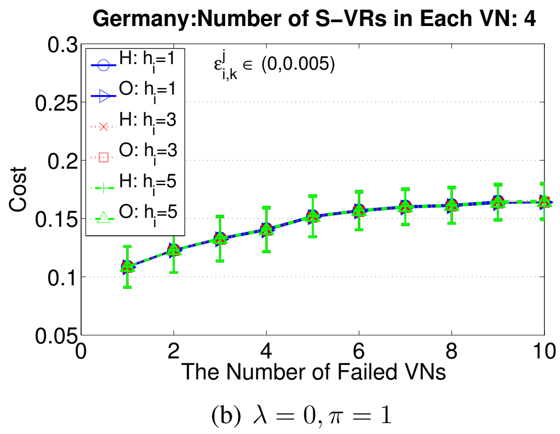


(b) $\lambda = 0, \pi = 1$

Figure 23: *Nobel-EU*: Independent VR Failures (Cost v.s. Failures)



(a) $\lambda = 1, \pi = 0$



(b) $\lambda = 0, \pi = 1$

Figure 24: *Germany50*: Independent VR Failures (Cost v.s. Failures)

5.4.2 Weight Parameters

We now discuss the impact of the weight parameters related to the connectivity cost function (5.15). To avoid interference with the LB objective, we set $\lambda = 1, \pi = 0$ for this discussion. Recall that the connectivity cost function has two tuples of weight parameters. $w_a = \langle w_{a1}, w_{a2} \rangle$ is associated with geographical distance to the failed VR and to the failed VR's original neighbors, respectively. $w_b = \langle w_{b1}, w_{b2} \rangle$ is associated with distance-relevant cost and RTT-relevant cost, respectively.

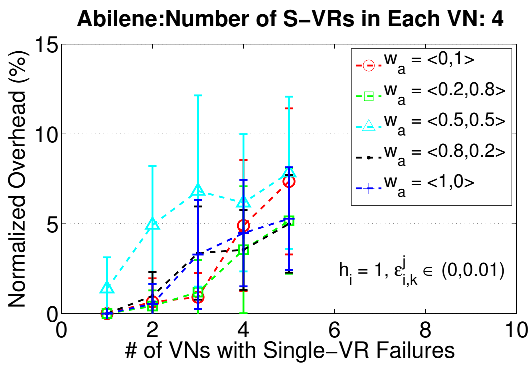
5.4.2.1 Impact of w_a

To study the impact of geographical location (w_a), we first set w_b = $\langle$1,0$\rangle$ (i.e., the RTT's impact was set to be zero.) Table 10 presents five different combinations of $\langle w_{a1}, w_{a2} \rangle$. For extreme cases, when a VR failure occurs, we may select an S-VR closest to the failed VR (w_a = $\langle$1,0$\rangle$); or select an S-VR that has a minimum average distance to the failed VR's neighbors in the VN (w_a = $\langle$0,1$\rangle$). The rest of the sets of w_a considered a joint impact of the two intuitive cases, as shown in Table 10.

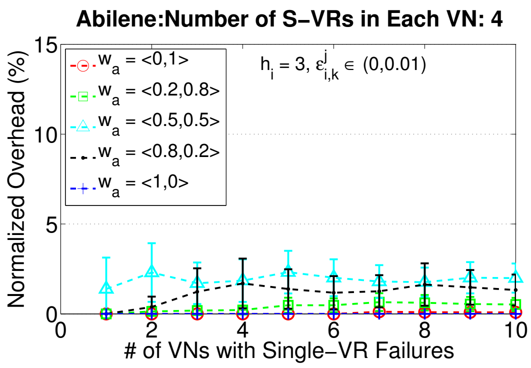
Table 10: w_a Values
($w_\theta = \langle 0, 1 \rangle$, $w_b = \langle 1, 0 \rangle$)

	1	2	3	4	5
w_{a1}	0	0.2	0.5	0.8	1
w_{a2}	1	0.8	0.5	0.2	0

We used the Abilene network with 10 overlaid VNs to illustrate the impact of w_a to the normalized overhead (as shown in Fig. 25). In order to study whether the w_a value will make different influences under various constraints on the substrate resource usage,



(a) Abilene: $h_i = 1$



(b) Abilene: $h_i = 3$

Figure 25: Impact of w_a (Abilene Network): 4 S-VRs per VN

we set $h_i = 1$ and 3, and the maximum virtual link bandwidth demand was set to 0.01 and 0.08 (i.e., $\epsilon_{ik}^j \in (0, 0.01)$ and $\epsilon_{ik}^j \in (0, 0.08)$), respectively.

When $h_i = 1$ and $\epsilon \in (0, 0.01)$, the S-VRs for restoring any two VNs are not allowed to be from the same substrate node. In general, given a specific amount of S-VRs reserved for each VN, when the number of affected VNs was increased, the normalized overhead of the heuristic over the optimal cost grew as the number of failures increased; in such a situation, there may not be feasible solutions for the heuristic (e.g., six or more failures as shown in Fig. 25(a)). By looking at the 95% CI of the average cost overhead for each w_a pair, under a feasible solution, we noticed a number of behaviors of the increment of the normalized cost overhead. When $w_a = \langle 0.5, 0.5 \rangle$, the geographical distance between an S-VR and the failed VR or the neighboring VRs are considered equally important. In this case, the 95% CI shows a higher average normalized overhead with bigger swings. For example, to restore 4 VNs, the cost overhead was 6.1939 ± 3.8162, whereas it was 3.5491 ± 3.5314 for $w_a = \langle 0.2, 0.8 \rangle$. On the other hand, compared with the extreme cases ($w_a = \langle 1, 0\rangle$ and $w_a = \langle 0, 1\rangle$), slightly increasing one value (i.e., w_{a1} or w_{a2}) from 0 reduced the incremental rate of the normalized overhead.

When $h_i = 3$ and $\epsilon \in (0, 0.01)$, each substrate node has the capability to support more S-VRs to be activated at the same time. As shown in Fig. 25(b), the normalized overhead has been reduced to be lower than 2.5%. For $w_a = \langle 0.5, 0.5\rangle$, the normalized overhead was still higher than other w_a tuples. On the other hand, the extreme cases showed zero average normalized overhead, and the associated 95% CI showed zero swing. This indicates that if substrate nodes have sufficient resources to support multiple S-VRs to be activated concurrently, assigning $w_{a1} = 0$ or $w_{a2} = 0$, the heuristics performs as well as the optimization model.

We also increased the bandwidth demand range for each VN ($\epsilon^j_{i,k} \in (0, 0.08)$), and the results did not showed much difference if we compared the patterns between different w_a combinations.

From the discussion above, we found that the selection of the S-VR relying on one type of distance (i.e. either to the failed VR or to the neighboring VRs) resulted in a lower normalized overhead.

5.4.2.2 Impact of w_b

Given the values we have discussed about the w_a tuple, we now discuss the composite impact of geographical location and the RTT associated with an S-VR using a variety of combinations of w_b tuples (see Table 11). Fig. 26 and Fig. 27 present the impact of w_b when $w_a = \langle 0.2, 0.8\rangle$ and $w_a = \langle 0.8, 0.2\rangle$, respectively. In addition, we set

Table 11: w_b Values
($w_\theta = \langle 0, 1\rangle$, $w_a = \langle 0.2, 0.8\rangle$ and $\langle 0.8, 0.2\rangle$)

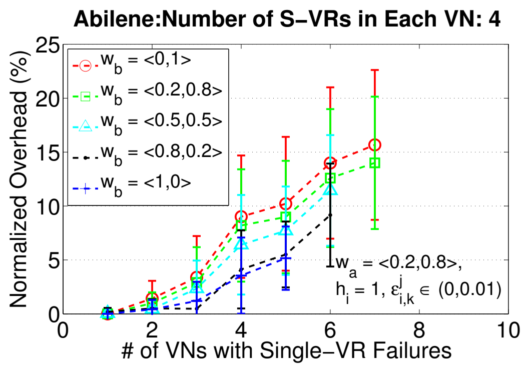
	1	2	3	4	5
w_{b1}	0	0.2	0.5	0.8	1
w_{b2}	1	0.8	0.5	0.2	0

$\epsilon^j_{i,k} \in (0, 0.01)$.

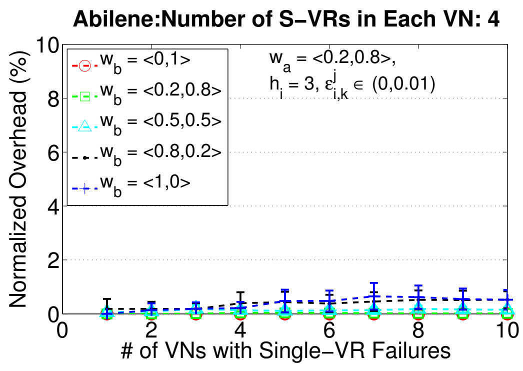
When $h_i = 1$, compared with $w_b = \langle 1, 0\rangle$, if the S-VR selection relied more on the RTT than the geographical distance factor (i.e. $0 \leq w_{b1} < 1$), feasible solutions were obtained for restoring more VNs. For example, for the blue line for $w_b = \langle 1, 0\rangle$ in Fig. 26, if we tuned w_{b2} from 0 to 1, we noticed that the number of VNs that can be restored increased (from 5 to 7 VNs). However, the normalized overhead was increased from $5.16{\pm}2.94$ to $15.66{\pm}6.95$. Thus, there was a trade off between getting more feasible solutions and reducing normalized cost overhead. For $w_a = \langle 0.8, 0.2\rangle$, Fig. 27 shows the trade off in the same pattern.

When $h_i = 3$, the normalized overhead was reduced as w_b2 was increased from 0 to 1. For example, at 10 VNs in Fig. 26(b), the 95% CI of the average normalized overhead was 0.522 ± 0.367 for $w_b = \langle 1, 0\rangle$, whereas it was 0.018 ± 0.025 for $w_b = \langle 0.2, 0.8\rangle$. Compared with Fig. 27(b) with $w_a = \langle 0.8, 0.2\rangle$, for all w_b tuples given that the $w_a = \langle 0.2, 0.8\rangle$, Fig. 26(b) shows that the average normalized overhead was less than 1% at 10 VNs failures.

In summary, if all VRs (including S-VRs) have sufficient virtual interfaces and all virtual interfaces are homogeneous, the MNC goal depends on the Type-II cost. When



(a) Abilene: $h_i = 1$



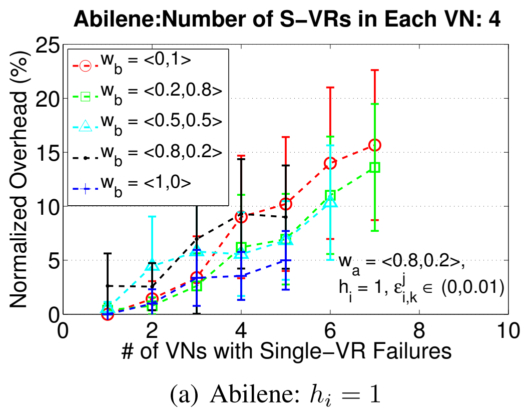
(b) Abilene: $h_i = 3$

Figure 26: Impact of w_b (Abilene Network): 4 S-VRs per VN
($w_a = \langle 0.2, 0.8\rangle$, $\epsilon^j_{i,k} \in (0, 0.01)$)

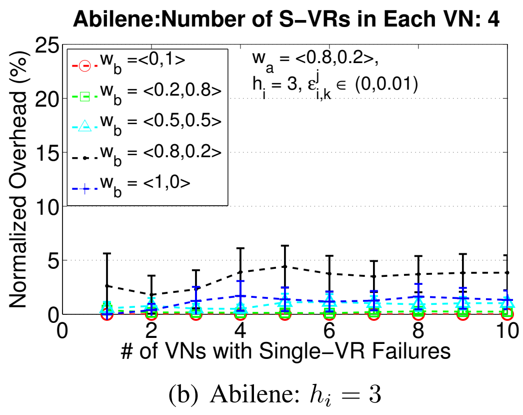
considering the geographical location of a candidate S-VR, in order to reduce the normalized overhead of the heuristic over the optimization model, choosing the S-VR as close as to the failed VR's neighbors could be a good strategy. If a real-time RTT is available from the SNP, with few VNs affected, distance can be considered as a main selection criterion. However, if a large number of VNs are affected, although the normalized overhead will increase, the heuristic approach can restore more VNs.

5.4.3 Independent Virtual Router Failures

Although an independent VR failure does not affect other VNs, there is a possibility that multiple VNs concurrently have independent VR failures. As listed in Table 9, we are interested in how the normalized overhead is influenced by a variety of factors. Consider one or more VNs with independent VR failures, we next present the normalized overhead of the heuristic cost over the optimal cost, when the substrate networks were *Abilene*, *Nobel-EU*, and *Germany50* topologies (Fig. 20), respectively. In the rest of



(a) Abilene: $h_i = 1$



(b) Abilene: $h_i = 3$

Figure 27: Impact of w_b (Abilene Network): 4 S-VRs per VN
($w_a = \langle 0.8, 0.2\rangle$, $\epsilon^j_{i,k} \in (0, 0.01)$)

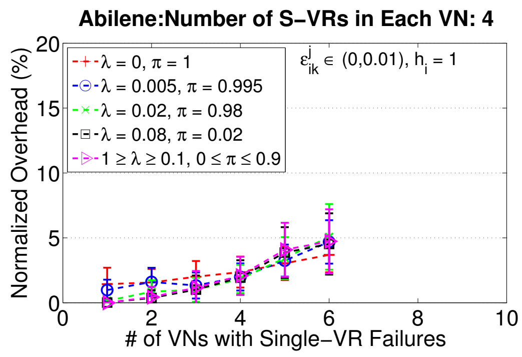
discussion, we assigned $w_a = \langle 0.2, 0.8\rangle$ and $w_b = \langle 0.2, 0.8\rangle$.

5.4.3.1 Abilene Topology

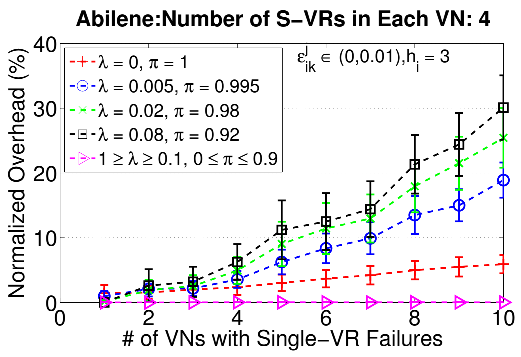
Abilene Topology has only 12 nodes. According to the first assumption we made in Chapter 3: any two VRs in a VN are not provisioned by a common substrate node, the maximum VN we created over the *Abilene* network consisted of 6 VRs, so that each VN may have at most 6 S-VRs.

Recall the inequality relation between π and ρ λ we introduced in the heuristic, we set $\rho = 10$, so that the impact of $t \in (0, 1)$ will not be eliminated. Given that $\lambda + \pi = 1$, so if $0.1 \leq \lambda \leq 1$ or $0 \leq \pi \leq 0.9$, the overall goal is to minimize the network cost, otherwise the LB should be considered along with the MNC objective.

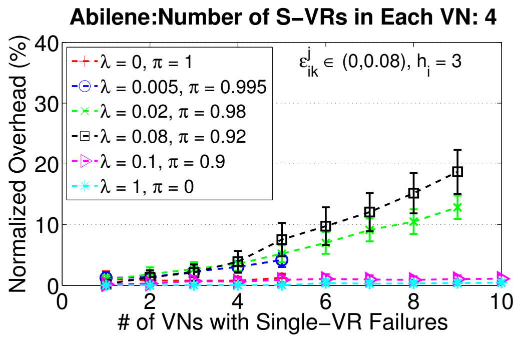
To find proper combinations of λ and π, we increased λ in three different steps (1) $\lambda \in \{0, 0.005\}$ (2) from 0.02 to 0.08 with step 0.02; (3) from 0.1 to 1 with step 0.1, and we can compute the corresponding value for π. The first two ranges indicate



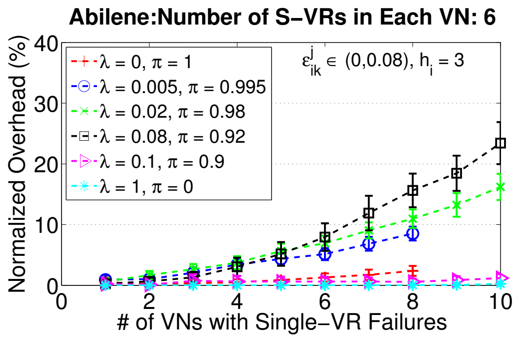
(a) $h_i = 1$, 4 S-VRs/VN and L-BW



(b) $h_i = 3$, 4 S-VRs/VN and L-BW



(c) $h_i = 3$, 4 S-VRs/VN and H-BW

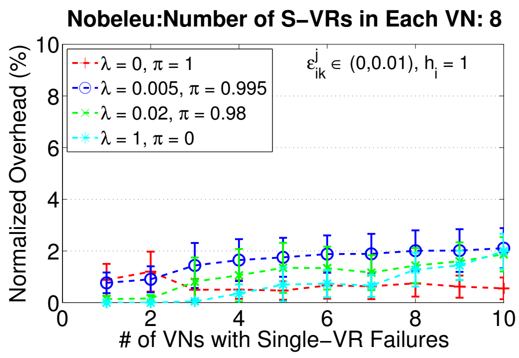


(d) $h_i = 3$, 6 S-VRs/VN and H-BW

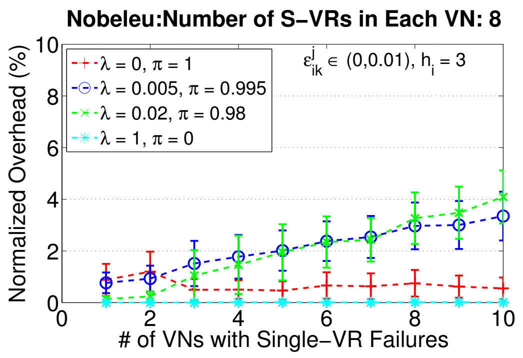
Figure 28: *Abilene*: Independent VR Failures (L-BW: Low Bandwidth Request; H-BW: High Bandwidth Request)

an LB-based selection, and the third one indicates an MNC dominated selection. As we discussed earlier in the scenario design, for each substrate network, four different factors might affect the S-VR selection by the heuristic and the optimization model, and they are the number of VR failures, the maximum number of S-VRs can be activated from a common substrate node (h_i), the bandwidth demand by the virtual links and the size of S-VR pool reserved for each VN. We next use Fig. 28 to illustrate the impacts under these factors.

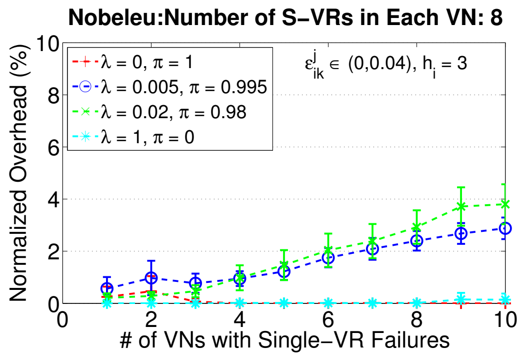
Two general observations can be identified by comparing Fig. 28(b) – Fig. 28(d)



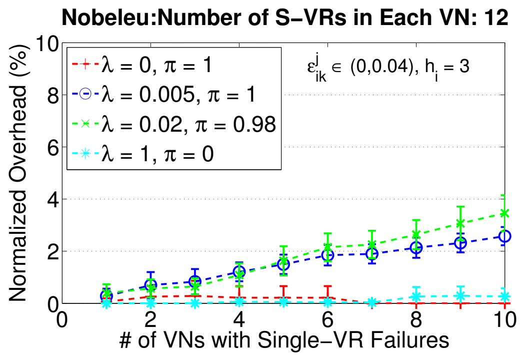
(a) $h_i = 1$, 8 S-VRs/VN and L-BW



(b) $h_i = 3$, 8 S-VRs/VN and L-BW



(c) $h_i = 3$, 8 S-VRs/VN and H-BW



(d) $h_i = 3$, 12 S-VRs/VN and H-BW

Figure 29: *Nobel-EU*: Independent VR Failures (L-BW: Low Bandwidth Request; H-BW: High Bandwidth Request)

with Fig. 28(a). First, when the substrate nodes had limited resources to support multiple collocated S-VR candidates to be activated concurrently (e.g., $h_i = 1$), the heuristic could not find feasible solutions when more VNs fails, no matter whether the overall goal was MNC or LB. As presented in Fig. 28(a), the heuristic did not find feasible solutions for more than 6 failures. Consider the feasible solutions, the 95% CI of the average normalized overhead under various weight tuples mostly overlapped with each other; they were overall less than 5%. Thus, the impact of λ & π is less significant. However, if the capability of activating multiple collocated S-VRs is increased (e.g., $h_i = 3$), to restore the same number of VNs, the normalized overhead did not change much when the dominant objective was MNC (i.e., $\lambda \in [0.1, 1]$); whereas the overhead increased as λ was increased from 0 to 0.08. On the other hand, under the LB dominant selection (i.e., $\lambda < 0.1$), as more VNs suffered failures, the incremental ratio of normalized overhead was getting bigger when λ increased from 0 to 0.08. For example, as the number of failure grew from 1 to 10, the 95% CI of the average overhead was increased from $(1.43 \pm 1.27)\%$ to $(5.92 \pm 1.41)\%$ for $\lambda = 0$, and it was from $(0.97 \pm 0.80)\%$ to $(18.90 \pm 2.71)\%$ for $\lambda = 0.005$.

For $h_i = 3$, we next discuss the cases where the VNs had higher bandwidth demand (i.e., $\epsilon_{ik}^{j} \in (0, 0.08)$). Compared with Fig. 28(b), Fig. 28(c) shows that when the selection was purely dependent on the LB (i.e., $\lambda = 0$), the heuristic did not find feasible solution for more than five failures (see the red plot). If we raised λ from 0 to 0.005, the feasible solutions were not improved much, and the overhead was increased. However, if we further increased λ to 0.02, although the overhead grew more, the heuristic algorithm obtain feasible solutions for up to 9 failures. When $\lambda = 0.08$, the number of feasible solutions was not improved, but the cost overhead increased. Hence, by tuning the value of λ, the heuristic approach was able to restore more VNs, with the trade off by increasing the cost overhead. On the other hand, when the selection was purely dependent on the MNC, the heuristic performed as well as the optimization model, and the average cost overhead was less than 0.5% at 10 failures. Compared to Fig. 28(c), Fig. 28(d), Under the same bandwidth request, provisioning more S-VRs helped the heuristic algorithm to restore more VNs. For example, for a purely LB dependent selection ($\lambda = 0, \pi = 1$), the maximum number of failed VNs can be restored was increased from 5 to 8. Therefore, if virtual links have relative high bandwidth demand, to run the heuristic algorithm, the SNP can consider either allocating more S-VRs to each VN without bringing up the cost overhead much, or assigning a proper value to λ instead of 0.

5.4.3.2 Nobel-EU Topology

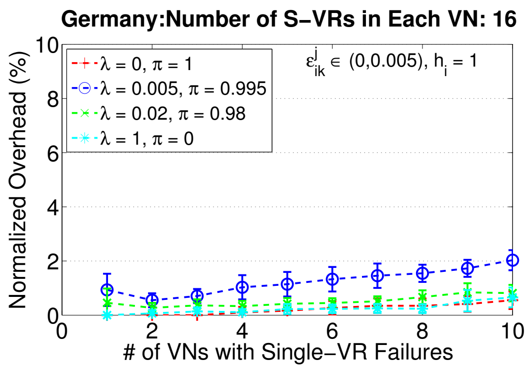
For *Nobel-EU* topology, we varied the number of S-VRs for each scenario from 8 to 12, and we present the result for 8 and 12 S-VRs reserved per VN, respectively, which not only allowed customers to request VNs with a maximum 16 VRs, but also provided a proper amount of S-VR candidates for each VN. This way, based on our discussion on the feasibility of a heuristic algorithm, we can always find both a feasible heuristic and an optimal S-VR selection.

Fig. 29 presents the results in the same way as we did for *Abilene* topology. When $h_i = 1$ with low bandwidth requests (see Fig. 29(a)), for the same amount of failures, we do not see significant difference on the average normalized overhead, and the maximum overhead was about than 2%. As we expected, the normalized overhead went up as more failed VNs were to be restored. For example, for $\lambda = 0, \pi = 1$, the normalized overhead was about 0% for 1 failure, and it was $(1.99 \pm 0.68)\%$ for 10 failures. If the substrate nodes had more resources (i.e., $h_i = 3$), we found that the heuristic algorithm showed better performance when the goal was pure MNC ($\lambda = 1, \pi = 0$) or LB ($\lambda = 0, \pi = 1$) than the other composited objectives, and the cost overhead for $\lambda = 1$ was about 0.01% even at 10 failures. If the VN bandwidth demand was higher, the maximum bandwidth utilization on the substrate network increased from $(15{\pm}1)\%$ to $(57{\pm}5)\%$ for restoring 10 VNs, but the pattern that the normalized overhead grows did not change. If we increased the reserved S-VRs for each VN from 8 to 12, cost overhead for the cases with $\lambda = 0.005$ and $\lambda = 0.02$ was reduced. Hence, for the *Nobel-EU* substrate network, if the VN size is not bigger than 16 VRs, reserving 8 S-VRs for each VN is sufficient to restore at least 10 VNs.

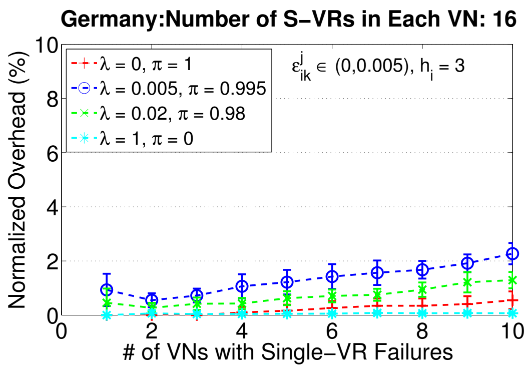
5.4.3.3 Germany50 Topology

For the *Germany50* topology, the number of S-VRs for each VN was varied from 4 to 22. In particular, we report the 95% CI of the normalized overhead in Fig. 30 and Fig. 31, where the virtual link bandwidth request was $(0 - 0.5)\%$. In particular, Fig. 30 presents the composite impact of MNC and LB objectives, and we observed the same pattern as we discussed for *Abilene* topology and *Nobel-EU* topology.

Fig. 31 showed the impact of the number of S-VRs reserved for each VN and h_i, respectively. Specifically, we notice that reserving more S-VRs improved the heuristic's performance for the LB based selection (see Fig. 31(a)), and the improvement was not obvious for the MNC based selection. On the other hand, we found the heuristic algorithm's performance was improved by increasing h_i values (see Fig. 31(b)). This is reasonable, as MNC objective is based on computing minimum distance or RTT. Hence, if multiple failed VRs are in the same area, then a substrate node who has the capability to support more than one S-VR to be activated concurrently will produce less aggregated network cost, rather than selecting S-VRs from other substrate node with longer distance



(a) $h_i = 1$, 16 S-VRs/VN



(b) $h_i = 3$, 16 S-VRs/VN

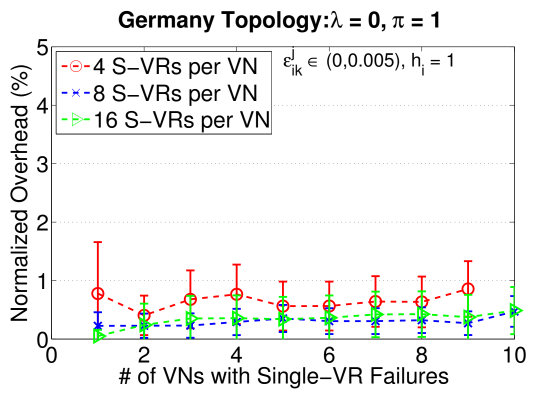
Figure 30: *Germany50*: Independent VR Failures with L-BW Requests (Composite Impact of MNC and LB Objectives)

or latency. However, we also notice that increasing h_i would not help much to reduce the overhead for the LB based selection.

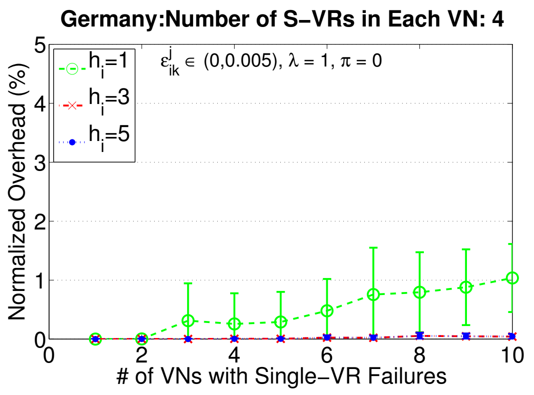
5.4.4 Dependent Virtual Router Failure

In a VNE, a substrate node failure may have critical impacts to the VNs. In particular, if a substrate node that hosts VRs belonging to n VNs fails, this failure immediately affects these n VNs at the same time. In the worst case, one substrate node failure will cause all overlaid VNs to fail. Hence, it is important to restore those VNs by quickly selecting proper S-VRs. We use the term dependent VR failure to denote the VRs failures caused by the substrate node failure. In this section, we use the *Nobel-EU* topology to discuss the S-VR selection for restoring VNs from dependent VR failures in the worst case (i.e., all 10 VNs failed).

Given the condition that $h_i = 3$ and $\epsilon^j_{ik} \in (0, 0.04)$ (i.e., high bandwidth request), Table 12 presents the 95% CI of the average network cost calculated by the MILP model



(a) $h_i = 3$



(b) 4 S-VRs/VN

Figure 31: *Germany50*: Independent VR Failures with L-BW Requests (Impact of the size of S-VR pool and h_i)

and the heuristic algorithm, respectively; as well as the 95% CI of the average normalized overhead for various scenarios where we varied the size of S-VR set and the $\langle \lambda, \pi \rangle$ values.

When $\lambda = 0$ or 0.02, the S-VR selection was impacted more by the LB objective. Specifically, $\lambda = 0$, as more S-VRs are provided to the VNs, the maximum bandwidth utilization on the substrate nodes slightly increased.

When $\lambda = 1$, the S-VR selection was purely dependent on the MNC objective. Thus, as more S-VRs were reserved, the cost was reduced. However, with MNC as the dominant criterion, the maximum bandwidth utilization of the substrate nodes could be higher than the LB as the dominant criterion. For instance, we found that $t = 0.854$ for $\lambda = 1$, whereas $t = 0.810$ for $\lambda = 0$ and $t = 0.812$ for $\lambda = 0.02$. For the independent VR failure scenario with 10 VN failures, we only notice about 1% increment on the maximum bandwidth utilization t.

For the *Germany50* topology, we observed similar pattern for the network cost and corresponding normalized overhead.

Table 12: Dependent VR Failures with *Nobel-EU* Topology ($h_i = 3$)
Worst Case: One Substrate Node Failure Affected all 10 VNs

Size of the S-VR Set	$\langle \lambda, \pi \rangle$	High vlink Bandwidth Request		
		95% CI of Average Network Cost		95% CI of Average Normalized Overhead
		MILP	Heuristic	
8	$\langle 0.00, 1.00 \rangle$	0.570 ± 0.047	0.570 ± 0.047	0
	$\langle 0.02, 0.98 \rangle$	0.929 ± 0.065	0.961 ± 0.068	$(3.390 \pm 0.501)\%$
	$\langle 1.00, 0.00 \rangle$	18.271 ± 1.519	18.278 ± 1.518	$(0.037 \pm 0.033)\%$
10	$\langle 0.00, 1.00 \rangle$	0.577 ± 0.046	0.577 ± 0.046	$(0.028 \pm 0.056)\%$
	$\langle 0.02, 0.98 \rangle$	0.919 ± 0.065	0.949 ± 0.067	$(3.324 \pm 0.713)\%$
	$\langle 1.00, 0.00 \rangle$	17.478 ± 1.460	17.486 ± 1.461	$(0.046 \pm 0.040)\%$
12	$\langle 0.00, 1.00 \rangle$	0.579 ± 0.048	0.579 ± 0.047	$(0.028 \pm 0.056)\%$
	$\langle 0.02, 0.98 \rangle$	0.913 ± 0.067	0.943 ± 0.069	$(3.261 \pm 0.533)\%$
	$\langle 1.00, 0.00 \rangle$	17.178 ± 1.451	17.208 ± 1.474	$(0.125 \pm 0.149)\%$

5.4.5 Computational Time

In the above case studies, the computational time for solving the MILP problem and the heuristic was extremely fast (less than 0.01 second in general). In order to study a more complicated case and compare the computation time of the heuristic and the optimal selection, we provisioned 50 VNs over the *Nobel-EU* topology, and allocated 8 S-VRs for each VN as a more strict S-VR provisioning, compared with the scenarios where 12 S-VRs provided for each VN over *Nobel-EU*.

Table 13 presented the computational time of running the MILP and the heuristic, along with the corresponding cost. When the dominant objective was LB (i.e., $\lambda = 0$), we see that both MILP and the heuristic produced very close network cost, and the computational time for both the heuristic and optimal solution was in the order of hundred milliseconds. Recall the computational complexity, when the heuristic considers the LB objective, the computational time increases as the number of affected VNs grows, as well as the number of substrate nodes is larger. On the other hand, when the MNC was dominant objective, we found that it only took 0.03 second to solve the problem, which was 15.7% of the computation time for running MILP model, and the heuristic only produced 2.5% cost overhead.

Table 13: Computation Time: *Nobel-EU* with 50 VNs
(8 S-VRs per VN; $h_i = 3$)

Dominant Objective	Computational Time (sec)		Network Cost	
	MILP	Heuristic	MILP	Heuristic
Load Balancing	0.13	0.23	0.600	0.602
Minimize the Cost	0.19	0.03	144.07	147.62

5.4.6 Observations

We summarize below the key observations:

1. In regard to the impact of the weight parameters associated with the subcomponents in the connectivity cost function (i.e., (5.15)):
    - If we consider the S-VR's geographical location as the only factor to the selection, the heuristic presented better performance when the selection mainly relied on one type of distance;
    - If we consider a composite impact of both the geographical distance and the RTT from an S-VR, there is a tradeoff between obtaining more feasible solutions and reducing the overhead of applying the heuristic algorithm.
2. For a LB dominant goal, the network cost increased sub-linearly, whereas it increased linearly for the MNC as the dominant goal.

3. Given the assumption that any two VRs within a VN do not share resources from the same substrate node, a smaller substrate network has limitations on provisioning proper amount of S-VRs for each VN, and this restriction may cause high overhead for the heuristic, or may not even get feasible solution to restore every affected VN, especially in cases of a large number of concurrent VR failures occur in the VNE. However, it is not necessary to provide more than sufficient S-VRs for each VN. Based on our experiments, we found that reserving VRs from about one third of the substrate nodes for each VN was sufficient.

4. Two types of VR failures were discussed. For independent VR failures, the failed VRs are more likely from different locations. With low bandwidth request, the heuristic algorithm showed good performance with purely LB oriented or MNC oriented selection (i.e., the overhead was small). If the bandwidth demand by the VNs was high, slightly tuning up the value of λ improved the heuristic's performance, where we observed more feasible solutions, with a trade off on the increment of the cost overhead; furthermore, if the SNP provided more S-VRs for the VNs, then more feasible solutions can be obtained by the heuristic and the cost overhead can be further reduced. With dependent VR failures, an LB-oriented selection helped to reduce the maximum bandwidth utilization on the substrate nodes.

5. The resource utilization on a substrate node is another important factor to the S-VR selection, and it is represented by the h_i value in our proposed model. To restore the same amount of VNs, if substrate nodes have more resources (e.g., less CPU usage or h_i is higher), the network cost can be reduced, and the heuristic showed

better performance, especially for the MNC based selection.

5.5 Summary

Dynamic network management becomes flexible and efficient in a virtualized network environment, especially with a centralized control system that has the capability to allocate substrate resources during the VN creation or reconfiguration. In this paper, we presented an optimization model and a heuristic to select optimal S-VRs by considering a variety of criteria to restore multiple VNs from VR failures that could be either independent or dependent failures.

The proposed optimization model considers both minimum network cost (MNC) and balancing of bandwidth utilization on substrate nodes, and the MNC consists of two cost functions from the operation and connectivity aspects, respectively. Thus, the model itself can be easily extended with more weighted criteria if needed. In this work, we considered the virtual interfaces to be homogenous, and studied the impact to the S-VR selection under a number of factors on three substrate topologies, such as the number of concurrent virtual router failures, the capability of activating multiple collocated S-VRs from a substrate node (h_i), the number of S-VRs provided for each VN and the bandwidth request from S-VRs. Our results showed that substrate nodes hosting S-VRs with sufficient resources or VNs with sufficient S-VRs helped to reduce the network cost as well as the overhead of the heuristic. For the dependent VR failures, a LB based selection shows a better load balance on the substrate network. On the other hand, the computation time of running both the optimization model and the heuristic was shown to be in the order of millisecond or seconds, and the heuristic algorithm was efficient for the MNC-oriented selection. Thus, our proposed model can be applied in a real-time manner.

CHAPTER 6

PROTOTYPE DESIGN AND IMPLEMENTATION ON GENI

GENI testbed is a wide area distributed testbed that support heterogeneous resource provisioning, and it allows experimenters to deploy and evaluate novel applications or protocols with arbitrary types of topologies. From the network virtualization perspective, it supports both layer-2 and layer-3 virtualization, so we design and implement a prototype of DRS to achieve survivable virtual network restoration against node failures.

6.1 Framework Design

Virtual network management can be automated as a life-cycle with autonomic computing. Fig. 32 shows the conceptual design for the autonomic virtual network management that automates three phases: resource provisioning, monitoring and reconfiguration. In particular, the resource provisioning mainly runs the virtual network embedding algorithm to allocate substrate resources and create virtual networks. For the monitoring phase, it is responsible for failure detection in both virtual networks and the underlying substrate network, and also report real-time performance of the networks and the resource utilization on the substrate network. The reconfiguration phase is triggered by two conditions. The first is a failure is detected, then network is dynamically restored by reconfiguration. The second case is for load balancing purpose, where the substrate network resources are to be reallocated when a new virtual network request comes.

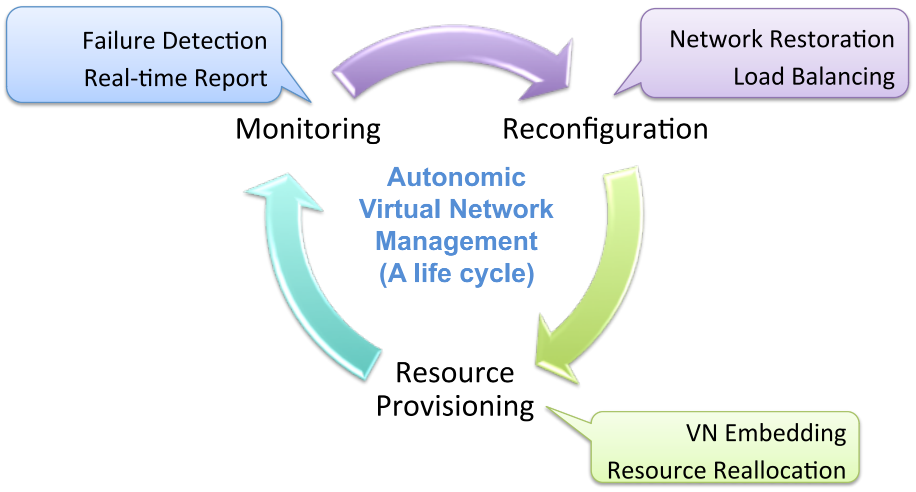


Figure 32: A Life-Cycle for Autonomic Virtual Network Management

In this section, we present the holistic design of a software-defined resilient VNE with the dynamic reconfiguration scheme against node failures.

6.1.1 Resilient Virtualized Networking Environment

Fig. 33 presents the architectural design of a resilient VNE, comprising the substrate network, the virtual network plane and the virtual network manager (VNM). Specifically, an SNP owns the core physical network infrastructure (i.e., the substrate network) and it also implements the VNM. According to the virtual network requests by the SPs, the SNP creates and configures the core VNs on the virtual network plane. Note that the core substrate network does not contain the edge routers connecting to the end host's domain. To support resiliency for each VN, in addition to the ordinary virtual topology, the SNP also reserves a set of S-VRs and logical links via VNM. These S-VRs are identical with other active VRs in terms of system configuration and software. For each VN that

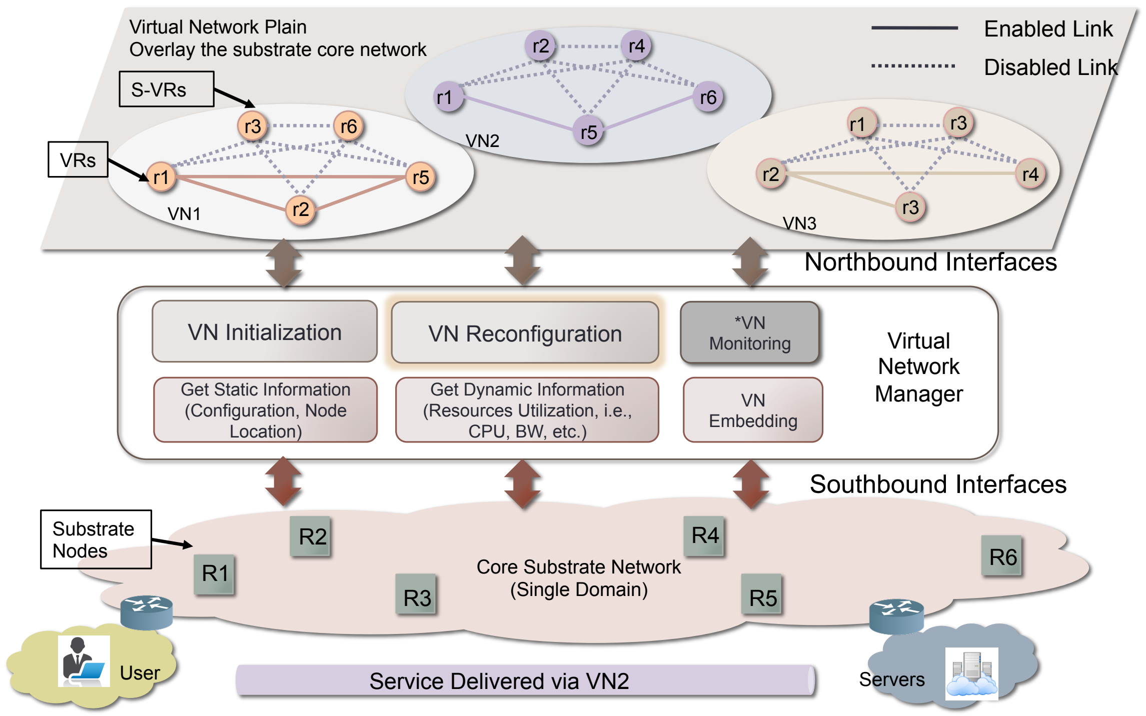


Figure 33: Dynamic Reconfiguration Scheme for Virtual Networks

is ready for delivering services, the S-VRs should be potentially capable to connect every other active VR easily. For instance, as presented with dotted lines in Fig. 33, logical links can be reserved between an S-VR and other VRs, so that if a VR fails, the selected S-VR can be easily added into the existing topology. On the other hand, logical links may be created between any pair of S-VRs, in case a secondary S-VR needs to be brought in and connected to the S-VR that is already connected.

We design a middleware called the Virtual Network Manager (VNM) operated by the SNP, and it logically resides between the SN and the virtual network plane. Hence, the VNM has both northbound interfaces and southbound interfaces. The northbound interfaces communicate with the VN plane to perform virtual network management including creation, configuration, reconfiguration, and monitoring. The southbound interfaces communicate with the substrate network to monitor and collect both static information and the real-time statistics about the substrate resources.

6.1.2 Virtual Network Manager

The VNM abstracts the virtual network control plane into a logically centralized system with a software-defined mechanism and it is designed with functional modules for autonomic management. In other words, to avoid the single-point failure, the SNP can implement a dual-VNM control, where the two VNM are synchronized with each other about the information collected from both the SN and the VN plane. Here, we introduce the modules communicating with the VN plane and the SN through the VNM's northbound interfaces and southbound interfaces, respectively.

6.1.2.1 Southbound Interfaces

There are three modules to communicate with the substrate network.

- Get static information: The static information of the SN is about the basic configuration on the physical network resources (e.g., the substrate node's location, hardware configuration, link capacity, topology, etc.), and such configuration can be used for virtual network embedding with regional considertaiton. The VNM periodically obtain these stationary information from the SN (e.g., every one month), or the SNP can notify the VNM with any specific updates.

- Get dynamic information: The dynamic information refers the real-time residual resource (e.g., CPU, bandwidth or interface statistics, etc) utilization on the substrate network, and they are significant factors to the embedding and restoration processes. For the DRS with the optical S-VR selection mode, the residual resource utilization is one of the constraints. The VNM periodically collects dynamics on the substrate network resources.

- VN Embedding: Having once learned about substrate network resources, the VNM runs a proper virtual network embedding algorithm to map a new VN to the substrate network, and stores the embedded result as a profile for this new VN. To support our proposed DRS, the VNM should run a survivable virtual network embedding algorithm such as [114] that allocate redundant nodes for each VN.

6.1.2.2 Northbound Interfaces

Towards the northbound interfaces, three modules are running in an autonomic manner, respectively.

- VN Initialization: Once the VN Embedding module generates a profile for each VN that has been allocated resources, the VNM will initialize a new VN based on its profile. The initialization includes booting up the virtual nodes with interfaces configured and setting up the virtual links.

- VN Reconfiguration: This module supports self-healing for the VNs and load balancing on the substrate network. Our goal is to achieve dynamic VN restoration after a VR failure is detected by the VN Monitor, and the optimal S-VR selection algorithm proposed in Chapter 5 will be triggered to select an optimal S-VR to replace the failed VR for each affected VN.

- VN Monitor: This module monitors the VN status, such as network congestions, virtual link failures, or virtual node failures. Note that monitoring is out of the scope of this dissertation.

6.2 Proof of Concept

We choose to use the GENI testbed [21] to implement a proof of concept for our proposed software-defined resilient VNE.

6.2.1 An Overview about GENI

GENI is a distributed experimental testbed for networking innovation research, and it provides computing and networking resources for experimenters to built isolated experiments. For example, InstaGENI [19] is a flavor of a standard GENI rack (i.e., aggregate) containing a consistent set of software and hardware including reservable computers, an OpenFlow data plane switch, and virtual machines (VMs). For the network resources, GENI supports tunneling mechanisms and VLAN techniques to create virtual links carry isolated traffic, and it also supports high-speed data plane virtual links using AL2S (Advanced Layer-2 Service) provided by Internet2.

GENI resources are specified in a standard XML document called *Resource Specification (RSpec)*. There are three types of RSpec files: *advertisement RSpec*, *request*

RSpec, and *manifest RSpec* [21]. In specific, the advertisement RSpec of an aggregate contains the information about its resources, and the experimenters can sent a request to obtain this RSpec to check the detailed information about a GENI aggregate. A request RSpec contains the resource specification desired by an experimenter for a particular slice, such as aggregates that provide VMs, configuration information, etc. Once a request RSpec is submitted, GENI aggregates allocate the requested resources to the experimenter's corresponding slice, and the experimenter has full control of the reserved VMs to build experiments in his/her own slice. There is a command-line tool called `omni` to assist experimenters to check GENI resources or reserve resources from GENI testbed. More details about GENI RSpecs can be found in Appendix A.

Given the flexibility and the tools provided by GENI, we are able to implement a prototype of the software-defined resilient VNE. Table 14 shows the relation between the proposed VNE and the GENI context. In particular, when creating a VN is created, the SP only sees the requested topology. However, from the VNM's perspective, the actual VN also includes the potential virtual links connecting S-VRs to active VRs and the ones between the S-VRs. Since the virtual links from an S-VR are already created, it is easy to add them to the existing active virtual topology by enabling the associated virtual interfaces. Regarding the middleware VNM, by taking the advantage of the GENI API, we can implement it as a third-party tool that is able to generate request RSpec and send it to the GENI testbed to request resources on behalf of the SPs. In this way, virtual network embedding and reconfiguration can be implemented within the VNM.

Table 14: VNE under GENI Context

GENI Context	Proposed VNE
GENI testbed	Substrate Networks
Slices	Virtual Networks
VMs installed with routing software	VRs or S-VRs
GRE/EGRE tunnels or stitched links or LANs	Virtual Links

6.2.2 Autonomic Virtual Network Initilization

The initialization of a survivable VN with S-VRs includes two parts under GENI context. First is to automate the generation of a request RSpec, and the second is to automate the software router configuration process.

6.2.2.1 RSpec Generation

As RSpecs are important for VN creation and configuration, generating RSpecs in an automated fashion is desirable. Since GENI aggregates automatically allocate resources to the requested VN based on the request RSpec, to achieve autonomic VN creation, the first challenge is to generate the request RSpec for complex virtual topologies. Fortunately, GENI provides `geni-lib` [4], a python library for building scripted tools.

We developed a scripted tool relying on `geni-lib` to create and configure arbitrary virtual topologies and generate the request RSpec automatically for experiments, to request resources from InstaGENI aggregates. This scripted tool is now distributed with `geni-lib` as the `ScaleUp` tool, which aims to provide following flexibilities to experimenters through a single configuration file, and its structure is presented in Fig. 34.

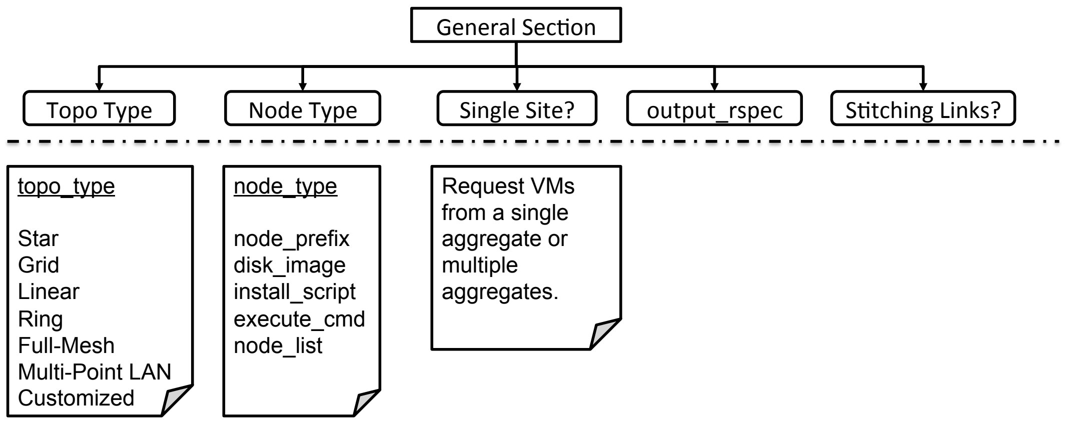


Figure 34: Structure of the Configuration File for `ScaleUp` Tool

- Arbitrary topology creation: the `ScaleUp` tool allow users to create VN in arbitrary topologies in various size consisting of Xen-VMs on InstaGENI aggregates. For standard topology types (i.e., ring, full-mash, star, grid and multi-point LAN), users just needs to specify the parameters associated with a particular topology class. For instance, to create an $m \times n$ grid topology, the dimension information (i.e. m and n) should be provided. For customized topology, users will have to provide the details about the topology, in the format of `[(a,b), (b,c), (b,d),...]`, where `(a,b)` represents a bidirectional virtual link between node `a` and `b`.

- Flexible node configuration: According to the pattern of the segments defining nodes in a request RSpec, we would like to provide an elastic configuration that allows any combination of the features that the requested nodes are expected to have. A requested VM can be differentiated from other VMs by a composite of characteristics, such as node category, OS image, post-boot installation procedures (i.e., downloading scripts and executing commands) and the list of nodes that belong to this category. In our case, we only need to configure nodes into two categories (i.e. end hosts and VRs), where the end hosts require a standard Ubuntu OS with `iperf` installed and the VRs require a customized image that has `XORP` installed already.

```
[router]
node_prefix=rt
disk_image=<URL to the customized OS image with XORP installed>
install_script=<URL to the scripts that creates OSPF configuration
and starts XORP>
execute_cmd=<the command to start XORP>
node_list=1,2,3


[host]
node_prefix=host
disk_image=<URL to the standard Ubuntu OS image>
install_script=<URL to the scripts to install iperf>
execute_cmd=<the command to install iperf>
node_list=4,5
```

Figure 35: Configuration for VRs and End-Host Nodes with the `ScaleUp` Tool

- Auto-IP Assignment: `ScaleUp` automatically assigns IP addresses to the interfaces for each Xen-VM once the topology is defined, and it provides efficiency for configuring VMs in a large-scale complex topology.

- Link Types Auto-Detection: Under GENI context, the link types are determined by the VM types and the source of the VMs. For example, if two VMs are requested from different aggregates, then the virtual link between these two VMs should be either the GRE/EGRE tunnel or the stitched link configured by VLAN techniques, otherwise, the virtual link is just a regular LAN type within a single aggregate.

Hence, we use a combination of three sections defined in the configuration file for the `ScaleUp` tool, which are `am_nodes` and `single_am` and `use_stitching`. For the node setup presented in Fig. 35, if we would like to have the nodes distributed across multiple aggregate, and create EGRE tunnels between them, then we will specify these three sections as shown in Fig. 36.

```
single_am=no
use_stitching=no
[am_nodes]
ig-illinois:2,3,5
ig-wisconsin:1,4
```

Figure 36: Determine Virtual Link Type with the `ScaleUp` Tool: An Example

- RSpec Generation: Once a user has entered all the request to the configuration file to run the `ScaleUp` tool, an RSpec file will be generated based on these speficialed features for the user's *slice*. Fig. 37 shows a sample request RSpec generated by `scaleup` according to the configuration we presented above.

6.2.2.2 VR Configuration

The `ScaleUp` tool is designed to be a generalized tool for to create request RSpec file for efficiently setting up experiments on the GENI testbed. Now, we present how to automate the configuration procedure for VRs.

The VR configuration procedure consists of routing software installation and routing protocol configuration. We first tried to install `XORP v1.8` on a standard Ubuntu 12.04 OS image, and it took us more than 40 minutes to complete the installation and pass all the tests. Hence, it is not efficient to install `XORP` at the post-boot stage. GENI

```
<node client_id="rt-2" exclusive="false"
component_manager_id="urn:publicid:IDN+instageni.illinois.edu
+authority+cm">
    <sliver_type name="emulab-xen">
      <disk_image name="https://www.instageni.clemson.edu/
image_metadata.php?uuid=5417bcef-224e-11e4-aa5a-000000000000"/>
      <ns0:xen xmlns:ns0="http://www.protogeni.net/resources/rspec/
ext/emulab/1" cores="1" ram="256" disk="8"/>
    </sliver_type>
    <interface client_id="rt-2:if1">
      <ip address="192.168.1.2" netmask="255.255.255.0" type="ipv4"/>
    </interface>
    …………
    <services>
      <install url="http://www.gpolab.bbn.com/exp/sysexpr/
xorp_autostart.tar.gz" install_path="/local"/>
      …………
      <execute shell="sh" command="/bin/bash /local/xorp_autostart/
start-xorp.sh"/>
      …………
    </services>
</node>
…………
<link client_id="egre0">
    <interface_ref client_id="rt-2:if1"/>
    <interface_ref client_id="rt-1:if1"/>
    <link_type name="egre-tunnel"/>
</link>
```

Figure 37: Node and Link Segments in a Request RSpec

allows experimenters to create customized OS image and save it online to avoid such time-consuming post-boot situation, and the image can be loaded from a given URL like other standard OS image. Therefore, we created a customized Ubuntu 12.04 image with `XORP v1.8` pre-installed, and specified the URL to this customized image when generating the request RSpec with the `scaleup` tool.

Secondly, we configured every VR as an `OSPF` VR, and the `OSPF` configuration file is automatically generated based on the virtual interface information right after the

VR is up, including those interfaces configured for inactive virtual links. Since all S-VRs are provisioned with inactive virtual links reserved to the active VRs, the `OSPF` protocol will be configured automatically for each S-VR at the post-boot stage as well.

6.2.3 Collect Substrate Network Information

In Chapter 5, we proposed a heuristic multi-criteria selection algorithm to select an S-VR to replace a failed VR in a VN, where one of the factors to the optimal selection is the geographical location of the S-VRs we also consider the real-time traffic statistics on the interfaces of the S-VR's substrate host. For the geographical information about an S-VR, we can run `omni` to periodically pull the *advertisement RSpec* from the corresponding aggregate that provisioning resources for the S-VRs.

For the real-time monitoring, GENI provides two ways to view the performance of GENI resources. The first is *GENI Monitoring* framework [5] that provides a web-based interactive interface for experimenters to view the real-time monitoring for a specific *slice* or for GENI hardware (i.e., nodes, links, traffic statistics on interfaces, etc). Currently, this monitoring system does not provide a command-line API for us to pull the monitoring data at this time, so we are not be able to integrate the real-time resource utilization and traffic statistics monitoring module with the VNM, but we believe this will be a great enhancement for the VNM implementation in future. The second way is to use `geni-lib`, which has a sample script to pull the VM availability from all GENI aggregates at real-time. There, we modified this sample code to get the VM availability on each PC of the InstaGENI aggregates that provisions resources for S-VRs, and use this information as the resource utilization on the substrate nodes.

Thus, if we consider two criteria: the distance from a S-VR to the failed VR and the VM load of the S-VR's host, a simplified version of the objective function proposed in Chapter 5 can be defined as:

$$\min \{\lambda \cdot \text{dist\_to\_fail} + \pi \cdot \text{VM\_load}\} \tag{6.1}$$

where λ and π are the weight parameters to determine the emphasis of selection criterion.

6.2.4 Results Discussion

Now we present an implementation of the prototype of VNM on GENI. Since our work is on basis of the VN embedding procedure that has allocated resources to a VN request, assuming that the VNM received a VN request of five VRs in the given topology, and the VNM also provisioned three S-VRs (labeled as 8, 9, 10). Thus, overall the regular VRs and the S-VRs were distributed over seven different InstaGENI aggregates, as presented in Fig. 38. In this VN request, there were three VRs (labeled as 2,3,5) performing as the core VRs. Based on this topology and S-VR mapping, we are able to create the configuration file as the input for the `ScaleUp` to generate the request RSpec, and then sent the RSpec to get the resources reserved from GENI testbed. In particular, the request RSpec also included the links between the S-VRs and all other core VRs, so we can quickly replace the failed VR with one of the S-VR by activating the proper virtual interfaces. As a result, the VNM had to request at least $3 \times 3 = 9$ extra virtual links. This indicating the tradeoff between quick restoration and the number of spare links, so

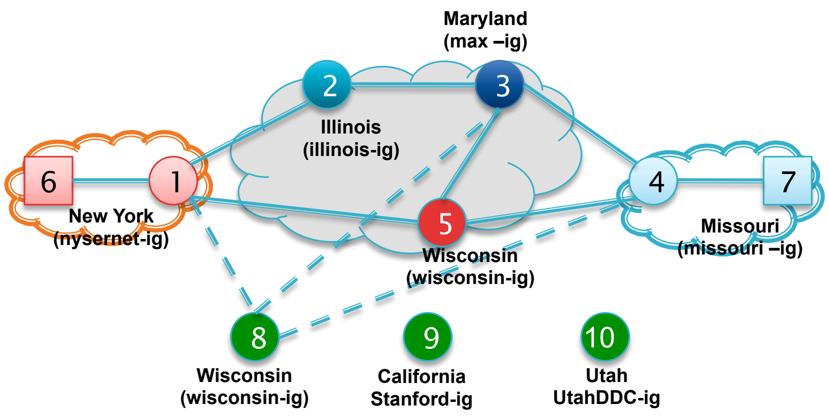


Figure 38: Preliminary Setup

allocating a proper number of S-VRs is necessary to reduce unnecessary virtual link provisioning.

To construct the function (6.1), we parsed the latitude & longitude value for each S-VR from the associated *advertisement RSpec*, and pulled the real-time VM availability information. Table 15 displays the geographic location information (i.e., latitude and longitude value) and the VM load information for each S-VR's host for the topology presented in Fig. 38, which was collected in Oct. 2014. Consider VR-5 failed, the current monitoring data available on GENI testbed allowed us to consider two extreme cases: if the VNM considers to use the S-VR from the nearest site to the failed VR's site, it may assign $\lambda = 1, \pi = 0$, so S-VR from Wisconsin (ID = 8) will be selected based on Table 15. However, if the VNM considers to balance the VM load on the substrate network, it may assign $\lambda = 0, \pi = 1$, so the S-VR from the UtahDDC aggregate will be selected, as only 4% of the VMs have been reserved. In other words, this substrate node at UtahDDC has more resources than the other two substrate nodes. Therefore, the S-VR selection algorithm based on (6.1) will be triggered at the reconfiguration module of the VNM, and once an S-VR is selected, it should connect VR-5's neighbors (i.e., VR-1, VR-3, and VR-4) for topology conservation. For each VR or S-VR, We have installed the scripts that can quickly manipulate the virtual interfaces so that a S-VR can be added to the existing topology immediately (i.e., less than 1 sec), depending on the selection strategy.

Table 15: Selection Criteria for S-VR Selection

S-VR	InstaGENI	latitude	longitude	Dist (mi.)	VM Load
8	wisconsin-ig	43.075	-89.410	0.000	6%
9	stanford-ig	37.430	-122.170	1760.078	14%
10	UtahDDC-ig	40.751	-111.890	1164.032	4%

6.3 Summary

In this chapter, we presented the design of a software-defined resilient virtualized networking environment, where the VNM is the core component to fulfill dynamic VN restoration with the optimal S-VR selection scheme. We presented an proof of concept implementation for our design on the GENI testbed, where we focused on two of the major modules at the VNM: the VN initilizaiton and substrate network information collection, and presented a simplified composite objective based on (5.12).

We next plan to focus on evaluating the impact of the software-defined autonomic VN restoration from the VR failures, which implements the heuristic multi-criteria selection algorithm, on multiple VNs while focusing on the independent and dependent VR failures.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conslusion

This dissertation discussed a problem of survivability against node failures in a virtualized networking environment using redundant nodes with dynamic restoration approach, where the redundant nodes are presented as standby virtual routers (S-VRs) reserved for each virtual network (VN). The objective is to design a dynamic reconfiguration scheme (DRS) with an optimal S-VR selection model for simultaneous VN restoration from either *independent* or *dependent* VR failures.

To understand and solve this problem, we divided the work into three phases. The first phase is to understand the types of cost and constraint to choose an S-VR and replace a failed VR with this S-VR. We conducted an experimental study by employing the DRS on a realistic distributed wide-area virtual network we previously built over the GpENI network. In this experiment, we ran the DRS for a single VN with a shared-S-VR pool in three different virtual topologies, and measured the recovery time and the instantaneous network performance (e.g., bandwidth, packet loss, etc.) during and after the reconfiguration process. The S-VR selection in this experiment was based on the geographical distance. The results showed that with a centralized management, the affected VN can be quickly restored with the proposed DRS, and the geographical distance based selection showed a better network performance after the restoration than a random S-VR selection.

Furthermore, we also notice the resource utilization on the substrate nodes that host the S-VRs and the latency between the S-VR to other active VRs could also be possible factors to the selection.

In the second phase we designed an optimal S-VR selection model that considers all possible factors and costs types in a weighted manner, and the model is able to handle selections for multiple VNs with VR failures. We also proposed a heuristic multi-criteria selection algorithm. By considering a variety of factors (e.g., virtual link bandwidth request, the number of S-VRs per VN, resource utilization on the S-VR's substrate host, etc.), the numerical results showed that the heuristic cost showed very less overhead than the optimal cost when the VNs were provisioned sufficient S-VRs or the substrate nodes have sufficient sources. If the virtual link bandwidth request was higher, the optimization model showed better performance in terms of the restoration ratio. On the other hand, when there are large number of VNs with VR failures simultaneously, the heuristic algorithm ran faster than solving the optimization model when the S-VR selection was to minimize the selection cost. The proposed optimal S-VR selection model for dynamically restore VNs from node failures in a VNE loosely couples multiple selection criteria. The cost functions are general formulations containing a variety of cost components, and they can be easily extended by adding other potential cost components with corresponding weight values. The model is also can be simplified by dropping some components if these components are not critical to the selection.

The third phase is to design a resilient VNE with a software-defined management, where the proposed DRS and the S-VR selection model can be employed. The core component we presented was called *virtual network manager (VNM)*, which has the global view of both the substrate network and the virtual network plane. The VNM is responsible for automating VN initialization and configuration, and dynamic restoration against VR failures by employing our proposed model in the second phase. We implemented a prototype of VNM that implemented VN creation and the DRS on the GENI testbed. Due to the limited monitoring data can be accessed from the GENI testbed by the time we implemented the prototype, we simplified the S-VR model by considering only two factors: geographical location of the S-VRs and the VM load on the substrate nodes. Our ultimate goal is to fully implement the multi-criteria S-VR selection model and the heuristic algorithm on the GENI testbed when more monitoring data are released in future, and understand how this model performed for simultaneously restoring multiple VNs.

7.2 Future Work

There are several aspects we would like explore in the future research. First, we would like to extend the prototype implementation to emulate simultaneous VN auto-restorations from either *independent* or *dependent* VR failures. Second, current GENI testbed has not released the public API for the monitoring system yet, so the challenge for us is to access the real-time raw data about the testbed (i.e., CPU utilization, interface statistics, etc.) that can be used in our proposed model. Once we have the access to the real-time monitoring data, we can apply the multi-criteria heuristic selection, and compare the results with the optimization model. Third, the model we proposed in this dissertation was based on the IP-layer network virtualization environment, we believe it is possible to adapt it to the software-defined networking environment (i.e., OpenFlow Networks), where the backup resources could be ports and switches. Thus, we would like to study the virtualization used in the OpenFlow network environment and explore the protection and restoration model for node failures based on our experiences in the virtualized IP networks.

APPENDIX A

RESOURCE SPECIFICATION (RSPEC) IN GENI

In the GENI context, there are three types of resource specification (RSpec) presented as standard XML documents: *advertisement RSpec*, *request RSpec* and *manifest RSpec*.

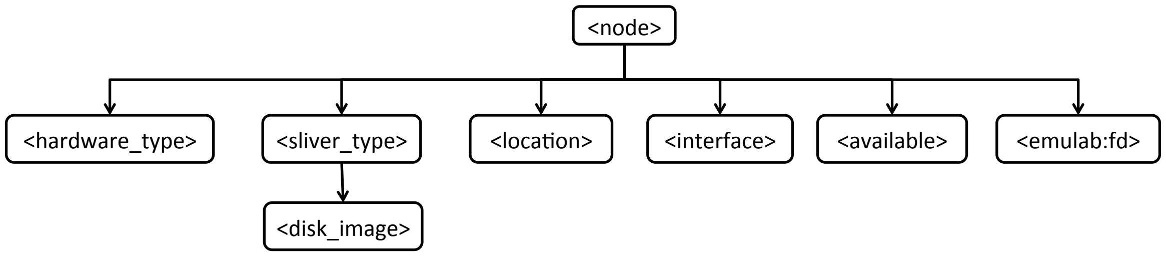
A.1 Advertisement RSpec

An advertisement RSpec (A-RSpec) presents the resources information and status of a GENI rack or a GENI aggregate, and its structure may vary depending on the components of a particular type of GENI rack. We use the advertisement RSpec for two kinds of standard GENI racks (i.e., InstaGENI rack and ExoGENI rack) to illustrate what information we can obtained for this type of RSpec. In this section, we mainly introduce the schemas for the node and link sections in an A-RSpec file.

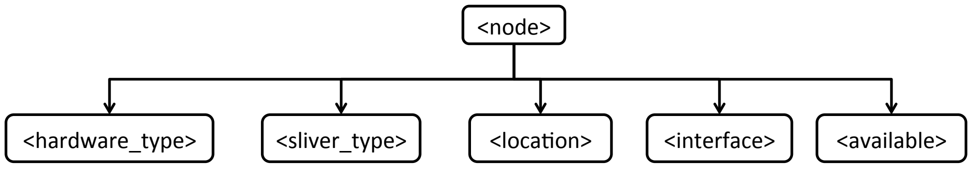
A.1.1 Node Information

Usually a GENI rack consists of a number of nodes that support VM provisioning or network connectivity for an experimenter's slice, and the A-RSpec reveals what resources are available. The schema of the node segment in the A-RSpec can be decomposed as shown in Fig. 39, and we notice that there are slight differences between the node schema of the A-RSpec for the InstaGENI and ExoGENI racks.

- `hardware_type` tells diverse hardwares in the node. For example, if the node



(a) InstaGENI Rack



(b) ExoGENI Rack

Figure 39: Node Section Schema of the A-RSpec

is for computing resource provisioning, the hardware type may be ”pc” or ”vm” related; if the node is for networking resource provisioning, the hardware type could be ”lan” or other interconnectivity.

- `sliver_type` tells the possible computing resources can be provisioned by a computing node. For an InstaGENI rack, it could be raw PCs, OpenVZ or Xen-VMs, according to the virtualization technologies. For an ExoGENI rack, there are more options (e.g., ”EC2M1Small”, ”XOMedium”, etc.) depends on disk and memory size of the VMs. For InstaGENI rack, there is a second level field `disk_image` shows the operating system image can be loaded to the *slivers*.

- *location* indicates the geographical information about the node (i.e., latitude and longitude information).

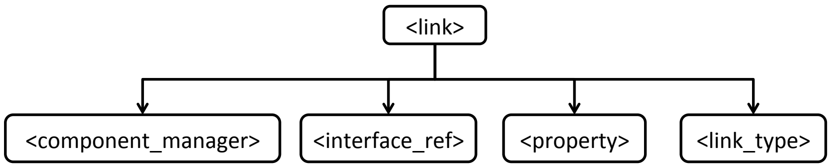


Figure 40: Link Section Schema of the A-RSpec

- `interface` defines the name of the interfaces on the node.
- `available` indicates whether the node is available or not.
- `emulab:fd` is defined for InstaGENI rack only, and it specifies the CPU and memory information of the node.

A.1.2 Link Information

Fig. 40 shows the schema used by the link section in an A-RSpec file. In particular, the `component_manager` indicates the end of the link, i.e., the identifier of the associate GENI rack to the link. Each `property` tag specifies the capacity, latency and loss rate of an interface associated with the link, and the `link_type` tag specifies whether the link is Ethernet or other types.

Two complete examples for GPO-ExoGENI and UMKC-InstaGENI can be found at:

`https://docs.google.com/document/d/1544b8TfYICAsY1qACTNhQuE-dGWVqVVFIdod_8FpQis/edit?usp=sharing`

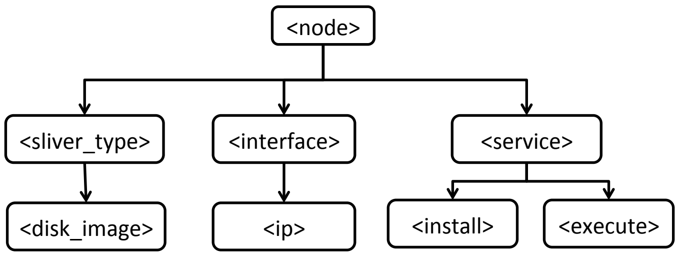


Figure 41: Node Section Schema of the R-RSpec

A.2 Request RSpec

The request RSpec (R-RSpec) is used to request resources for a specific *slice*. Here we briefly introduced the basic components of an R-RSpec file (i.e. node section and link section) for InstaGENI racks. It is important to understand the schema of the request RSpec to a particular type of GENI rack, so that we can automate the R-RSpec generation relying on `geni-lib`.

Fig. 41 shows the basic schema of the node section in a R-RSpec. Usually we need to specify the `sliver_type` and `interface` information. For *sliver* type, we should indicate the type of node resources to be reserved for a *sliver*, such as Xen VMs, OpenVZ or raw PCs, and then indicate the disk image to be loaded to the *sliver*. In regard to the interface information, when we are interested in creating a virtual network, we will have to configure IP address for each interface of a *sliver*. Furthermore, if we would like to install any software or packages to the *sliver* or execute particular command and the post-boot stage, we will have to add `service` segment to the node section in the R-RSpec.

For the link section in an R-RSpec, there are two basic tags: `interface_ref` and `link_type`. The `interface_ref` tags specify the identifiers of the *sliver*'s interface and the associated interface on a remote *sliver*. The `link_type` indicate whether the link is a GRE/EGRE tunnel, LAN or stitching link. Sometimes we may also add the *property* tag to specify the capacity and latency for the link. For a link connecting two *slivers* from two different GENI racks, we have to include `component_manager` to indicate the identifier of the corresponding GENI racks.

A list of example R-RSpec used for GENI tutorials can be found at:

`http://www.gpolab.bbn.com/experiment-support/`

A.3 Manifest RSpec

Once we have resources reserved for our *slice* on the GENI testbed, GENI API will return us a manifest RSpec (M-RSpec) file to indicate the actual GENI hardwares that provisioning resources to our *slice*. For instance, if we do not specify particular node ID on the GENI racks, or we do not specify which GENI rack we would like to request resource from, once the resources are allocated and ready, the M-RSpec tells us the identifiers of the nodes and particular GENI rack.

A list of M-RSpec along with the associated R-RSpec can be found at:

`http://groups.geni.net/geni/wiki/GIMIv3/SampleManifests`

APPENDIX B

OPTIMAL S-VR SELECTION MODEL IN AMPL

```
# ========= Index bound Declaration ======#
# the number of physical routers on the substrate
param num_pr > 0 integer;
# the number of virtual networks
param num_vn > 0 integer;

# ========== Set declaration ============= #


# the set of phyiscal resources (CPU & RAM)
set RES;

# the set of physical routers: R
set PRouter = 1 .. num_pr by 1;

# the set of virtual networks overlay over the substrate: G
set VNet = 1 .. num_vn by 1;

# the number of ports on a physical router
param num_p {i in PRouter} > 0 integer;

# the set of physical ports on the physical rouer i:P_{i}
set Port{i in PRouter} = 1 .. num_p[i] by 1;

# the number of virtual interfaces on a virtual router: r_{i,j}
param num_viface {i in PRouter, j in VNet} > 0 integer;

# the set of virtual interfaces on a virtual router r_{i,j}: M^j_i
set VIface {i in PRouter, j in VNet} = 1 .. num_viface[i,j] by 1;

# the set of failed virtual router r_{i,j}: F^j
set Fail {VNet};


# ========== Parameter Declaration ======= #
# The available bw of port p on physical router i: b_{i,p}
```

```
param b {i in PRouter, p in Port[i]} >= 0;

# Indecates whether a virtual router r_{i,j} is connected: alpha^j_i
param alpha {i in PRouter, j in VNet} binary;

# Indicating whether a virtual router r_{i,j} is failed: beta^j_i
param beta {i in PRouter, j in VNet} binary;

# Indicating mapping relation between virtual interface m and potential
# connected remote virtual router r_{j,k}:(gamma^{j,f}_{i,m,k})
param gamma { j in VNet, f in Fail[j], i in PRouter, m in VIface[i,j], k in PRouter}
= if m = k then 1 else 0;


# the upper limit on how many S-VRs can be created on a substrate node: h_i
param limit {i in PRouter} >=0 integer;


# Indicating whether a virtual router r_{i,j} is a standby: delta^j_i
param delta {i in PRouter, j in VNet} binary;

# number of failed router in virtual network j: n^j
param num_fail {j in VNet} = card {Fail[j]} >= 0;

# indicating whether r_{i,j} and r_{k,j} is connected (equals to 1): e^j_{i,k}
param e { j in VNet, i in PRouter, k in PRouter: i <> k} binary;

# requested bw on the virtual interface m of
# standby virtual router r_{i,j}: c^{j,f}_{i,m,k}
param c { j in VNet, f in Fail[j], i in PRouter, m in VIface[i,j],
k in PRouter: i <> k} >= 0;


# ========== Start Link-Path Formulation ===== #
# number of substrate links
param L > 0 integer;

# the set of substrate links
set slink := 1 .. L by 1;

# number of demands
param D > 0 integer;

# the demand set in terms of index
```

```
set demand := 1 .. D by 1;

# number of paths by demand d
param Pd{d in demand} > 0 integer;

# Paths sets per demand tuple
set Q{d in demand} := 1 .. Pd[d] by 1;

# slink parameters
param slink_src{l in slink} within PRouter;
param slink_dst{l in slink} within PRouter;
param slink_capacity{l in slink} >= 0;

# demand parameters
param demand_src{d in demand} within PRouter;
param demand_dst{d in demand} within PRouter;
param demand_vn{d in demand} within VNet;
param demand_fnode{f in demand} within PRouter;

# demand volumn: c[j,f,i,m,k]
param demand_c{d in demand} > 0;

# Paths set
set paths{d in demand, q in Q[d]} within slink;

# link-path indicator, set to 1 if path q for demand tuple d uses link l, otherwise is 0;
param Delta{d in demand, q in Q[d], l in slink} = if l in paths[d,q] then 1 else 0;

# Flow amount on path q for demand d
var x{d in demand, q in Q[d]} >= 0;
# ========== END Link-Path formulation ========== #

# ========== Cost Function Parameters ========== #
### virtual interface operation cost ###
# the cost of enabling an interface: s^-
param enable >= 0;

# the cost of disabling an interface: s^+
param disable >= 0;

# the cost of configure IP addresses: l
param ipConfig >= 0;

# the bool value that indicates whether IP configure is needed on a remote
# virtual router r^j_k (failed VR's neighbor): tao_{j,k}
param tau{k in PRouter, j in VNet} binary;


### Virtual router connection cost ###
# distance between two routers: d_{i,k}
param dist{i in PRouter, k in PRouter: i <> k} >= 0;

# round trip time between two routers: rtt_{i,k}
param rtt {i in PRouter, k in PRouter: i <> k} >= 0;

# loss rate between two routers
param pktloss {i in PRouter, k in PRouter: i <> k} := 0;

# weight parameters for type-II cost function
param a1 >= 0;
param a2 >= 0;
param b1 >= 0;
param b2 >= 0;
param b3 >= 0;

### residual physical resource
# residual hardware resources (i.e. CPU, Memory): res_{i,r}
param res {i in PRouter, r in RES} >=0;




# the available resources on physical router i
param xi {i in PRouter} = sum {p in Port[i]} b[i,p];


# the cost of connecting a virtual router r_{k,j} to standby r_{i,j}:
# \sigma^{j,f}_{i,m,k}
param sigma {j in VNet, f in Fail[j], i in PRouter, m in VIface[i,j], k in PRouter:
i <> k && i <> f && k <> f}
   =  b1 * (a1 * dist[i,f] + a2 * dist[i,k]) + b2 * rtt[i,k] + b3 * pktloss[i,k];

# the cost of enable virtual interface m: \eta^{j,f}_{i,m,f}
param eta {j in VNet, f in Fail[j], i in PRouter, m in VIface[i,j],
k in PRouter: i <> k && i <> f}
   = 2 * (disable + enable) + tau[k,j] * ipConfig;

# check how many s-vr created on a substrate router
```

```
param sum_delta{i in PRouter} := sum{j in VNet, f in Fail[j]} delta[i,j];

#### objective function weight parameter ####
param m1 >= 0;
param m2 >= 0;
param m3 >= 0;

#### Varaibles Declaration ####

# Indicate whether a virtual router r_{i,j} is selected to replace
# failed router r_{f,j}: u^{j,f}_{i}
var u {j in VNet, f in Fail[j], i in PRouter} binary;

# indicate whether connected virtual router r_{k,j} is connected to
# standby virtual router r_{i,j}: v^{j,f}_{i,m,k}
var v {j in VNet, f in Fail[j], i in PRouter, m in VIface[i,j],
k in PRouter: i <> k && i <> f} binary;

# indicate resource utilization  ------- r
var RR >= 0;


# ========== objective function ========= #

minimize Total_Cost:
sum { j in VNet, f in Fail[j], i in PRouter, m in VIface[i,j], k in PRouter:
i <> k && i <> f && k <> f} (m1 * eta[j,f,i,m,k] +
m2 * sigma[j,f,i,m,k]) * delta[i,j] * gamma[j,f,i,m,k] * v[j,f,i,m,k] + m3 * RR;

# Constraint (1)
subject to IsStandby {j in VNet, f in Fail[j], i in PRouter: delta[i,j] == 1}:
    u[j,f,i] <= delta[i,j]*(1 - beta[i,j]);

# Constraint (2)
subject to IsSelect {j in VNet, f in Fail[j], i in PRouter, m in VIface[i,j],
k in PRouter: i <> k && i <> f && f <> k && delta[i,j] == 1
&& e[j,f,k] == 1 && gamma[j,f,i,m,k] == 1}: v[j,f,i,m,k] <= u[j,f,i];

# Constraint (3)
subject to OneSPerF {j in VNet, f in Fail[j]}:
sum{i in PRouter} delta[i,j] * u[j,f,i] = 1;

# Constraint (4)
subject to OneFPerS {i in PRouter, j in VNet: delta[i,j]==1}:
```

```
sum{f in Fail[j]} delta[i,j] * u[j,f,i] <= 1;

# Constraint (5)
subject to max_select {i in PRouter}:
    sum{j in VNet, f in Fail[j]} delta[i,j] * u[j,f,i] <= limit[i];

# Constraint (6)
subject to IsConnect {j in VNet, f in Fail[j], i in PRouter: delta[i,j] == 1
&& i <> f}:
    sum{m in VIface[i,j], k in PRouter: k <> f && k <> i} e[j,f,k] *
     gamma[j,f,i,m,k] * v[j,f,i,m,k]
    - sum{k in PRouter: k <> i && k <> f} e[j,f,k] * beta[f,j] * u[j,f,i]= 0;

# Constraint (7)
subject to bw {i in PRouter, j in VNet, f in Fail[j]: delta[i,j] == 1 && i <> f}:
    sum {m in VIface[i,j], k in PRouter: i <> k && k <> f}
        delta[i,j] * c[j,f,i,m,k] * gamma[j,f,i,m,k] * v[j,f,i,m,k]
        <= sum {p in Port[i]} delta[i,j] * b[i,p] * u[j,f,i];

# Constraint (8)
subject to res_util {i in PRouter: sum_delta[i] > 0}:
    sum {j in VNet, f in Fail[j]: i <> f} delta[i,j] * u[j,f,i] *
    sum {k in PRouter, m in VIface[i,j]: k <> i && k <> f} gamma[j,f,i,m,k]
     * c[j,f,i,m,k] + sum {p in Port[i]} (1 - b[i,p])
     <= sum{p in Port[i]} 1 * RR;

# Constraint (9)
subject to d_pair{d in demand, j in VNet, f in Fail[j], i in PRouter:
i<>f && delta[i,j] == 1 && demand_fnode[d] == f && demand_vn[d] == j
&& demand_src[d] == i}:
sum{q in Q[d]} x[d,q] = demand_c[d] * u[j,f,i];

# Constraint (10)
subject to cl{l in slink}: sum{d in demand, q in Q[d]} Delta[d,q,l] * x[d,q]
<= slink_capacity[l];

subject to upRRbd:
    RR <= 0.9;
```

REFERENCE LIST

[1] Emulab. http://emulab.net/.

[2] FITS:Future Internet Testbed with Security. www.gta.urfj.br/fits.

[3] GENI. www.geni.net.

[4] geni-lib. https://bitbucket.org/barnstorm/geni-lib.

[5] GENI Monitoring Wiki. http://genimon.uky.edu/.

[6] GpENI. www.gpeni.net.

[7] Introducing Linux Network Namespaces. http://blog.scottlowe.org/2013/09/04/introducing-linux-network-namespaces/.

[8] iperf. http://sourceforge.net/projects/iperf/.

[9] OFELIA. http://www.fp7-ofelia.eu/.

[10] OpenVZ. https://openvz.org/Main_Page.

[11] PlanetLab. www.planet-lab.org.

[12] User Controlled LightPaths Project (UCLP). http://uclp.uwaterloo.ca/index.htm.

[13] Ahlgren, B., Dannewitz, C., Imbrenda, C., Kutscher, D., and Ohlman, B. A survey of information-centric networking. *Communications Magazine, IEEE 50*, 7 (2012), 26–36.

[14] Andersen, D., Balakrishnan, H., Kaashoek, F., and Morris, R. Resilient Overlay Networks. *SIGOPS Oper. Syst. Rev. 35*, 5 (Oct. 2001), 131–145.

[15] Ata, S., Huang, D., Liu, X., Wada, A., Xing, T., Juluri, P., Chung, C.-J., Sato, Y., and Medhi, D. SeRViTR: A framework, implementation, and a testbed for a trustworthy future Internet. *Computer Networks 63* (2014), 128–146.

[16] Banerjee, S., Griffin, T. G., and Pias, M. The interdomain connectivity of PlanetLab nodes. In *Passive and active network measurement*. Springer, 2004, pp. 73–82.

[17] Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. Xen and the art of virtualization. *ACM SIGOPS Operating Systems Review 37*, 5 (2003), 164–177.

[18] Bari, M. F., Roy, A. R., Chowdhury, S. R., Zhang, Q., Zhani, M. F., Ahmed, R., and Boutaba, R. Dynamic Controller Provisioning in Software Defined Networks. In *CNSM* (2013), pp. 18–25.

[19] Bastin, N., Bavier, A., Blaine, J., Chen, J., Krishnan, N., Mambretti, J., McGeer, R., Ricci, R., and Watts, N. The InstaGENI Initiative: An Architecture for Distributed Systems and Advanced Programmable Networks. *Computer Networks 61* (2014), 24–38.

[20] Bavier, A., Feamster, N., Huang, M., Peterson, L., and Rexford, J. In VINI veritas: realistic and controlled network experimentation. In *ACM SIGCOMM Computer Communication Review* (2006), vol. 36, ACM, pp. 3–14.

[21] Berman, M., Chase, J. S., Landweber, L., Nakao, A., Ott, M., Raychaudhuri, D., Ricci, R., and Seskar, I. GENI: A Federated Testbed for Innovative Network Experiments. *Comput. Netw. 61* (Mar. 2014), 5–23.

[22] Bhatia, S., Motiwala, M., Muhlbauer, W., Mundada, Y., Valancius, V., Bavier, A., Feamster, N., Peterson, L., and Rexford, J. Trellis: A platform for building flexible, fast virtual networks on commodity hardware. In *Proceedings of the 2008 ACM CoNEXT Conference* (2008), ACM, pp. 72:1–72:6.

[23] Butt, N. F., Chowdhury, M., and Boutaba, R. Topology-Awareness and Reoptimization Mechanism for Virtual Network Embedding. *NETWORKING 2010 6091* (2010), 27–39.

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

[25] Cetinkaya, E. K., and Sterbenz, J. P. A taxonomy of network challenges. In *Design of Reliable Communication Networks (DRCN), 2013 9th International Conference on the* (2013), IEEE, pp. 322–330.

[26] Chan, T.-J., Chan, C.-K., Chen, L.-K., and Tong, F. A self-protected architecture for wavelength-division-multiplexed passive optical networks. *IEEE photonics technology letters 15*, 11 (2003), 1660–1662.

[27] Chaparadza, R., Wodczak, M., Ben Meriem, T., De Lutiis, P., Tcholtchev, N., and Ciavaglia, L. Standardization of resilience & survivability, and autonomic fault-management, in evolving and future networks: an ongoing initiative recently launched in ETSI. In *Design of Reliable Communication Networks (DRCN), 2013 9th International Conference on the* (2013), IEEE, pp. 331–341.

[28] Chen, D., Qiu, X., Qu, Z., Zhang, S., and Li, W. Algorithm for virtual nodes reconfiguration on network virtualization. In *2011 International Conference on Advanced Intelligence and Awareness Internet (AIAI 2011)*.

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

[30] Chiueh, S. N. T.-c., and Brook, S. A survey on virtualization technologies. *RPE Report* (2005), 1–42.

[31] Chowdhury, M., Rahman, M. R., and Boutaba, R. Vineyard: Virtual network embedding algorithms with coordinated node and link mapping. *IEEE/ACM Transactions on Networking (TON) 20*, 1 (2012), 206–219.

[32] Chowdhury, N. M. K., and Boutaba, R. A survey of network virtualization. *Computer Networks 54*, 5 (2010), 862 – 876.

[33] Chowdhury, N. M. K., Rahman, M. R., and Boutaba, R. Virtual network embedding with coordinated node and link mapping. In *IEEE INFOCOM 2009* (2009), pp. 783–791.

[34] Chun, B., Culler, D., Roscoe, T., Bavier, A., Peterson, L., Wawrzoniak, M., and Bowman, M. Planetlab: an overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication Review 33*, 3 (2003), 3–12.

[35] Clark, D., Lehr, B., Bauer, S., Faratin, P., Sami, R., and Wroclawski, J. Overlay Networks and the Future of the Internet. *Communications and Strategies 63, 3rd quater* (2006), 1–21.

[36] Collins, A. *The Detour framework for packet rerouting*. PhD thesis, Citeseer, 1998.

[37] Committee, L. S. IEEE Standard for Local and metropolitan area networks – Virtual Bridged Local Area Networks. Tech. rep., IEEE, 2005.

[38] Des Ligneris, B. Virtualization of Linux based computers: the Linux-VServer project. In *High Performance Computing Systems and Applications, 2005. HPCS 2005. 19th International Symposium on* (2005), IEEE, pp. 340–346.

[39] Doriguzzi Corin, R., Gerola, M., Riggio, R., De Pellegrini, F., and Salvadori, E. Vertigo: Network virtualization and beyond. In *Software Defined Networking (EWSDN), 2012 European Workshop on* (2012), IEEE, pp. 24–29.

[40] Doshi, B. T., Dravida, S., Harshavardhana, P., Hauser, O., and Wang, Y. Optical network design and restoration. *Bell Labs Technical Journal 4*, 1 (1999), 58–84.

[41] Du, P., Chen, M., and Nakao, A. Port-space isolation for multiplexing a single IP address through open vswitch. In *Testbeds and Research Infrastructures. Development of Networks and Communities*. Springer, 2011, pp. 113–122.

[42] Edwards, S., Liu, X., and Riga, N. Creating Repeatable Computer Science and Networking Experiments on Shared, Public Testbeds. *ACM SIGOPS Operating Systems Review 49*, 1 (2015), 90–99.

[43] Ferguson, P., and Huston, G. What is a VPN?, 1998.

[44] Fischer, A., Botero, J. F., Till Beck, M., De Meer, H., and Hesselbach, X. Virtual network embedding: A survey. *Communications Surveys & Tutorials, IEEE 15*, 4 (2013), 1888–1906.

[45] Fonseca, P., Bennesby, R., Mota, E., and Passito, A. A Replication Component for Resilient OpenFlow-based Networking. In *Network Operations and Management Symposium (NOMS)* (2012), pp. 933–939.

[46] Garcia, J., and Duato, J. An algorithm for dynamic reconfiguration of a multicomputer network. In *Parallel and Distributed Processing, 1991. Proceedings of the Third IEEE Symposium on* (dec 1991), pp. 848 –855.

[47] Gleeson, B., Lin, A., Heinanen, J., Armitage, G., and Malis, A. A Framework for IP based Virtual Private Networks, RFC 2764. Tech. rep., IETF, 2000.

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

[49] Golab, W., and Boutaba, R. Path Selection in User-controlled Circuit-switched Optical Networks. *Opt. Switch. Netw. 5*, 2-3 (June 2008), 123–138.

[50] Guo, B., Qiao, C., Wang, J., Yu, H., Zuo, Y., Li, J., Chen, Z., and He, Y. Survivable virtual network design and embedding to survive a facility node failure. *Lightwave Technology, Journal of 32*, 3 (2014), 483–493.

[51] Guo, T., Wang, N., Moessner, K., and Tafazolli, R. Shared backup network provision for virtual network embedding. In *Communications (ICC), 2011 IEEE International Conference on* (2011), IEEE, pp. 1–5.

[52] Habib, I. Virtualization with KVM. *Linux J. 2008*, 166 (Feb. 2008).

[53] Hamzeh, K., Pall, G., Verthein, W., Taarud, J., Little, W., and Zorn, G. RFC-2637: Point-to-point tunneling protocol (PPTP). *Network Working Group* (1999).

[54] Hanks, S., Meyer, D., Farinacci, D., and Traina, P. Generic routing encapsulation (GRE). Tech. rep., IETF, 2000.

[55] Helsley, M. LXC: Linux container tools. *IBM devloperWorks Technical Library* (2009).

[56] Herker, S., Khan, A., and An, X. Survey on survivable virtual network embedding problem and solutions. In *ICNS 2013, The Ninth International Conference on Networking and Services* (2013), pp. 99–104.

[57] Hibler, M., Ricci, R., Stoller, L., Duerig, J., Guruprasad, S., Stack, T., Webb, K., and Lepreau, J. Large-scale virtualization in the Emulab network testbed. In *USENIX 2008 Annual Technical Conference on Annual Technical Conference* (2008), pp. 113–128.

[58] Hock, D., Hartmann, M., Gebert, S., Jarschel, M., Zinner, T., and Tran-Gia, P. Pareto-optimal resilient controller placement in SDN-based core networks. In *Teletraffic Congress (ITC), 2013 25th International* (2013), IEEE, pp. 1–9.

[59] Houidi, I., Louati, W., Zeghlache, D., Papadimitriou, P., and Mathy, L. Adaptive virtual network provisioning. In *Proceedings of the second ACM SIGCOMM workshop on Virtualized infrastructure systems and architectures* (2010), ACM, pp. 41–48.

[60] Hu, Q., Wang, Y., and Cao, X. Location-constrained survivable network virtualization. In *Sarnoff Symposium (SARNOFF), 2012 35th IEEE* (2012), IEEE, pp. 1–5.

[61] Huang, M. VNET: PlanetLab virtualized network access. Tech. rep., Tech. Rep. PDN–05–029, PlanetLab Consortium, 2005.

[62] Huang, M., and Parmentelat, T. MyPLC User's Guide. `https://svn.planet-lab.org/wiki/MyPLCUserGuide`.

[63] Ishiguro, K., Takada, T., Ohara, Y., Zinin, A., Natapov, G., and Mizutani, A. Quagga routing suite, 2007.

[64] Kent, S., and Seo, K. RFC 4301: Security Architecture for the In. temet Protocol, 2005.

[65] Kephart, J. O. An Architectural Blueprint for Autonomic Computing, 3rd Edition. Tech. rep., IBM, June 2005.

[66] Liu, X., Edwards, S., Riga, N., and Medhi, D. Design of a software-defined resilient virtualized networking environment. In *Design of Reliable Communication Networks (DRCN), 2015 11th International Conference on the* (2015), IEEE, pp. 111–114.

[67] Liu, X., Juluri, P., and Medhi, D. An experimental study on dynamic network reconfiguration in a virtualized network environment using autonomic management. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on* (2013), IEEE, pp. 616–622.

[68] Liu, X., and Medhi, D. Optimal standby virtual routers selection for node failures in a virtual network environment. In *Network and Service Management (CNSM), 2014 10th International Conference on* (2014), IEEE, pp. 28–36.

[69] Liu, X., Mohanraj, S., Pióro, M., and Medhi, D. Multipath Routing From a Traffic Engineering Perspective: How Beneficial is It? In *Network Protocols (ICNP), 2014 IEEE 22nd International Conference on* (2014), IEEE, pp. 143–154.

[70] Liu, X., Wada, A., Xing, T., Juluri, P., Sato, Y., Ata, S., Huang, D., and Medhi, D. SeRViTR: A framework for trust and policy management for a secure Internet and its proof-of-concept implementation. In *Network Operations and Management Symposium (NOMS), 2012 IEEE* (2012), IEEE, pp. 1159–1166.

[71] Liu, Y., and Tipper, D. Successive survivable routing for node failures. In *Global Telecommunications Conference, 2001. GLOBECOM'01. IEEE* (2001), vol. 4, IEEE, pp. 2093–2097.

[72] Lua, E. K., Crowcroft, J., Pias, M., Sharma, R., and Lim, S. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys & Tutorials, IEEE 7*, 2 (2005), 72–93.

[73] Maier, G., Pattavina, A., De Patre, S., and Martinelli, M. Optical network survivability: protection techniques in the WDM layer. *Photonic Network Communications 4*, 3-4 (2002), 251–269.

[74] Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, C.-N., and Diot, C. Characterization of failures in an IP backbone. In *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies* (2004), vol. 4, IEEE, pp. 2307–2317.

[75] Markopoulou, A., Iannaccone, G., Bhattacharyya, S., Chuah, C.-N., Ganjali, Y., and Diot, C. Characterization of failures in an operational IP backbone network. *IEEE/ACM Transactions on Networking (TON) 16*, 4 (2008), 749–762.

[76] Masti, S. B., and Raghavan, S. V. Vna: An enhanced algorithm for virtual network embedding. In *Computer Communications and Networks (ICCCN), 2012 21st International Conference on* (2012), IEEE, pp. 1–9.

[77] McKeown, N. Software-defined networking. *INFOCOM keynote talk 17*, 2 (2009), 30–32.

[78] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and Turner, J. OpenFlow: enabling innovation in campus networks. *ACM SIGCOMM Computer Communication Review 38*, 2 (2008), 69–74.

[79] Medhi, D., Ramamurthy, B., Scoglio, C., Rohrer, J. P., Çetinkaya, E. K., Cherukuri, R., Liu, X., Angu, P., Bavier, A., Buffington, C., and Sterbenz, J. P. The GpENI Testbed: Network Infrastructure, Implementation Experience, and Experimentation. *Computer Networks 61* (2014), 51–74.

[80] Medhi, D., and Ramasamy, K. *Network Routing*. Morgan Kaufmann, 2007, ch. 18, pp. 634–641.

[81] Mohan, G., Murthy, C., and Somani, A. K. Efficient algorithms for routing dependable connections in WDM optical networks. *IEEE/ACM Transactions on Networking (TON) 9*, 5 (2001), 553–566.

[82] Moraes, I. M., Mattos, D. M., Ferraz, L. H. G., Campista, M. E. M., Rubinstein, M. G., Costa, L. H. M., de Amorim, M. D., Velloso, P. B., Duarte, O. C. M.,

and Pujolle, G. FITS: A flexible virtual network testbed architecture. *Computer Networks 63* (2014), 221–237.

[83] Nakao, A., Ozaki, R., and Nishida, Y. CoreLab: an emerging network testbed employing hosted virtual machine monitor. In *Proceedings of the 2008 ACM CoNEXT Conference* (2008), ACM, pp. 73:1–73:6.

[84] Nejabati, R., Escalona, E., Peng, S., and Simeonidou, D. Optical network virtualization. In *Optical Network Design and Modeling (ONDM), 2011 15th International Conference on* (2011), IEEE, pp. 1–5.

[85] Office., G. P. Stitcher. `http://trac.gpolab.bbn.com/gcf/wiki/Stitcher`.

[86] Orlowski, S., Wessäly, R., Pióro, M., and Tomaszewski, A. SNDlib 1.0Ñsurvivable network design library. *Networks 55*, 3 (2010), 276–286.

[87] Peterson, L., Bavier, A., Fiuczynski, M. E., and Muir, S. Experiences building planetlab. In *Proceedings of the 7th symposium on Operating systems design and implementation* (2006), USENIX Association, pp. 351–366.

[88] Peterson, L., and Roscoe, T. The design principles of PlanetLab. *ACM SIGOPS operating systems review 40*, 1 (2006), 11–16.

[89] Pfaff, B., Pettit, J., Amidon, K., Casado, M., Koponen, T., and Shenker, S. Extending Networking into the Virtualization Layer. In *Hotnets* (2009), pp. 1–6.

[90] Quétier, B., Neri, V., and Cappello, F. Scalability comparison of four host virtualization tools. *Journal of Grid Computing 5*, 1 (2007), 83–98.

[91] Rahman, M., and Boutaba, R. SVNE: Survivable Virtual Network Embedding Algorithms for Network Virtualization. *IEEE Transactions on Network and Service Management 10* (June 2013), 105–118.

[92] Rahman, M. R., Aib, I., and Boutaba, R. Survivable virtual network embedding. In *NETWORKING 2010*. Springer, 2010, pp. 40–52.

[93] Ramamurthy, S., Sahasrabuddhe, L., and Mukherjee, B. Survivable WDM mesh networks. *Lightwave Technology, Journal of 21*, 4 (2003), 870–883.

[94] Ravindran, R., Liu, X., Chakraborti, A., Zhang, X., and Wang, G. Towards software defined ICN based edge-cloud services. In *Cloud Networking (CloudNet), 2013 IEEE 2nd International Conference on* (2013), IEEE, pp. 227–235.

[95] Riondato, M. Jails. `https://www.freebsd.org/doc/handbook/jails.html`.

[96] Ripeanu, M. Peer-to-peer architecture case study: Gnutella network. In *Peer-to-Peer Computing, 2001. Proceedings. First International Conference on* (2001), IEEE, pp. 99–100.

[97] Rosen, E., and Rekhter, Y. BGP/MPLS IP Virtual Private Networks (VPNs), RFC 4364. Tech. rep., IETF, February 2006.

[98] Savage, S., Anderson, T., Aggarwal, A., Becker, D., Cardwell, N., Collins, A., Hoffman, E., Snell, J., Vahdat, A., Voelker, G., and Zahorjan, J. Detour: Informed Internet routing and transport. *Micro, IEEE 19*, 1 (1999), 50–59.

[99] Schaffrath, G., Werle, C., Papadimitriou, P., Feldmann, A., Bless, R., Greenhalgh, A., Wundsam, A., Kind, M., Maennel, O., and Mathy, L. Network virtualization architecture: proposal and initial prototype. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures* (2009), ACM, pp. 63–72.

[100] Schroeder, M., Birrell, A., Burrows, M., Murray, H., Needham, R., Rodeheffer, T., Satterthwaite, E., and Thacker, C. Autonet: a high-speed, self-configuring local area network using point-to-point links. *Selected Areas in Communications, IEEE Journal on 9*, 8 (oct 1991), 1318 –1335.

[101] Shamsi, J., and Brockmeyer, M. QoSMap: QoS aware mapping of virtual networks for resiliency and efficiency. In *Globecom Workshops, 2007 IEEE* (2007), IEEE, pp. 1–6.

[102] Shamsi, J., and Brockmeyer, M. QoSMap: achieving quality and resilience through overlay construction. In *Internet and Web Applications and Services, 2009. ICIW'09. Fourth International Conference on* (2009), IEEE, pp. 58–67.

[103] Sherwood, R., Gibb, G., Yap, K.-K., Appenzeller, G., Casado, M., McKeown, N., and Parulkar, G. Flowvisor: A network virtualization layer. *OpenFlow Switch Consortium, Tech. Rep* (2009).

[104] Sivakumar, M., Shenai, R. K., and Sivalingam, K. M. A survey of survivability techniques for optical WDM networks. In *Emerging Optical Network Technologies*. Springer, 2005, pp. 297–331.

[105] Soltesz, S., Pötzl, H., Fiuczynski, M. E., Bavier, A., and Peterson, L. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. In *ACM SIGOPS Operating Systems Review* (2007), vol. 41, ACM, pp. 275–287.

[106] Sundararaj, A. I., and Dinda, P. A. Towards Virtual Networks for Virtual Machine Grid Computing. In *Virtual machine research and technology symposium* (2004), pp. 177–190.

[107] Telesis, A. AlliedWare Plus OS Overview: VLANs (Virtual LANs). `http://www.alliedtelesis.com/userfiles/file/overview_vlans.pdf`.

[108] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and Palter, B. RFC 2661-Layer Two Tunneling Protocol(L2TP). *IETF, August* (1999).

[109] Vakali, A., and Pallis, G. Content delivery networks: Status and trends. *Internet Computing, IEEE 7*, 6 (2003), 68–74.

[110] Wada, A., Sato, Y., Liu, X., Xing, T., Ata, S., Medhi, D., Huang, D., and Oka, I. A behavior based policy management for adaptive trustworthiness assignment in

future network. In *Integrated Network Management (IM 2013), 2013 IFIP/IEEE International Symposium on* (2013), IEEE, pp. 784–787.

[111] Xing, T., Liu, X., Chung, C.-J., Wada, A., Ata, S., Huang, D., and Medhi, D. Constructing a virtual networking environment in a geo-distributed programmable layer-2 networking environment (G-PLaNE). In *Communications (ICC), 2012 IEEE International Conference on* (2012), IEEE, pp. 5879–5884.

[112] Xylomenos, G., Ververidis, C. N., Siris, V., Fotiou, N., Tsilopoulos, C., Vasilakos, X., Katsaros, K. V., Polyzos, G. C., et al. A survey of information-centric networking research. *Communications Surveys & Tutorials, IEEE 16*, 2 (2014), 1024–1049.

[113] Yeow, W.-L., Westphal, C., and Kozat, U. Designing and Embedding Reliable Virtual Infrastructures. In *Proceedings of the Second ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures* (New York, NY, USA, 2010), VISA '10, ACM, pp. 33–40.

[114] Yu, H., Anand, V., Qiao, C., and Sun, G. Cost efficient design of survivable virtual infrastructure to recover from facility node failures. In *Communications (ICC), 2011 IEEE International Conference on* (2011), IEEE, pp. 1–6.

[115] Yu, M., Yi, Y., Rexford, J., and Chiang, M. Rethinking virtual network embedding: substrate support for path splitting and migration. *SIGCOMM Comput. Commun. Rev. 38*, 2 (Mar. 2008), 17–29.

[116] Zhang, X., Chen, X., and Phillips, C. Achieving effective resilience for qos-aware application mapping. *Computer Networks 56*, 14 (2012), 3179–3191.

[117] Zhang, X., Phillips, C., and Chen, X. An overlay mapping model for achieving enhanced QoS and resilience performance. In *Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), 2011 3rd International Congress on* (2011), IEEE, pp. 1–7.

[118] Zhou, D., and Subramaniam, S. Survivability in optical networks. *IEEE network 14*, 6 (2000), 16–23.

[119] Zhou, Y., Li, Y., Jin, D., Su, L., and Zeng, L. A virtual network embedding scheme with two-stage node mapping based on physical resource migration. In *Communication Systems (ICCS), 2010 IEEE International Conference on* (2010), IEEE, pp. 761–766.

VITA

Xuan Liu was born on June 29, 1985 in Wuhan, Hubei Province, China. She was educated in local public schools and graduated from Hubei Shuiguohu Senior High School in Wuhan, Hubei. After graduating from the public school, she attended China University of Geosciences in Wuhan, Hubei and graduated in June 2003 with a Bachelor degree in Communication Engineering.

After obtaining the undergraduate degree, Ms. Liu joined University of Missouri - Kansas City to pursue Masters in Computer Science. She graduated with a Masters degree in December, 2010. In August 2010, she joined the interdisciplinary PhD program in University of Missouri - Kansas City with Telecommunication & Computer Networking as her Coordinating discipline and Mathematics as her Co-discipline, respectively.

Ms. Liu was an ICN Architectural SW Intern at Futurewei Technologies from August, 2012 to January, 2013, and from May to August, 2013. Ms. Liu was a Network System Intern at Raytheon BBN Technologies from June to August, 2014.

Upon completion of her degree requirement, Ms. Liu will join AT&T Research Labs in Bedminster, NJ.