Identity Management and Resource Allocation in the Network Virtualization Environment

by

N.M. Mosharaf Kabir Chowdhury

A thesis
presented to the University of Waterloo
in fulfilment of the
thesis requirement for the degree of

Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2008

AUTHOR'S DECLARATION

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

N.M. Mosharaf Kabir Chowdhury

Abstract

Due to the existence of multiple stakeholders with conflicting goals and policies, alterations to the existing Internet architecture are now limited to simple incremental updates; deployment of any new, radically different technology is next to impossible. To fend off this ossification, *network virtualization* has been propounded as a diversifying attribute of the future internetworking paradigm. However, many technical issues stand in the way toward its successful realization. In this thesis, we address two basic problems in the network virtualization environment.

The *identity management problem* is primarily concerned with ensuring interoperability across heterogeneous identifier spaces for locating and identifying end hosts in different virtual networks. We propose a novel identity management framework (*iMark*) that enables end-to-end connectivity across heterogeneous virtual networks without revoking their autonomy. We describe the architectural and the functional components of iMark accompanied by the procedures that manipulate these components and validate it through numerical evaluation.

The *virtual network embedding problem* deals with the mapping of virtual nodes and links onto physical network resources. We argue that the separation of the node mapping and the link mapping phases in the existing algorithms considerably reduces the solution space and degrades embedding quality. We propose coordinated node and link mapping, based on a mathematical programming formulation, to devise two algorithms (*D-ViNE* and *R-ViNE*) for the online version of the problem under realistic assumptions. Extensive simulation experiments show that the proposed algorithms significantly outperform the existing heuristics by increasing the acceptance ratio and the revenue while decreasing the cost incurred by the substrate network.

Acknowledgements

First and foremost I thank Allah, the Lord of the Worlds, the Beneficent, and the Merciful, for giving me the light and for enabling me to complete this task.

I express my deepest gratitude to my supervisor, Professor Raouf Boutaba, for his support, guidance, encouragement, and his confidence in me. He has always allowed me to pursue my ideas and provided invaluable feedback to keep my focus and to improve the quality of my research. He is not only a true mentor to me in the avenues of research but also an invaluable friend in the ways of life. I also thank my readers, Professor Ashraf Aboulnaga and Professor Johnny Wong, for their insightful comments and constructive criticisms.

I am truly grateful to my parents for inspiring me throughout my life and for paving a secured way for my endeavors. I cannot thank Raki, my younger brother, enough for constantly being with me over the Internet to ensure a sense of home.

I express my deep appreciation to the members of the UWaterloo Network Virtualization Project. Specially, I would like to thank my collaborators, Fida and Muntasir, for their hard work and support during the *iMark* and the *ViNE-Yard* projects. I would also like to thank Sonia and all other members of the Networking Lab. I am thankful to all my friends and colleagues, specially Mustaq bhai, Shamim bhai, and Dolar, for their invaluable support.

I have been spared the horror of wrestling with LaTeX, and worries about the University of Waterloo typesetting requirements, thanks to the thesis template created and made publicly available by Matthew Skala.

Finally, I would like to thank the University of Waterloo for providing me with the required financial and technical support as well as an excellent academic environment throughout the program.

To my family

Table of Contents

List of Tables

List of Figures

Citations to Previous Publications

Large portions of this thesis appeared in the following publications:

- N.M. Mosharaf Kabir Chowdhury, Muntasir Raihan Rahman, and Raouf Boutaba. Virtual Network Embedding with Coordinated Node and Link Mapping. To appear in *Proceedings of IEEE INFOCOM'2009*, 9 pages, (Accepted: December, 2008).
- N.M. Mosharaf Kabir Chowdhury, Fida-E Zaheer, and Raouf Boutaba. iMark: An Identity Management Framework for Network Virtualization Environment. To appear in *Proceedings of IFIP/IEEE IM'2009*, 8 pages, (Accepted: October, 2008).
- N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba. A Survey of Network Virtualization. Technical Report CS-2008-25, University of Waterloo, October, 2008.
- N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba. Network Virtualization: The Past, The Present, and The Future. Under revision in *IEEE Communications Magazine*, 8 pages, (Submitted: May, 2008).

Chapter 1

Prologue

The Internet has been stunningly successful in modeling the way we access and exchange information in the modern world. Over the course of past three decades, the Internet architecture has proven its worth by supporting multitude of distributed applications and a wide variety of network technologies over which it currently runs. However, its popularity has become the biggest impediment to its further growth. Due to its multi-provider nature, adopting a new architecture or modification of the existing one requires consensus among competing stakeholders. As a result, alterations to the Internet architecture have become restricted to simple incremental updates and deployment of new network technologies have become increasingly difficult [13, 107].

1.1 Network Virtualization

To fend off this ossification once and for all, *network virtualization* has been propounded as a diversifying attribute of the future inter-networking paradigm. Even though architectural purists view network virtualization as a means for evaluating new architectures, the pluralist approach considers virtualization as a fundamental attribute of the architecture itself [13]. By allowing multiple heterogeneous network architectures to cohabit on a shared physical substrate, network virtualization provides flexibility, promotes diversity, and promises security and increased manageability. They believe that network virtualization can eradicate the so-called *ossifying forces* of the current Internet that have restricted changes to incremental updates, and consequently stimulate innovation [13, 107].

To introduce flexibility, decoupling of the functionalities is a well-known principle in the computing literature. Similar approach has been propounded for virtualizing networks [107, 22, 46]. In this case, the role of the traditional ISPs has been divided into two independent entities: *infrastructure providers*, who manage the physical infrastructure,

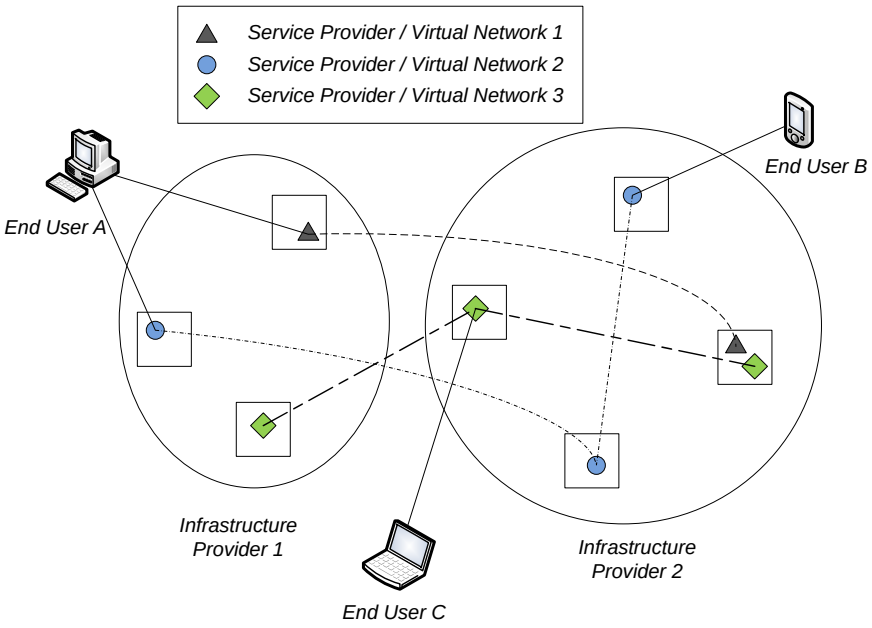

Figure 1.1: Network Virtualization Architecture

and *service providers*, who create virtual networks by aggregating resources from multiple infrastructure providers and offer end-to-end services.

Figure 1.1 depicts three possibly heterogeneous virtual networks that span over two different infrastructure provider domains. The owner of a virtual network, i.e., a service provider, is free to implement end-to-end services by selecting custom packet formats, routing protocols, forwarding mechanisms, and other control and management protocols. End users can opt-in to any virtual network, or even multiple ones at the same time. For example, end user A in Figure 1.1 is connected to two different virtual networks.

1.2 Challenges and Opportunities

While network virtualization promises extended flexibility and increased heterogeneity, several technical challenges in terms of the instantiation, operation, and management of this environment must be handled first to realize those pledges. Examples of instantiation related problems include interfacing, signaling, bootstrapping, and embedding of virtual networks on the shared physical infrastructure. On the other hand, implementation of virtual routers and virtual links as well as resource scheduling among coexisting virtual resources are a few of many operation related challenges. Finally, failure handling, mobility management, virtual network configuration and monitoring are some examples of the management problems in the network virtualization environment.

However, most of the existing research works related to network virtualization can at best be described as attempts to fix some existing problems, rather than a conscious and focused push to build a complete network virtualization environment. As a result, several aspects of network virtualization remain unexplored till today; many others, although touched, can use further improvement. This presents a broad range of open problems and unique opportunities, both theoretical and practical, to the researchers working in this area.

1.3 Contributions

The contributions of this thesis are threefold. First and foremost, we present a comprehensive survey of network virtualization and related research. We put network virtualization in a historical perspective, present a categorical overview of previous and on-going projects related to network virtualization, and most importantly, we present an elaborate enumeration of open challenges and opportunities in this research area with an aim to stoke wide interests among networking researchers.

From the open problems, we select two particular problems of the network virtualization environment as the main focus of this thesis. In order to manage identities in the heterogeneous network virtualization environment, we propose an identity management framework (*iMark*) that ensures end-to-end connectivity through interoperability between heterogeneous identifier spaces, allows flexibility of the choice of identifier spaces by different virtual networks, and enables mobility and Überhoming of the end hosts.

Finally, we present one deterministic (*D-ViNE*) and one randomized (*R-ViNE*) virtual network embedding algorithms based on a mathematical formulation of the embedding problem. Both the proposed algorithms outperform the existing embedding algorithms in terms of virtual network request acceptance ratio, utilization of resources, and cost as well as revenue of the infrastructure providers. We also present *D-ViNE-LB*, an improved and load-balanced version of the deterministic algorithm, that significantly increases the acceptance ratio, utilization, and revenue with relatively low increase of the provisioning cost.

1.4 Thesis Organization

The remainder of this thesis can be outlined as follows. Chapter 2 presents a review of the concepts similar to network virtualization and summarizes related past and ongoing projects. Chapter 3 provides a conceptual overview of network virtualization and enumerates open research challenges in this area. Chapter 4 motivates and defines the identity management problem and presents a framework to address related issues in this context. Chapter 5

addresses the most prevalent resource allocation problem in the network virtualization environment: virtual network embedding, and presents two algorithms that outperform the existing heuristics using improved correlation between the node mapping and the link mapping phases during the embedding process. Finally, Chapter 6 concludes this thesis with a summary of our contributions and possible future work.

Chapter 2

Literature Review

2.1 Introduction

The concept of multiple co-existing logical networks appeared in the networking literature several times in the past in different capacities. They can broadly be categorized into four main classes: *Virtual Local Area Networks (VLAN)*, *Virtual Private Networks (VPN)*, *active and programmable networks*, and *overlay networks*. A VLAN is a group of logically networked hosts with a single broadcast domain that provides the semblance of a physical LAN regardless of physical connectivity. A VPN, on the other hand, is a trunked VLAN that is a specialized virtual network connecting multiple distributed sites through tunnels over shared or public networks. An overlay network is yet another form of network virtualization which is typically implemented in the application layer, though various implementations at lower layers of the network stack do exist. It has been extensively used as a weak but effective tool to deploy new features and fixes in the Internet. Finally, active and programmable networks is a concept that enables customization of network elements through programmability based on service providers' requirements.

Capitalizing on these ideas, many projects had been established in the last decade that championed the core concept of coexisting, possibly heterogeneous, logical networks on a shared physical infrastructure. The goal of this chapter is to provide an overview as well as a historical perspective of network virtualization for a better understanding of this thesis.

2.2 Chapter Organization

The remainder of this chapter is organized as follows. Section 2.3 provides an overview of the four incarnations of network virtualization: VLANs, VPNs, active and programmable networks, and overlay networks. Following that, Section 2.4 discusses the past and the

ongoing projects directly and indirectly linked to network virtualization related concepts. Finally, Section 2.5 summarizes the discussion in tabular forms (Table 2.1 and Table 2.2).

2.3 Historical Perspective

2.3.1 Virtual Local Area Network

A virtual local area network (VLAN) [35] is a group of logically networked hosts with a single broadcast domain regardless of their physical connectivity. All frames in a VLAN bear a VLAN ID in the MAC header, and VLAN-enabled switches use both the destination MAC address and the VLAN ID to forward frames. Since VLANs are based on logical instead of physical connections, network administration, management, and reconfiguration of VLANs are simpler than that of their physical counterparts. In addition, VLANs provide elevated levels of trust, security, and isolation.

2.3.2 Virtual Private Network

A virtual private network (VPN) [48, 88, 89] is a dedicated network connecting multiple sites using private and secured tunnels over shared or public communication networks like the Internet. In most cases, VPNs connect geographically distributed sites of a single corporate enterprise.

Each VPN site contains one or more customer edge (CE) devices that are attached to one or more provider edge (PE) routers. Typically a VPN is managed and provisioned by a VPN service provider (SP), and it is known as provider-provisioned VPN (PPVPN) [14]. Based on the protocol used in the VPN data plane, PPVPN technologies can be classified into three broad categories:

Layer 3 PPVPN

Layer 3 VPN (L3VPN) [32, 29] is characterized by its use of Layer 3 protocols in the VPN backbone to carry data between the distributed CEs. There are two types of L3VPNs:

In *CE-based* VPN approach, the SP network is completely unaware of the existence of a VPN. CE devices create, manage, and tear up the tunnels between themselves. Tunneling requires three different protocols:

1. *Carrier protocol* (e.g. IP), used by the SP network to carry the VPN packets.
2. *Encapsulating protocol*, used to wrap the original data. It can range from very simple wrapper protocols (e.g. GRE [45], PPTP [56], L2TP [105]) to secure protocols (e.g. IPSec [67]).

3. *Passenger Protocol*, which is the original data in customer networks.

Sender CE devices encapsulate the passenger packets and route them into carrier networks; when these encapsulated packets reach the end of the tunnel, i.e., receiver CE devices, they are extracted and actual packets are injected into receiver networks.

On the other hand, all the states in *PE-based* L3VPNs are maintained in the PE devices, and a connected CE device may behave as if it were connected to a private network.

Layer 2 VPN

Layer 2 VPNs (L2VPNs) [15, 16] transport Layer 2 (typically Ethernet) frames between participating sites. The advantage is that it is agnostic about the higher-level protocols and consequently, more flexible than L3VPN. On the downside, there is no control plane to manage reachability across the VPN.

There are two fundamentally different kinds of Layer 2 VPN services that a service provider could offer to a customer: Virtual Private Wire Service (VPWS) and Virtual Private LAN Service (VPLS). A VPWS is a VPN service that supplies an L2 point-to-point service. A VPLS is a point-to-multipoint L2 service that emulates LAN service across a WAN. There is also the possibility of an IP-only LAN-like Service (IPLS), which is similar to VPLS except that CE devices are hosts or routers instead of switches and only IP packets are carried (either IPv4 or IPv6).

Layer 1 VPN

Layer 1 VPN (L1VPN) [19, 100] framework emerged in recent years from the need to extend L2/L3 packet-switching VPN concepts to advanced circuit-switching domains. It provides a multi-service backbone where customers can offer their own services, whose payloads can be of any layer (e.g., ATM, IP, and TDM). This ensures that each of the service networks has independent address space, independent Layer 1 resource view, separate policies, and complete isolation from other VPNs.

L1VPN can be of two types: Virtual Private Wire Services (VPWS) and Virtual Private Line Services (VPLS). VPWS services are point-to-point, while VPLS can be point-to-multipoint.

2.3.3 Active and Programmable Networks

Active and programmable networks research was motivated by the need to create, deploy, and manage novel services on the fly in response to user demands. But the ideas never materialized to real implementations due to concerns about technical feasibility and economical viability as well as lack of willingness from network operators.

Two separate schools of thought emerged on how to actually implement such concepts: one from telecommunications community and the other from IP networks community.

Open Signaling Approach

Open signaling [3] takes a telecommunication approach with a clear distinction between transport, control, and management planes that constitute programmable networks and emphasizes on QoS guarantees. An abstraction layer is proposed for physical network devices to act as distributed computing environments with well-defined open programming interfaces allowing service providers to manipulate network states.

Active Networks Approach

Active networks [102, 101, 116, 82] promote dynamic deployment of new services at runtime within the confinement of existing IP networks. Routers or switches in these networks can perform customized computations based on the contents of the *active* packets and can also modify them. Active networks allow the customization of network services at packet transport granularity and offer more flexibility than the open signaling approach at the expense of a more complex programming model.

Different levels of programmability have been suggested over the years. At one end, ANTS [112] offers a Turing-complete machine model at the active router enabling each user to execute any new code. At the other end of the spectrum, DAN [40] only allows the user to call functions already installed at a particular node. Calvert et al. [30] classify the proposed architectures based on the granularity of control, statefullness, and language expressive power.

2.3.4 Overlay Networks

An overlay network is a logical network built on top of one or more existing physical networks. The Internet itself started off as an overlay on top of the telecommunication network. Overlays in the existing Internet are typically implemented in the application layer; however, various implementations at lower layers of the network stack do exist.

Overlays do not require, nor do they cause any changes to the underlying network. As a consequence, overlays have long been used as relatively easy and inexpensive means to deploy new features and fixes in the Internet. A multitude of application layer overlay designs have been proposed in recent years to address diverse issues, which include: ensuring performance [91] and availability [10] of Internet routing, enabling multicasting [41, 62, 33], providing QoS guarantees [97], protecting from denial of service attacks [68, 11], and for

content distribution [72], file sharing [75] and even in storage systems [39]. Overlays have also been used as testbeds (e.g., PlanetLab [5]) to design and evaluate new architectures.

However, Anderson et al. [13] point out that standard overlays falter as a deployment path for radical architectural innovation in at least two ways. First, overlays have largely been seen as a way to deploy narrow fixes to specific problems without any holistic view of the interactions between different overlays. Second, most overlays have been designed in application layer on top of IP and hence are not capable of supporting radically different concepts.

2.4 Network Virtualization Projects

Over the years, the term *"virtual network"* has been used to describe different projects on virtual private networks, overlay networks, and active or programmable networks. But very few of them actually followed the pluralist view of network virtualization. We present an overview of a number of virtual network architectures and related projects (e.g. overlay, programmable network or VPN inspired designs) that have emerged in the literature in this section.

2.4.1 Characteristics

We summarize the most significant past and ongoing projects directly or indirectly related to network virtualization based on the following set of characteristics:

- ***Networking technology*:** A handful of network virtualization prototypes have been developed for specific networking technologies with an aim to exploit unique characteristics of those networks to enable virtualization. Such projects include, X-Bone for IP networks, Tempest targeting ATM networks, and the very recent GENI initiative that will be agnostic to any specific technology.
- ***Layer of virtualization*:** Influenced by the presence of the existing Internet, researchers have naturally approached network virtualization in a layered manner. As a result, many projects have attempted to virtualize different layers of the network stack, starting from the physical layer (UCLP) and continuing up to the application layer (VIOLIN).
- ***Architectural domain*:** Most projects have focused on particular architectural domains, which dictates the design choices taken in the construction of architectures and services that can be offered on those platforms. Examples include, network management (VNRMS), virtual active networks (NetScript), and spawning networks (Genesis).

- ***Level of virtualization*:** To enable network virtualization, one must virtualize the nodes, the links, and every other resource in the network. The level of virtualization refers to the granularity at which each virtual network can administer itself. At one end of this spectrum, node virtualization creates VNs by connecting virtual machines on different nodes (e.g., PlanetLab). At the other end, CABO proposes the concept of true plurality where each VN has a semblance of the native network.

2.4.2 Networking Technology

IP Networks: X-Bone

X-Bone [103] was first proposed as a system for rapid and automated deployment and management of overlay networks using encapsulation to enable virtual infrastructure. Later this idea was extended to the concept of *Virtual Internet (VI)* [104], which is an IP network composed of tunneled links among a set of virtual routers and hosts, with dynamic resource discovery, deployment, and monitoring support.

A VI virtualizes all the components of the Internet: hosts, routers and links between them. A single network node may participate as a virtual host (VH), a virtual router (VR), or multiple of them simultaneously in a VI. All components participating in the VI must support multihoming, since even a base host with a single VH is necessarily a member of at least two networks: the Internet and the VI overlay. Addresses within each VI is unique and can be reused in another overlay, unless there is no shared common node in the underlying network between the two VIs.

VIs completely decouple the underlying physical network from the overlays and multiple VI can *coexist* together. VIs also support control recursion to allow divide-and-conquer network management and network recursion to stack one VI on another.

ATM Networks: Tempest

Tempest [109] is a network control architecture that allows multiple heterogeneous control architectures to run simultaneously over single physical ATM network. It is defined as a set of policies, algorithms, mechanisms, and protocols to control and manage various devices on the network.

Tempest is based on the concept of *switchlets* [108], which allows a single ATM switch to be controlled by multiple controllers by strictly partitioning the resources of that switch between those controllers. The set of switchlets that a controller or group of controllers possess forms its virtual network. Third parties can lease such virtual networks from the Tempest network operator to use them for any purpose as they see fit.

Programmability in Tempest is supported at two levels of granularity: first, switchlets support the introduction of alternative control architectures in the network; and second, services can be refined by dynamically loading programs into the network that customize existing control architectures. This allows the users to have *application-specific* control.

2.4.3 Layer of Virtualization

Physical Layer: UCLP

UCLP [6] is a distributed network control and management system for CA*NET 4 network that allows end users to treat network resources as software objects, and lets them provision as well as dynamically reconfigure optical networks (at Layer 1). Users are able to join or divide lightpaths within a single domain, or across multiple independent management domains to create customized logical IP networks.

UCLP takes a modular approach to resource management by introducing three distinct service layers [24, 25, 114]. Customers and administrators configure and use end-to-end UCLP resources through the *user access layer*. The *service provisioning layer* is managed with a grid application. Finally, the *resource management layer* deals with actual physical resources.

UCLPv1.4 [86] introduced dynamic topology discovery process and enabled auto-routing through intelligent algorithms alongside already available manual lightpath configuration capabilities. Later, UCLPv2 [79] extended UCLP with the use of Service Oriented Architecture (SOA) and workflow technologies with an aim to form the underpinning architectural framework for extending UCLP to allow the interconnection of instruments, time slices, and sensors; and for incorporating virtual routers and switches.

Link Layer: VNET

VNET [9, 98] is a Layer 2 overlay network for virtual machines (VMs) that implements a virtual LAN (VLAN) spread over a wide area using Layer 2 tunneling protocol (L2TP). Each physical machine hosting a virtual machine (VM) runs a VNET process that intercepts VM traffic and tunnels it to the appropriate destination. The destination is either another VM that can be contacted directly through VNET or an address external to the overlay. Traffic destined for an external address is routed through the overlay to a VNET proxy node, which is responsible for injecting the packets onto the appropriate network. The overlay thus consists of a set of TCP connections or UDP peers (VNET links) and a set of rules (VNET routes) to control routing on the overlay.

Since VNET operates at Layer 2, it is agnostic to Layer 3. As a result, protocols other than IP can be used. In addition, VNET also supports migration of a VM from one machine to

another without any participation from the VM's OS and all connections remain open after migration.

Network Layer: AGAVE

The main objective of the AGAVE [22, 23, 110] project is to provide end-to-end QoS-aware service provisioning over IP networks following the theme of QoS forwarding mechanisms such as IntServ [26] and DiffServ [21]. To achieve this, AGAVE proposes a new inter-domain architecture based on the novel concept of Network Planes (NPs), which allows multiple IP Network Providers (INPs) to build and provide Parallel Internets (PIs) tailored to end-to-end service requirements.

NPs are internal to INPs and are created based on the service requirements described by the SPs. An NP can be engineered for routing, forwarding, or resource management. To enable end-to-end services over multi-provider environment, NPs from different INPs are connected together to form PIs based on inter-INP agreements. One of the interesting feature of AGAVE is that it does not require all the NPs participating in a PI to be homogeneous resulting in greater flexibility.

Application Layer: VIOLIN

VIOLIN [64, 90] is an application-level virtual network architecture, where isolated virtual networks are created in software on top of an overlay infrastructure(e.g., PlanetLab). Capitalizing on the advances in VM technologies, VIOLIN extends the idea of single node isolation in VMs to provide completely isolated virtual networks.

A VIOLIN consists of virtual routers (vRouters), LANs (vLANs) and end hosts (vHosts), all being software entities hosted by overlay hosts. Both vHosts and vRouters are virtual machines running in physical overlay hosts. A vLAN is created by connecting multiple vHosts using virtual switches (vSwitches), while vRouters connect multiple vLANs to form the total network.

VIOLIN provides network isolation with respect to: 1) administration, 2) address space and protocol, 3) attack and fault impact, and 4) resources. The combined effect is a confined, secured, and dedicated environment that can be used to deploy untrusted distributed applications and perform risky network experiments.

2.4.4 Architectural Domain

Network Management: VNRMS

VNRMS [65, 66, 80] is a flexible and customizable virtual network (VN) management architecture, which provides a programmable networking environment to generate multiple levels of virtual networks through nesting from a single physical network (PN). A virtual network is composed of several virtual network resources (VNRs), where each VNR is a subset of a physical network resource (PNR) in the underlying network. VNRMS lets the customers to customize the VNRs through active *resource agents* using a customer-based management system (CNRMS). While the provider VNRMS has access to all the resource agents, a customer can access only those that belong to its VN.

In order to allow a CNRMS to manage only a subset of resources in a PNR, the management information base (MIB) of that PNR is logically partitioned into multiple disjoint MIBs, known as MIBlets [81]. MIBlets provide *abstract* and *selective* views of the resources that are allocated to a particular VN. An abstract view hides the details of the resource interface that are not relevant to the CNRMS. A selective view restricts the CNRMS to access only the resources allocated to it.

Virtual Active Networks: NetScript

NetScript [38] is a language system for dynamically programming and deploying protocol software in an active network. It is a strongly typed language that creates universal language abstractions to capture network programmability. Unlike other active network architectures, where packets contain active programs, NetScript packets are passive. These packets are processed by protocol software or hardware when they flow through the network. In this architecture, active packet processing applications and standardized protocols can be composed together, interoperate, and utilize each other's services. Consequently, NetScript can be used to systematically compose, provision, and manage virtual active network abstractions [37].

NetScript supports creation of arbitrary packet formats, dynamic composition of standard and active protocol, and can operate on any type of packet stream. NetScript communication abstractions consider network nodes as collections of Virtual Network Engines (VNEs) interconnected by Virtual Links (VLs) that constitute NetScript Virtual Networks (NVNs) [115].

Spawning Networks: Genesis

The Genesis Kernel [71] is a *spawning network* [73, 31], a variant of open programmable networks, that automates the life cycle process for the creation, deployment, management, and designing of network architectures. It allows multiple heterogeneous child virtual networks to operate on top of subsets of their parent's resources, and provides isolation among them. The Genesis Kernel also supports nesting of virtual networks and inheritance of architectural components from parent to child networks.

A virtual network in the Genesis Kernel is characterized by a set of routelets interconnected by a set of virtual links. Routelets represent the lowest level of operating system support dedicated to a virtual network, and are designed to operate over a wide variety of networking technologies including IP and ATM technology. They process packets along a programmable data path at the internetworking layer, while virtual network kernel makes control algorithms support programmability.

2.4.5 Level of Virtualization

Node Virtualization: PlanetLab

PlanetLab [5, 83, 95] is an overlay-based testbed that was developed to design, evaluate, and deploy geographically distributed network services with support for researchers and users. Its goal is to create a *service-oriented network architecture* combining the best of both the distributed systems community and the networks community.

PlanetLab is built upon four design principles. First, it supports *sliceability*. That is, each application acquires and runs in a slice of the overlay. Virtual machine monitors (VMMs) running on each node allocate and schedule slices of the nodes' resources to create a distributed virtualized environment. Second, it supports a highly decentralized control structure, enabling nodes to act according to local policies. Third, overlay management is divided into sub-services that run on their own slices, instead of a centralized one. Finally, overlay supports an existing and widely adopted programming interface, with internal changes over time keeping the API intact, to promote actual long-term service development instead of just being a temporary testbed.

GENI

Based on the experience accumulated from using PlanetLab and other similar testbeds, the Global Environment for Network Innovations (GENI) [1, 54] is a major planned initiative of the US National Science Foundation (NSF) to build an open, large-scale, realistic experimental facility for evaluating new network architectures, carrying real traffic on behalf of end

users, and connecting to the existing Internet to reach external sites. The purpose of GENI is to give researchers the opportunity to create customized virtual network and experiment unfettered by assumptions or requirements of the existing Internet.

Main design goals of GENI [54] include: sliceability to share resources, generality to give an initial flexible platform for the researchers, fidelity, diversity and extensibility, wide deployment and user access for testing and evaluation purposes as well as actual use of deployed services and prototypes, controlled isolation and monitoring facilities.

GENI proposes virtualization in the form of slices of resources in space and time. If resources are partitioned in time, a given resource might not sustain real user workload, thereby limiting its feasibility for deployment studies. On the other hand, if resources are partitioned in space, only a limited number of researchers might be able to include a given resource in their slices. In order to maintain balance, GENI proposes to use both types of virtualization based on resource type. If sufficient capacity is available to support deployment studies, GENI uses time-based slicing; otherwise, it partitions resources in space to support a handful of high priority projects instead of making those resources available to everyone.

VINI

VINI [8, 18] is a virtual network infrastructure allowing network researchers to evaluate their protocols and services in a realistic environment with high degree of control. It can be viewed as an extension to PlanetLab toward GENI, that will be able to provide infrastructure like PlanetLab along with the support for virtual networks as in X-Bone or VIOLIN.

VINI offers more latitude to researchers than PlanetLab at routing level. It provides the ability to create real complex networks and to inject exogenous events to create more realistic alternative to simulation and emulation of proposed network architectures.

Initial prototype of VINI (PL-VINI) was implemented on PlanetLab by synthesizing a collection of available software components. It can be considered as a specific instantiation of an overlay network that runs software routers and allows multiple such overlays to exist in parallel. In particular, it used *XORP* for routing [57], *Click* for packet forwarding and network address translation [69], and *OpenVPN* servers to connect with end users [4].

Recently a software platform for hosting multiple virtual networks on shared physical network infrastructure, Trellis [20], has been developed. Trellis synthesizes container-based virtualization technologies together with a tunneling mechanism into a coherent platform to achieve the following design goals: performance, scalability, flexibility, and isolation. It allows each virtual network to define its custom topology, routing protocols, and forwarding tables.

Full Virtualization: CABO

At present, Internet service providers (ISPs) manage their network infrastructure as well as provide network service to end users. Adopting a new architecture not only requires change in hardware and host software, but also it requires that ISPs jointly agree on any architectural change [13]. CABO [46] is one of the very recent proposals toward network virtualization that promotes separation between infrastructure providers and service providers to end this deadlock. CABO exploits virtualization to allow service providers to simultaneously run multiple end-to-end services over equipment owned by different infrastructure providers.

To allow multiple virtual networks to share the same physical equipments, CABO virtualizes the nodes and the links. Virtual nodes are connected using virtual links to form a virtual network. These virtual nodes are created by the service providers and hosted by infrastructure providers' equipments using a subset of available resources. Similarly, virtual links are formed from a path in the underlying physical network and include portion of the resources along the path.

CABO has introduced and achieved some significant developments in terms of virtual routers and routing in virtualized networks in general. It supports automatic migration of virtual routers from one physical node to another [111] using migration technologies in the underlying virtual machines. It also proposes a new multi-layer routing scheme that is scalable as well as quick to react to any changes in network conditions [120]. In supporting programmable routers, CABO resembles the theme introduced in active networks research, except that it does not enable users to program the network; rather service providers can customize their networks to provide end-to-end service to the end users.

2.5 Summary

Amid current trends of virtualizing practically every aspect of computing, ranging from operating systems, storage systems to servers, and even large data centers (e.g., cloud computing), network virtualization stands at a unique point in the virtualization design space. In one hand, it is necessary to have a virtualized network to interconnect all other virtualized appliances to give each of the virtual entities a complete semblance of their native counterparts.

On the other hand, after enjoying years of rapid growth, the progress of the Internet and networking in general has come to a standstill. Most researchers now agree that a redesign is a bare necessity, not luxury [47]. Network virtualization can take the leading role in this scenario to promote innovation, to provide flexibility, and to introduce heterogeneity. From Table 2.1 it is evident that over time research regarding network virtualization has shifted focus toward this same direction of creating a holistic and generalized network virtualiza-

Table 2.1: Characteristics of Different Network Virtualization Projects

Project	Architectural Domain	Networking Technology	Layer of Virtualization	Level of Virtualization	References
VNRMS	Virtual network management	ATM/IP		Node/Link	[65, 66, 80]
Tempest	Enabling alternate control architectures	ATM	Link		[109, 108]
NetScript	Dynamic composition of services	IP	Network	Node	[38, 37]
Genesis	Spawning virtual network architectures		Network	Node/Link	[71, 73, 31]
VNET	Virtual machine Grid computing		Link	Node	[9, 98]
VIOLIN	Deploying on-demand value-added services on IP overlays	IP	Application	Node	[64, 90]
X-Bone	Automating deployment of IP overlays	IP	Network	Node/Link	[103, 104]
PlanetLab	Deployment and management of overlay-based testbeds	IP	Application	Node	[5, 83]
UCLP	Dynamic provisioning and reconfiguration of lightpaths	SONET	Physical	Link	[6, 86, 79]
AGAVE	End-to-end QoS-aware service provisioning	IP	Network		[22, 23, 110]
GENI	Creating customized virtual network testbeds	Heterogeneous			[1, 54]
VINI	Evaluating protocols and services in a realistic environment		Link		[8, 18]
CABO	Deploying value-added end-to-end services on shared infrastructure	Heterogeneous		Full	[46]

Table 2.2: Recent Network Virtualization Related Projects

Project	Originated In	Link
4WARD	Europe	http://www.4ward-project.eu/
AKARI	Japan	http://akari-project.nict.go.jp/
CABO	USA	http://www.cs.princeton.edu/~jrex/virtual.html
Clean Slate	USA	http://cleanslate.stanford.edu/
GENI	USA	http://www.geni.net/
NouVeau	Canada	http://netlab.cs.uwaterloo.ca/virtual/
PlanetLab	USA	http://www.planet-lab.org/
Trilogy	Europe	http://www.trilogy-project.org/
UCLP	Canada	http://www.uclp.ca/
VINI	USA	http://www.vini-veritas.net/

tion environment that features a completely virtualized (virtualization of all the network elements), highly customizable (virtualization at lower layers), and technology-agnostic (creation of virtual networks over heterogeneous combination of underlying networks) networking facility for the future Internet. And all these research works have finally culminated into projects all over the world that are directly or indirectly related to network virtualization (Table 2.2).

Chapter 3

Network Virtualization: Concepts and Challenges

3.1 Introduction

Unlike the existing all-IP Internet, a virtualized networking environment is a collection of multiple heterogeneous network architectures from different network service providers. Each service provider leases resources from one or more infrastructure providers to create virtual networks and deploys customized protocols to deliver end-to-end services to the end users. By decoupling service providers from infrastructure providers, network virtualization introduces much sought flexibility for the researchers. From a commercial point of view, this decoupling amortizes high fixed cost of maintaining a physical presence by sharing capital and operational expenditure across multiple infrastructure providers.

However, several technical challenges in terms of instantiation, operation, management, and interactions must be resolved to realize such an environment. Most of the existing research works related to network virtualization can at best be described as attempts to fix some existing problems, rather than a conscious and focused push to build a complete network virtualization environment. This presents a broad range of open problems and unique opportunities, both theoretical and practical, to the researchers working in this area.

3.2 Chapter Organization

This chapter is divided into two major parts: one about the present and the other about the future. Section 3.3 presents a conceptual overview of the network virtualization environment along with its architectural principles and design goals that we follow throughout this work. After that, Section 3.4 delivers an enumeration of the future research challenges in the

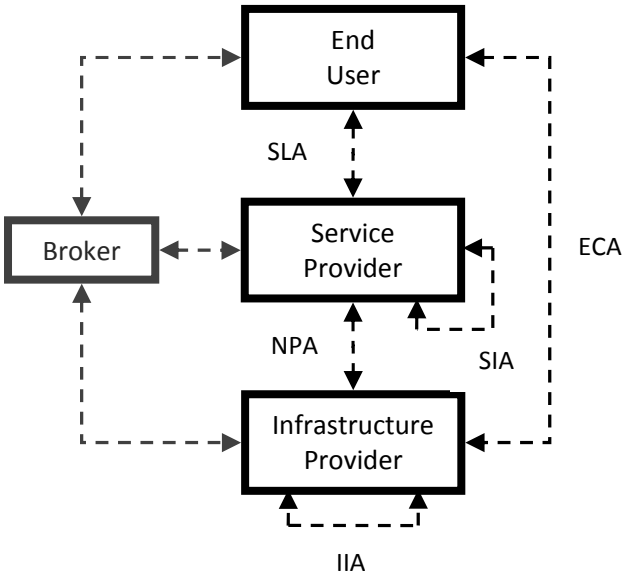

Figure 3.1: Network Virtualization Business Model: Relationship between Players

network virtualization landscape by broadly classifying the problems into several categories, and it acts as a reference point for our choice of addressed problems in this thesis.

3.3 Network Virtualization Environment (NVE)

3.3.1 Reference Business Model

The main distinction between the participants in the network virtualization model (Figure 3.1) and the traditional model is the presence of two different roles: infrastructure providers and service providers, as opposed to the single role of the ISPs [107, 23, 46]. It should be noted that business roles do not necessarily map one-to-one to distinct business entities (i.e., any business entity can assume multiple roles).

1. **Infrastructure Provider (InP):** Infrastructure providers deploy and actually manage the underlying physical network resources in the network virtualization environment. They are in charge of the operations and maintenance of the physical infrastructure and offer their resources through programmable interfaces to different service providers. Infrastructure providers distinguish themselves through the quality of resources they provide, the freedom they delegate to their customers (i.e. service providers), and the tools they provide to exploit that freedom.
   Multiple InPs communicate and collaborate, based on InP interconnection agreements (IIAs), to create end-to-end physical infrastructure. Those who offer connectivity

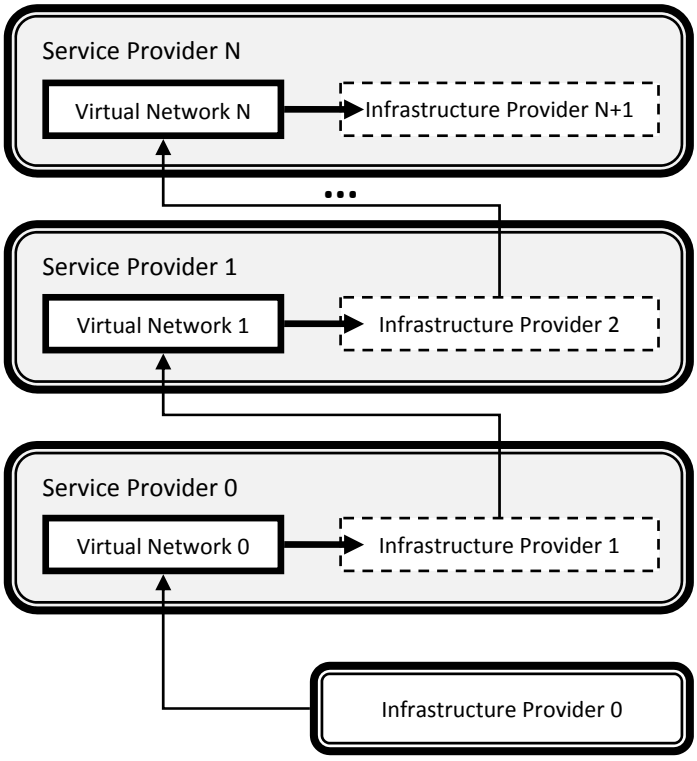

Figure 3.2: Hierarchy of Roles

to service providers through different networking technologies, e.g., optical fiber, satellite etc., are known as *facilities providers*. On the other hand, infrastructure providers connecting customer premise equipments (CPEs) to the core network are the *access providers* [22].

2. **Service Provider (SP):** Service providers lease resources from multiple facilities providers to create virtual networks and deploy customized protocols, if required, by programming the allocated network resources to offer end-to-end services to end users. Relationship between service providers and infrastructure providers are regulated by *network provisioning agreements (NPAs)*.
   SPs can have peering relationship between themselves on the basis of *SP interconnection agreements (SIAs)*. An SP can also create child virtual networks by partitioning its resources. It can then lease those child networks to other SPs, virtually taking the role of an infrastructure provider creating a hierarchy of roles (Figure 3.2).

3. **End User:** End users in the network virtualization environment are similar to the end users in the existing Internet, except that the existence of multiple virtual networks from competing service providers enables them to choose from a wide range of

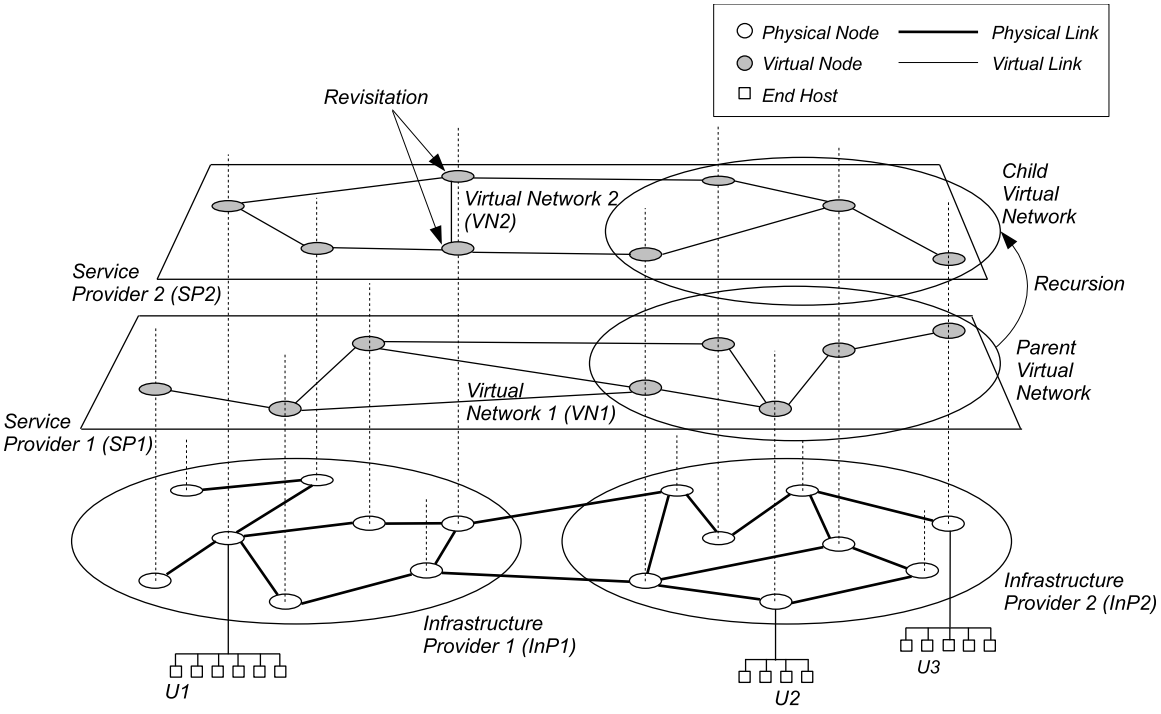

Figure 3.3: Network Virtualization Architecture Overview

services. Any end user may simultaneously connect to multiple service providers for different services. Services are offered on the basis of terms and conditions defined in *service level agreements (SLAs)* between the service providers and the customers. End users connect to the physical infrastructure through *end user connectivity agreements (ECAs)* with the access providers.

4. **Broker:** Brokers play a pivotal role in the network virtualization economy. They act as mediators between InPs, SPs, and end users in the network virtualization marketplace. SPs buy (lease) resources from InPs to create VNs and sell services deployed on those VNs to interested end users through brokers. Their presence simplifies the process of matching SPs' requirements to available resources by aggregating offers from multiple InPs. Similarly, they also allow end users to select desirable services from a wide range of SPs. It should be noted that a marketplace can also be formed without any broker or mediator through peering relationship between concerned parties.

3.3.2 Architectural Overview

In the *network virtualization environment* (NVE), the basic entity is a *virtual network* (VN). A VN is a collection of virtual nodes connected together by a set of virtual links to form

a virtual topology, which is essentially a subset of the underlying physical topology. Each virtual node is hosted on a particular physical node, whereas a virtual link spans over a path in the physical network and includes a portion of the network resources along the path.

Each VN is composed and managed by a single SP, even though the underlying physical resources might be aggregated from multiple InPs. Figure 3.3 depicts two virtual networks, VN1 and VN2 created by the service providers SP1 and SP2, respectively. SP1 composed VN1 on top of the physical resources managed by two different infrastructure providers (InP1 and InP2) and provides end-to-end services to the end users U2 and U3. SP2, on the other hand, deployed VN2 by combining resources from infrastructure provider InP1 with a child VN from service provider SP1. End users U1 and U3 are connected through VN2.

The owner of a VN is free to implement end-to-end services by selecting custom packet formats, routing protocols, forwarding mechanisms, as well as control and management planes. As mentioned earlier, end users have the choice to opt-in to any VN. For example, end user U3 is subscribed to two virtual networks VN1 and VN2 managed by SP1 and SP2, respectively.

3.3.3 Architectural Principles

Network virtualization propounds the following principles for the next-generation networking paradigm: *coexistence* of multiple heterogeneous VNs to introduce diversity; *recursion* of virtual resources to enable reselling; *inheritance* of architectural attributes to promote value-addition; and finally, *revisitation* to simplify network operations and management.

1. **Coexistence:** Coexistence of multiple VNs is the defining characteristic of the NVE [13, 107, 46]. It refers to the fact that multiple VNs from different service providers can coexist together, spanning over part or full of the underlying physical networks provided by one or more infrastructure providers. VN1 and VN2 in Figure 3.3 are examples of two coexisting VNs.

2. **Recursion:** When one or more VNs are spawned from another VN creating a virtual network hierarchy with *parent-child* relationships, it is known as recursion as well as *nesting* of virtual networks [71]. In Figure 3.2, 'Service Provider 0' has created a VN on top of an actual physical network provided by 'Infrastructure Provider 0', and has leased away a portion of the allocated resources to 'Service Provider 1', to whom it appears as 'Infrastructure Provider 1'. This hierarchical construct can continue until cumulative overhead of creating child VNs makes further subdivision impossible.

3. **Inheritance:** Child VNs in the NVE can inherit architectural attributes from their parents, which also means that the constraints on the parent VN automatically translate to similar constraints on its children [71]. For example, in Figure 3.3,

constraints imposed by InP2 will automatically be transferred to VN2 from VN1 through inheritance. Inheritance allows a SP to add value to the spawned child VNs before reselling them to other SPs [46].

4. **Revisitation:** Revisitation [104] allows a physical node to host multiple virtual nodes of a single VN. Use of multiple logical routers to handle diverse functionalities in a large complex network allows a SP to logically rearrange its network structure and to simplify the management of a VN. Revisitation can also be useful for creating testbed networks. Figure 3.3 provides an example of revisitation in the virtual network VN2.

3.3.4 Design Goals

The overall goal of enabling multiple heterogeneous virtual networks to coexist together on a shared physical infrastructure can be subdivided into several smaller objectives. In order to materialize a viable network virtualization environment, each of these design goals must be fulfilled. These goals also provide guidelines for designing protocols and algorithms for virtual networks.

- **Flexibility:** Network virtualization must provide flexibility at every aspect of networking. Each SP should be able to use arbitrary network topology, routing or forwarding functions as well as customized control protocols independent of the underlying physical network and other coexisting VNs.
  For example, deploying source routing in today's Internet is immensely difficult because of the lack of consensus among the ISPs; in a virtualized environment, the owner of a VN should be able to offer source routing without having to coordinate with any other parties.

- **Manageability:** By separating SPs from InPs, network virtualization will modularize network management tasks and introduce accountability at every layer of networking [46, 23, 119, 107]. InPs will be in total control of the management and operations of physical entities in the network and provide access to resources. SPs, on the other hand, will lease subsets of resources from different infrastructure providers, create virtual networks on top of the allocated resources following specific policies, and provide actual services to end users. This separation of accountability will provide complete, end-to-end control of the VNs to the SPs obviating the requirement of coordination across administrative boundaries as seen in the existing Internet.

- **Scalability:** Coexistence of multiple networks is one of the fundamental motivations behind network virtualization. Scalability comes as an indispensable part of this equation. InPs should try to maximize the number of coexisting VNs without affecting

their performance. This will increase utilization of resources and amortize capital expenditure (CAPEX) and operational expenditure (OPEX) of individual VNs.

- **Isolation:** Network virtualization must ensure complete logical and physical isolation between co-existing VNs to improve fault-tolerance, security, and privacy. Network protocols are often misconfigured and subject to implementation errors. Virtualization must ensure that misconfigurations in one VN are contained within itself and do not affect other co-existing VNs.
- **Stability and Convergence:** Isolation ensures that faults in one VN do not affect other coexisting VNs, but errors and misconfigurations in the underlying physical network can also destabilize the NVE. Moreover, instability in the InPs (e.g., routing oscillation) can lead to instability of all the hosted VNs. Virtualization must ensure the stability of the NVE, and in case of any instability the affected VNs must be able to successfully converge to their stable states.
- **Programmability:** To ensure flexibility and manageability, programmability of the network elements is an indispensable requirement. Only through programmability, SPs can implement customized protocols and deploy diverse services. Hence, two pressing questions: *"how much programmability should be allowed"*, and *"how it should be exposed"* must have satisfactory answers. A win-win situation must be found where programmability is easy, effective, as well as secure at the same time.
- **Heterogeneity:** Heterogeneity in the context of network virtualization comes mainly from two fronts: first, heterogeneity of the underlying networking technologies (e.g., optical, wireless, sensor etc); second, each end-to-end VN, created on top of that heterogeneous combination of underlying networks, can also be heterogeneous. SPs must be allowed to compose and run cross-domain end-to-end VNs without the need for any technology specific solutions. Underlying infrastructures must also be capable of supporting heterogeneous protocols and algorithms implemented by different SPs. In addition, heterogeneity of end user devices must also be taken into account.
- **Experimental and Deployment Facility:** Before deployment, any geographically distributed network service is typically designed and evaluated in test labs under controlled environment. Since it is very expensive to mimic a production network, tests are limited to simple topologies and traffic patterns that do not necessarily represent the real-world environment. Moreover, migration of a network to a different condition can also be extremely painstaking. By developing the service in a separate virtual network from the very beginning can effectively alleviate these problems. In addition, deploying new end-to-end services could not be more easier than deploying it on a separate virtual network of its own [5, 8].

- **Legacy Support:** Legacy support or backward compatibility has always been a matter of deep concern while deploying any new technology. Conceptually, network virtualization can easily integrate legacy support by considering the existing Internet as just another VN into its collection of networks. This will ensure that the existing distributed applications, services, and technologies need not be changed and redeveloped overnight; instead, we can keep using them until their counterparts are available or they are ported to the newer networks.
  For example, use of IPv6 would have been much faster if it could be implemented in another virtual network without having to deal with the preeminent IPv4.

3.4 Key Research Directions

Most of the existing research works related to network virtualization can at best be described as attempts to fix existing problems, rather than a conscious and focused push to build a complete NVE. As a result, several aspects of network virtualization remain unexplored till today, and many others require modification and improvement. In this section, we summarize the key issues to be resolved for the successful realization of the NVE by broadly categorizing them into four different classes: *instantiation*, *operation*, *management*, and *interactions*.

3.4.1 Instantiation

Interfacing

Every InP must provide an interface, following some standard, so that SPs can communicate with them and express their requirements. In addition, standard interfaces are also required to make programmability of the network elements available to the SPs. On a similar note, appropriate interfaces between end users and SPs, as well as among multiple InPs, and among SPs must also be identified and standardized.

Signaling and Bootstrapping

Before creating a VN, an SP must already have network connectivity to the InPs in order to issue its requests. This introduces circularity where network connectivity is a prerequisite to itself [46]. As long as the NVE is not mature enough to support itself, signaling must be handled by other means of communication, e.g. the current Internet. There must also be bootstrapping capabilities to allow SPs to customize the virtual nodes and virtual links allocated to them through appropriate interfaces. Both requirements call for at least another

network that will always be present to provide connectivity to handle these issues, or an *out-of-band* mechanism to perform signaling and bootstrapping.

Admission Control and Usage Policing

In order to uphold QoS guarantees, InPs must ensure that resources are not overbooked to SPs. Consequently, they have to perform accurate accounting and implement admission control algorithms to ensure that resources allocated to the VNs do not exceed the physical capacity of the underlying network. Instead of performing admission control for individual nodes or links as in the current Internet, admission control in this context must be performed on virtual networks.

In order to avoid constraint violations by globally distributed VNs, distributed policing mechanisms must be employed to make sure that SPs cannot overflow the amount of resources allocated to them by direct or indirect means. Raghavan et al. [85] present such a global rate limiting algorithm coordinated across multiple sites in the context of cloud-based services in the existing Internet. Similar concepts need to be developed in the context of network virtualization too.

Virtual Network Embedding or Mapping

Since a virtual link may span over multiple physical links, there may be many possible mappings for any given VN. In order to maximize the number of co-existing VNs, it is very important to determine how to embed a SP's request onto the physical network. But the embedding problem, with constraints on nodes and links, can be reduced to the $\mathcal{NP}$-hard *multi-way separator problem* [12] even when all the requests are known in advance.

Existing heuristic-based solutions can broadly be categorized into two major categories based on the offline and online versions of the problem they deal with. In the offline problem, all the SPs' requests are known in advance. Zhu and Ammar [119] aim at achieving load balancing in the underlying physical infrastructure assuming unlimited resources. Lu and Turner [74] provide a solution for mapping only one VN with an aim to minimize cost. Other solutions for the offline problem based on multi-commodity flow exist in the VPN context [99, 55].

For the online problem, Fan and Ammar [43] present a solution for determining dynamic topology reconfiguration for service overlay networks with dynamic communication requirement. Zhu and Ammar [119] handle the problem by calculating the whole mapping periodically. In both solutions, it is assumed that infrastructure resource is unlimited. Yu et al. [117] take a different approach by assuming that path splitting is supported by the underlying network, and employ path migration periodically to re-optimize the utilization of

the InPs' resources. Some of the mentioned algorithms also consider admission control as an integral part of the solution.

Even though various constraints and objectives make this problem computationally intractable, presence of multifarious topologies and possible opportunities to exploit them still leave enough room for research on customized solutions and better approximation algorithms.

3.4.2 Operation

Virtual Nodes

Virtual nodes allow multiple SPs to share the same set of physical resources and implement separate customized control protocols on them. Up until now, router vendors have promoted virtual nodes as a tool for simplifying core network design, decreasing capital expenditure (CAPEX), and for VPN purposes [42]. Similar concept can be extended with programmability to create substrate routers that will allow each service provider to customize their virtual nodes. A conceptual construct of such substrate routers can be found in [107].

Scalability of the NVE is closely tied to the scalability of the physical elements used by the InPs. Commercial router vendors have already implemented routers that can hold multiple logical routers [70]. Fu and Rexford [51] present a mechanism that improves scalability by capitalizing on the commonality of address prefixes in multiple FIBs from different virtual routers to decrease memory requirements and lookup times. Research in this direction should focus on increasing the number of virtual nodes any single physical router can hold.

Performance of virtual routers on existing virtual machine systems should also be explored. Specifically, how different system virtualization techniques, e.g. full virtualization, or paravirtualization, affect the performance requires serious attention. Design and performance of virtual routers implemented on top of Xen virtual machine systems as well as the impact of current multi-core processors on their performance has been studied in [76].

To increase network manageability, and to handle network failures to some extent, migration of virtual routers can be an effective solution [111]. But probable destinations of a migrating virtual router are restricted by some physical constraints, like change of latency, link capacity, platform compatibility issues, and even capabilities of destination physical routers. The obvious question: *"how to cope with these issues"*, remains open to be answered.

Virtual Links

To realize network virtualization, links between virtual nodes must also be virtualized. The ability to create tunnels over multiple physical links already exists in the context of VPNs. Similar tunneling mechanisms can also be used in case of VNs.

The speed of transporting packets across a virtual link should be comparable to that of a native link, which translates into minimum encapsulation and multiplexing cost. In addition, link scheduling algorithms must also be considered to better utilize the idle periods in virtual links. Finally, virtual links must also be flexible enough to carry packets of any protocol.

Naming and Addressing

While network virtualization provides immense flexibility in terms of creation and deployment of radical technologies, such flexibility does not come without cost. Due to the potential heterogeneity of the coexisting networks, end-to-end communication and universal connectivity in the NVE becomes a major challenge. In the NVE, it will be very hard to keep a system like DNS working due of scalability concerns and administrative issues. Same is true for addressing. Mapping between different address contexts is a well-known problem in current literature. But in the presence of different, often incompatible, addressing requirements in heterogeneous virtual networks the problem gets more complicated.

In the NVE, any end user can simultaneously connect to multiple virtual networks through multiple infrastructure providers using heterogeneous technologies to access different services. We refer to this phenomena by über-homing. Any naming framework for the network virtualization environment must provide additional level of indirection to support über-homing.

Network virtualization introduces a dynamic environment at all strata of networking. On the one hand, at macro level, VNs providing basic services or VNs with shared interests can be dynamically aggregated together to create compound VNs, and even hierarchy of VNs. On the other hand, dynamic join, leave, and mobility of the end users within and in between VNs add a micro level component to the dynamic characteristic of the NVE.

Resource Scheduling

When establishing a VN, a SP requires specific guarantees for the virtual routers' attributes, as well as the virtual links' bandwidth allocated to its network. For virtual routers, a SP might request guarantees for a minimum packet processing rate of the CPU, specific disk requirements, and a lower bound on the size of the memory. On the other hand, virtual link requests may range from best-effort service to fixed loss and delay characteristics found in dedicated physical links. To provide such guarantees, and to create an illusion of an isolated and dedicated network to each SP, InPs must employ appropriate scheduling algorithms in all of the network elements.

Efficient and effective resource scheduling mechanisms become more important when resources are not statically allocated to multiple VNs; instead, they are dynamically dis-

tributed to increase utilization of the resources as well as the revenue of the infrastructure providers. DaVinci [60] presents such a dynamic allocation framework where each substrate link periodically reassigns bandwidth shares between the virtual links. However, dynamic allocation gives a hint of best-effort mechanisms found in the existing Internet, and a careful investigation is required to validate such measure.

Topology Discovery

In order to allocate resources for requests from different SPs, InPs must be able to determine the topology of the networks they manage as well as the status of the corresponding network elements (i.e., physical nodes and interconnections between them). Furthermore, two adjacent InPs must also be able to establish links between their networks to enable cross-domain VN instantiation.

UCLP promotes a combination of *Event-based* and *Periodic* topology discovery, with an additional topology database; whereas, CABO argues for the use of a separate discovery plane run by the InPs as proposed in the 4D network management architecture [52].

3.4.3 Management

VN Configuration and Monitoring

To enable individual SPs configure, monitor, and control their VNs irrespective of others, considerable changes are required from the level of NOCs to intelligent agents at lower level network elements. The concept of MIBlets, i.e., partitioned MIBs, used in VNRMS to gather and process performance statistics for each of the coexisting VNs instead of using a common MIB can be a good starting point. But a full-fledged, robust monitoring framework needs more attention and efforts.

VN Management Frameworks

Since a VN can span over multiple underlying physical networks, management frameworks must also be developed to aggregate information from diverse management paradigms followed by participating InPs. Introducing a common abstraction layer, which will be followed by all the management softwares, can be an effective solution [49]. Clearly identifying and drawing a line between the scope of management for InPs and SPs is also a very important task.

Mobility Management

In an NVE, mobility of the devices must be supported congenitally, not using makeshift solutions as in the existing Internet. Mobility in this context does not just refer to its simplest form, i.e., geographic mobility of the end user devices, but routers in the core network can also move around using migration techniques. As a result, finding the exact location of any device at a particular moment and routing packets accordingly is a complex issue that needs simple solutions. In addition, end users can also move logically from one VN to another in order to access different services, which further complicates the problem.

Failure Handling

Failures in the underlying physical network components can give rise to complicated problems, such as a cascading series of failures in all the VNs directly hosted on those components and others that are recursively spawned from the affected ones. Detection, propagation, and isolation of such failures, as well as protection and restoration from them are all open research challenges.

3.4.4 Interaction between Players

Networking Technology Agnostic Virtualization

Network virtualization on different technologies face challenges that require specific solutions for provisioning, operation, and maintenance. For instance, with the advent of next-generation SONET/SDH and optical switching along with GMPLS, Layer 1 VPN is now a reality [19]. UCLP virtualizes optical networks capitalizing on the property of lightpaths that can be physically sub-divided into smaller lightpaths.

Virtual Sensor Networks (VSN) [63], on the other hand, deal with providing protocol support for the formation, usage, adaptation, and maintenance of subsets of sensors collaborating on specific tasks. Dynamic leave/join behavior of sensors and unique power constraints pose different challenges for VSN that will never occur in an optical network. Similarly, virtualization of wireless networks using different multiplexing techniques creates different complications, e.g. node synchronization and managing device states [94].

End-to-end VNs can span across multiple domains, each with possibly heterogeneous networking technologies. Interactions between such contrasting underlying infrastructures, while providing a generic and transparent interface for SPs to easily compose and manage VNs remains a daunting task.

Inter-VN Communication

Even though one of the main inspirations behind network virtualization is complete isolation between co-existing VNs, there are cases when two VNs need to share resources or information. For example, a large multinational corporation might deploy a VN across the globe, with child VNs for each of the continents, to manage its operations. It is very likely that those child VNs will need to communicate with the global one, as well as among themselves. In some cases, communicating VNs might not even be under the same administrative domain. Hence, the necessity, scope, and required interface for such interconnections among SPs and corresponding VNs deserve close scrutiny.

Network Virtualization Economics

Unlike the traditional networks where bandwidth is the chief commodity, virtual nodes are equally important as virtual links in an NVE. SPs are the buyers in this economy, whereas InPs are the sellers. There can also be brokers who act as mediators between the buyers and the sellers. End users also participate as buyers of services from different SPs.

Traditionally, there are two general types of marketplaces: centralized and decentralized. Centralized marketplaces are efficient; but vulnerable and not scalable. On the other hand, fully decentralized marketplaces are extensible and fault-tolerant; but prone to malicious behavior and inefficiency. To find a trade-off between these two options, existing works (e.g., PeerMart [58, 59]) on p2p marketplaces can be extended to the domain of network virtualization.

3.5 Summary

In this chapter, we have presented a conceptual model of the NVE that we refer to throughout this thesis. The architectural principles of the NVE guide us in designing and developing the framework and the algorithms in the later chapters, while the design goals of the NVE act as major motivations and drive us through the process.

We have also enumerated the most significant challenges to be resolved for successfully realizing the NVE. We have selected two problems to address in this thesis from the presented collection. In Chapter 4, we address the identity management problem that is concerned with naming and addressing in the NVE in presence of dynamism and mobility of the end hosts. Later, in Chapter 5, we propose one deterministic and one randomized VN embedding algorithms using better coordination between the node mapping and the link mapping phases based on a mathematical formulation of the problem.

Chapter 4

iMark: An Identity Management Framework for Network Virtualization Environment

4.1 Introduction

Each virtual network in the network virtualization environment is free to implement its own naming, addressing, routing, and transport mechanisms. While such flexibility allows fast and easy deployment of diversified applications and services, it does not come without cost. Due to the potential heterogeneity of the concerned physical and virtual networks, ensuring end-to-end communication and universal connectivity poses a daunting challenge in the network virtualization environment.

We believe that the first logical step toward universal connectivity is to make heterogeneous namespaces[1] (or identifier spaces) in different physical and virtual networks interoperable. Once it becomes possible to uniquely identify and locate the end hosts irrespective of their physical and logical locations, enabling end-to-end communication boils down to creating connections with necessary address/protocol translators in place.

This chapter presents iMark, an identity management framework for the network virtualization environment, which focuses on interoperability of heterogeneous identifier spaces. It does not put any restriction on an individual network's choice of local naming mechanism; instead, iMark defines a globally agreed upon identifier space for the end hosts and provides mechanisms to translate back and forth between local and global identifiers through a set of mappings placed in iMark controllers. Such explicit separation of the identity of an end host from its physical and logical locations allows heterogeneous networks to interoperate without sacrificing their autonomy.

[1] The words 'name' and 'identifier' are used interchangeably to refer to the identity of an entity.

4.2 Chapter Organization

The remainder of the chapter is organized as follows. We present the motivation behind the design of iMark in Section 4.3. In Section 4.4 we describe the design choices and a high-level overview of iMark, followed by a detailed specification of the basic iMark operations in Section 4.5. Section 4.6 presents experimental results from initial simulations. Section 4.7 summarizes related works, and we conclude in Section 4.8.

4.3 Motivation

To understand the intricacies of the naming problem in the network virtualization environment, consider the following scenario:

Alice is the North American continental manager of a corporate giant G with her head-quarters located in Toronto. G maintains separate VNs for every continent and a parent VN connecting all of its continental head-quarters. As a continental manager, Alice holds frequent video conferences with the regional managers of her continent and has to visit different offices occasionally. In addition, she has to participate in monthly meetings with her counterparts in other continents regarding the global objectives and progress of the company. As a result, she must be able to connect to appropriate VNs from her home, local office, or even when she is in transit or in foreign offices. She can try to access a VN using different devices, including personal phone or laptop, or her office desktop and through different access networks. In each case, the corresponding VN must be able to identify her with a single identity irrespective of her physical location, device, or the access network.

Now consider Bob, who is the continental manager of S, the biggest supplier of G. His job requires him to reach Alice from his own VN to wherever she is at a particular moment; not to mention that he himself can connect from different places and with different devices. Since Alice's VN and Bob's VN might not use compatible naming and addressing systems, finding one another in different VNs is not as simple as it is in the existing Internet.

In addition, the InPs that are hosting Alice's and Bob's VNs can also move the virtual nodes of those VNs around, to handle failures, or to upgrade equipment, or for regular maintenance. Even though the physical locations of the virtual nodes are changing, they must maintain their identity to keep themselves reachable from other nodes in the same VN or from other VNs.

Therefore, the naming requirements for the NVE boils down to something that will be able to handle the following phenomena:

4.3.1 Dynamism in the Network Virtualization Environment

Network virtualization introduces a dynamic environment at all strata of networking, which starts from individual end users or network elements and continues up to the level of complete VNs. We can broadly categorize such dynamism into two classes:

1. **Macro Level:** VNs providing basic services or VNs with shared interests can be dynamically aggregated together to create compound VNs. This is known as *federation* of VNs. Multiple federations and VNs can also come together to create hierarchy of VNs. Even though the level of dynamism is expected to be very low at this level, the complexity of adding a VN to a collection, or removing one, can be quite high.
2. **Micro Level:** This is the more influential of the two classes discussed here and, hence, requires more attention. Micro level dynamic behavior can basically be attributed to two broad sets of activities:
   - Dynamic join, leave, and *mobility* of the end users within and in between multiple VNs.
   - Dynamism incurred by the migration of virtual routers for different purposes [113, 111].

Mobility of the end users or virtual resources can again be of two types:

- *Geographical mobility* from one physical access network to another (e.g., Alice connecting to her office VN using her laptop from her home, in transit, or from her office)
- *Logical mobility* from one VN to another (e.g., Alice moving from her office VN to an online gaming VN in her spare time)

A naming framework for the NVE must, therefore, be flexible enough to handle such high level of mobility of end users while preserving their identities. Moreover, it should also provide support for federation and hierarchy of VNs to deploy complex end-to-end services.

4.3.2 Scale

Every day the number of users is increasing rapidly, and it is expected to continue along this line in the near future. Any new naming infrastructure, whether for the NVE or for something else, must be scalable enough to accommodate huge influx of end users.

4.3.3 Interactions Between Multiple Heterogeneous Parties

One of the most important issues in an NVE is the way multiple players interact among themselves. Such an interaction can be between two SPs (i.e. VNs), or two InPs, or an SP and an InP, and, in the most trivial form, between an end user and an SP. Moreover, each

party can have heterogeneous naming, addressing and routing mechanisms. To identify a particular node (physical or virtual) or an end user in this complex web, a naming framework must be expressive.

4.3.4 Über-homing

In the NVE, any end user can simultaneously connect to multiple VNs through multiple InPs using heterogeneous technologies to access different services. We refer to this phenomenon by *über-homing*. Über-homing has significant impact in cross VN routing. In that case, multiple routes might exist to reach a particular node through different VNs and InPs. The decision to prefer one over another can be taken based on the agreements between concerned SPs and InPs. Any naming framework for an NVE must provide additional level of indirection to support über-homing.

4.4 Architectural Overview

In this section, we discuss the decisions we have made in designing the iMark framework, followed by an architectural overview of its components. A detailed description of how iMark works can be found in the next section (Section 4.5).

4.4.1 Design Principles

The design choices made for iMark are inspired by the three key tenets of a next-generation architecture described in [34] and aim toward separation of identity and location, isolation of conflicting interests, and minimizing global functionalities.

1. **Separation of Identity and Location:** In the existing Internet, IP addresses denote both the identity of a node and its topological location. But mixing identity with location limits host mobility and restricts multihoming among many other problems [34]. Several proposals exist in the literature that separates a host's identity from its location. iMark takes a similar stance with a focus on supporting logical and physical mobility, federation and hierarchy of VNs, and überhoming.
2. **Local Autonomy:** iMark does not impose any requirements or restrictions on individual physical or virtual networks; rather it provides a set of defined interfaces and mechanisms to enable end-to-end connectivity across heterogeneous physical and virtual networks.
3. **Global Identifier Space:** Since each VN can implement its own naming mechanism, local identifiers have little end-to-end significance. Hence, in order to provide end-to-end communication between nodes in different VNs, there must be a globally agreed

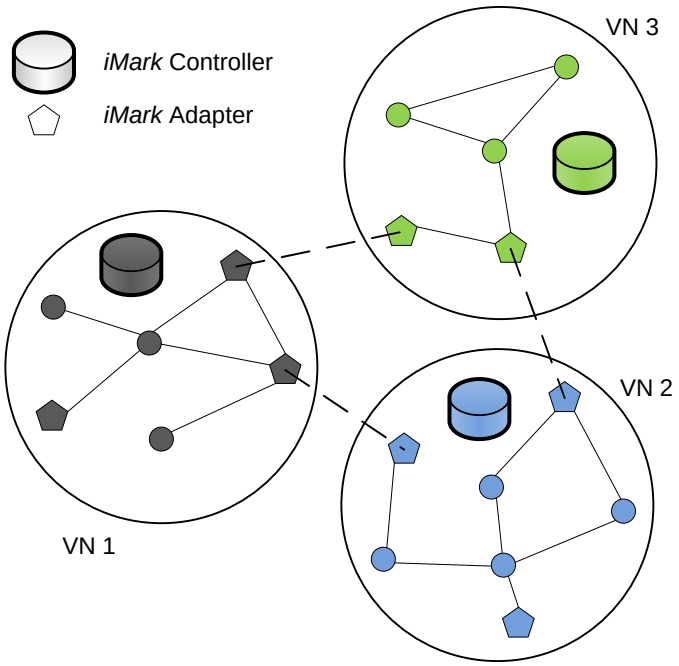

Figure 4.1: Overview of the iMark Framework

upon identification mechanism. Moreover, in order to ensure trust and security, end hosts must have unique identities that always remain the same, irrespective of whichever VN they are in or however they are connected. This requirement calls for the only globally agreed state in iMark. iMark does not impose any structure on these identifiers though; it only requires them to be unique.

4.4.2 iMark Components and Concepts

In order to identify nodes in corresponding VNs and to locate them in the underlying physical networks, iMark defines several entities and corresponding identifier spaces. To enable connectivity between heterogeneous identifier spaces, iMark stores mappings between different identifiers and keeps those mappings updated for address/protocol translation. This allows all the networks to be completely autonomous in their internal choices of naming, addressing, and routing.

Figure 4.1 depicts the essential components of iMark, which are discussed in the following:

Controllers

Controllers are logical entities in each VN that provide traditional control functionalities, e.g., address allocation, name resolution etc., along with other network specific additional services. A controller can be centralized (e.g., DNS) or distributed (e.g., DHT) based on the design of its VN.

Adapters

Adapters are special entities that act as gateways between two adjoining VNs. When adjoining VNs use different addressing schemes and/or protocol suites, adapters perform required address and protocol translations to relay traffic between them. If both networks use the same mechanism, adapters just forward data without modifying them.

Entities and Identifier Spaces

Given the NVE concepts presented previously, we identify the major entities that constitute iMark as follows:

1. **Service Provider:** Service providers create and manage one or more VNs by aggregating virtual resources from multiple InPs and provide deployed services to end users based on specific agreements.
2. **Virtual Network:** Any VN is instantiated and managed by a single SP. A VN has a finite timespan associated with it and is dissolved after that period.
3. **Virtual Resource:** Virtual resources belong to a single VN at a given time. Any end user device connected to a particular VN is logically considered to be a virtual resource of that VN.
4. **Infrastructure Provider/Physical Network:** Infrastructure providers are in charge of the underlying networks and all the physical resources contained within them. InPs have one-to-one relationships with the physical networks they manage; hence, they can be considered a single entity.
5. **Physical Resource:** Physical resources are actual network elements, e.g., routers and switches, that host the virtual resources.
6. **End User:** End users connect to VNs provided by different SPs through *access networks* managed by the InPs.

Figure 4.2 depicts the relationships between these entities using standard notations.

Based on the proposed entities, we define multiple identifier spaces (IDSes) to provide identifiers for those entities. Each IDS provides different types of identifiers to uniquely identify an entity in particular contexts. We summarize the IDSes below:

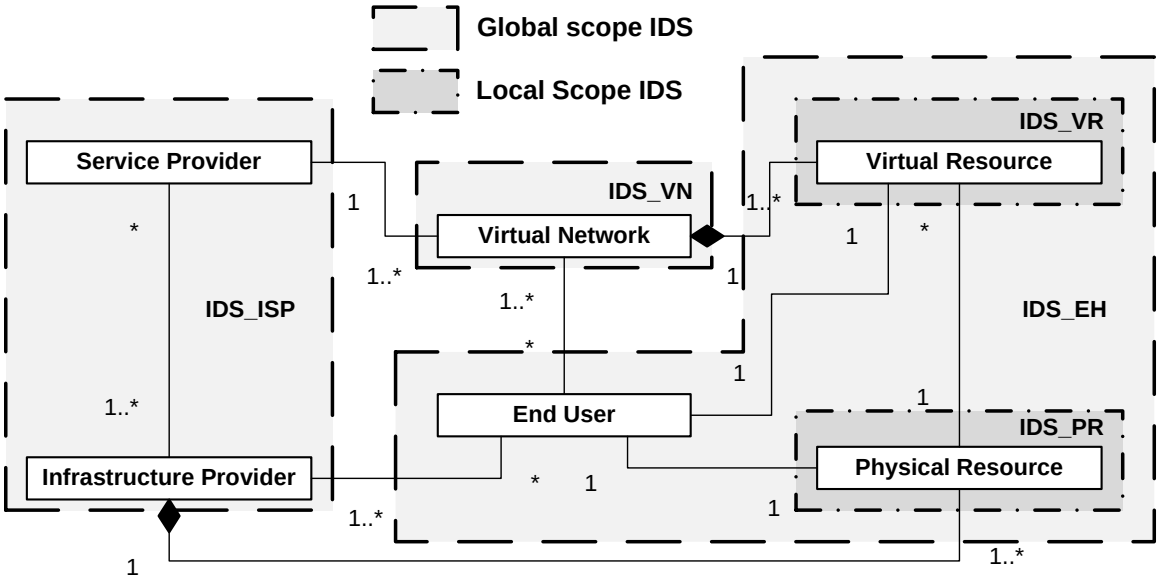

Figure 4.2: iMark Entities and Relationships between them. Shaded Rectangles Denote the Identifier Spaces.

1. *IDS_ISP* identifies all the SPs and InPs using unique *g_isp_id* for each one of them. A common IDS for both SPs and InPs enables them to participate in a common environment, e.g., a resource trading marketplace. An *isp_type* is used to differentiate SPs from InPs.
2. *IDS_VN* provides identifiers (*g_vn_id*) for all the virtual networks. Each VN also has a set of characterizing attributes that can be used to search for VNs with particular properties.
3. *IDS_VR* identifies all the virtual resources connected to and contained within a VN using l_vr_id. These identifiers are unique *within* a virtual network. Each virtual resource has an associated *vr_type* that defines whether it is an end user or an actual virtual resource inside the VN.
   If any end user is simultaneously connected to multiple VNs at a particular time, it will have multiple local l_vr_ids. Each VN is free to use its own control and data plane protocols with its own set of l_vr_ids irrespective of other VNs.
4. *IDS_PR* specifies l_pr_id to locally identify physical network elements and connected end user devices. Each physical resource also has a *pr_type* to distinguish between end user devices and internal network elements.
   If any end user is simultaneously connected to multiple physical networks, i.e., multi-homed, it will have multiple l_pr_ids.
5. *IDS_EH* provides globally unique location-independent identifiers, g_eh_id, for every

Table 4.1: Mappings between Different Identifiers

$(\langle from\_id \longleftrightarrow to\_id \rangle)$	Purpose
$\langle g\_eh\_id \longleftrightarrow l\_vr\_id \rangle$	Identifies any resource within a virtual network and vice versa.
$\langle g\_eh\_id \longleftrightarrow l\_pr\_id \rangle$	Identifies any resource within a physical network and vice versa.
$\langle g\_eh\_id \rightarrow g\_vn\_id \rangle$	Stores the virtual network an end host is connected to.
$\langle l\_vr\_id \rightarrow l\_pr\_id \rangle$	Finds the local identifier of the physical host of a virtual resource within a physical network.
$\langle g\_vn\_id \rightarrow \{l\_pr\_id\} \rangle$	Gets the local identifiers of the access nodes of a virtual network inside a physical network.
$\langle g\_vn\_id \rightarrow g\_isp\_id \rangle$	Finds the owner SP of a virtual network.
$\langle g\_vn\_id \rightarrow \{g\_isp\_id\} \rangle$	Obtains the set of InPs that host the virtual network in the underlying network.

end user and nodes that a particular network wants to expose to the outside world.

Mappings

In order to locate all the entities in an NVE and to route to their current locations based on their global identifiers, a set of mappings between different IDSes are required. Mappings are stored at controllers and updated based on micro-level events (e.g., node join, leave, and mobility) as well as macro-level ones (e.g., VN creation, expiration etc.). Table 4.1 presents a list of mappings required by iMark.

Federation and Hierarchy of iMark Controllers

Federation allows multiple autonomous VNs in the NVE to connect and logically merge together to provide end-to-end services. An example of federation is the peering relationship between Alice's VN and Bob's VN described in Section 4.3. iMark creates a common control space, known as *controller network*, connecting controllers of each of the participant VNs to support federations. The controller network can itself be a VN. Adapters ensure interoperation between possibly heterogeneous controllers by performing necessary translations. It should be noted that federation is just a mechanism to share control information between VNs, and it does not disrupt their local autonomy.

Multiple federations and VNs can also create a logical hierarchy of VNs for different

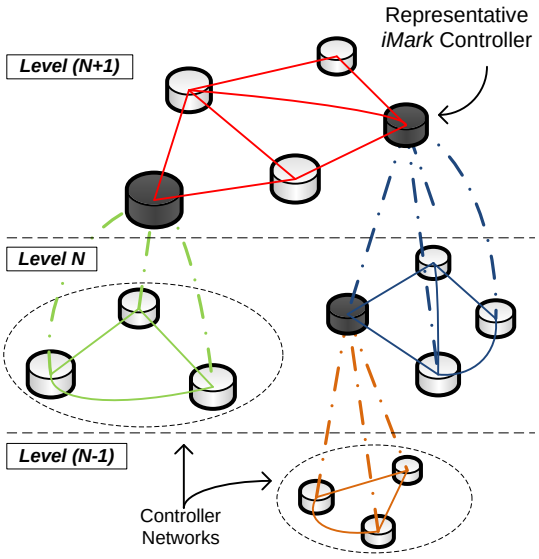

Figure 4.3: Federation and Hierarchy of iMark Controllers

reasons. For example, Alice's corporation G in Section 4.3 has created a hierarchy of VNs for administrative purposes. iMark proposes the concept of *representative controllers* of federations to support such *controller hierarchy*. A representative controller can either be an elected member of the federation, or it can be a separate entity altogether. Each representative controller has knowledge of all the end hosts that belong to any of the VNs in its subtree.

4.5 iMark Operations

Due to the autonomy and potential heterogeneity of the VNs in an NVE, ensuring end-to-end connectivity across VN boundaries is a nontrivial task. Since local identifiers (l_vr_id) are not meaningful outside a VN's domain, connectivity is provided based on the global identifiers (g_eh_id) of the end hosts using different iMark mappings mentioned earlier. In order to create these mappings, a joining procedure is required that binds end hosts to physical access networks as well as to VNs of their choice. To communicate across VN boundaries, an explicit connection setup procedure is followed that looks up the destination host and sets up relaying states. This section describes these basic procedures along with the compound ones like *überhoming* and *mobility* handling.

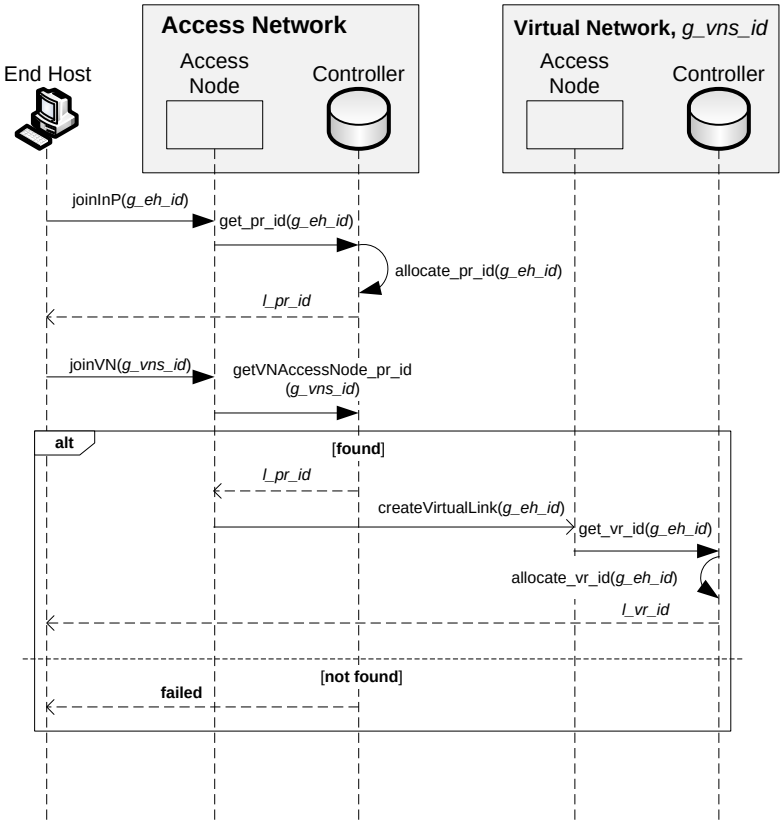

Figure 4.4: Sequence Diagram: Join Operation

4.5.1 Macro Level Operations

In order to let end hosts join different VNs and communicate between themselves, VNs must be instantiated first. Here we briefly describe how SP, InP, and VN specific mappings accommodate VN instantiation as well as formation of federation and hierarchy in an NVE. How VNs are provisioned before being instantiated is addressed in the later chapter.

It is well understood that in order to create VNs, SPs and InPs must have a common marketplace to trade resources [58] which itself can be a VN. iMark provides globally unique identifiers (g_isp_id) for the SPs and the InPs to participate in such an environment. We refer to this VN as the *administrative VN* and its controller as the *administrative controller*.

When an SP wants to create a VN, it contacts one or more InPs and provides its requirements. Once a VN is provisioned and instantiated, it is assigned a unique identifier (g_vn_id) which is used for later identification, e.g., during the join operation. Two mappings, $\langle g\_vn\_id \rightarrow g\_isp\_id \rangle$ and $\langle g\_vn\_id \rightarrow \{g\_isp\_id\} \rangle$ are stored in the administrative controller at this point; the first one identifies the owner of the VN and is required to form federations; the later identifies the InPs that host the VN's resources and is used by the

InPs to setup cross-InP virtual links for the VN. Physical networks store $\langle l\_vr\_id \rightarrow l\_pr\_id \rangle$ mappings to route in the underlay to create virtual links for the VN request.

Each VN has several access nodes that are located in different physical access networks and are used as gateways to that VN. Access networks store information about these access nodes using $\langle g\_vn\_id \rightarrow \{l\_pr\_id\} \rangle$ mapping and use this mapping during the join procedure.

4.5.2 Join

In order to connect to a VN, an end host must join a physical access network first. The controller of the access network assigns and stores an l_pr_id corresponding to the g_eh_id of the end host based on its naming and addressing mechanism. This l_pr_id will be used in the underlay to create virtual links between the end host and VN access nodes.

Next, the end host provides a globally unique identifier of the VN (g_vn_id) it wants to connect to. The access network finds out the l_pr_id of the access node of that particular VN and forwards the request. The controller of the VN then assigns the end host an l_vr_id, stores a $\langle g\_eh\_id \leftrightarrow l\_vr\_id \rangle$ mapping, and assimilates the end host.

Figure 4.4 provides further details of the joining procedure through a sequence diagram.

4.5.3 Lookup and Connection Setup

When an end host e_s wants to initiate communication with another end host e_d, it gives the global identifier of e_d (g_eh_id) to its local iMark controller with a request to setup a connection. The controller first looks up its tables to see whether e_d belongs to its own network. In that case, it sets up a connection and returns the l_vr_id of e_d to e_s

If e_d does not belong to the same VN and the VN belongs to a VN federation or hierarchy, the controller communicates with other controllers in the controller network (first horizontally, then vertically toward the topmost level of VN hierarchy). If any controller can resolve g_eh_id, it returns a positive response to the originating controller. Consequently, a cross VN connection is setup by creating necessary states in the inter-VN adapters.

Since lookup is expensive, after every lookup operation, the originating VN's controller caches the $\langle g\_eh\_id \rightarrow g\_vn\_id \rangle$ mapping for a certain time period as a performance optimization measure.

In case e_d is simultaneously connected to multiple VNs, one is chosen as the destination VN based on inter-VN agreements and VN-specific policies.

Figure 4.5 depicts the lookup and connection setup procedure using a sequence diagram. Note that, searching in the VN hierarchy as well as the destination host is omitted from the diagram for brevity.

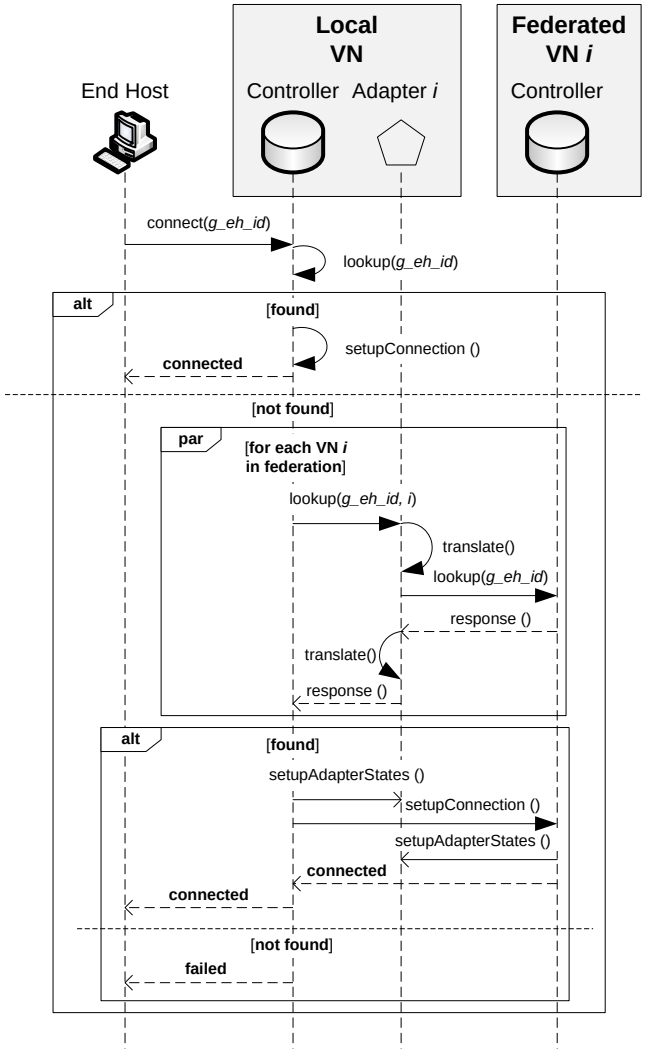

Figure 4.5: Sequence Diagram: Connecting to an End Host

4.5.4 Leave

Whenever an end host wants to leave a VN, it notifies the concerned controller, and all the corresponding mappings stored during the join procedure are removed. In addition, controllers can implement heart beat protocols to periodically check the availability of the connected virtual resources. It also allows controllers to handle failures as a normal leave events.

4.5.5 Über-homing

When an end host is über-homed, it can have multiple l_vr_ids in each of the connected VNs along with multiple l_pr_ids, if necessary, in each of the access networks it used to connect

to those VNs. Unlike the multihoming scenario in the existing Internet where different IP addresses might be assigned to the same node by different ISPs, in an NVE each end user has a unique identifier g_eh_id which is free from its logical and physical location.

Once an end-to-end connection to an end host is setup through a particular pair of physical and virtual networks, l_vr_id and l_pr_id corresponding to that g_eh_id in those physical and virtual networks are used to locate the end host and to perform routing.

4.5.6 Mobility

As mentioned earlier, mobility in an NVE can be of two types: geographical mobility of the end host physical devices from one access network to another, and logical mobility of the end hosts from one VN to another. iMark supports both types of mobility through necessary manipulations of the related mappings, with some assistance from the überhoming capability of the NVE.

In case of geographical mobility, an end host moves from one access network to another by *soft handoff*. First, it *joins* the new access network without leaving the old one and gets a new l_pr_id. Then it requests the new access network to create a connection to the same VN that it is already connected to through the old access network. When the controller of the VN gets the new request, it updates its $\langle g\_eh\_id \leftrightarrow l\_vr\_id \rangle$ mapping with a new l_vr_id based on the l_pr_id of the new access network. The end host finally leaves the old access network to complete a seamless transition.

Logical mobility can be handled by a simpler two step process: *leave* from the old VN, and then *join* a new one.

4.6 Simulation Results

iMark can face performance challenges from two main sources: size of the mappings stored at different controllers, and lookup frequency at different levels of the controller hierarchy. In this section, we evaluate iMark's performance in both cases through simulation.

4.6.1 Experimental Setup

We developed an in-house simulator and performed the simulations on a quad CPU Sun V440 Server with 8GB of memory. In order to explore the problem space, we ran a large set of experiments by varying the total number of VNs from 1000 to 15000, the size of federations from 30 to 300, and the total number of end hosts from 10 thousand to 10 million. To explore the impact of controller hierarchies, we experimented with two different types of hierarchies: in a balanced hierarchy, we allowed VNs to form federations only among

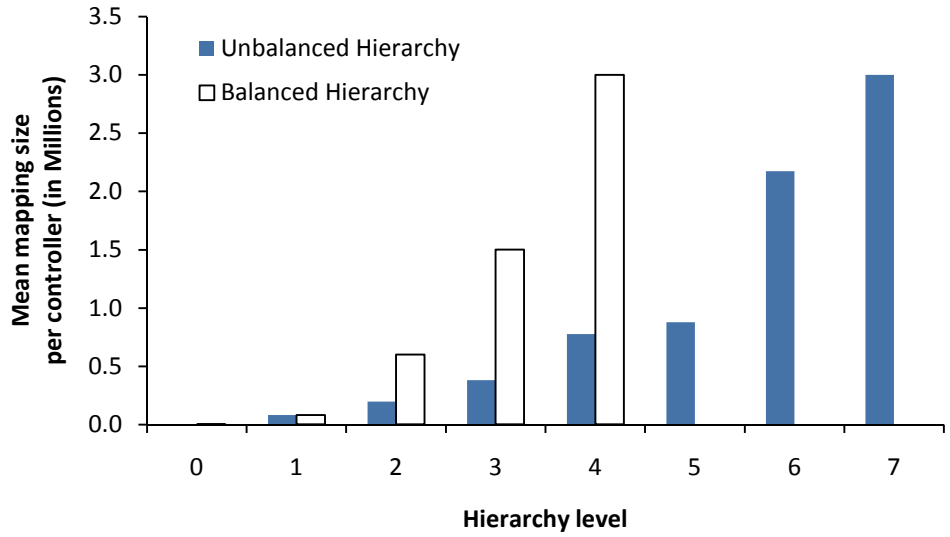

Figure 4.6: Mean Mapping Size per Controller at Different Levels of Balanced and Unbalanced Controller Hierarchies

themselves; whereas, in an unbalanced hierarchy, we let VNs join randomly at any level of the hierarchy. Of the two, balanced ones resulted in shorter hierarchies.

Our main goal was to show iMark's correctness and to provide an indication of its performance trends in heterogeneous NVE. Hence, we did not put any restrictions on the use of any particular protocol or algorithm in individual VNs. For simplicity, we did not employ any optimization, e.g., caching, and did not consider mobility and überhoming of end hosts.

After running a large set of experiments by varying different parameters, we observed definite trends in the size of the mappings stored and the lookup frequency. We picked one representative result of each case to discuss our findings.

4.6.2 Mapping Size

When considering the total amount of mapping information stored in a representative controller, the contribution of $\langle g\_eh\_id \rightarrow g\_vn\_id \rangle$ easily dominates the rest, since each representative controller aggregates this mapping from all of its child controllers. We, therefore, focused on finding out how the size of this mapping increases as we move upward in the controller hierarchy.

Figure 4.6 depicts the simulation results showing the mean mapping size per controller at different levels of the controller hierarchy, level 0 being the lowest level consisting only of individual VNs. For this particular experiment, we considered 3000 VNs with an average of 80 VNs per federation, and 3 million end hosts with an average of 1000 end hosts per VN.

As expected, the size of the mapping increases gradually from lower to higher levels of

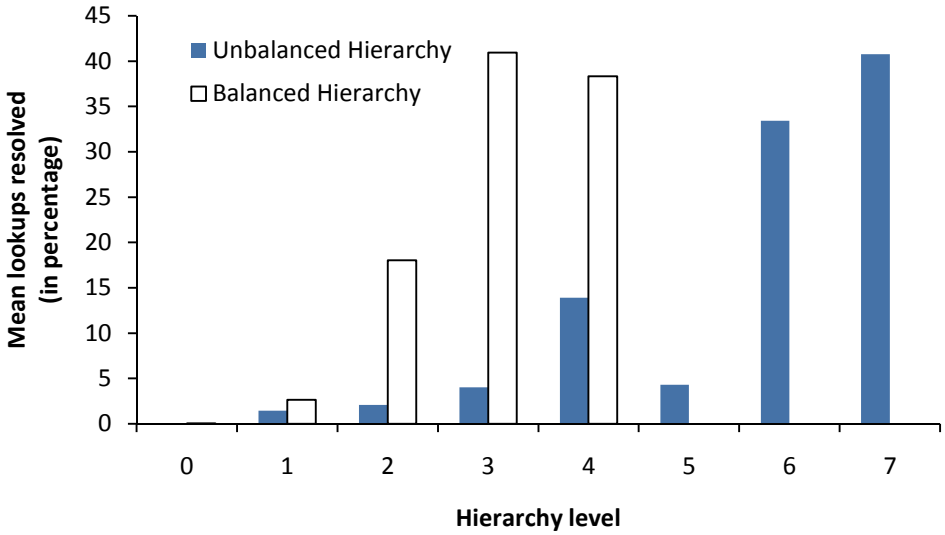

Figure 4.7: Mean Lookups Resolved at Different Levels of Balanced and Unbalanced Controller Hierarchies

the hierarchy with the topmost level controllers having the maximum. Since the unbalanced hierarchy has more levels than its balanced counterpart, each level has fewer participating VNs and federations; this results in a greater number of controllers, each with smaller loads.

4.6.3 Lookup

Whenever there is a lookup request, a controller first tries to resolve it using its own mapping information. In case of a failure, it forwards the request to its peers in the federation before resorting to the upper level controllers. The more requests a controller forwards to the upper level, the higher the number of messages generated. This also results in higher lookup resolution time. So we examined the percentage of lookup requests that are resolved at different levels of the controller hierarchy to gain an insight into the performance of iMark.

Figure 4.7 presents the simulation results showing the mean percentage of lookups resolved at different levels of the controller hierarchy after 1 million lookup operations. For this experiment, we considered 3000 VNs with an average of 80 VNs per federation, and 50000 end hosts.

Since the upper level controllers store more information, a large percentage of the lookup requests find their way to the top two layers of the hierarchy. The smaller height of the balanced variant gives it a competitive advantage over its unbalanced counterpart, because in this case requests can reach the topmost levels faster.

A sudden decrease in the lookup resolution percentage is observed in level 5 of the unbalanced hierarchy. Due to the randomness in the formation phase, we believe that in this

particular instance, more individual VNs than federations ended up forming the federations of level 5, which resulted in less information being stored at this level on the average.

4.7 Related Work

To the best of our knowledge, there is no existing work in the literature that addresses the exact problems posed by the unique naming and addressing requirements of an NVE. But there are several proposals that share some common aspects with our one and have influenced as well as motivated us.

Similar to our proposal, TRIAD [53] uses location independent identifiers instead of addresses for node identification. But TRIAD relies on the presence of IPv4 in all network domains, which is completely in contrast with the basic requirement of heterogeneity. In addition, TRIAD's dependence on semantics and hierarchy of domain names is completely opposite to our choice of flat global identifier space.

Plutarch [36], on the other hand, provides explicit support for heterogeneity through the concept of *contexts* and uses *interstitial functions* to translate communication between them, which is similar to our proposal. But Plutarch does not consider mobility of end hosts between multiple contexts and überhoming.

IPNL [50] and 4+4 [106] try to isolate independent IP-based networks through loose integration. IPNL provides three stage communication path consisting of originating and terminating *private realms* and a global *middle realm*. 4+4 generalizes it by supporting multiple middle realms. But both schemes primarily focus on the address depletion problem faced by the existing Internet and are not concerned about the requirements of the NVE.

Host Identity Protocol (HIP) [78] is also related to iMark in its concept of separating the end-point identifier and locator roles even though HIP targets a uniform IP networks without any notion of virtualization. It introduces a new Host Identity (HI) namespace based on public keys, which are normally self-generated. While in this work we do not explicitly address how the unique global identifier space can be formed, the concept of public key based namespace can be a good choice for future exploration in this direction.

TurfNet [92, 84] is the most closely related proposal to our work in the existing literature. Conceptually, it supports heterogeneous autonomous network domains; separation of identifiers from locators; encapsulation of internal naming, addressing and routing mechanism and policies of an autonomous domain; and dynamic network composition (vertical and horizontal). But since TurfNet does not consider network virtualization, it is free from the issues arising from InP-SP interactions. Also it does not consider mobility of end hosts and virtual resources.

In addition, our work is motivated by the recent works on flat identifiers and location

independent (or identity-based) naming and routing mechanisms [96, 17, 27, 28]. Last but not the least, our mapping mechanism and identifier space selection was highly influenced by the P2P-based naming architecture proposed in [44] for autonomic networks.

4.8 Summary

In this chapter, we have presented iMark, a novel identity management framework for the network virtualization environment. iMark manages identifiers for entities at different levels: at macro level, it assists creation of independent VNs and formation of VN federations and hierarchy of VNs by ensuring cooperation between SPs and InPs; whereas, at micro level, iMark enables end-to-end communication between end hosts in different VNs. iMark separates identity of the end hosts from their physical and logical locations, and, with the help of a global identifier space, it provides universal connectivity without revoking the autonomy of the concerned physical and virtual networks. To demonstrate iMark's correctness and to provide an indication of its performance, we have done a simulation-based study of the framework. Current experience with iMark suggests that it can indeed enable end-to-end connectivity in a highly heterogeneous NVE.

Chapter 5

ViNE-Yard: Virtual Network Embedding with Coordinated Node and Link Mapping

5.1 Introduction

One of the basic challenges in network virtualization is the embedding [1] of virtual network requests from different service providers onto the underlying physical network resources. In order to provision a virtual network, constraints on both the virtual nodes and the virtual links must be satisfied. But each virtual node can be mapped to multiple physical nodes and each virtual link onto multiple physical paths. As a result, many possible mappings exist for any given virtual network request. Moreover, multiple virtual networks share the same underlying physical resources. Hence, efficient and effective embedding of each of the *online* virtual network requests is of utmost importance in order to increase the *utilization* of the substrate network resources and consequently the revenue of the infrastructure provider.

However, the *virtual network embedding problem*, with constraints on virtual nodes and virtual links, can be reduced to the $\mathcal{NP}$-hard *multi-way separator problem* [12], even if all the requests are known in advance. Even when all the virtual nodes are already mapped, embedding the virtual links with bandwidth constraints onto substrate paths is still $\mathcal{NP}$-hard in the *unsplittable flow* scenario. As a result, a number of heuristic-based algorithms have appeared in the relevant literature [43, 74, 119, 117]. Most of these proposals focused primarily on edge mapping (using, for example, shortest path, k-shortest paths, and multi-commodity flow algorithms) after employing greedy methods to *preselect* the node mappings. However, preselection in the node mapping stage without considering its relation to the link mapping stage restricts the solution space, and may result in poor performance.

[1] The words 'embedding', 'mapping', and 'assignment' are used interchangeably.

In this chapter, we introduce better correlation between the node mapping and the link mapping phases by proposing two new VN embedding algorithms *D-ViNE* (**D**eterministic **Vi**rtual **N**etwork **E**mbedding) and *R-ViNE* (**R**andomized **Vi**rtual **N**etwork **E**mbedding). In these algorithms, we map the virtual nodes to substrate nodes in a way that facilitates the mapping of the virtual links to physical paths in the subsequent phase. To this end, we extend the physical network graph by introducing meta-nodes for each virtual node and connect the meta-nodes to a selected subset of physical nodes (Section 5.4.1). We then treat each virtual link with bandwidth constraints as a commodity consisting of a pair of meta-nodes. As a result, finding an optimal flow for the commodity is equivalent to the mapping of the corresponding virtual link in an optimal way. We introduce additional binary constraints that force only one meta-edge to be selected for each meta-node, effectively selecting exactly one substrate node for each meta-node corresponding to a particular virtual node.

We use mixed integer programming (MIP) formulation [93] to solve the embedding problem with binary constraints on the meta-edges and linear constraints on the actual substrate network edges. Since solving an MIP is known to be $\mathcal{NP}$-hard [93], finding an optimal VN embedding using MIP becomes $\mathcal{NP}$-hard as well. As a result, we relax the integer program to obtain a linear programming formulation which can be solved in polynomial time. We then use deterministic and randomized rounding techniques on the solution of the linear program to approximate the values of the binary variables in the original MIP. Once all the virtual nodes have been mapped, we use the multi-commodity flow algorithm to map the virtual links onto the substrate network between the mapped virtual nodes [99, 117]. This can also be solved in polynomial-time since we assume that path splitting is supported by the substrate network [117].

5.2 Chapter Organization

The rest of this chapter is organized as follows. In Section 5.3, we formalize the network model and the virtual network embedding problem itself. Section 5.4 provides the optimal MIP formulation for the virtual network embedding problem using substrate network augmentation, and Section 5.5 relaxes the MIP formulation to obtain a linear program and presents D-ViNE and R-ViNE using deterministic and randomized rounding techniques. Section 5.6 presents simulation results that evaluate the proposed algorithms and quantify the benefits of correlated node and link mapping. In Section 5.7, we compare our algorithm with the related work, and we conclude in Section 5.8.

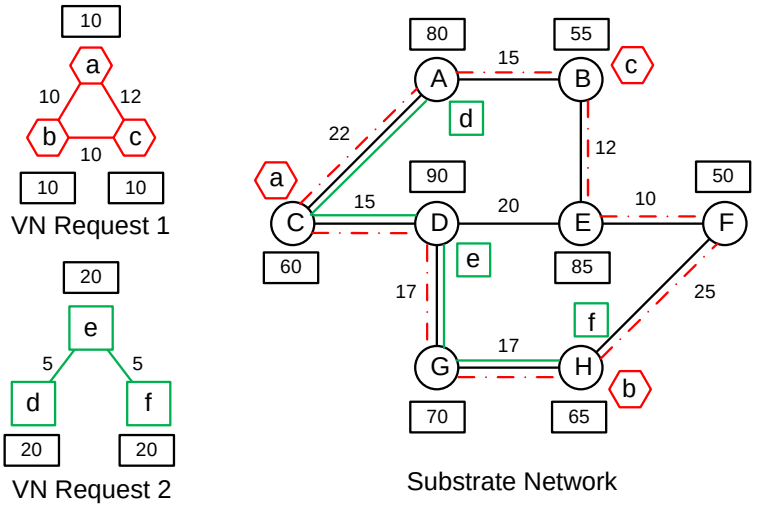

Figure 5.1: Mapping of VN Requests onto a Shared Substrate Network

5.3 Network Model and Problem Description

5.3.1 Substrate Network

We model the substrate network as a weighted undirected graph and denote it by $G^S = \left(N^S, E^S\right)$, where N^S is the set of substrate nodes and E^S is the set of substrate links. Each substrate node $n^S \in N^S$ is associated with the CPU capacity weight value $c\left(n^S\right)$ and geographic location $loc\left(n^S\right)$. Each substrate link $e^S\left(i,j\right) \in E^S$ between two substrate nodes i and j is associated with the bandwidth capacity weight value $b\left(e^S\right)$ denoting the total amount of bandwidth.

We denote the set of all substrate paths by $\mathcal{P}^S$, and the set of substrate paths from the source node s to the destination node t by $\mathcal{P}^S\left(s,t\right)$.

Figure 5.1 shows a substrate network, where the numbers over the links represent available bandwidths and the numbers in rectangles represent available CPU resources [2].

5.3.2 Virtual Network Request

Similar to the substrate network, we model VN requests as weighted undirected graphs and denote a VN request by $G^V = \left(N^V, E^V\right)$. We express the requirements on virtual nodes and virtual links in terms of the attributes of the nodes and links of the substrate network. Each VN request has an associated non-negative value D^V expressing how far a virtual node $n^V \in N^V$ can be placed from the location specified by $loc\left(n^V\right)$. D^V can be expressed in terms

[2] We use notations similar to [117] to denote capacities and requirements.

of physical distance or in terms of permissible delay from $loc\left(n^V\right)$. Figure 5.1 shows two VN requests with node and link constraints.

5.3.3 Stress and Residual Capacity of Substrate Resources

In order to quantify the resource usage of the substrate network, we use the notion of *stress*. The substrate node stress, $S_N\left(n^S\right)$ is defined as the total amount of CPU capacity allocated to different virtual nodes hosted on the substrate node $n^S \in N^S$.

$$S_N\left(n^S\right) = \sum_{\forall n^V \uparrow n^S} c(n^V) \tag{5.1}$$

where $x \uparrow y$ denotes that the virtual node x is hosted on the substrate node y.

Similarly, the substrate link stress, $S_E\left(e^S\right)$ is defined as the total amount of bandwidth reserved for the virtual links whose substrate paths passes through the substrate link $e^S \in E^S$.

$$S_E\left(e^S\right) = \sum_{\forall e^V \uparrow e^S} b(e^V) \tag{5.2}$$

where $x \uparrow y$ denotes that the substrate path of the virtual link x passes through the substrate link y.

The definitions of *node stress* and *link stress* are similar to that in [119] with the difference that the number of virtual links and nodes are used to measure the resources, not the actual amount of CPU and bandwidth resources.

The *residual* or the available capacity of a substrate node, $R_N\left(n^S\right)$ is defined as the available CPU capacity of the substrate node $n^S \in N^S$.

$$R_N\left(n^S\right) = c\left(n^S\right) - S_N\left(n^S\right)$$

Similarly, the residual capacity of a substrate link, $R_E\left(e^S\right)$ is defined as the total amount of bandwidth available on the substrate link $e^S \in E^S$.

$$R_E\left(e^S\right) = b\left(e^S\right) - S_E\left(e^S\right)$$

The available bandwidth capacity of a substrate path $P \in \mathcal{P}^S$ is given by

$$R_E(P) = \min_{e^S \in P} R_E\left(e^S\right)$$

5.3.4 Virtual Network Assignment

When a VN request arrives, the substrate network has to determine whether to accept the request or not. If the request is accepted, the substrate network then determines a suitable assignment for the VN and allocates network resources on the substrate nodes and paths selected by that assignment. The allocated resources are released once the VN expires.

The assignment of the VN request V to the substrate network can be decomposed into two major components:

Node assignment

Each virtual node is assigned to a *different* substrate node by a mapping $\mathcal{M}_N : N^V \rightarrow N^S$ from virtual nodes to substrate nodes such that for all $n^V, m^V \in N^V$,

$$\mathcal{M}_N\left(n^V\right) \in N^S$$
$$\mathcal{M}_N\left(m^V\right) = \mathcal{M}_N\left(n^V\right), \textit{iff } m^V = n^V$$

subject to

$$c\left(n^V\right) \leq R_N\left(\mathcal{M}_N\left(n^V\right)\right) \tag{5.3a}$$
$$dis\Big(loc\left(n^V\right), loc\left(\mathcal{M}_N\left(n^V\right)\right)\Big) \leq D^V \tag{5.3b}$$

where $dis(i, j)$ measures the distance between the location of two substrate nodes i and j. We consider end-to-end delay as the measure of distance in this work.

In Figure 5.1, the first VN request has the node mapping $\{a \rightarrow C, b \rightarrow H, c \rightarrow B\}$ and the second VN request has $\{d \rightarrow A, e \rightarrow D, f \rightarrow H\}$. Note that two virtual nodes b and f from different VN requests are mapped onto the same substrate node H.

Link assignment

Each virtual link is mapped to a substrate path (unsplittable flow) or a set of substrate paths (splittable flow) between the corresponding substrate nodes that host the end virtual nodes of that virtual link. It is defined by a mapping $\mathcal{M}_E : E^V \rightarrow \mathcal{P}^S$ from virtual links to substrate paths such that for all $e^V = \left(m^V, n^V\right) \in E^V$,

$$\mathcal{M}_E\left(m^V, n^V\right) \subseteq \mathcal{P}^S\Big(\mathcal{M}_N\left(m^V\right), \mathcal{M}_N\left(n^V\right)\Big)$$

subject to

$$R_E(P) \geq b\left(e^V\right), \forall P \in \mathcal{M}_E\left(e^V\right) \tag{5.4}$$

The first VN request in Figure 5.1 has been assigned the link mapping $\Big\{(a,b) \rightarrow \{(C,D),(D,G),(G,H)\},(a,c) \rightarrow \{(C,A),(A,B)\},(b,c) \rightarrow \{(H,F),(F,E),(E,B)\}\Big\}$, and the second one has the mapping $\Big\{(d,e) \rightarrow \{(A,C),(C,D)\},(e,f) \rightarrow \{(D,G),(G,H)\}\Big\}$.

Objectives

Our main interest in this chapter is to propose online VN embedding algorithms that map multiple VN requests with node and link constraints. We also want to increase revenue and decrease cost of the InP in the long run, in addition to balancing load of the substrate network resources.

Similar to the previous works in [119, 117], we define the revenue of a VN request as:

$$\mathbb{R}(G^V) = \sum_{e^V \in E^V} b\left(e^V\right) + \sum_{n^V \in N^V} c\left(n^V\right) \tag{5.5}$$

While revenue gives an insight into how much an InP will gain by accepting a VN request, it is not very useful without knowing the cost the InP will incur for embedding that request. We define the cost of embedding a VN request as the sum of total substrate resources allocated to that VN:

$$\mathbb{C}(G^V) = \sum_{e^V \in E^V} \sum_{e^S \in E^S} f_{e^S}^{e^V} + \sum_{n^V \in N^V} c(n^V) \tag{5.6}$$

where $f_{e^S}^{e^V}$ denotes the total amount of bandwidth allocated on the substrate link e^S for virtual link e^V. We use a modified version of (5.6) as the objective function of our MIP formulation.

5.4 Mixed Integer Programming Formulation for Optimal VN Embedding

5.4.1 Augmented Substrate Graph Construction

In order to coordinate the node mapping phase with its link mapping counterpart, the base substrate network must be extended to create an *augmented substrate graph* using the location requirement of the virtual nodes as the basis for the extension. Since each $n^V \in N^V$ has an associated constraint $loc\left(n^V\right)$ on its possible placement, we can create as many *clusters* in the substrate network with radius D^V. We denote such a cluster by $\Omega(n^V)$ and call it the Ω-Set of the virtual node n^V :

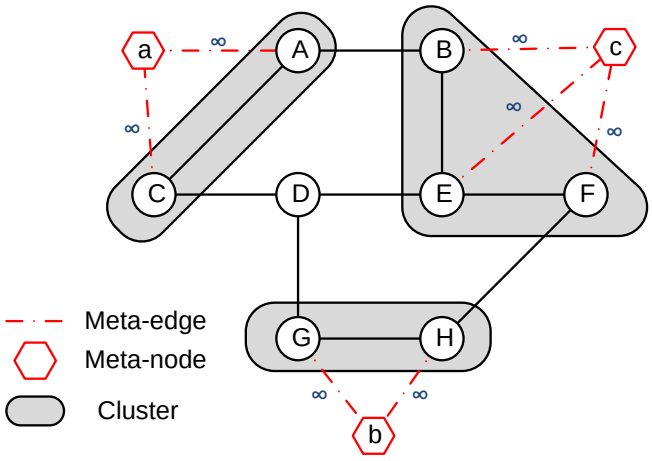

Figure 5.2: Construction of an Augmented Substrate Graph with Meta-nodes and Meta-edges for a VN Request

$$\Omega(n^V) = \left\{ n^S \in N^S | dis\left(loc\left(n^V\right), loc\left(n^S\right)\right) \leq D^V \right\}$$

In Figure 5.2, substrate nodes B, E, and F are within D^V distance of the virtual node c, hence $\Omega(c) = \{B, E, F\}$.

For each $n^V \in N^V$ we create a corresponding *meta-node* $\mu(n^V)$, and we connect $\mu(n^V)$ with all the substrate nodes belonging to $\Omega(n^V)$ using *meta-edges* with infinite bandwidth. Throughout the rest of this chapter, we sometimes write the Ω-Set as $\Omega(m)$ instead of $\Omega(n^V)$, where $m = \mu(n^V)$. We combine all the meta-nodes and meta-edges with the substrate graph to create the augmented substrate graph $G^{S'} = (N^{S'}, E^{S'})$, where

$$N^{S'} = N^S \cup \left\{ \mu(n^V) \mid n^V \in N^V \right\}$$
$$E^{S'} = E^S \cup \{(\mu(n^V), n^S) \mid n^V \in N^V, n^S \in \Omega(n^V)\}$$

An example of the augmented graph construction has been shown in Figure 5.2.

5.4.2 MIP Formulation

The virtual network embedding problem can now be formulated as a mixed integer $|E^V|$-commodity flow problem. We consider each virtual edge e_i^V $\left(1 \leq i \leq |E^V|\right)$ as a commodity with source and destination nodes s_i and t_i, respectively $\left(\forall i, s_i, t_i \in N^{S'} \setminus N^S\right)$. Each flow, in this setting, starts from a meta-node and ends in another meta-node. By introducing restrictions on the meta-edges, each meta-node $\mu(n^V)$ can be forced to choose only one

meta-edge to connect itself to an actual substrate node in $\Omega(n^V)$. This effectively selects a substrate node for each meta-node, i.e., maps the virtual node corresponding to that meta-node to a substrate node. At the same time, all the virtual edges (i.e., flows) are also mapped efficiently inside the substrate network with the help of path splitting. We present the MIP formulation in the following manner.

Program 5.1 (Mixed Integer Program for Virtual Network Embedding)

Variables:

- f_{uv}^i: A flow variable denoting the total amount of flow in the $u \rightarrow v$ direction on the substrate edge (u, v) for the i'th virtual edge.
- x_{uv}: A binary variable, which has the value '1' if $\sum_i \left(f_{uv}^i + f_{vu}^i\right) > 0$; otherwise, it is set to '0'.

Objective:

$$\begin{aligned} \text{minimize} \quad & \sum_{uv \in E^S} \frac{\alpha_{uv}}{R_E(u,v) + \delta} \sum_i f_{uv}^i \\ & + \sum_{w \in N^S} \frac{\beta_w}{R_N(w) + \delta} \sum_{m \in N^{S'} \setminus N^S} x_{mw}\, c(m) \end{aligned} \tag{5.7}$$

Constraints:

- Capacity Constraints:

$$\sum_i (f_{uv}^i + f_{vu}^i) \leq R_E(u,v)\, x_{u,v}, \forall u, v \in N^{S'} \tag{5.8}$$

$$R_N(w) \geq x_{mw}\, c(m), \forall m \in N^{S'} \setminus N^S, \forall w \in N^S \tag{5.9}$$

- Flow Related Constraints:

$$\sum_{w \in N^{S'}} f_{uw}^i - \sum_{w \in N^{S'}} f_{wu}^i = 0, \forall i, \forall u \in N^{S'} \setminus \{s_i, t_i\} \tag{5.10}$$

$$\sum_{w \in N^{S'}} f_{s_i w}^i - \sum_{w \in N^{S'}} f_{w s_i}^i = b(e_i^V), \forall i \tag{5.11}$$

$$\sum_{w \in N^{S'}} f_{t_i w}^i - \sum_{w \in N^{S'}} f_{w t_i}^i = -b(e_i^V), \forall i \tag{5.12}$$

- Meta and Binary Constraints:

$$\sum_{w \in \Omega(m)} x_{mw} = 1, \forall m \in N^{S'} \setminus N^S \tag{5.13}$$

$$\sum_{m \in N^{S'} \setminus N^S} x_{mw} \leq 1, \forall w \in N^S \tag{5.14}$$

$$x_{uv} \leq R_E(u,v), \forall u,v \in N^{S'} \tag{5.15}$$

$$x_{uv} = x_{vu}, \forall u,v \in N^{S'} \tag{5.16}$$

- Domain Constraints:

$$f_{uv}^i \geq 0, \forall u,v \in N^{S'} \tag{5.17}$$

$$x_{uv} \in \{0,1\}, \forall u,v \in N^{S'} \tag{5.18}$$

Remarks:

- The objective function (5.7) of the MIP tries to minimize the cost of embedding the VN request as well as balance the load. By dividing the cost with the residual capacity, it is ensured that the resources with more residual capacities are preferred over the resources with less residual capacities. $1 \leq \alpha_{uv} \leq R_E(u,v)$ and $1 \leq \beta_w \leq R_N(w)$ are parameters to control the importance of load balancing while embedding a request. $\delta \to 0$ is a small positive constant to avoid dividing by zero in computing the objective function.
- Constraint set (5.8) and (5.9) contains the node and edge capacity bounds. Summing up f_{uv}^i and f_{vu}^i in (5.8) ensures that the summation of flows on both directions of the undirected edge (u,v) remains within its available bandwidth.
- Constraint sets (5.13) and (5.14) are related to the augmented portion of the substrate graph. Constraint set (5.13) makes sure that only one substrate node is selected for each meta-node, whereas constraint set (5.14) ensures that no more than one meta-node is placed on a substrate node.
- Constraint sets (5.15) and (5.16) together with (5.4) ensure that x_{uv} is set whenever there is any flow in either direction of the substrate edge (u,v).

5.5 LP Relaxation and Rounding-Based Algorithms

Since solving an MIP is known to be computationally intractable [93], simultaneous node and link embedding using Program 5.1 is practically infeasible. Hence we relax the integer constraints (5.18) of the MIP, and obtain the following linear program (Program 5.2). Once we have the LP solution, we use deterministic and randomized rounding techniques to obtain

integer values for the variable x and embed VN requests.

Program 5.2 (Relaxed Linear Program for Virtual Network Embedding)
Objective:

$$\begin{aligned} \text{minimize} \quad & \sum_{uv \in E^S} \frac{\alpha_{uv}}{R_E(u,v)+\delta} \sum_i f_{uv}^i \\ & + \sum_{w \in N^S} \frac{\beta_w}{R_N(w)+\delta} \sum_{m \in N^{S'} \setminus N^S} x_{mw}\, c(m) \end{aligned} \tag{5.19}$$

Constraints:

- Capacity Constraints:

$$\sum_i (f_{uv}^i + f_{vu}^i) \leq R_E(u,v)\, x_{u,v}, \forall u, v \in N^{S'}$$
$$R_N(w) \geq x_{mw}\, c(m), \forall m \in N^{S'} \setminus N^S, \forall w \in N^S$$

- Flow Related Constraints:

$$\sum_{w \in N^{S'}} f_{uw}^i - \sum_{w \in N^{S'}} f_{wu}^i = 0, \forall i, \forall u \in N^{S'} \setminus \{s_i, t_i\}$$
$$\sum_{w \in N^{S'}} f_{s_i w}^i - \sum_{w \in N^{S'}} f_{w s_i}^i = b(e_i^V), \forall i$$
$$\sum_{w \in N^{S'}} f_{t_i w}^i - \sum_{w \in N^{S'}} f_{w t_i}^i = -b(e_i^V), \forall i$$

- Meta and Relaxed Binary Constraints:

$$\sum_{w \in \Omega(m)} x_{mw} = 1, \forall m \in N^{S'} \setminus N^S$$
$$\sum_{m \in N^{S'} \setminus N^S} x_{mw} \leq 1, \forall w \in N^S x_{uv} \leq R_E(u,v), \forall u, v \in N^{S'}$$
$$x_{uv} = x_{vu}, \forall u, v \in N^{S'}$$

- Domain Constraints:

$$f_{uv}^i \geq 0, \forall u, v \in N^{S'}$$
$$x_{uv} \geq 0, \forall u, v \in N^{S'} \tag{5.20}$$

Remarks:

- The domain constraint set (5.20) on the x_{uv} variables has been relaxed.

5.5.1 Deterministic Rounding-based VN Embedding Algorithm (D-ViNE)

D-ViNE (Algorithm 5.3) takes online VN requests as inputs and maps them onto the substrate network one at a time. It takes decisions based only on the past VN requests that it has already seen, i.e., D-ViNE uses no look-ahead. Since the integer domain constraints (5.18) on the x variables have already been relaxed, we no longer get integer values for the x variables. Instead, we employ deterministic rounding technique to get integer values for x. We introduce $\varphi : N^S \rightarrow \{0,1\}$, which is initially set to zero for all $n^S \in N^S$ signifying that all the substrate nodes are initially unused. Whenever a virtual node is mapped to a particular physical node n^S, we set $\varphi(n^S)$ to 1 to ensure that no substrate node is used twice for the same VN request.

Algorithm 5.3 D-ViNE: Deterministic Rounding-based Virtual Network Embedding Algorithm

procedure D-ViNE($G^V = (N^V, E^V)$)
  Create augmented substrate graph $G^{S'} = \left(N^{S'}, E^{S'}\right)$
  Solve Program 5.2
  for all $n^S \in N^S$ **do**
    $\varphi(n^S) \leftarrow 0$
  end for
  for all $n \in N^V$ **do**
    if $\Omega(n) \cap \{n^S \in N^S | \varphi(n^S) = 1\} = \emptyset$ **then**
      VN request cannot be satisfied
      return
    end if
    for all $z \in \Omega(n)$ **do**
      $p_z \leftarrow (\sum_i f^i_{\mu(n)z} + f^i_{z\mu(n)}) x_{\mu(n)z}$
    end for
    Let $z_{max} = \arg\max_{z \in \Omega(n)} \{p_z | \varphi(z) = 0\}$ ▷ break ties arbitrarily
    set $\mathcal{M}_N(n) \leftarrow z_{max}$ ▷ Map n to z_{max}
    $\varphi(z_{max}) \leftarrow 1$
  end for
  Solve MCF to map virtual edges.
  Update residual capacities of the network resources.
end procedure

Description and Discussion

The procedure begins by creating an augmented substrate graph, $G^{S'} = \left(N^{S'}, E^{S'}\right)$ for the VN request $G^V = \left(N^V, E^V\right)$ using the augmentation method described in Section 5.4.1. Next it solves Program 5.2 to get a fractional solution which is at least as good as the integer solution of Program 5.1. For each virtual node, D-ViNE first checks whether there are any unmapped substrate nodes within its feasible region (the substrate nodes in the virtual nodes Ω-Set). If the corresponding Ω-Set is empty, D-ViNE stops the embedding process immediately and rejects the VN request. Otherwise the *deterministic rounding procedure* is initiated in line 12.

For each virtual node n, D-ViNE calculates a value p_z for each substrate node $z \in \Omega(n)$ in its cluster. p_z is calculated as the product of the value $x_{\mu(n)z}$ and the total flow passing through the meta-edge $(\mu(n), z)$ in both directions. The reason behind using this multiplication instead of just $x_{\mu(n)z}$ is as follows:

In the MIP solution x_{uv} is set to binary values based on the presence of flows in either direction in the edge (u, v). When the binary constraint x is relaxed, one might expect that the fractional values of x_{uv} would also be proportional to the total flow in the edge (u, v). But during the LP relaxation process, the correlation between the flow variable f and the binary variable x is lost. It is because a linear program tries to optimize the objective function without violating the constraints; it does not care about the values as long as they are within their permitted domains. As a result, in the relaxed linear program, it is possible that the f values are very high and the corresponding x values are very low or vice versa. Multiplying the f and x values thwarts the possibility of selecting a substrate node based on high x value but very low f value on its corresponding meta-edge and vice versa. The ones that have better values for both the variables f and x are more likely to be in the solution of the MIP than others. D-ViNE maps the virtual node n onto the unmapped substrate node z (i.e., $\varphi(z) = 0$) with the highest p_z value, breaking ties arbitrarily.

Once all the virtual nodes have been mapped to different substrate nodes, D-ViNE applies the multi-commodity flow algorithm to map the virtual edges in E^V onto the substrate paths. One can also use shortest path algorithms when path splitting is not supported by the substrate network. Finally, D-ViNE updates the residual capacities of the substrate nodes and links to prepare for the next request.

Time Complexity

An important aspect of D-ViNE is that the multi-commodity flow algorithm is executed twice; first, during the node mapping phase (since Program 5.2 is a linear programming relaxation of the original mixed integer multi-commodity flow problem), and second, during the edge mapping phase. It can easily be seen that D-ViNE runs in polynomial-time, since the

multi-commodity flow algorithm can be solved in polynomial-time using either the ellipsoid algorithm or Karmarkar's interior point algorithm for linear programming [93].

Algorithm 5.4 R-ViNE: Randomized Rounding-based Virtual Network Embedding Algorithm

procedure R-ViNE($G^V = (N^V, E^V)$)
  Create augmented substrate graph $G^{S'} = \left(N^{S'}, E^{S'}\right)$
  Solve Program 5.2
  for all $n^S \in N^S$ **do**
    $\varphi(n^S) \leftarrow 0$
  end for
  for all $n \in N^V$ **do**
    if $\Omega(n) \cap \{n^S \in N^S | \varphi(n^S) = 1\} = \emptyset$ **then**
      VN request cannot be satisfied
      return
    end if
    for all $z \in \Omega(n)$ **do**
      $p_z \leftarrow (\sum_i f^i_{\mu(n)z} + f^i_{z\mu(n)}) x_{\mu(n)z}$
    end for
    $p_{sum} \leftarrow \sum_{z \in \Omega(n)} p_z$
    for all $z \in \Omega(n)$ **do**
      $p_z \leftarrow p_z / p_{sum}$
    end for
    set $\mathcal{M}_N(n) \leftarrow z$ with probability p_z ▷ $\sum_{z \in \Omega(n)} p_z = 1$
    $\varphi(z) \leftarrow 1$ with probability p_z
  end for
  solve MCF to map virtual edges.
  Update residual capacities of the network resources.
end procedure

5.5.2 Randomized Rounding-based VN Embedding Algorithm (R-ViNE)

R-ViNE is quite similar to D-ViNE except that it uses randomized rounding instead of deterministic rounding. Once the p_z values are calculated as in D-ViNE, R-ViNE normalizes those values within 0 to 1 range. The normalized values for each $z \in \Omega(n)$ correspond to the probabilities of n being mapped to z by the optimal MIP. R-ViNE selects a substrate node $z \in \Omega(n)$ to map a virtual node n with probability p_z. The remainder of this algorithm is similar to its deterministic counterpart, and it is clear that this algorithm also runs in polynomial-time.

5.6 Performance Evaluation

In this section, we first describe the evaluation environment, and then present our main evaluation results. Our evaluation focuses primarily on quantifying the advantage of coordinating node mapping and link mapping phases in terms of acceptance ratio, revenue and cost. We also compare D-ViNE and R-ViNE with existing algorithms modified to fit into our model. We also present a study of the affect of arrival rate changes on the performance of different algorithms.

5.6.1 Simulation Settings

We have implemented a discrete event simulator to evaluate the performance of our algorithms which is freely available at [7]. Unless otherwise specified, we use the following settings in our simulation experiments:

The substrate network topologies in our experiments are randomly generated with 50 nodes using the GT-ITM tool [118] in (25×25) grids. Each pair of substrate nodes are randomly connected with probability 0.5. The cpu and bandwidth resources of the substrate nodes and links are real numbers uniformly distributed between 50 and 100. We assume that VN requests arrive in a Poisson process with an average rate of 4 VNs per 100 time units, and each one has exponentially distributed lifetime with an average of $\mu = 1000$ time units. In each VN request, the number of virtual nodes is randomly determined by a uniform distribution between 2 and 10 following similar setups to previous works [119, 117]. The average VN connectivity is fixed at 50%. The cpu requirements of the virtual nodes are real numbers uniformly distributed between 0 to 20 and the bandwidth requirements of the virtual links are uniformly distributed between 0 to 50. We have used the open source mixed integer programming library glpk [2] to solve Program 5.2.

5.6.2 Performance Metrics

We have used the following five metrics in our evaluations to measure the performance of our algorithms against the existing ones:

1. **Acceptance Ratio:** Acceptance ratio of an algorithm measures the percentage of total VN requests accepted by that algorithm over a given period. While it gives a sense of how well an algorithm is performing, it cannot completely capture the performance when the ultimate goal of an InP is to increase its revenue. An algorithm can accept many smaller or less profitable VN requests to increase this ratio without actually maximizing the overall revenue and leaving the resources under-utilized.

Table 5.1: Summary of Compared VN Embedding Algorithms

Notation	Algorithm Description
D-ViNE	Deterministic Node Mapping with Splittable Link Mapping using MCF
R-ViNE	Randomized Node Mapping with Splittable Link Mapping using MCF
G-SP [119]	Greedy Node Mapping with Shortest Path Based Link Mapping
G-MCF [117]	Greedy Node Mapping with Splittable Link Mapping using MCF
D-ViNE-SP	Deterministic Node Mapping with Shortest Path Based Link Mapping
D-ViNE-LB	Deterministic Node Mapping with Splittable Link Mapping using MCF, where $\alpha_{uv} = \beta_w = 1, \forall\, u, v, w \in N^S$

2. **Generated Revenue ($\mathbb{R}$):** We also measure the generated revenue (defined in (5.5)) of an embedding algorithm over time. An algorithm can be considered to be performing better than its counterparts, when it generates more revenue in addition to a higher acceptance ratio.
3. **Provisioning Cost ($\mathbb{C}$):** We measure the cost (defined in (5.6)) that an algorithm incurs to embed a particular VN request, which is particularly useful to calculate the cost-revenue ratio of an embedding. This ratio can later be used for admission control purposes.
4. **Average Node Utilization:** Average node utilization of the substrate network is measured by averaging the stress (defined in (5.1)) of all the substrate nodes.
5. **Average Link Utilization:** Similarly, average link utilization is the average of the link stresses (defined in (5.2)) of all the substrate edges.

5.6.3 Compared Algorithms

In our evaluation, we have compared six algorithms that combine different node mapping and link mapping strategies including our contributions and algorithms from previous research [119, 117] modified to fit into our model (i.e., no reassignment). The notations that we use to refer to different algorithms are enumerated in Table 5.1.

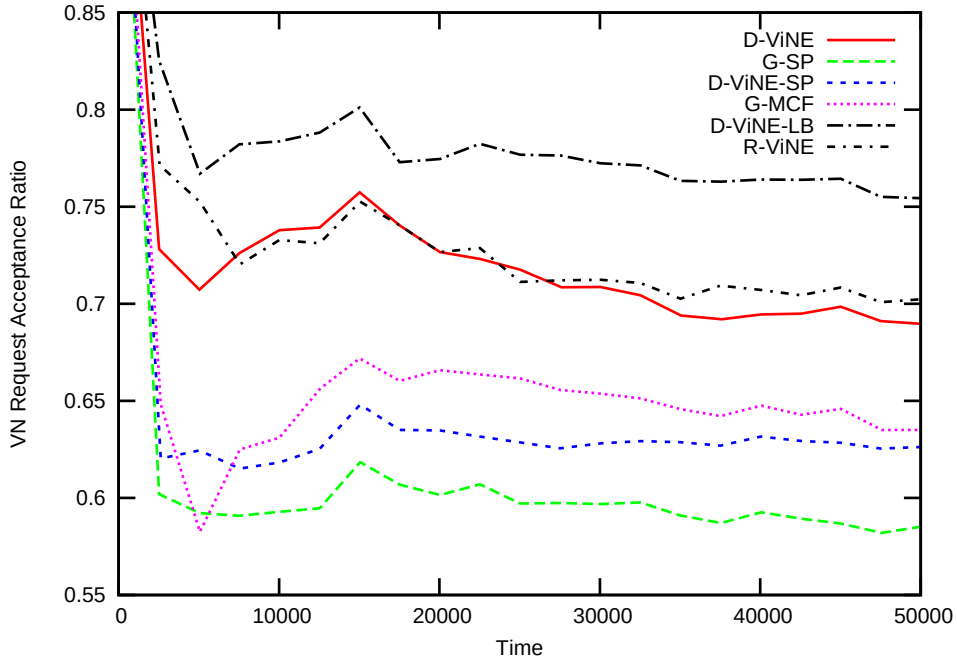

Figure 5.3: VN Request Acceptance Ratio Over Time

5.6.4 Time-based Evaluation

We plot the performance metrics defined in Section 5.6.2 against time to show how each of the compared algorithms (Table 5.1) actually perform in the long run. We summarize our key observations in the following:

(1) Coordinated Node and Link Mapping Leads to Higher Acceptance Ratio and Larger Revenue: Figure 5.3 and Figure 5.4 depict that the proposed algorithms, D-ViNE and R-ViNE, lead to better acceptance ratio as well as higher revenue than the existing algorithms (G-SP and G-MCF) through coordinated node and link mapping. Having higher revenue along with better acceptance ratio implies that our proposed algorithms actually embed VN requests that generate more revenue, instead of only embedding smaller VN requests just to increase the acceptance ratio.

(2) Load Balancing Further Increases the Acceptance Ratio and the Revenue: From Figure 5.3 and Figure 5.4, it is also evident that D-ViNE-LB generates more revenue and accepts more VN requests than the basic D-ViNE algorithm. In D-ViNE-LB, the value of the objective function (5.19) of Program 5.2 depends on the residual capacity of the network resources in addition to the provisioning cost (α and β values are set to 1 here). The lower the residual capacity of a particular node or link, the higher the value of the objective function. As a result, D-ViNE-LB tries to avoid highly utilized nodes and links as long as it

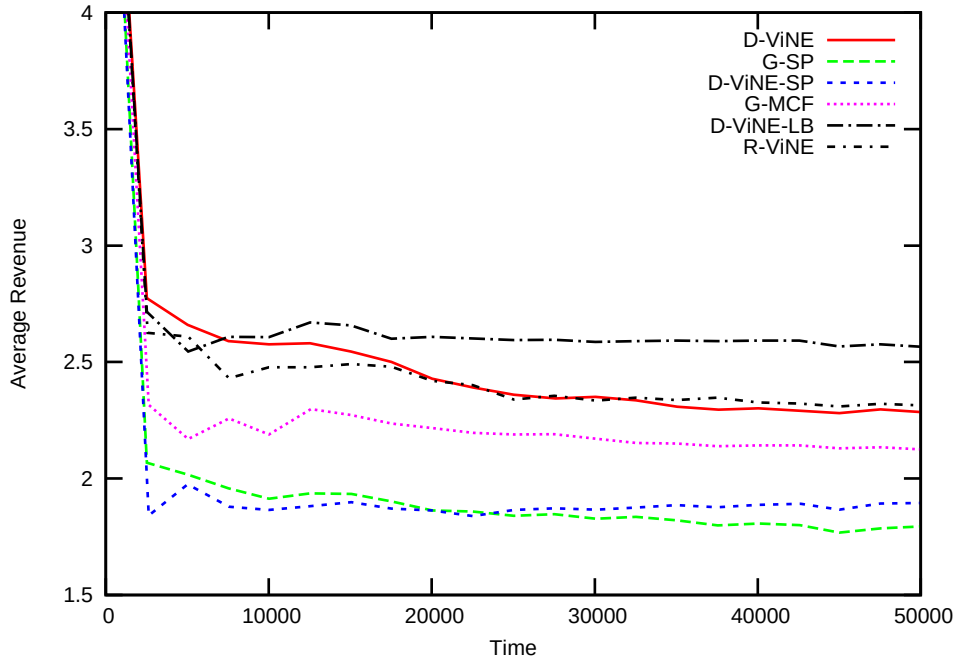

Figure 5.4: Time Average of Generated Revenue

can, leaving those critical resources available for the VN requests that absolutely need them.

(3) Randomization Leads to Better Performance: It is well established in the algorithm design literature that randomization allows efficient solutions to many intractable problems in polynomial time with low probability of error. Our experiments show that the randomized version of our VN embedding algorithm (R-ViNE) performs better than its deterministic counterpart (D-ViNE) in terms of acceptance ratio and revenue generation (Figure 5.3 and Figure 5.4).

In addition to that, for networks with large number of nodes randomization has been shown to be effective for load balancing [77]. This phenomena is also visible in our experiments, since R-ViNE performs similar to D-ViNE-LB in most scenarios.

(4) Load Balancing Slightly Increases the Average Provisioning Cost: While load balancing increases revenue and acceptance ratio by avoiding highly utilized resources, it runs the risk of increasing the average provisioning cost as shown in Figure 5.5. Since D-ViNE-LB tries to avoid highly utilized resources, sometimes it ends up suggesting a longer path to map a particular virtual edge which eventually sums up to slightly higher average provisioning cost in the long run.

(5) Coordination Increases Resource Utilization: Figure 5.6 and Figure 5.7 depict the average utilization of substrate nodes and substrate links for different VN embedding

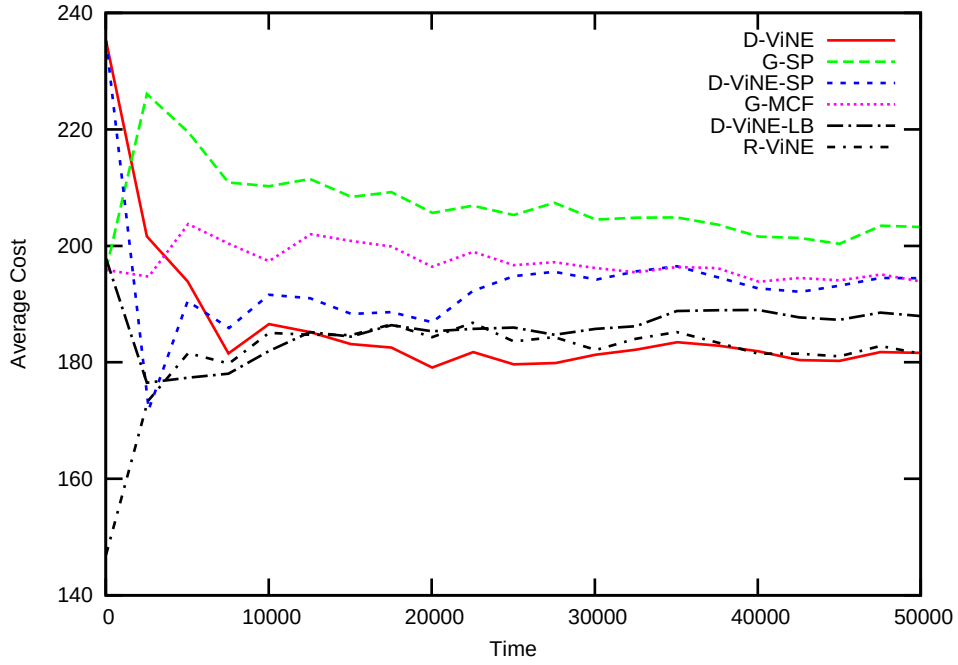

Figure 5.5: Average Cost of Accepting VN Requests Over Time

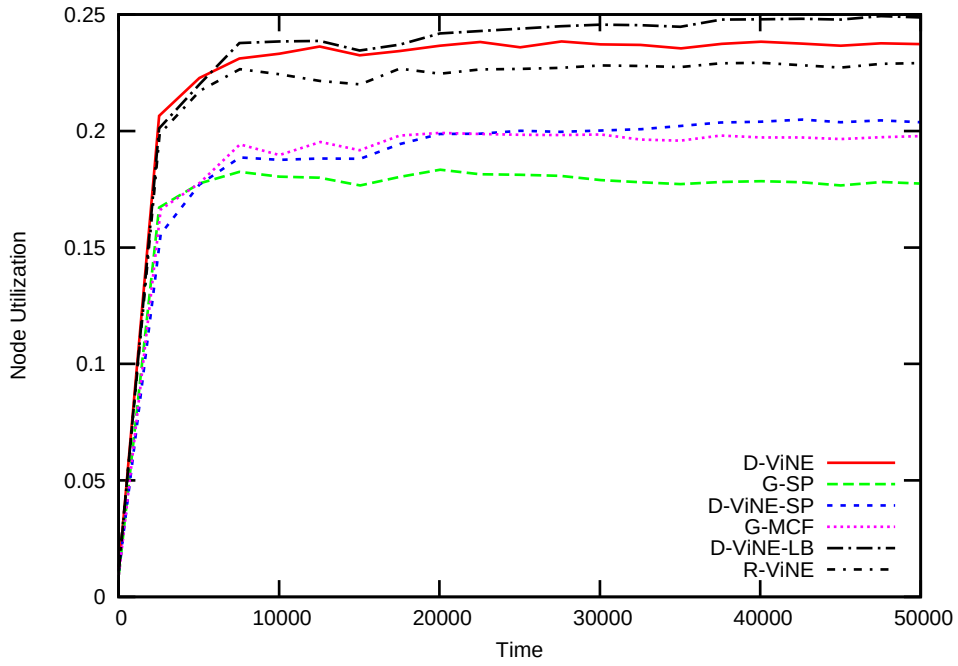

Figure 5.6: Average Node Utilization

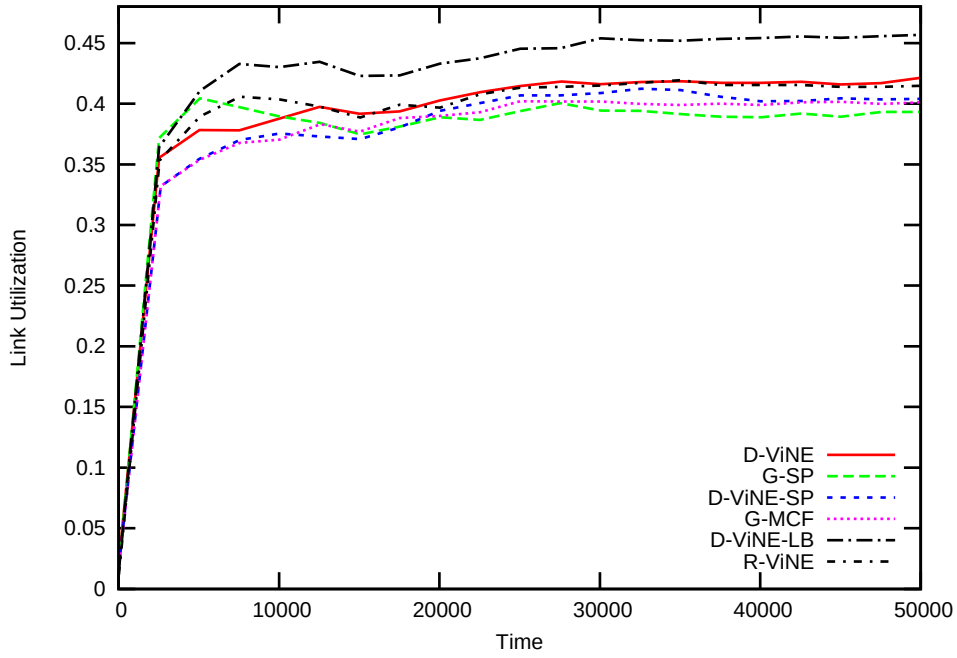

Figure 5.7: Average Link Utilization

algorithms. Since D-ViNE-LB has the highest acceptance ratio, naturally it also has the highest node and link utilization.

However, D-ViNE-LB achieves a relatively higher gain in link utilization over its counterparts than in node utilization. We believe that the reason behind this is the distributive nature of D-ViNE-LB algorithm. In order to avoid links with lower residual capacities, i.e., in order to minimize (5.7), D-ViNE-LB uses longer paths containing more substrate links with higher residual capacities to embed virtual links.

5.6.5 Effect of Increasing Load on VN Embedding Algorithms

In order to evaluate the scalability of the proposed algorithms with increasing load, we change the VN arrival rate from 4 requests per 100 time units to 8 requests per 100 time units while keeping the average VN lifetime fixed. For each arrival rate, we run the simulations for sufficiently long time until they reach a stable state and plot each of the performance metric's stable value against arrival rate (Figure 5.8). Our findings are:

(1) Dominance of the Proposed Algorithms is Not Diminished by Load: As evident from Figure 5.8, all the proposed algorithms (D-ViNE, R-ViNE, and D-ViNE-LB) maintain their relative superior performance in terms of the total number of accepted VN requests, revenue, cost, and average node and link utilization with increasing load. As we have noticed before,

D-ViNE and R-ViNE maintain a close resemblance while D-ViNE-LB brings in more revenue without any significant changes with load.

(2) Increasing Load Leads to Slightly Better Cost-Revenue Ratio: With increasing load all the algorithms under consideration tend to achieve better cost-revenue ratios (Figure 5.8(b) and Figure 5.8(c)). However, the proposed algorithms have steeper decreases in their average costs with more or less similar increases in their revenues, resulting in higher cost-revenue improvement than the existing algorithms.

(3) Relative Link Utilization of D-ViNE-LB Worsens with Load: As the arrival rate is increased from 4 to 8 VN requests per 100 time units, average link utilization of D-ViNE-LB increases much quickly than the total number of VN requests accepted by it in comparison with D-ViNE and R-ViNE.

For example, in terms of the total number of accepted VN requests D-ViNE-LB's advantage over D-ViNE diminishes by 3% when the arrival rate is increased from 4 to 8 requests in 100 time units (Figure 5.8(a)), while the extra link resource usage increases by almost 2% (Figure 5.8(e)). At this rate, D-ViNE-LB will run out of resources faster than D-ViNE when the load (i.e., the arrival rate) is too high.

5.7 Related Work

The VN assignment problem is similar to the previous works on embedding VPNs in a shared provider topology and the network testbed mapping problem [55, 87]. However, a typical VPN request consists only of bandwidth requirements, specified in terms of a traffic matrix, without any constraint on its nodes. As a result, most VPN designing algorithms come down to finding paths for source/destination pairs. On the other hand, the *Assign* algorithm [87] used in the Emulab testbed considers bandwidth constraints alongside constraints on exclusive use of nodes, i.e., different VNs cannot share a substrate node. But in network virtualization, there are capacity and placement requirements on both the virtual nodes and the virtual links; in addition, substrate nodes and links can be shared by multiple VNs.

In order to reduce the hardness of the VN assignment problem and to enable efficient heuristics, existing research has been restricting the problem space in different dimensions, which include:

1. considering offline version of the problem (i.e., all the VN requests are known in advance) [74, 119],
2. ignoring either node requirements or link requirements [43, 74],
3. assuming infinite capacity of the substrate nodes and links to obviate admission control [43, 74, 119], and

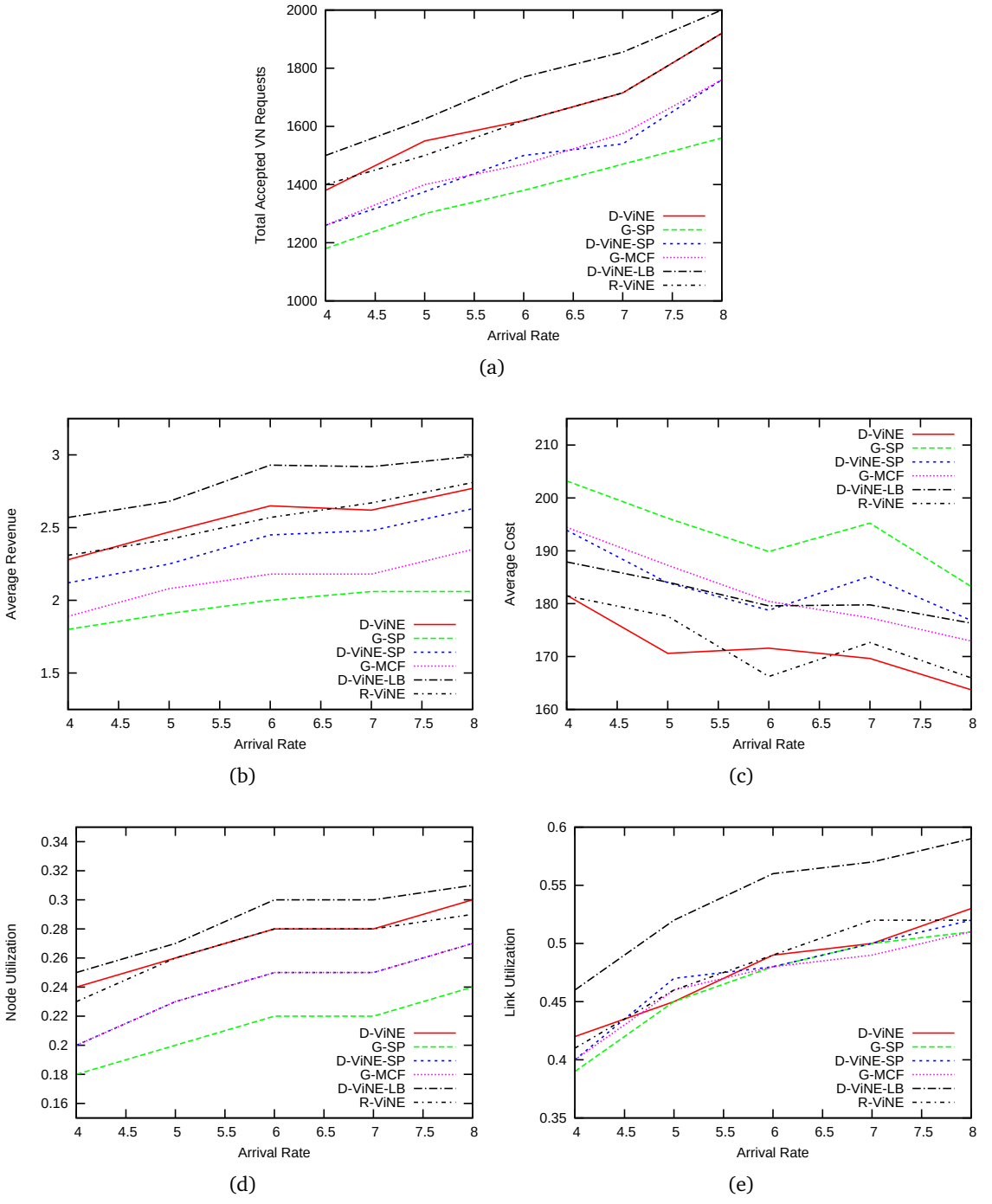

Figure 5.8: Effect of Increasing Load on Compared VN Embedding Algorithms in Terms of: (a) Total Number of Accepted VN Requests; (b) Revenue; (c) Provisioning Cost; (d) Average Node Utilization; and, (e) Average Link Utilization.

4. focusing on specific VN topologies [74].

The authors in [117] consider all these issues, except for the location constraints on the virtual nodes, by envisioning support from the substrate network through node and link migration as well as multi-path routing.

However, contrary to the algorithms proposed in this chapter, all the existing algorithms can clearly be separated into two basic phases:

1. assigning virtual nodes using some greedy heuristics, e.g., assign virtual nodes with higher processing requirements to substrate nodes with more available resources [119, 117], and
2. embedding virtual links onto substrate paths using shortest path algorithms [119] in case of unsplittable flows, or using *multi-commodity flow* algorithms in case of splittable flows [99, 117].

The authors in [61] have proposed a distributed algorithm that simultaneously maps virtual nodes and virtual links without any centralized controller, but the scalability and performance of their algorithm is still not comparable with the centralized ones.

In this chapter, we take a formal approach to solve the online VN embedding problem using a MIP formulation under realistic assumptions of node and link constraints. To the best of our knowledge, this is the first attempt to apply mathematical programming to this problem in the network virtualization context. We do not restrict the problem space by assuming infinite capacity of the substrate network resources, nor do we assume any specialized VN topologies.

5.8 Summary

To make network virtualization an integral part of the future Internet architecture, efficient and practical algorithms for VN embedding are specially required. In this chapter, we proposed algorithms for VN embedding that differ from the previous algorithms by introducing coordination between node and link mapping phases. We argued that this coordination greatly increases the solution space and the quality of the heuristic algorithms. To this end we first formulated the embedding problem as a mixed integer program. We then relaxed the integer constraints and used deterministic and randomized rounding techniques to obtain polynomial-time solvable algorithms for node mapping. The node mapping phase combined with the multi-commodity flow based link mapping phase in our algorithms outperformed the existing approaches in terms of acceptance ratio, revenue, and provisioning cost, as shown through extensive simulation.

Chapter 6

Epilogue

Amid current trends of virtualizing practically every aspect of computing, ranging from operating systems, storage systems to servers, and even large data centers (e.g., cloud computing), network virtualization stands at a unique point in the virtualization design space. On the one hand, it is necessary to have a virtualized network to interconnect all other virtualized appliances to give each of the virtual entities a complete semblance of their native counterparts.

On the other hand, after enjoying years of rapid growth, the progress of the Internet and networking in general has come to a standstill. Most researchers now agree that a redesign is a bare necessity, not luxury [47]. Network virtualization can take the leading role in this scenario to promote innovation, to provide flexibility, and to introduce heterogeneity.

However, realization of the network virtualization environment must satisfy the requirements set by its characteristics and design goals. Even though these requirements will ensure an open, flexible, and heterogeneous networking environment, they will also pose a string of challenges that will require coordinated attention from the researchers working in networking and other related fields.

In this thesis, we have addressed one inter-domain and another intra-domain problem in the network virtualization environment. We have proposed *iMark*, an identity management framework for the network virtualization environment, to manage the identities in a heterogeneous virtualized environment without enforcing any constraint on the naming and addressing choices of the participant physical and virtual networks. For resource allocation in a one infrastructure provider environment, we have devised two virtual network embedding algorithms (*D-ViNE* and *R-ViNE*) based on a mathematical formulation of the embedding problem with an aim to increase the utilization of the physical network resources and the revenue of the infrastructure provider. Through extensive simulation we have demonstrated the validity and the performance of both the contributions. In addition, we have presented a

comprehensive survey of the past, the present, and the future of network virtualization with an objective to stoke wide interests among the researchers in this field.

6.1 Summary of Contributions

The contributions of this thesis to the network virtualization literature are summarized below:

1. The **Survey of Network Virtualization** provides the following:
   - *Historical perspective* of network virtualization under the light of existing research areas (e.g., VLANs, VPNs etc.) that are closely related to the concepts of flexibility, heterogeneity, programmability, isolation and other design goals of network virtualization.
   - *Categorization* of the previous and on-going research projects that address different aspects of network virtualization.
   - *Enumeration of the open problems* in the network virtualization environment accompanied by the summary of the existing works on those problems in the networking literature.
2. The **Identity Management Framework** delivers the following features:
   - *Interoperability: iMark* ensures end-to-end connectivity in the network virtualization environment through interoperability between heterogeneous identifier spaces.
   - *Flexibility:* No explicit requirements are placed on the choice of internal naming and addressing mechanisms of the concerned networks.
   - *Mobility and Überhoming: iMark* inherently supports mobility and überhoming for all the end hosts.
3. The **Virtual Network Embedding Algorithms** offer the following:
   - *Better Embedding Quality:* Both *D-ViNE* and *R-ViNE* outperform the existing embedding algorithms in terms of VN request acceptance ratio, utilization of resources, and cost as well as revenue of the infrastructure providers.
   - *Load Balancing: D-ViNE-LB*, the load-balanced version of *D-ViNE*, significantly increases the acceptance ratio, utilization, and revenue with relatively low increase of the provisioning cost.
   - *Mathematical Foundation:* A MIP formulation of the embedding problem has been presented with the help of graph augmentation and clustering techniques. This formulation can be used to develop other embedding algorithms in addition to the ones presented here.

6.2 Future Research

There are several possible research avenues that directly follow from our work. In the following, we will discuss some of the most important ones:

iMark Prototype Development

In this thesis, we have justified iMark's design and architecture through numerical simulation studies due to the absence of a realistic prototype of the network virtualization environment. However, such experiments can only provide a coarse overview of the internal functionalities; they do not capture the packet level behavior that might arise in a realistic scenario.

- In order to investigate such micro-level behavior, design and implementation of a full-fledged prototype of the network virtualization environment is of utmost importance. Such a prototype will not only facilitate iMark's validation, but also allow design, development, and validation of many other projects on network virtualization.
- In addition to validating iMark, we have demonstrated its scalability over a large number of end hosts. Additional performance and robustness aspects of iMark along with possible optimizations, e.g., caching, to improve its scalability can be investigated. The effects of end host mobility and überhoming on iMark in a heterogeneous NVE.

Theoretical Analysis of D-ViNE and R-ViNE

While our approach of coordinating the node mapping and the link mapping phases to improve VN embedding quality is, to the best of our knowledge, the first of its kind, a number of issues remain unresolved in this work and can be good starting points for further research in this direction.

- First and foremost is the analysis of the theoretical approximation factors of the proposed algorithms in the worst case. Developing a primal-dual based analysis framework to obtain lower bounds on the performance of D-ViNE and R-ViNE can be an interesting research topic from theoretical computer science perspective.
- Finding out advanced economic models, instead of the simple revenue model used in the existing literature, for pricing in the network virtualization environment can be another important research topic that needs further attention.
- Finally, available approaches (e.g., column generation technique) to directly solve integer and mixed integer programs can be employed to develop efficient algorithms to obtain optimal or near-optimal solutions for the original mixed integer formulation (Program 5.1) without any relaxation.

Cross Domain Virtual Network Embedding

In this thesis, we have addressed the online *intra-domain* virtual network embedding problem, i.e., we have considered only a single infrastructure provider. However, an end-to-end virtual network can span over administrative domains of multiple infrastructure providers.

- In order to create end-to-end virtual networks over geographically distributed regions, service providers need to contact multiple infrastructure providers who have available resources in those regions.
  In order to enable free trading and fair pricing, a framework is required in the form of an open marketplace that will enable service providers and infrastructure providers to trade resources at competitive prices.
- Each infrastructure provider is interested only in maximizing its own profit; usually it means, getting requests for their high-margin equipments while offloading unprofitable work onto their competitors. Such selfish behavior can give rise to tussles between adjoining infrastructure providers.
  Theoretical studies from *game theoretic* and *mechanism design* perspectives can give insights into the behavior trends, which can later be used to formulate strategies as found in the existing traffic engineering literature.

Bibliography

The numbers at the end of each entry list pages where the reference was cited. In the electronic version, they are clickable links to the pages.

[1] GENI: Global Environment for Network Innovations. http://www.geni.net/. 14, 17

[2] GNU Linear Programming Kit. http://www.gnu.org/software/glpk/. 64

[3] Open Signaling working group. http://comet.columbia.edu/opensig/. 8

[4] OpenVPN: An open source SSL VPN solution. http://openvpn.net/. 15

[5] PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services. http://www.planet-lab.org/. 9, 14, 17, 25

[6] UCLP: User controlled lightpaths project. http://uclp.uwaterloo.ca/. 11, 17

[7] ViNE-Yard. http://www.mosharaf.com/ViNE-Yard.tar.gz. 64

[8] VINI: A virtual network infrastructure. http://www.vini-veritas.net/. 15, 17, 25

[9] Virtuoso: Resource management and prediction for distributed computing using virtual machines. http://virtuoso.cs.northwestern.edu/. 11, 17

[10] David Andersen, Hari Balakrishnan, Frans Kaashoek, and Robert Morris. Resilient overlay networks. *SIGOPS Operating Systems Review*, 35(5):131–145, 2001. 8

[11] David G. Andersen. Mayday: Distributed filtering for Internet services. In *Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems (USITS'03)*, Berkeley, CA, USA, 2003. 8

[12] D.G. Andersen. Theoretical approaches to node assignment. Unpublished Manuscript, http://www.cs.cmu.edu/~dga/papers/andersen-assign.ps, 2002. 27, 51

[13] Thomas Anderson, Larry Peterson, Scott Shenker, and Jonathan Turner. Overcoming the Internet impasse through virtualization. *Computer*, 38(4):34–41, 2005. 1, 9, 16, 23

[14] L. Andersson and T. Madsen. Provider Provisioned Virtual Private Network (VPN) Terminology. RFC 4026, March 2005. 6

[15] L. Andersson and E. Rosen. Framework for Layer 2 Virtual Private Networks (L2VPNs). RFC 4664, September 2006. 7

[16] W. Augustyn and Y. Serbest. Service Requirements for Layer 2 Provider-Provisioned Virtual Private Networks. RFC 4665, September 2006. 7

[17] Hari Balakrishnan, Karthik Lakshminarayanan, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Michael Walfish. A layered naming architecture for the Internet. In *ACM SIGCOMM*, pages 343–352, 2004. 49

[18] Andy Bavier, Nick Feamster, Mark Huang, Larry Peterson, and Jennifer Rexford. In VINI veritas: Realistic and controlled network experimentation. In *Proceedings of SIGCOMM*, pages 3–14, 2006. 15, 17

[19] Driss Benhaddou and Wesam Alanqar. Layer 1 virtual private networks in multidomain next-generation networks. *IEEE Communications Magazine*, 45(4):52–58, April 2007. 7, 31

[20] Sapan Bhatia, Murtaza Motiwala, Wolfgang Mühlbauer, Vytautas Valancius, Andy Bavier, Nick Feamster, Larry Peterson, and Jennifer Rexford. Hosting virtual networks on commodity hardware. Technical Report GT-CS-07-10, Georgia Tech, January 2008. 15

[21] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An architecture for differentiated services. RFC 2475, December 1998. 12

[22] M. Boucadair, Bruno Decraene, M.L. Garcia-Osma, A. J. Elizondo, J. Rodriguez Sanchez, B. Lemoine, E. Mykoniati, P. Georgatsos, D. Griffin, J. Spencer, J. Griem, N. Wang, M. Howarth, G. Pavlou, S. Georgoulas, and B. Quoitin. Parallel Internets framework. AGAVE Deliverable (Id: `AGAVE/WP1/FTRD/D1.1/public`), 2006. 1, 12, 17, 21

[23] M. Boucadair, P. Levis, D. Griffin, N. Wang, M. Howarth, G. Pavlou, E. Mykoniati, P. Georgatsos, B. Quoitin, J. Rodriguez Sanchez, and M.L. Garcia-Osma. A framework for end-to-end service differentiation: Network planes and parallel Internets. *IEEE Communications Magazine*, 45(9):134–143, September 2007. 12, 17, 20, 24

[24] R. Boutaba, W. Golab, Y Iraqi, and B. St-Arnaud. Grid-controlled lightpaths for high performance grid applications. *Journal of Grid Computing*, 1(4):387–394, December 2003. 11

[25] R. Boutaba, W. Golab, Y Iraqi, and B. St-Arnaud. Lightpaths on demand: A web services-based management system. *IEEE Communications Magazine*, 42(7), July 2004. 11

[26] B. Braden, D. Clark, and S. Shenker. Integrated services in the Internet architecture: An overview. RFC 1633, June 1994. 12

[27] Matthew Caesar, Miguel Castro, Edmund B. Nightingale, Greg O'Shea, and Antony Rowstron. Virtual ring routing: network routing inspired by dhts. In *Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'06)*, pages 351–362, 2006. 49

[28] Matthew Caesar, Tyson Condie, Jayanthkumar Kannan, Karthik Lakshminarayanan, and Ion Stoica. ROFL: Routing on flat labels. In *ACM SIGCOMM*, pages 363–374, 2006. 49

[29] R. Callon and M. Suzuki. A Framework for Layer 3 Provider-Provisioned Virtual Private Networks (PPVPNs). RFC 4110, July 2005. 6

[30] K.L. Calvert, S. Bhattacharjee, E. Zegura, and J. Sterbenz. Directions in active networks. *IEEE Communications Magazine*, 36(10):72–78, October 1998. 8

[31] A. T. Campbell, M. E. Kounavis, D. A. Villela, J. Vicente, K. Miki, H. G. De Meer, and K. S. Kalaichelvan. Spawning networks. *IEEE Network Magazine*, 13(4):16–30, 1999. 14, 17

[32] M. Carugi and D. McDysan. Service Requirements for Layer 3 Provider Provisioned Virtual Private Networks (PPVPNs). RFC 4031, April 2005. 6

[33] Yang Chu, Sanjay Rao, Srinivasan Seshan, and Hui Zhang. Enabling conferencing applications on the internet using an overlay muilticast architecture. *SIGCOMM Computer Communication Review*, 31(4):55–67, 2001. 8

[34] David D. Clark, Karen Sollins, John Wroclawski, and Ted Faber. Addressing reality: An architectural response to real-world demands on the evolving Internet. *SIGCOMM Computer Communication Review*, 33(4):247–257, 2003. 36

[35] LAN/MAN Standards Committee. IEEE standard for local and metropolitan area networks– virtual bridged local area networks. IEEE Std 802.1Q-2005, May 2006. 6

[36] Jon Crowcroft, Steven Hand, Richard Mortier, Timothy Roscoe, and Andrew Warfield. Plutarch: An argument for network pluralism. In *Proceedings of the ACM SIGCOMM workshop on Future Directions in Network Architecture (FDNA'03)*, pages 258–266, 2003. 48

[37] Sushil da Silva, Danilo Florissi, and Yechiam Yemini. NetScript: A language-based approach to active networks. Technical report, Columbia University, January 1998. 13, 17

[38] Sushil da Silva, Yechiam Yemini, and Danilo Florissi. The NetScript active network system. *IEEE Journal on Selected Areas in Communication*, 19(3):538–551, 2001. 13, 17

[39] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. In *Proceedings of the eighteenth ACM symposium on Operating systems principles (SOSP'01)*, pages 202–215, 2001. 9

[40] D. Decasper and B. Plattner. DAN: Distributed code caching for active networks. In *Proceedings of the IEEE INFOCOM'98*, volume 2, pages 609–616, 1998. 8

[41] Hans Eriksson. MBone: The multicast backbone. *Communications of the ACM*, 37(8):54–60, 1994. 8

[42] Danny McPherson et al. Core network design and vendor prophecies. In *NANOG 25*, June 2003. 28

[43] J. Fan and M. Ammar. Dynamic topology configuration in service overlay networks - a study of reconfiguration policies. In *Proceedings of IEEE INFOCOM*, 2006. 27, 51, 70

[44] Ramy Farha and Alberto Leon-Garcia. A novel peer-to-peer naming infrastructure for next generation networks. In *IPOM*, pages 1–12, 2007. 49

[45] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Generic Routing Encapsulation (GRE). RFC 2784, March 2000. 6

[46] Nick Feamster, Lixin Gao, and Jennifer Rexford. How to lease the Internet in your spare time. *ACM SIGCOMM Computer Communication Review*, 37(1):61–64, 2007. 1, 16, 17, 20, 23, 24, 26

[47] Anja Feldmann. Internet clean-slate design: What and why? *SIGCOMM Computer Communication Review*, 37(3):59–64, July 2007. 16, 73

[48] P. Ferguson and G. Huston. What is a VPN? Technical report, Cisco Systems, 1998. 6

[49] Metin Feridan, Michael Moser, and Axel Tanner. Building an abstraction layer for management systems integration. In *Proceedings of the 1st IEEE/IFIP International Workshop on End-to-End Virtualization and Grid Management (EVGM'2007)*, pages 57–60, October 2007. 30

[50] Paul Francis and Ramakrishna Gummadi. IPNL: A NAT-extended Internet architecture. In *ACM SIGCOMM*, pages 69–80, 2001. 48

[51] Jing Fu and Jennifer Rexford. Efficient IP-address lookup with a shared forwarding table for multiple virtual routers. In *ACM CoNEXT*, 2008. 28

[52] Albert Greenberg, Gisli Hjalmtysson, David A. Maltz, Andy Myers, Jennifer Rexford, Geoffrey Xie, Hong Yan, Jibin Zhan, and Hui Zhang. A clean slate 4d approach to network control and management. *SIGCOMM Computer Communication Review*, 35(5):41–54, 2005. 30

[53] M. Gritter and D. Cheriton. An architecture for content routing support in the Internet. In *USITS*, pages 37–48, 2001. 48

[54] GENI Planning Group. GENI design principles. *Computer*, 39(9):102–105, 2006. 14, 15, 17

[55] Anupam Gupta, Jon M. Kleinberg, Amit Kumar, Rajeev Rastogi, and Bulent Yener. Provisioning a virtual private network: A network design problem for multicommodity flow. In *Proceedings of ACM STOC*, pages 389–398, 2001. 27, 70

[56] K. Hamzeh, G. Pall, W. Verthein, J. Taarud, W. Little, and G. Zorn. Point-to-Point Tunneling Protocol (PPTP). RFC 2637, July 1999. 6

[57] Mark Handley, Eddie Kohler, Atanu Ghosh, Orion Hodson, and Pavlin Radoslavov. Designing extensible IP router software. In *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation (NSDI'05)*, pages 189–202, 2005. 15

[58] David Hausheer and Burkhard Stiller. Auctions for virtual network environments. In *Workshop on Management of Network Virtualisation*, 2007. 32, 42

[59] David Hausheer and Burkhard Stiller. PeerMart: Decentralized auctions for bandwidth trading on demand. http://ercim-news.ercim.org/content/view/100/254/, January 2007. 32

[60] Jiayue He, Rui Zhang-Shen, Ying Li, Cheng-Yen Lee, Jennifer Rexford, and Mung Chiang. Davinci: Dynamically adaptive virtual networks for a customized internet. In *ACM CoNEXT*, 2008. 30

[61] Ines Houidi, Wajdi Louati, and Djamal Zeghlache. A distributed virtual network mapping algorithm. In *Proceedings of IEEE ICC*, pages 5634–5640, 2008. 72

[62] John Jannotti, David K. Gifford, Kirk L. Johnson, M. Frans Kaashoek, and Jr. James W. O'Toole. Overcast: Reliable multicasting with an overlay network. In *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation (OSDI'00)*, pages 197–212, 2000. 8

[63] Anura P. Jayasumana, Qi Han, and Tissa H. Illangasekare. Virtual sensor networks: A resource efficient approach for concurrent applications. In *Proceedings of the International Conference on Information Technology (ITNG'07)*, pages 111–115, 2007. 31

[64] X. Jiang and D. Xu. VIOLIN: Virtual internetworking on overlay infrastructure. Technical Report TR-03-027, Purdue University, 2003. 12, 17

[65] A. Jun and A. Leon-Garcia. A virtual network approach to network resources management. In *Proceedings of the Canadian Conference on Broadband Research (CCBR'98)*, June 1998. 13, 17

[66] A. Jun and A. Leon-Garcia. Virtual network resources management: A divide-and-conquer approach for the control of future networks. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'98)*, volume 2, pages 1065–1070, 1998. 13, 17

[67] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301, December 2005. 6

[68] A. Keromytis, V. Misra, and D. Rubenstein. SOS: Secure overlay services. In *Proceedings of the ACM SIGCOMM Conference (SIGCOMM'02)*, August 2002. 8

[69] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 18(3):263–297, 2000. 15

[70] Matt Kolon. Intelligent logical router service. Technical Report 200097-001, Juniper Networks, October 2004. 28

[71] M.E. Kounavis, A.T. Campbell, S. Chou, F. Modoux, J. Vicente, and Hao Zhuang. The Genesis Kernel: A programming system for spawning network architectures. *IEEE Journal on Selected Areas in Communications*, 19(3):511–526, 2001. 14, 17, 23

[72] Balachander Krishnamurthy, Craig Wills, and Yin Zhang. On the use and performance of content distribution networks. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement (IMW'01)*, pages 169–182, 2001. 9

[73] Aurel A. Lazar and Andrew T. Campbell. Spawning networking architectures (White Paper). Technical report, Columbia University, 1998. 14, 17

[74] Jing Lu and Jonathan Turner. Efficient mapping of virtual networks onto a shared substrate. Technical Report WUCSE-2006-35, Washington University, 2006. 27, 51, 70, 72

[75] Eng Keong Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, 7(2):72–93, 2005. 9

[76] Laurent Mathy, Norbert Egi, Mickael Hoerdt, Adam Greenhalgh, and Mark Handley. (Some) implementation issues for virtual routers. In *Workshop on Management of Network Virtualisation*, 2007. 28

[77] Michael Mitzenmacher, Andrea W. Richa, and Ramesh Sitaraman. The power of two random choices: A survey of techniques and results. In *Handbook of Randomized Computing*, pages 255–312. Kluwer Academic Press, 2001. 67

[78] R. Moskowitz, P. Nikander, P. Jokela, and T. Henderson. Host Identity Protocol. RFC 5201 (Experimental), April 2008. 48

[79] Biswajit Nandy, Don Bennett, Imran Ahmad, Shikharesh Majumdar, and Bill St.Arnaud. User controlled lightpath management system based on a service oriented architecture. `http://www.solananetworks.com/UCLP/files/UCLPv2-SOA.pdf`, 2006. 11, 17

[80] W. Ng, R. Boutaba, and A. Leon-Garcia. Provision and customization of ATM virtual networks for supporting IP services. In *Proceedings of the IEEE ATM Workshop'1999*, pages 205–210, 1999. 13, 17

[81] W. Ng, D. Jun, H. Chow, R. Boutaba, and A. Leon-Garcia. Miblets: A practical approach to virtual network management. In *Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management (IM'99)*, pages 201–215, 1999. 13

[82] Dragos Niculescu. Survey of active network research. http://www.research.rutgers.edu/~dnicules/research/other/active_survey.pdf, 1999. 8

[83] Larry Peterson, Tom Anderson, David Culler, and Timothy Roscoe. A blueprint for introducing disruptive technology into the Internet. *SIGCOMM Computer Communication Review*, 33(1):59–64, 2003. 14, 17

[84] J. Pujol, S. Schmid, L. Eggert, M. Brunner, and J. Quittek. Scalability analysis of the TurfNet naming and routing architecture. In *DIN*, 2005. 48

[85] Barath Raghavan, Kashi Vishwanath, Sriram Ramabhadran, Kenneth Yocum, and Alex C. Snoeren. Cloud control with distributed rate limiting. In *Proceedings of SIGCOMM'07*, pages 337–348, 2007. 27

[86] J. Recio, E. Grasa, S. Figuerola, and G. Junyent. Evolution of the user controlled lightpath provisioning system. In *Proceedings of 7th International Conference on Transparent Optical Networks*, volume 1, pages 263–266, July 2005. 11, 17

[87] R. Ricci, C. Alfeld, and J. Lepreau. A solver for the network testbed mapping problem. *ACM SIGCOMM Computer Communication Review*, 33(2):65–81, April 2003. 70

[88] E. Rosen and Y. Rekhter. BGP/MPLS VPNs. RFC 2547, March 1999. 6

[89] E. Rosen and Y. Rekhter. BGP/MPLS IP Virtual Private Networks (VPNs). RFC 4364, February 2006. 6

[90] Paul Ruth, Xuxian Jiang, Dongyan Xu, and Sebastien Goasguen. Virtual distributed environments in a shared infrastructure. *Computer*, 38(5):63–69, 2005. 12, 17

[91] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: A case for informed Internet routing and transport. *IEEE Internet Computing*, 19(1):50–59, January 1999. 8

[92] S. Schmid, L. Eggert, M. Brunner, and J. Quittek. TurfNet: An architecture for dynamically composable networks. In *Proceedings of the First IFIP TC6 WG6.6 International Workshop on Autonomic Communication (WAC'04)*, 2004. 48

[93] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, Inc., New York, NY, USA, 1986. 52, 59, 63

[94] Gregory Smith, Anmol Chaturvedi, Arunesh Mishra, and Suman Banerjee. Wireless virtualization on commodity 802.11 hardware. In *Proceedings of the second ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WinTECH'07)*, pages 75–82, 2007. 31

[95] Neil Spring, Larry Peterson, Andy Bavier, and Vivek Pai. Using PlanetLab for network research: Myths, realities, and best practices. *SIGOPS Operating Systems Review*, 40(1):17–24, 2006. 14

[96] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet indirection infrastructure. In *Proceedings of the ACM SIGCOMM Conference (SIGCOMM'02)*, pages 73–88, 2002. 49

[97] Lakshminarayanan Subramanian, Ion Stoica, Hari Balakrishnan, and Randy Katz. OverQoS: An overlay based architecture for enhancing internet QoS. In *Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI)*, pages 71–84, March 2004. 8

[98] A. Sundararaj and P. Dinda. Towards virtual networks for virtual machine grid computing. In *Proceedings of the 3rd USENIX Virtual Machine Research and Technology Symposium (VM'04)*, 2004. 11, 17

[99] W. Szeto, Y. Iraqi, and R. Boutaba. A multi-commodity flow based approach to virtual network resource allocation. In *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'03)*, pages 3004–3008, 2003. 27, 52, 72

[100] T. Takeda. Framework and Requirements for Layer 1 Virtual Private Networks. RFC 4847, April 2007. 7

[101] David L. Tennenhouse, Jonathan M. Smith, W. David Sincoskie, David J. Wetherall, and Gary J. Minden. A survey of active network research. *IEEE Communications Magazine*, 35(1):80–86, January 1997. 8

[102] David L. Tennenhouse and David J. Wetherall. Towards an active network architecture. *ACM Computer Communication Review*, 26(2), 1996. 8

[103] Joe Touch and Steve Hotz. The X-Bone. In *In Proceedings of the Third Global Internet Mini-Conference at GLOBECOM'98*, pages 44–52, 1998. 10, 17

[104] Joseph D. Touch, Yu-Shun Wang, Lars Eggert, and Gregory Finn. A virtual internet architecture. Technical Report TR-570, USC/Information Sciences Institute, 2003. 10, 17, 24

[105] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter. Layer Two Tunneling Protocol "L2TP". RFC 2661, August 1999. 6

[106] Zoltán Turányi, András Valkó, and Andrew T. Campbell. 4+4: An architecture for evolving the Internet address space back toward transparency. *SIGCOMM CCR*, 33(5):43–54, 2003. 48

[107] J.S. Turner and D.E. Taylor. Diversifying the Internet. In *IEEE GLOBECOM*, volume 2, 2005. 1, 20, 23, 24, 28

[108] Jaco E. van der Merwe and Ian M. Leslie. Switchlets and dynamic virtual ATM networks. In *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM'97)*, pages 355–368, 1997. 10, 17

[109] Jacobus E. van der Merwe, Sean Rooney, Ian Leslie, and Simon Crosby. The Tempest—A practical framework for network programmability. *IEEE Network Magazine*, 12(3):20–28, 1998. 10, 17

[110] N. Wang, D. Griffin, J. Spencer, J. Griem, J. Rodriguez Sanchez, M. Boucadair, E. Mykoniati, B. Quoitin, M. Howarth, G. Pavlou, A. J. Elizondo, M. L. Garcia Osma, and P. Georgatsos. A framework for lightweight QoS provisioning: Network planes and parallel Internets. In *Proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management (IM'07)*, pages 797–800, 2007. 12, 17

[111] Yi Wang, Eric Keller, Brian Biskeborn, Jacobus van der Merwe, and Jennifer Rexford. Virtual routers on the move: Live router migration as a network-management primitive. In *Proceedings of the ACM SIGCOMM'08*, pages 231–242, 2008. 16, 28, 35

[112] D.J. Wetherall, J.V. Guttag, and D.L. Tennenhouse. ANTS: A toolkit for building and dynamically deploying network protocols. In *IEEE OPENARCH'98*, pages 117–129, 1998. 8

[113] Timothy Wood, Prashant Shenoy, Arun Venkataramani, and Mazin Yousif. Black-box and gray-box strategies for virtual machine migration. In *NSDI*, pages 229–242, 2007. 35

[114] Jing Wu, Hanxi Zhang, S Campbell, M. Savoie, G.V. Bochmann, and B. St Arnaud. A grid oriented lightpath provisioning system. In *Proceedings of the GLOBECOM'04*, pages 395–399, 2004. 11

[115] Yechiam Yemini and Sushil da Silva. Towards programmable networks. In *IFIP/IEEE International Symposium on Distributed Systems: Operations and Management*, October 1997. 13

[116] Adel A. Youssef. A survey of active networks. Technical Report CS-TR-4422, University of Maryland, 1999. 8

[117] Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking virtual network embedding: Substrate support for path splitting and migration. *ACM SIGCOMM Computer Communication Review*, 38(2):17–29, April 2008. 27, 51, 52, 53, 56, 64, 65, 72

[118] E.W. Zegura, K.L. Calvert, and S. Bhattacharjee. How to model an Internetwork. In *Proceedings of IEEE INFOCOM*, pages 594–602, 1996. 64

[119] Y. Zhu and M. Ammar. Algorithms for assigning substrate network resources to virtual network components. In *Proceedings of IEEE INFOCOM*, 2006. 24, 27, 51, 54, 56, 64, 65, 70, 72

[120] Yaping Zhu, Jennifer Rexford, Andy Bavier, and Nick Feamster. UFO: A resilient layered routing architecture. Technical Report TR-780-07, Princeton University, 2007. 16