A Survey of Network Virtualization

N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba

David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1
{nmmkchow, rboutaba}@cs.uwaterloo.ca

Technical Report: CS-2008-25

October 15, 2008

Abstract

Due to the existence of multiple stakeholders with conflicting goals and policies, alterations to the existing Internet are now limited to simple incremental updates; deployment of any new, radically different technology is next to impossible. To fend off this ossification once and for all, *network virtualization* has been propounded as a diversifying attribute of the future inter-networking paradigm. 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. In this paper, we present a network virtualization model with a set of quintessential design goals, survey the past and the state-of-the-art of network virtualization, and discuss the future challenges that must be addressed to realize a viable network virtualization model.

1 Introduction

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 instead of radical changes by introducing and deploying new network technologies [1, 2].

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 [1]. According to them, network virtualization can extenuate the ossifying forces of the current Internet and stimulate innovation by enabling diverse network architectures to cohabit on a shared physical substrate. To introduce diversity, separation of policy from mechanism is a well-tested principle in computing literature. Similar approach has been propounded for virtualizing networks [2, 3, 4]. In this case, the role of the traditional ISPs has been divided into two: *infrastructure providers*, who manage the physical infrastructure, and *service providers*, who create virtual networks by aggregating resources from multiple infrastructure providers and offer end-to-end services to the end users. Such an environment will foster deployment of multiple coexisting heterogeneous network architectures that are not bounded by the inherent limitations found in the existing Internet.

This paper examines the past and the state-of-the-art in network virtualization and identifies key issues for future exploration. The rest of this article is organized as follows. In Section 2, we review three existing ideas: virtual private networks, programmable networks, and overlay networks, that are closely related

to the concept of virtual networks. Next, in Section 3, we present a reference business model of a network virtualization environment and identify critical design factors to materialize it. Following this, we summarize a number of past and present projects on network virtualization and related concepts in Section 4. In Section 5, we identify key research issues for further exploration based on our analysis of the surveyed work. We conclude in Section 6.

2 Historical Perspective

The concept of multiple co-existing networks is not necessarily new. It appeared in the networking literature several times in the past in different capacities. In this section, we discuss three such incarnations that are closely related to the concept of network virtualization. A virtual private network, also known as a VPN, is a specialized virtual network that connects 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. Active and programmable networks, on the other hand, is a concept that enables customization of network elements based on service providers' requirements.

2.1 Virtual Private Network (VPN)

A virtual private network (VPN) [5, 6, 7] can be thought of as a dedicated communications network of one or more enterprises that are distributed over multiple sites and connected through tunnels over shared or public communication networks like the Internet. If all the sites in a VPN are owned by the same enterprise, the VPN is known as a corporate *Intranet*. And if the sites are owned by different enterprises, the VPN is known as an *Extranet*. Most of the VPNs in practice are examples of intranets that connect geographically distributed sites of large corporate enterprises.

Each VPN site must contain one or more Customer Edge (CE) devices (e.g. hosts or routers). Each CE device is attached, via some sort of attachment circuit, to one or more Provider Edge (PE) routers. Routers in the SP's network that do not attach to CE devices are known as 'P' routers. Normally a VPN is managed and provisioned by a VPN service provider (SP) and known as Provider-provisioned VPN (PPVPN) [8]. PPVPN technologies can be classified into three broad categories based on the protocol used in the VPN data plane:

2.1.1 Layer 3 PPVPN

The distinguishing feature of Layer 3 VPN (L3VPN) [9, 10] is the use of layer 3 protocols (e.g. IP or MPLS) in the shared network infrastructure (VPN backbone) to carry data between the distributed CEs.

L3VPNs can again be classified into two categories: CE-based and PE-based VPNs. In the CE-based VPN approach, the shared service provider network does not have any knowledge of the customer VPN. CE devices create, manage, and tear up the tunnels. The SP network is completely unaware of the existence of a VPN and treats packets as normal IP packets. 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 [11], PPTP [12], L2TP [13]) to secure protocols (e.g. IPSec [14]).
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 tunnels, i.e. receiver CE devices, they are extracted and actual packets are injected into receiver networks.

On the other extreme, 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. In this case, the PE devices know that certain traffic is VPN traffic and they process that accordingly. When a PE-based VPN maintains

a complete logical router with unique forwarding table and unique routing protocol set for each VPN, it is known as Virtual Router PPVPN. If a single BGP instance is shared between VPNs, with separate forwarding environment and a separate forwarding table for each of them, it is known as BGP/MPLS IP VPN [7]. In this case route advertisements are marked with attributes to identify their VPN context.

2.1.2 Layer 2 VPN

Layer 2 VPNs (L2VPNs) [15, 16] transport Layer 2 (typically Ethernet but also ATM and Frame Relay) frames between participating sites. The advantage of L2VPN is that it is agnostic to the higher-level protocols and simpler. But the disadvantage is that 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). There is also the possibility of an IP-only LAN-like Service (IPLS). 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. And an IPLS 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.

2.1.3 Layer 1 VPN

Accompanied by the rapid advances in next-generation SONET/SDH and optical switching along with GMPLS [17] control, the Layer 1 VPN (L1VPN) [18, 19] framework emerged from the need to extend L2/L3 packet-switching VPN concepts to advanced circuit-switching domains. It enables multiple virtual client-provisioned transport networks over a common layer 1 core infrastructure.

The fundamental difference between L1VPNs and L2 or L3 VPNs is that in L1VPNs, data plane connectivity does not guarantee control plane connectivity (and vice versa). But CE-PE control plane connectivity is required for L1VPN services provisioned through the control plane, and CE-CE data plane connectivity is maintained by signaling mechanisms based on this control plane connectivity.

The main characteristics of L1VPN is that it provides a multi-service backbone where customers can offer their own services, whose payloads can be of any layer (e.g., ATM, IP, TDM). This ensures that each of the service networks have independent address space, independent layer 1 resource view, independent policies and complete isolation from other virtual networks.

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.

2.2 Active and Programmable Networks

The need to create, deploy and manage novel services on the fly in response to user demands was the key driving factor behind the uprise of programmable networks research. To enable on-demand services, the separation of communications hardware from control software is a fundamental requirement. If that separation is in place, software can be programmed irrespective of the underlying hardware to deliver necessary functionalities. Programmable networking community discusses on how this separation can be achieved.

The authors in [20] present a survey of programmable networks along with a generalized programmable networking model, where programmability is achieved by introducing computation inside the network and by extending the amount and scope of computation performed in existing routers and switches. Collectively, the computation and communication models make up a programmable network, allowing a network architect to program individual layers across the transport, control and management planes.

The computation model is realized by a set of distributed node kernels and network programming environment. The node kernel represents the lowest level of programmability, providing a small set of node interfaces supporting the manipulation of node states. Network programming environments support the construction of networks enabling dynamic deployment of network services and protocols. They offer a set of open interfaces and services to network designers to program customized network architecture.

Two separate schools of thought emerged on how to actually implement this concept; one from telecommunications community and the other from IP networks community.

2.2.1 Open Signaling Approach (Opensig)

Open signaling [21] takes a telecommunication approach to the problem with a clear distinction between transport, control, and management planes that constitute programmable networks and emphasize on QoS guarantees for created services. It argues for modeling communication hardware using a set of open programmable network interfaces, thereby enabling open access to switches and routers by third party software providers. It creates an abstraction layer for physical network devices to act as distributed computing objects with well-defined open programming interfaces, which allow service providers to manipulate the network states.

2.2.2 Active Networks Approach (AN)

Active networks [22, 23, 24, 25] community promotes dynamic deployment of new services at runtime within the confinement of existing networks. These networks are active in the sense that routers or switches can perform customized computations based on the packet contents and can also modify them. Active networks approach allows customization of network services at packet transport granularity instead of doing so through a programmable control plane; hence, it offers more flexibility than Opensig approach at the cost of more complex programming model.

Different levels of programmability have been suggested over the years. At one end, ANTS [26] 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 [27] only allows the user to call functions already installed at a particular node. Calvert et al. [28] classify the proposed architectures based on the granularity of control, statefullness, and language expressive power.

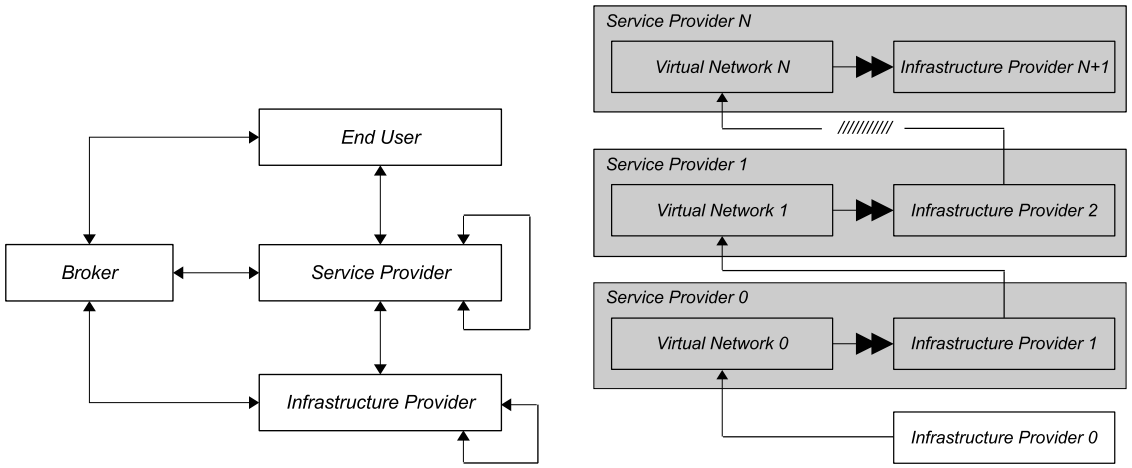
2.3 Overlay Networks

An overlay network is a virtual computer network which creates a virtual topology on top of the physical topology of another network. Nodes in an overlay network are connected through virtual links which may correspond to a path, connected by multiple physical links in the underlying network. Overlays are not geographically restricted and participation is completely voluntary. Since participants voluntarily lend their resources to the network, overlays do not typically involve significant expenditures. Moreover, they are flexible and adaptable to changes and easily deployable in comparison to any other network.

As a consequence, overlay networks have long been used to deploy new features and fixes in the Internet. A multitude of overlay designs have been proposed in recent years to address diverse issues, which include: ensuring performance [29] and availability [30] of Internet routing, enabling multicasting [31, 32, 33], providing QoS guarantees [34], protecting from denial of service attacks [35, 36], and for content distribution [37], file sharing [38] and even in storage systems [39].

The Detour [29] study noted that re-routing packets through virtual tunnels could often improve end-to-end performance in terms of loss, latency and throughput, than the direct Internet path. The Resilient Overlay Network (RON) [30] project experimentally showed that an overlay network that performs its own network measurements can provide fast recovery from failure and improved latency and loss-rates even over short time-scales. The overlay-based Internet Indirection Infrastructure (i3) [40] aims to simplify network services' deployment and management by decoupling the acts of sending and receiving. In i3, sources send packets to a logical identifier and receivers express interest in packets sent to an identifier. This additional level of indirection allows for more flexibility in node mobility and in service location and deployment. In all three cases, routing performance of the Internet is strengthened by introducing an overlay on top of the existing routing substrate.

"Routing as a Service" [41] introduces third party Routing Service Providers (RSPs) that buy virtual links from different ASes connecting some number of virtual routers. Hosts desiring customized routes contract with an RSP, which would then set up an appropriate end-to-end overlay path along its virtual links based on the global view of the topology. This idea of introducing third party providers is particularly of interest here. OverQoS [34] provides a mechanism to establish overlay links with certain loss and delay guarantees. Service Overlay Network (SON) [42] is designed to use overlay technique to provide value-added Internet services. A SON can purchase bandwidth with certain QoS guarantees from different ISPs to build a logical end-to-end service delivery overlay.



(a) Relationship between Players (b) Hierarchy of Roles

Figure 1: Network Virtualization Business Model

Most importantly, overlays have been used as testbeds. PlanetLab(§4.5.1) is such an example. With the advent of PlanetLab, researchers can easily create and manage customized environments to design and evaluate new architectures, protocols, and algorithms.

However, Anderson et al. [1] 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.

3 Network Virtualization Environment (NVE)

Based on the pluralist approach, we define network virtualization as an integral part of the diversified Internet architecture, which supports multiple coexisting heterogeneous network architectures from different service providers, sharing a common physical substrate managed by multiple infrastructure providers. By decoupling service providers from infrastructure providers, network virtualization introduces flexibility for innovation and change.

In this section, we present a conceptual view of the network virtualization architecture along with a reference business model and identify the principal design factors to materialize the architecture.

3.1 Reference Business Model

Players in the network virtualization model (Figure 1(a)) are different from those in the traditional networking model. The main distinction is the presence of two different roles: *infrastructure providers* and *service providers*, which are represented by a single role: Internet Service Provider (ISP) in the conventional model [2, 3, 4]. 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. 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).

3.1.1 Infrastructure Provider

Infrastructure providers (InPs) 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. They do not offer direct services to end users. 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.

Infrastructure providers communicate and collaborate among themselves, based on specific agreements, to create the complete underlying network. Those offering connectivity to service providers through different networking technologies, e.g. optical fiber, or satellite, are known as the *facilities providers*. On the other hand, Infrastructure providers connecting customer premise equipments (CPEs) to the network are the *access providers* [3].

3.1.2 Service Provider

Service providers (SPs) lease resources from multiple facilities providers to create virtual networks and deploy customized protocols by programming the allocated network resources to offer end-to-end services to the end users. A service provider can also create child virtual networks by partitioning its resources. It can then lease those child networks to other service providers, practically taking the role of an infrastructure provider creating a hierarchy of roles (Figure 1(b)).

3.1.3 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 services. Any end user may connect to multiple service providers for different services. End users are the target recipients of the services provided by the SPs. Services are offered on the basis of terms and conditions on behalf of both the SPs and the customers.

3.1.4 Broker

Brokers play a pivotal role in the network virtualization economy. They act as *mediators* between infrastructure providers, service providers, and end users in the network virtualization marketplace. Service providers buy (lease) resources from infrastructure providers to create virtual networks and sell services deployed on those virtual networks to interested end users through brokers. Their presence simplify the process of matching service providers' requirements to available resources by aggregating offers from multiple infrastructure providers. Similarly, they also allow end users to select desirable services from a wide range of service providers.

3.2 Architecture

In the network virtualization environment (NVE), the basic entity is a virtual network (VN). Each virtual network is composed and managed by a single service provider. It is a collection of virtual nodes connected together by a set of virtual links forming a virtual topology. Once provisioned a virtual network has the semblance of an actual physical network. Figure 2 depicts two possibly heterogeneous virtual networks VN1 and VN2 created by 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 its virtual network VN2 by combining resources from infrastructure provider InP1, with a child virtual network from service provider SP1. End users U1 and U3 are connected through VN2.

The owner of a virtual network is free to implement end-to-end services by selecting custom packet formats, routing protocols, forwarding mechanisms, as well as control and management planes. End users can opt-in to any virtual network. For example, end user U3 is subscribed to both VN1 and VN2.

3.2.1 Concepts

Here we formally define the common terminologies and notations that frequently appear to describe or refer to the different aspects of a network virtualization architecture.

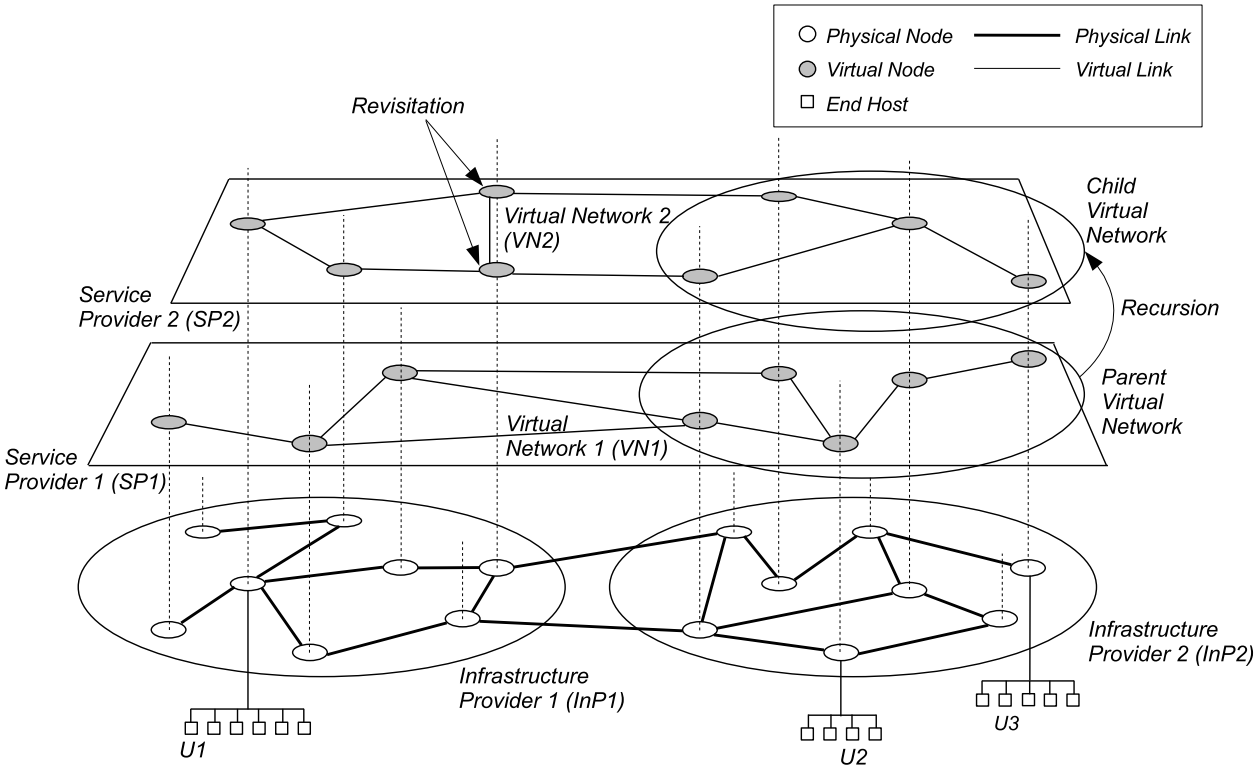


Figure 2: Network Virtualization Architecture

Physical Topology A weighted undirected graph $G_P = (V_P, E_P)$, where each node in the network is a vertex $v_P \in V_P$, with a set of attributes A_{v_P}. Similarly, each **physical link** between two nodes is represented by an edge $e_P \in E_P$, with corresponding set of attributes A_{e_P}.

Virtual Topology Another weighted undirected graph $G_V = (V_V, E_V)$, where V_V is the set of virtual nodes and E_V is the set of virtual links. It is also known as **logical topology**. Virtual nodes, v_V and virtual links, e_V also have their corresponding set of attributes A_{v_V} and A_{e_V} respectively, that are provided by the service provider when placing a request for a virtual network.

Virtual Node A virtual node can either be a virtual host or a virtual router. A **virtual host** act as a packet source or a sink and is never located inside the network. A **virtual router**, on the other hand, acts in the same way as a router in the physical network. Its main functionality is to forward packets according to the protocols of that virtual network.

Virtual Link An edge, $e_V \in E_V$, in the virtual network that may span over one or more connected physical links i.e. a path in the underlying physical topology. As mentioned earlier, e_V has a set of attributes A_{e_V} that characterize it. Bandwidth, delay, loss etc. are the examples of attributes of a virtual link.

Concurrence Concurrence of virtual networks means that multiple virtual networks 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 2 are examples of concurrent virtual networks. To put it simply, an infrastructure provider might cater to multiple service providers and a service provider might use resources from different infrastructure providers.

Recursion While virtual networks can be concurrent, in some cases, it might also be necessary to create and maintain one or more virtual networks within another virtual network creating a virtual network hi-

erarchy with *parent-child* relationship. This is known as recursion as well as **nesting** of virtual networks. In Figure 1(b), 'Service Provider 0' has created a virtual network 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 virtual networks makes further subdivision impossible.

Inheritance Child virtual networks, i.e. networks derived from other networks, can inherit architectural components from their parents. Also, constraints on a parent virtual network automatically translate to similar constraints on its children. In Figure 2, constraints due to InP2 will automatically be transferred to SP2 from SP1 through inheritance.

Revisitation Revisitation allows a physical node in the underlying infrastructure to host multiple virtual nodes of a single virtual network. Use of multiple logical routers to handle diverse functionalities in a complex network can be a great relief for network operators. It can also be useful for creating testbed networks. In Figure 2, we can see an illustration of revisitation in VN2.

3.3 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. We believe that in order to materialize the proposed architecture, each of these design goals must be fulfilled. These goals also provide a guideline to design a protocol or an algorithm for virtual networks. In addition, they can also be used to compare multiple network virtualization schemes.

3.3.1 Flexibility

Network virtualization must provide flexibility at every aspect of networking. Each service provider 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 virtual networks.

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 virtual network should be able to offer source routing without having to coordinate with any other parties.

3.3.2 Manageability

By separating service providers from infrastructure providers, network virtualization will modularize network management tasks and introduce accountability at every layer of networking [4, 43, 44, 2]. Infrastructure providers will be in total control of the management and operations of physical entities in the network and provide access to resources. Whereas, service providers 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 virtual networks to the service providers obviating the requirement of coordination across administrative boundaries as seen in the existing Internet.

3.3.3 Scalability

Coexistence of multiple networks is one of the fundamental motivations behind network virtualization. Scalability comes as an indispensable part of this equation. Optimally, there should be as many virtual networks as the underlying infrastructure can manage resource for. But due to conflicting requirements and different constraints it might not be possible to accommodate so many of them in practice. Infrastructure providers should try to maximize the number of coexisting virtual networks without degrading performance of any one of them. This will increase utilization of resources and amortize capital expenditure (CAPEX) and operational expenditure (OPEX) of individual virtual networks.

3.3.4 Isolation, Security, and Privacy

Network virtualization must provide isolation (both logical and resource) between different co-existing virtual networks, which will result in networks that are resilient to faults and attacks. Failure or security breach in one virtual network should not be able to affect another. Also the risk involved in carrying untrusted services and applications along with confidential information can be reduced by using separate virtual networks for each of them. Network protocols are often misconfigured and are subject to implementation bugs. Virtualization will ensure that misconfigurations or bugs in one network do not hamper the operations of another.

3.3.5 Programmability

To make virtual networks flexible and manageable, programmability of the network elements is of utmost importance. Only through programmable network elements, it will be possible for the service providers to implement customized protocols and deploy diverse services. Hence, the design decisions: *"how much programmability should be allowed"*, and *"how it should be exposed"* must have satisfactory answers. Once again the disagreement between the active networks community and the open programmable networks community becomes eminent. We must find a win-win situation where programmability will be easy, effective, as well as secure at the same time.

3.3.6 Heterogeneity

Heterogeneity in the context of network virtualization comes from two fronts. First, heterogeneity of the underlying networking technologies (e.g., optical, wireless, sensor etc.). Since each networking technology has its unique characteristics, enabling virtualization in each of them call for different solutions. Second, each end-to-end virtual network, created on top of that heterogeneous combination of infrastructure networks, can also be heterogeneous. Service providers must be allowed to compose and run end-to-end virtual networks without any restrictions and across multiple domains without the need for technology specific solutions. Moreover, underlying infrastructures must be capable of supporting heterogeneous protocols and algorithms implemented by different service providers. In addition, heterogeneity of end user devices must also be taken into account.

3.3.7 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 [45, 46].

3.3.8 Legacy Support

Legacy support or backward compatibility has always been a matter of deep concern while deploying any new technology. In fact, the biggest impedance against a complete makeover of the current Internet is the fear of losing all the advancements we have had in last three decades. Network virtualization must be able to replace the existing Internet without breaking it. Theoretically, network virtualization can easily integrate legacy support by considering the existing Internet as just another virtual network 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.

4 Network Virtualization Projects

The term *"virtual networks"* has been extravagantly used by different research groups to describe their works on virtual private networks, overlay networks, and active or programmable networks. Until recently, very few of them actually followed the pluralist view of network virtualization. In this section, we survey a number of virtual network architectures and related projects (e.g. overlay, programmable network or VPN inspired designs) that have emerged in the literature. We categorize the projects based on a set of characteristics and select a representative project for each characteristic. At the end of the section, we summarize the surveyed projects in Table 1.

4.1 Characteristics

In the motley of multifarious designs, one can observe a set of characteristics that govern the construction of different network virtualization architectures by careful examination. We use the following characteristics to better understand the already crowded field:

- Networking technology
- Layer of virtualization
- Architectural domain
- Level of virtualization

4.2 Networking Technology

A handful of network virtualization prototypes have been designed targeting specific networking technologies. The motivation behind these projects is to exploit unique characteristics of these networks to enable virtualization. We consider two representative projects enabling virtualization in IP and ATM networks.

4.2.1 *IP:* X-Bone

The X-Bone [47] was first proposed as a system for the rapid, 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)* [48], 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.

The VI virtualizes all the components of the Internet: hosts, routers and links between them. A single network node may participate as virtual host (VH), virtual router (VR) or multiple of them simultaneously in a VI. VHs act as data sources and sinks, while VRs act as data transits. Virtual links are part of VI deployment and VIs use IP in IP encapsulation [49] to transmit data through these links and thus avoid the need for new, non-ubiquitous protocol support. 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.

VIs completely decouple underlying physical network from the overlays and support *concurrence* and *recursion* as well as *revisitation*. Recursion is supported in two forms: control recursion and network recursion. Control recursion allows divide-and-conquer network management. And network recursion is true stacking of a VI on another VI, where packets on the uppermost VI have additional layers of header information when traversing inside the upper VI. Support for revisitation requires that Internet hosts and routers support forwarding based on the incoming interface, as well as packet header [50].

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. The addresses are managed by separate entity that is used in deployment of overlays.

Recently, P2P-XBone [51], a peer-to-peer based fusion of X-Bone, was proposed to enable dynamic join/leave of participating nodes from a VI. It also allows creation and release of dynamic IP tunnels, and customized routing table configuration.

4.2.2 *ATM:* Tempest

The Tempest [52] 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 following the open signaling (§2.2.1) school of thought of network programmability. It is based on the concept of *switchlets* [53], 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.

The Tempest supports programmability at two levels of granularity. First, switchlets support the introduction of alternative control architectures in the network. 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. And the association between user defined control policy and the allocated network resources is known as *connection closure*.

The basic components of the Tempest are as follows: first, a control architecture communicates with a switch independent control interface *Ariel*. Then it goes to the *Prospero* switch divider, which logically divides an ATM switch into switchlets and assigns resources to the client. There is an SNMP based switch management interface, *Caliban*, that provides basic functionalities to manage the switchlets. A basic bootstrapping virtual network is provided for the components to interact between themselves to create the actual virtual network. Afterward, network builder creates the topology of the virtual network based on a given specification. Network builder is also responsible for the modification and maintenance of virtual networks. And finally, the *Hollowman* control architecture [54] provides an out-of-band signaling mechanism for communication between ATM enabled workstations to devolve control from the ATM switches into an application level distributed processing environment.

4.3 Layer of Virtualization

Different research groups have considered virtualizing different layers of the network stack, starting from the physical layer and continuing up to the application layer. We summarize four such projects in the following.

4.3.1 *Physical Layer:* UCLP

UCLP [55] 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 (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, also known as *Articulated Private Networks (APNs)*. Furthermore, users can publish complete or partial network resources allocated to them for the availability of other users.

UCLP takes a modular approach toward resource management by introducing the concept of three distinct service layers [56, 57, 58]: User access, Service provisioning, and Resource management layers. The service provisioning layer is managed with a grid application. Customers and administrators configure and use end-to-end UCLP resources through the user access layer. Resource management layer deals with actual physical resources.

UCLPv1.4 [59] introduced dynamic topology discovery process and enabled auto-routing through intelligent algorithms alongside already available manual lightpath configuration capabilities. Later, UCLPv2 [60] 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.

4.3.2 *Link Layer:* VNET

VNET [61, 62] 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). Using VNET, VMs have no network presence at all on a remote site. Instead, VNET ensures that a distributed collection of VMs

each maintain a network presence on a given LAN. As a result, the management process is concerned with managing VMs present in the same LAN, instead of dealing with heterogeneous management policies of different sites.

Each physical machine hosting a 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. Furthermore, MAC address of a VM's virtual Ethernet adapter and the LAN it appears in are kept fixed for the lifetime of that VM. As a result, a VM can be migrated from one machine to another without any participation from the VM's OS and all connections remain open after migration.

4.3.3 *Network Layer:* AGAVE

The main objective of the AGAVE [3, 43, 63] project is to provide end-to-end QoS-aware service provisioning over IP networks following the theme of previously defined, but never implemented, QoS forwarding mechanisms such as IntServ [64] and DiffServ [65]. To achieve this, it proposes a new inter-domain architecture based on the novel concept of Network Planes (NPs), which will allow multiple IP Network Providers (INPs) to build and provide Parallel Internets (PIs) tailored to end-to-end service requirements. AGAVE specifies an open connectivity provisioning interface to allow Service Providers (SPs) to interact with underlying INP(s) infrastructure for the provision of their IP-based services.

AGAVE introduces the concept of NPs to differentiate the delivery behaviors experienced by IP flows when crossing an IP realm managed by a single INP. 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.

AGAVE replaces node-centric provisioning / configuration approach in favor of a more centralized network-based configuration, which ensures configuration consistency between participating INPs and reduces misconfiguration errors. Also, it supports an NP emulation function that assesses the status of the network and evaluates the impact of introducing new NPs before accepting new IP-connectivity provisioning requests.

4.3.4 *Application Layer:* VIOLIN

VIOLIN [66, 67] is an application-level virtual network architecture, where isolated virtual networks are created in software on top of an overlay infrastructure(e.g. PlanetLab(§4.5.1)). Capitalizing on the advances in virtual machine (VM) technologies [68, 69, 70], 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.

All VIOLIN entities are implemented as virtual machines in overlay hosts. User-Mode Linux(UML) [70], an user-space implementation of Linux, is used as the VM technology. UML's virtual NICs and virtual links are limited within the same host; so a nontrivial extension using UDP tunneling is implemented to connect VMs in different hosts. A vSwitch is created for each vLAN and all the hosts in that vLAN open UDP tunnels to get connected together. vRouters are similar to vHosts except that they can forward packets and enforce different routing policies.

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.

4.4 Architectural Domain

Most virtual networking projects are related to the introduction of new services. However, typically each one focuses on a particular architectural domain (e.g. management, transport, application etc.). We discuss four projects that address virtualization in the context of network management, resource management, and service composition through virtual active networks (VANs), and spawning networks.

4.4.1 *Network Management:* VNRMS

VNRMS [71, 72, 73] 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). 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 [74]. 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. These MIBlets are manipulated by the customer using SNMP [75] through CNRMS for configuration and monitoring of the VN [76].

4.4.2 *Resource Management:* Darwin

Darwin [77] presents a collection of customizable resource management mechanisms that support value-added services. It considers a network environment that comprises not only communication services, but storage and computation resources as well. By packaging storage/computation resources together with communication services, service providers will be able to compose sophisticated value-added services in a hierarchical fashion. It argues for customization of resource management policies along space, time and organizational constraints.

Architecturally, it includes *Xena* [78], a high level resource broker that maps user requirements to a set of local resources, local resource managers that communicate with Xena using the *Beagle* [79] signaling protocol, and hierarchical scheduling disciplines based on service profiles. Xena is made programmable in a restricted fashion with a set of service delegates that provide support for active packets. Delegates can be dynamically injected into IP routers or hosts to support application specific processing and real-time management of resources. Xena expresses the resource requirements in terms of virtual networks and passes it through to Beagle. Beagle then establishes a core, based on the input from Xena, and initiates individual flow setups by the senders or receivers that rendezvous with the core in designated routers. For an individual physical resource, sharing and contention exist at multiple levels. Darwin uses the Hierarchical Fair Service Curve (H-FSC) scheduler for the purpose.

4.4.3 *Virtual Active Networks:* NetScript

NetScript [80] is a language system for dynamically programming and deploying protocol software in an active network (§2.2.2). 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, as in current network model. 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 [81].

NetScript consists of three integrated domain-specific language components that reside above a dataflow kernel. The packet presentation language defines the syntax of a NetScript protocol, while the composition language helps implementing the semantics of that protocol. Finally, the classification language acts as a bridge between syntax of a protocol and its semantics. These language components work together to

create a programmable stream-processing system that enables creation of arbitrary packet formats, dynamic composition of standard and active protocol, and can operate on any type of packet stream defined by the presentation language.

In essence, NetScript is analogous to PostScript for printers. Just as PostScript captures the programmability of printer engines, NetScript creates a software abstraction to capture the programmability of network node functions. NetScript communication abstractions consider network nodes as collections of Virtual Network Engines (VNEs) interconnected by Virtual Links (VLs) that constitute NetScript Virtual Networks (NVNs) [82].

4.4.4 *Spawning Networks:* Genesis

The Genesis Kernel [83] is a *spawning network* [84, 85], 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 idea is similar to spawning a child process from its parent process in the operating system literature. 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. The set of virtual links create the transport environment connected by routelets. The Programming environment based on CORBA is used to control the distributed controllers across virtual networks.

The virtual network life cycle process is automated comprising of four distinct phases [85]: 1) profiling, 2) spawning, 3) management, and 4) architecting. The profiling phase captures the blueprint of a network architecture in terms of a xml-based comprehensive profiling script. The spawning phase systematically sets up the topology and address space; allocates resources; and binds transport, control, and management objects to the physical network infrastructure. The management phase supports virtual network resource management [86], while the designing phase allows network designers to add, remove, or replace distributed network algorithms on-demand analyzing the pros and cons of the network design space.

4.5 Level of Virtualization

To enable network virtualization, one must virtualize the nodes, the links, and every other resources 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 virtual network by connecting virtual machines on different nodes (e.g. PlanetLab [45]). At the other end, CABO [4] proposes the concept of true plurality where each virtual network has a semblance of a native network.

4.5.1 *Node Virtualization:* PlanetLab

PlanetLab [45, 87, 88] 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. The overlay consists of two main software components: 1) a *virtual machine monitor* (VMM) running on each node providing a controlled interface to abstract resources, and 2) a *management service* used to control the overlay.

PlanetLab is built upon four design principles. First, it supports *sliceability*. That is, each application acquires and runs in a slice of the overlay. 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.

There are *resource monitor* and *resource broker* services that handle the resource management in PlanetLab. To obtain a slice a user first contacts a resource broker, then goes through admission control process in each of the nodes assigned by the broker and finally it launches its service in the resulting slice.

PlanetLab currently consists of over 800 nodes at around 400 sites [45], and hundreds of research experiments have been successfully conducted over the years that prove its worth.

4.5.2 VINI

VINI [46, 89] 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(§4.5.1) along with the support for virtual networks as in X-Bone(§4.2.1) or VIOLIN(§4.3.4).

VINI offers more latitude to researchers than PlanetLab at routing level; and 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. To achieve realism, VINI can be placed as a middle-ground of an end-to-end Internet path. The real traffic that traverse through VINI is then used by the research experiments running on VINI.

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 [90], *Click* for packet forwarding and network address translation [91], and *OpenVPN* servers to connect with end users [92].

Recently a software platform for hosting multiple virtual networks on shared physical network infrastructure, Trellis [93], 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. Initial evaluations show that virtual networks hosted on Trellis can transfer packets as fast as regular networks, and Trellis can host at least 64 such virtual networks without performance degradation.

4.5.3 GENI

Based on the experience accumulated from using PlanetLab and other similar testbeds, the Global Environment for Network Innovations (GENI) [94, 95] 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 [95] 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.

4.5.4 *Full Virtualization:* CABO

Current ISPs manage their network infrastructure as well as providing service to end users. Adopting a new architecture not only requires change in hardware and host software but it also requires that ISPs jointly agree on any architectural change [1]. CABO [4] 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. On one hand, CABO gives service providers direct control over the protocols and services that run on the virtual nodes and allows them to instantiate a virtual network on multi-provider infrastructure. On the other hand, it enables infrastructure providers to automatically discover and manage their physical infrastructure by running a discovery plane of their own [96].

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 [97] 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 [98]. In supporting programmable routers, CABO resembles the theme introduced in active networks research [22, 99], except for that it does not enable users to program the network; rather service providers can customize their networks according to their need.

4.6 Summary

A summary of the characteristics of the reviewed network virtualization projects is presented in Table 1.

5 Key Research Directions

Most of the research works, that has been floating around for many years under the banner of 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. In previous sections we presented a conceptual model of the network virtualization environment, concepts closely related to its origin, and discussed some of the most notable projects in this area. Based on that discussion, here we summarize the key issues to resolve under the light of past works and provide future directions for further exploration with an objective to spark renewed research interest in this area.

5.1 Interfacing

Service providers will synthesize physical infrastructure from one or more infrastructure providers to build their virtual networks. Hence, every infrastructure provider must provide a well-defined interface so that service providers can communicate with them and express their requirements. Ideally, every interface should follow a standard. Expressing a virtual network request in terms of virtual nodes, and virtual links along with their corresponding attributes can be a suitable solution.

Similarly, appropriate interfaces between end users and service providers, as well as among multiple infrastructure providers, and among service providers must also be identified and standardized. Examples of such interfaces and agreements between collaborating parties can be found in the AGAVE (§4.3.3) framework.

5.2 Signaling and Bootstrapping

Before creating a virtual network, a service provider must already have network connectivity to infrastructure providers in order to issue its requests. This introduces circularity where network connectivity is a prerequisite to itself [4]. As long as the network virtualization environment 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 service providers to customize the virtual nodes and the virtual links allocated to them to create their virtual networks. Both the requirements, i.e. signaling

Table 1: Characteristics of different network virtualization projects

	Architectural Domain	Networking Technology	Layer of Virtualization	Level of Virtualization	Ref.
X-Bone	Automating the deployment of IP overlays	IP	Application	Node & Link	[47, 48]
Tempest	Enabling alternate control architectures	ATM	Link		[52, 53]
Genesis	Spawning virtual network architectures		Network	Node & Link	[83, 84, 85]
UCLP	Dynamic provisioning and reconfiguration of lightpaths	SONET	Physical	Link	[55, 59, 60]
VNET	Virtual machine Grid computing		Link	Node	[61, 62]
AGAVE	End-to-end QoS-aware service provisioning	IP	Network		[3, 43, 63]
VIOLIN	Deploying on-demand value-added services on IP overlays	IP	Application	Node	[66, 67]
VNRMS	Virtual network management	ATM/IP		Node & Link	[71, 72, 73]
Darwin	Integrated resource management and value-added services	IP			[77]
NetScript	Dynamic composition of services	IP	Network	Node	[80, 81]
PlanetLab	Deployment and management of overlay-based testbeds	IP	Application	Node	[45, 87]
VINI	Evaluating protocols and services in a realistic environment		Link		[46, 89]
GENI	Creating customized virtual networks	Heterogeneous			[94, 95]
CABO	Deploying value-added end-to-end services on shared infrastructure	Heterogeneous		Full	[4]

and bootstrapping, call for at least another network that will always be present to provide connectivity to handle these issues. Genesis (§4.4.4) and Tempest (§4.2.2) follow this approach and provide a separate *bootstrapping* interface.

5.3 Accounting for Admission Control and Distributed Rate Limiting

In order to uphold QoS guarantees, infrastructure providers must ensure that resources are not overbooked to service providers. Consequently, they have to perform accurate accounting and implement admission control algorithms to ensure that resources allocated to the virtual networks 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 virtual networks, distributed policing mechanisms must be employed to make sure that service providers cannot overflow the amount of resources allocated to them by direct or indirect means. Raghavan et al. [100] 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.

5.4 Virtual Network Embedding or Mapping

Since a virtual link may span over multiple physical links, there may be many possible mappings for any given virtual network. In order to maximize the number of co-existing virtual networks it is very important to determine how to embed a service provider'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* [101] 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 service providers' requests are known in advance. Zhu and Ammar [44] aim at achieving load balancing in the underlying physical infrastructure assuming unlimited resources. Lu and Turner [102] provide a solution for mapping only one virtual network with an aim to minimize cost. Other solutions for the offline problem based on multi-commodity flow exist in the VPN context [103, 104].

For the online problem, Fan and Ammar [105] present a solution for determining dynamic topology reconfiguration for service overlay networks with dynamic communication requirement. Zhu and Ammar [44] handle the problem by calculating the whole mapping periodically. In both solutions, it is assumed that infrastructure resource is unlimited. Yu et al. [106] 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 infrastructure providers' resources. They also provide a modularized algorithm to optimize mapping for specific topologies. 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 newer approximation algorithms.

5.5 Resource Scheduling

When establishing a virtual network, a service provider requires specific guarantees for the virtual routers' attributes, as well as the virtual links' bandwidth allocated to its network. For virtual routers, a service provider 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 service provider, infrastructure providers 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 virtual networks; instead, they are dynamically distributed to increase utilization of the resources as well as the revenue of the infrastructure providers. DaVinci [107] 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.

Existing system virtualization technology provides scheduling mechanisms for CPU, memory, disk, and network interface in each of the VMs running on the host machine. Network virtualization can utilize these mechanisms to implement resource scheduling in the physical infrastructure. Previous results from research on packet scheduling algorithms for IP networks can also be useful in the design of schedulers.

5.6 Topology Discovery

In order to allocate resources for requests from different service providers, infrastructure providers 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 infrastructure providers must also be able to establish links between their networks to enable cross-domain virtual network instantiation.

UCLP (§4.3.1) promotes a combination of *event-based* and *periodic* topology discovery using an additional topology database [59]. Here, events update the topology database of an infrastructure provider; and a periodic refresh ensures that even if some events were not notified, the topology database is fresh. CABO (§4.5.4) argues for the use of a separate discovery plane run by the infrastructure providers as proposed in the 4D network management architecture [96]. Dissemination of gathered information in decision elements could be achieved via flooding across inter-provider links.

5.7 Virtual Nodes/Routers

One of the most fundamental issues in network virtualization, if not *the most*, is the virtualization of the nodes that constitute the underlying physical network. Virtual routers allow multiple service providers to share same set of physical resources, and implement customized control protocols on them. Up until now, router vendors have promoted virtual routers as a tool for simplifying core network design, decreasing CAPEX, and for VPN purposes [108]. Similar concept can be extended with programmability to create substrate routers that will allow each service provider to customize their virtual routers. A conceptual construct of such substrate routers can be found in [2]. Examples of extensible and flexible virtual router architectures include Click Modular Router [91], VERA [109] etc.

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 [110].

Scalability of network virtualization environment is closely tied to the scalability of the physical routers used by the infrastructure providers. Commercial router vendors have already implemented routers that can hold multiple logical routers [111]. Fu and Rexford [112] 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 routers any single physical router can hold.

To increase network manageability, and to handle network failures to some extent, migration of virtual routers can be an effective solution [97]. 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.

5.8 Virtual Links

Virtualization of links has similar importance as virtualization of routers, in implementing virtual networks. The ability to create tunnels over multiple physical links in Layers 3, 2, or 1 already exists in the context of L3,

L2, and L1 VPNs, respectively (§2.1). Similar protocols can be used in virtual networks too. Creating inter-infrastructure provider tunnels poses a great challenge, since it will require collaboration between multiple infrastructure providers. Whether the total tunneling procedure should be done without the knowledge of infrastructure providers of the tunnels' presence or should they know, is yet to be decided.

Speed matters for network links. The overhead for transporting packets across a virtual link must be minimal compared to that of transporting packets across a native link. This translates into minimum encapsulation and multiplexing cost. Optical fibers can be very useful in this regard, since they can be physically sub-divided into smaller lightpaths instead of encapsulating or multiplexing packets from different virtual networks to keep them isolated. Moreover, virtual links must also be flexible enough to carry packets of any protocol.

5.9 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 network virtualization environment becomes a major challenge. In the network virtualization environment, 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.

Unlike the existing Internet architecture, where IP addresses carry locations as well as (weak) identifications, naming and addressing should be decoupled in the network virtualization environment so that any end user can move from one service provider to another with a single identity. If identity gets linked to the service provider, then one will not be able to change providers without losing ones identity. This problem is similar, at a higher level, to the problem of people using ISP provided email addresses, who discover that they have to get new email addresses as soon as they change their ISPs.

In the network virtualization environment, 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.

iMark [113] is an identity management framework for the network virtualization environment that separates the identities of the end hosts from their physical and logical locations, and with the help of a global identifier space provides universal connectivity without revoking the autonomy of the concerned physical and virtual networks.

5.10 Dynamism and Mobility 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 virtual networks. Such dynamism can broadly be categorized into two classes:

1. Macro Level Dynamism: Virtual networks providing basic services or virtual networks with shared interests can be dynamically aggregated together to create compound virtual networks. This is known as *federation* of VNs. Multiple federations and virtual networks can also come together to create hierarchy of virtual networks. Even though the level of dynamism is expected to be very low at this level, the complexity of adding a virtual network to a collection, or removing one, can be quite high.

2. Micro Level Dynamism: This is the more influential of the two classes discussed here and requires more attention. Micro level dynamic behavior can basically be attributed to two broad sets of activities:

   - Dynamic join, leave, and *mobility* of end users within and in between virtual networks, and
   - Dynamism incurred by the migration (i.e., mobility) of virtual routers for different purposes [97].

Mobility of the end user devices and virtual resources in the network virtualization environment can again be divided into two categories:

- *Geographical mobility* from one physical network to another, and
- *Logical mobility* from one virtual network to another.

Finding the exact location of any resource at a particular moment and routing packets accordingly is a complex research challenge that needs close scrutiny.

5.11 Virtual Network Operations and Management

Network operations and management has always been a great challenge for the network operators. Smallest of misconfigurations can practically bring a fully-functioning network down to its knees. To prevent this, a large sum of money is spent yearly to make sure that a network is operational. Division of accountability and responsibilities among different participators in the network virtualization environment promises increased manageability, and reduced scopes for error. But this will require a complete makeover of the network management architecture, as we know it today. Considerable flexibility must be introduced from the level of network operations centers (NOCs) to intelligent agents at network elements, to enable individual service providers configure, monitor, and control their virtual networks irrespective of others'. The concept of MIBlets, i.e., partitioned MIBs, used in VNRMS (§4.4.1) to gather and process performance statistics for each of the coexisting virtual networks instead of using a common MIB can be a good starting point.

Since a virtual network can span over multiple underlying physical networks, applications must also be developed to aggregate information from diverse, often conflicting, management paradigms followed by participating infrastructure providers. Introducing a common abstraction layer, that will be followed by all the management softwares, can be an effective solution [114]. Identifying the scope of management for infrastructure providers and service providers is also a very important task.

5.12 Failure Handling and Event Notification

Failures in the underlying physical network components can give rise to tricky problems in the network virtualization environment. Any such failure can effectively cause a cascading series of errors in the virtual networks directly hosted on those components, and possibly in many others that are recursively spawned from the affected ones. For instance, a physical link failure will result in failures of all the virtual links that pass through it. Similarly, any physical node failure might require re-installations of all the service provider's custom softwares. Detection, propagation, and isolation of such failures, as well as prevention and recuperation from them are all open research challenges.

5.13 Enabling Virtualization across Heterogeneous Networking Technology

Each networking technology has its own set of unique characteristics. Virtualization of networks on each of these 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 (§2.1.3) is now a reality [18]. UCLP (§4.3.1) virtualizes optical networks capitalizing on the property of lightpaths that can be physically sub-divided into smaller lightpaths.

Virtual Sensor Networks (VSN) [115], 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 [116].

End-to-end virtual networks might span across multiple domains running over completely different type of network. Interaction between such contrasting underlying infrastructures, while providing a generic and transparent interface for service providers to easily compose and manage virtual networks remains a daunting task.

5.14 Inter Virtual Network Communication

Even though one of the main inspirations behind network virtualization is isolation between co-existing virtual networks, there are cases when two virtual networks need to share resources or information. For

Table 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/
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/

example, a large multinational corporation might deploy a virtual network across the globe, with child virtual networks for each of the continents, to manage its operations. It is very likely that those child virtual networks will need to communicate with the global one, as well as among themselves. In some cases, communicating virtual networks might not even be under the same administrative domain, i.e. managed by different service providers. Hence, the necessity, scope, and required interface for such interconnections among service providers and corresponding virtual networks deserve close scrutiny.

5.15 Network Virtualization Economics

In traditional networks, bandwidth is the chief commodity of interest. But in the network virtualization environment virtual nodes are also a very important commodity in addition to virtual links. Service providers are the buyers in this economy, whereas infrastructure providers are the sellers. There might also be brokers who will act as mediators between the buyers and the sellers. End users also participate as buyers of services from different service providers.

Traditionally, there are two general types of marketplaces: centralized, and decentralized. Centralized marketplaces are efficient; but vulnerable against attacks, and are not scalable. On the other hand, fully decentralized marketplaces are extensible and fault-tolerant; but prone to malicious behavior and inefficiency. PeerMart [117, 118], which combines both efficiency and scalability, is a semi-decentralized double-auction based marketplace for peer-to-peer systems. Virtuoso [119], developed for VNET, is another example of virtual network marketplace management system. Similar idea can also be developed for the network virtualization environment through further investigation.

6 Conclusion

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 [120]. Network virtualization can take the leading role in this scenario to promote innovation, to provide flexibility, and to introduce heterogeneity. This realization has given birth to several projects all over the world that are directly or indirectly related to network virtualization (Table 2).

However, realization of the network virtualization environment needs to satisfy the requirements set by its characteristics and design goals. Even though these requirements will ensure an open, flexible, and

heterogeneous networking environment, ensuring themselves is not so easy. We encourage more insight into the problems pointed out in this article and look forward to a motivated search for solutions to the open research issues from the researchers in this field.

References

[1] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *Computer*, vol. 38, no. 4, pp. 34–41, 2005.

[2] J. Turner and D. Taylor, "Diversifying the internet," in *Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM'05)*, vol. 2, 2005.

[3] M. Boucadair, B. Decraene, M. Garcia-Osma, A. J. Elizondo, J. R. 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.

[4] N. Feamster, L. Gao, and J. Rexford, "How to lease the internet in your spare time," *SIGCOMM Computer Communication Review*, vol. 37, no. 1, pp. 61–64, 2007.

[5] P. Ferguson and G. Huston, "What is a VPN?" Cisco Systems, Tech. Rep., 1998.

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

[7] ——, "BGP/MPLS IP Virtual Private Networks (VPNs)," RFC 4364, February 2006.

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

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

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

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

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

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

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

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

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

[17] E. Mannie, "Generalized Multi-Protocol Label Switching (GMPLS) Architecture," RFC 3945, October 2004.

[18] D. Benhaddou and W. Alanqar, "Layer 1 virtual private networks in multidomain next-generation networks," *IEEE Communications Magazine*, vol. 45, no. 4, pp. 52–58, April 2007.

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

[20] A. T. Campbell, H. G. D. Meer, M. E. Kounavis, K. Miki, J. B. Vicente, and D. Villela, "A survey of programmable networks," *SIGCOMM Computer Communication Review*, vol. 29, no. 2, pp. 7–23, 1999.

[21] "Open Signaling working group," http://comet.columbia.edu/opensig/.

[22] D. L. Tennenhouse and D. J. Wetherall, "Towards an active network architecture," *ACM Computer Communication Review*, vol. 26, no. 2, 1996.

[23] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall, and G. J. Minden, "A survey of active network research," *IEEE Communications Magazine*, vol. 35, no. 1, pp. 80–86, January 1997.

[24] A. A. Youssef, "A survey of active networks," University of Maryland, Tech. Rep. CS-TR-4422, 1999.

[25] D. Niculescu, "Survey of active network research," http://www.research.rutgers.edu/~dnicules/research/other/active_survey.pdf, 1999.

[26] D. Wetherall, J. Guttag, and D. Tennenhouse, "ANTS: A toolkit for building and dynamically deploying network protocols," in *IEEE OPENARCH'98*, 1998, pp. 117–129.

[27] D. Decasper and B. Plattner, "DAN: Distributed code caching for active networks," in *Proceedings of the IEEE INFOCOM'98*, vol. 2, 1998, pp. 609–616.

[28] K. Calvert, S. Bhattacharjee, E. Zegura, and J. Sterbenz, "Directions in active networks," *IEEE Communications Magazine*, vol. 36, no. 10, pp. 72–78, October 1998.

[29] 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*, vol. 19, no. 1, pp. 50–59, January 1999.

[30] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," *SIGOPS Operating Systems Review*, vol. 35, no. 5, pp. 131–145, 2001.

[31] H. Eriksson, "MBone: The multicast backbone," *Communications of the ACM*, vol. 37, no. 8, pp. 54–60, 1994.

[32] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. 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)*, 2000, pp. 197–212.

[33] Y. Chu, S. Rao, S. Seshan, and H. Zhang, "Enabling conferencing applications on the internet using an overlay muilticast architecture," *SIGCOMM Computer Communication Review*, vol. 31, no. 4, pp. 55–67, 2001.

[34] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz, "OverQoS: An overlay based architecture for enhancing internet QoS," in *Proceedings of the 1st Symposium on Networked Systems Design and Implementation (NSDI)*, March 2004, pp. 71–84.

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

[36] D. 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: USENIX Association, 2003.

[37] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the use and performance of content distribution networks," in *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement (IMW'01)*. New York, NY, USA: ACM, 2001, pp. 169–182.

[38] E. K. 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*, vol. 7, no. 2, pp. 72–93, 2005.

[39] F. Dabek, M. F. Kaashoek, D. Karger, R. Morris, and I. Stoica, "Wide-area cooperative storage with CFS," in *Proceedings of the eighteenth ACM symposium on Operating systems principles (SOSP'01)*. New York, NY, USA: ACM, 2001, pp. 202–215.

[40] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana, "Internet indirection infrastructure," in *Proceedings of the ACM SIGCOMM Conference (SIGCOMM'02)*, 2002, pp. 73–88.

[41] K. Lakshminarayana, I. Stoica, S. Shenker, and J. Rexford, "Routing as a service," UC Berkeley, Tech. Rep. UCB/EECS-2006-19, 2006.

[42] Z. Duan, Z.-L. Zhang, and Y. T. Hou, "Service overlay networks: SLAs, QoS, and bandwidth provisioning," *IEEE/ACM Transactions on Networking*, vol. 11, no. 6, pp. 870–883, 2003.

[43] M. Boucadair, P. Levis, D. Griffin, N. Wang, M. Howarth, G. Pavlou, E. Mykoniati, P. Georgatsos, B. Quoitin, J. R. Sanchez, and M. Garcia-Osma, "A framework for end-to-end service differentiation: Network planes and parallel Internets," *IEEE Communications*, vol. 45, no. 9, pp. 134–143, September 2007.

[44] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proceedings of the IEEE INFOCOM'06*, 2006.

[45] "PlanetLab: An open platform for developing, deploying, and accessing planetary-scale services," http://www.planet-lab.org/.

[46] "VINI: A virtual network infrastructure," http://www.vini-veritas.net/.

[47] J. Touch and S. Hotz, "The X-Bone," in *In Proceedings of the Third Global Internet Mini-Conference at GLOBECOM'98*, 1998, pp. 44–52.

[48] J. D. Touch, Y.-S. Wang, L. Eggert, and G. Finn, "A virtual internet architecture," USC/Information Sciences Institute, Tech. Rep. TR-570, 2003.

[49] C. Perkins, "IP encapsulation within IP," RFC 2003, October 1996.

[50] F. Baker, "Requirements for IPv4 routers," RFC 1812, June 1995.

[51] N. Fujita, J. D. Touch, V. Pingali, and Y.-S. Wang, "A dynamic topology and routing management strategy for virtual ip networks," *IEICE Transactions on Communications*, vol. E89-B, no. 9, pp. 2375–2384, 2006.

[52] J. E. van der Merwe, S. Rooney, I. Leslie, and S. Crosby, "The Tempest—A practical framework for network programmability," *IEEE Network Magazine*, vol. 12, no. 3, pp. 20–28, 1998.

[53] J. E. van der Merwe and I. M. Leslie, "Switchlets and dynamic virtual ATM networks," in *Proceedings of the IFIP/IEEE International Symposium on Integrated Network Management (IM'97)*, 1997, pp. 355–368.

[54] S. Rooney, "The hollowman: an innovative atm control architecture," in *Proceedings of the fifth IFIP/IEEE International Symposium on Integrated Network Management V : Integrated Management in a Virtual World*. London, UK, UK: Chapman & Hall, Ltd., 1997, pp. 369–380.

[55] U. of Waterloo, "User controlled lightpaths project," http://uclp.uwaterloo.ca/.

[56] R. Boutaba, W. Golab, Y. Iraqi, and B. St-Arnaud, "Grid-controlled lightpaths for high performance grid applications," *Journal of Grid Computing*, vol. 1, no. 4, pp. 387–394, December 2003.

[57] ——, "Lightpaths on demand: A web services-based management system," *IEEE Communications Magazine*, vol. 42, no. 7, July 2004.

[58] J. Wu, H. Zhang, S. Campbell, M. Savoie, G. Bochmann, and B. St Arnaud, "A grid oriented lightpath provisioning system," in *Proceedings of the GLOBECOM'04*, 2004, pp. 395–399.

[59] 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*, vol. 1, July 2005, pp. 263–266.

[60] B. Nandy, D. Bennett, I. Ahmad, S. Majumdar, and B. St.Arnaud, "User controlled lightpath management system based on a service oriented architecture," http://www.solananetworks.com/UCLP/files/UCLPv2-SOA.pdf, 2006.

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

[62] 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.

[63] N. Wang, D. Griffin, J. Spencer, J. Griem, J. R. Sanchez, M. Boucadair, E. Mykoniati, B. Quoitin, M. Howarth, G. Pavlou, A. J. Elizondo, M. L. G. 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)*, 2007, pp. 797–800.

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

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

[66] X. Jiang and D. Xu, "VIOLIN: Virtual internetworking on overlay infrastructure," Purdue University, Tech. Rep. TR-03-027, 2003.

[67] P. Ruth, X. Jiang, D. Xu, and S. Goasguen, "Virtual distributed environments in a shared infrastructure," *Computer*, vol. 38, no. 5, pp. 63–69, 2005.

[68] A. Whitaker, M. Shaw, and S. D. Gribble, "Scale and performance in the denali isolation kernel," in *Proceedings of the 5th symposium on Operating Systems Design and Implementation (OSDI'02)*, 2002, pp. 195–210.

[69] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield, "Xen and the art of virtualization," in *Proceedings of the nineteenth ACM symposium on Operating systems principles (SOSP'03)*. New York, NY, USA: ACM, 2003, pp. 164–177.

[70] J. Dike, "A user-mode port of the linux kernel," in *Proceedings of the 4th Annual Linux Showcase & Conference (ALS'00)*. Berkeley, CA, USA: USENIX Association, 2000, pp. 7–7.

[71] 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.

[72] ——, "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)*, vol. 2, 1998, pp. 1065–1070.

[73] 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*, 1999, pp. 205–210.

[74] 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)*, 1999, pp. 201–215.

[75] J. Case, M. Fedor, M. Schoffstall, and J. Davin, "Simple Network Management Protocol (SNMP)," RFC 1157, May 1990.

[76] R. Boutaba, W. Ng, and A. Leon-Garcia, "Web-based customer management of VPNs," *Journal of Network and Systems Management*, vol. 9, no. 1, pp. 67–87, 2001.

[77] P. Chandra, A. Fisher, C. Kosak, T. S. E. Ng, P. Steenkiste, E. Takahashi, and H. Zhang, "Darwin: Customizable resource management for value-added network services," *IEEE Network*, vol. 15, no. 1, pp. 22–35, 2001.

[78] P. Chandra, A. Fisher, C. Kosak, and P. Steenkiste, "Network support for application-oriented QoS," in *Proceedings of Sixth International Workshop on Quality of Service (IWQoS'98)*, May 1998, pp. 187–195.

[79] P. Chandra, A. Fisher, and P. Steenkiste, "Beagle - a resource allocation protocol for advanced services internet," Carnegie Mellon University, Tech. Rep. CMU-CS-98-150, August 1998.

[80] S. da Silva, Y. Yemini, and D. Florissi, "The NetScript active network system," *IEEE Journal on Selected Areas in Communication*, vol. 19, no. 3, pp. 538–551, 2001.

[81] S. da Silva, D. Florissi, and Y. Yemini, "NetScript: A language-based approach to active networks," Columbia University, Tech. Rep., January 1998.

[82] Y. Yemini and S. da Silva, "Towards programmable networks," in *IFIP/IEEE International Symposium on Distributed Systems: Operations and Management*, October 1997.

[83] M. Kounavis, A. Campbell, S. Chou, F. Modoux, J. Vicente, and H. Zhuang, "The Genesis Kernel: A programming system for spawning network architectures," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 3, pp. 511–526, 2001.

[84] A. A. Lazar and A. T. Campbell, "Spawning networking architectures (White Paper)," Columbia University, Tech. Rep., 1998.

[85] A. T. Campbell, M. E. Kounavis, D. A. Villela, J. Vicente, K. Miki, H. G. D. Meer, and K. S. Kalaichelvan, "Spawning networks," *IEEE Network Magazine*, vol. 13, no. 4, pp. 16–30, 1999.

[86] D. Villela, A. T. Campbell, and J. Vicente, "Virtuosity: Programmable resource management for spawning networks," *Computer Networks*, vol. 36, no. 1, pp. 49–73, 2001.

[87] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A blueprint for introducing disruptive technology into the Internet," *SIGCOMM Computer Communication Review*, vol. 33, no. 1, pp. 59–64, 2003.

[88] N. Spring, L. Peterson, A. Bavier, and V. Pai, "Using PlanetLab for network research: Myths, realities, and best practices," *SIGOPS Operating Systems Review*, vol. 40, no. 1, pp. 17–24, 2006.

[89] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In VINI veritas: Realistic and controlled network experimentation," in *Proceedings of SIGCOMM'06*. New York, NY, USA: ACM, 2006, pp. 3–14.

[90] M. Handley, E. Kohler, A. Ghosh, O. Hodson, and P. Radoslavov, "Designing extensible IP router software," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation (NSDI'05)*. Berkeley, CA, USA: USENIX Association, 2005, pp. 189–202.

[91] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click modular router," *ACM Transactions on Computer Systems*, vol. 18, no. 3, pp. 263–297, 2000.

[92] "OpenVPN: An open source SSL VPN solution," http://openvpn.net/.

[93] S. Bhatia, M. Motiwala, W. Mühlbauer, V. Valancius, A. Bavier, N. Feamster, L. Peterson, and J. Rexford, "Hosting virtual networks on commodity hardware," Georgia Tech, Tech. Rep. GT-CS-07-10, January 2008.

[94] "GENI: Global Environment for Network Innovations," http://www.geni.net/.

[95] G. P. Group, "GENI design principles," *Computer*, vol. 39, no. 9, pp. 102–105, 2006.

[96] A. Greenberg, G. Hjalmtysson, D. A. Maltz, A. Myers, J. Rexford, G. Xie, H. Yan, J. Zhan, and H. Zhang, "A clean slate 4d approach to network control and management," *SIGCOMM Computer Communication Review*, vol. 35, no. 5, pp. 41–54, 2005.

[97] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford, "Virtual routers on the move: Live router migration as a network-management primitive," in *Proceedings of the ACM SIGCOMM'08*, 2008, pp. 231–242.

[98] Y. Zhu, J. Rexford, A. Bavier, and N. Feamster, "UFO: A resilient layered routing architecture," Princeton University, Tech. Rep. TR-780-07, 2007.

[99] S. F. Bush, "Active virtual network management protocol," in *Proceedings of the Workshop on Parallel and Distributed Simulation*, 1999, pp. 182–192.

[100] B. Raghavan, K. Vishwanath, S. Ramabhadran, K. Yocum, and A. C. Snoeren, "Cloud control with distributed rate limiting," in *Proceedings of SIGCOMM'07*, 2007, pp. 337–348.

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

[102] J. Lu and J. Turner, "Efficient mapping of virtual networks onto a shared substrate," Washington University, Tech. Rep. WUCSE-2006-35, 2006.

[103] 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)*, 2003, pp. 3004–3008.

[104] A. Gupta, J. M. Kleinberg, A. Kumar, R. Rastogi, and B. Yener, "Provisioning a virtual private network: A network design problem for multicommodity flow," in *ACM Symposium on Theory of Computing*, 2001, pp. 389–398.

[105] J. Fan and M. Ammar, "Dynamic topology configuration in service overlay networks - a study of reconfiguration policies," in *Proceedings of the IEEE INFOCOM'06*, 2006.

[106] M. Yu, Y. Yi, J. Rexford, and M. Chiang, "Rethinking virtual network embedding: Substrate support for path splitting and migration," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 17–29, April 2008.

[107] J. He, R. Zhang-Shen, Y. Li, C.-Y. Lee, J. Rexford, and M. Chiang, "Davinci: Dynamically adaptive virtual networks for a customized internet," in *ACM CoNEXT*, 2008.

[108] D. M. et al., "Core network design and vendor prophecies," in *NANOG 25*, June 2003.

[109] S. Karlin and L. Peterson, "VERA: An extensible router architecture," *Computer Networks*, vol. 38, no. 3, pp. 277–293, 2002.

[110] L. Mathy, N. Egi, M. Hoerdt, A. Greenhalgh, and M. Handley, "(Some) implementation issues for virtual routers," in *Workshop on Management of Network Virtualisation*, 2007.

[111] M. Kolon, "Intelligent logical router service," Juniper Networks, Tech. Rep. 200097-001, October 2004.

[112] J. Fu and J. Rexford, "Efficient IP-address lookup with a shared forwarding table for multiple virtual routers," in *ACM CoNEXT*, 2008.

[113] M. K. Chowdhury, F. Zaheer, and R. Boutaba, "iMark: An identity management framework for network virtualization environment," in *The 11th IFIP/IEEE International Symposium on Integrated Network Management (IM)*, 2009.

[114] M. Feridan, M. Moser, and A. 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)*, October 2007, pp. 57–60.

[115] A. P. Jayasumana, Q. Han, and T. H. Illangasekare, "Virtual sensor networks - a resource efficient approach for concurrent applications," in *Proceedings of the International Conference on Information Technology (ITNG'07)*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 111–115.

[116] G. Smith, A. Chaturvedi, A. Mishra, and S. Banerjee, "Wireless virtualization on commodity 802.11 hardware," in *Proceedings of the the second ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization (WinTECH'07)*. New York, NY, USA: ACM, 2007, pp. 75–82.

[117] D. Hausheer and B. Stiller, "Auctions for virtual network environments," in *Workshop on Management of Network Virtualisation*, 2007.

[118] ——, "PeerMart: Decentralized auctions for bandwidth trading on demand," http://ercim-news.ercim.org/content/view/100/254/, January 2007.

[119] A. Shoykhet, J. Lange, and P. Dinda, "Virtuoso: A system for virtual machine marketplaces," Northwestern University, Tech. Rep. NWU-CS-04-39, July 2004.

[120] A. Feldmann, "Internet clean-slate design: What and why?" *SIGCOMM Computer Communication Review*, vol. 37, no. 3, pp. 59–64, July 2007.