Cabernet: Connectivity Architecture for Better Network Services

Yaping Zhu* Rui Zhang-Shen* Sampath Rangarajan† Jennifer Rexford*
*Princeton University †NEC Labs America

ABSTRACT

Deploying and managing distributed network services is exceptionally challenging, even for existing providers that have a large installed base of equipment. On one hand, a *network provider* can offer services over its own network, albeit with a relatively small geographic footprint. On the other hand, despite having servers at many locations, a *service provider* must rely on an underlying network that provides only best-effort packet delivery. In this paper, we propose Cabernet (Connectivity Architecture for Better Network Services), a three-layer network architecture that lowers the barrier for deploying wide-area services. We introduce the *connectivity* layer, which uses virtual links purchased from *infrastructure* providers to run virtual networks with the necessary geographic footprint, reliability and performance for the *service* providers. As an example, we present a cost-effective way to support IPTV delivery through wide-area IP multicast that runs on top of a reliable virtual network.

1. INTRODUCTION

Deploying and running wide-area network services is immensely challenging. Service providers typically must deploy servers in various geographic locations and purchase bandwidth from different network providers. If the service becomes successful, the service providers must rapidly grow their infrastructure to keep pace with demand. Moreover, since best-effort packet delivery is not sufficient for many real-time services, service providers must design their own application-layer mechanisms to adapt to network performance and reliability problems. Network providers, like large ISPs, can avoid these problems by offering services over their own dedicated network infrastructure [1, 2]. However, network providers have a limited geographic footprint, restricting the services to customers in their own domain.

In this paper, we present Cabernet (Connectivity Architecture for Better Network Services), a three-layer network architecture that lowers the barrier to deploying new wide-area services. The core of this architecture is the *connectivity layer* that enables service deployment in two main ways:

- **Large geographic footprint:** A connectivity provider constructs a wide-area virtual network, spanning equipment owned and managed by multiple infrastructure providers. This obviates the need for individual service providers to form their own business relationships with infrastructure providers. By carrying a large amount of traffic on behalf of many service providers, a connectivity provider can negotiate lower prices and make more efficient use of the underlying resources.
- **End-to-end control:** With complete control over its virtual network, a connectivity provider can run protocols and mechanisms that offer end-to-end performance and reliability tailored to a particular class of services (e.g., VoIP, gaming, and IPTV). Each service provider can then deploy end-to-end services on its own virtual network, constructed using the connectivity provider's nodes and links.

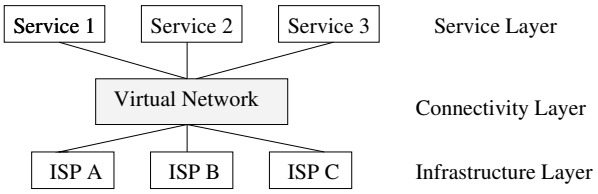

Figure 1: Hourglass Model of the Layered Architecture

Figure 1 illustrates the relationships between the different layers in the Cabernet architecture, where the connectivity provider's virtual network forms the "narrow waist." While the figure illustrates a single connectivity provider, multiple virtual networks operated by different connectivity providers could run independently of each other.

The Cabernet architecture simplifies service deployment by offering a simple abstraction for service providers. Each service provider has the illusion of a dedicated, wide-area virtual network that can easily expand as the service grows, which substantially lowers the barrier to deploying new services. The service provider can run "intradomain" protocols over its virtual network, without regard for the many underlying infrastructure networks. The service provider's virtual network is not affected by underlying performance and reliability problems, as those are handled by the lower layers.

Although quite different than today's architecture, Cabernet's model of hosting virtual networks is a logical extension to the existing model of hosting servers in data centers; in Cabernet, not only computing resources, but also the whole virtual network, are made available to service providers.

Realizing Cabernet introduces many challenges: How do the connectivity providers build virtual networks, and what do they need from the infrastructure providers? What is the functionality required at the infrastructure routers or servers to realize Cabernet with high performance? How can network services run on this layered architecture? The rest of the paper discusses how these challenges are addressed in Cabernet design. We start with an overview of the connectivity layer in Section 2, followed by the design of infrastructure node in Section 3 to implement the layered architecture. Then, we use IPTV as a case study to illustrate how Cabernet facilitates network service deployment in Section 4. We discuss related work in Section 5 and conclude in Section 6.

2. THE CONNECTIVITY LAYER

In this section, we present the design of the connectivity layer, beginning with an overview of its functionality. We next describe how the connectivity layer constructs a virtual network (from virtual nodes and links obtained from infrastructure providers). Then we discuss how the connectivity layer can provide end-to-end performance and reliability (tailored to the needs of the service layer).

2.1 Overview of Connectivity Providers

The main responsibility of the connectivity layer is to build virtual networks that span multiple infrastructure providers, so that a service provider can lease the exact virtual network it needs from a single connectivity provider, and have complete visibility and control within this virtual network. Thus, the connectivity layer significantly lowers the barrier of entry for service providers, as they do not need to interact with multiple infrastructure providers. Many service providers may have similar performance requirements, so a single connectivity provider can run a single virtual network with certain performance guarantee, and lease "slices" of this network to different service providers. This is more efficient than each service provider managing its own (smaller) network to achieve the desired performance. In addition, a service provider can easily have end-to-end control in the virtual network, even if the network's footprint spans multiple infrastructure providers.

A connectivity provider may obtain a large number of nodes and a rich mesh of links from infrastructure providers, and may use only a subset of them to provide a virtual network to a service provider. Figure 2 shows an example. In a virtual network (VN) customized for its needs, a service provider can run its own routing protocol among the nodes, and populate the forwarding tables accordingly to control traffic flow. Even though Figure 2 shows only one virtual network, a connectivity provider can support many virtual networks, presumably one for each service provider. To differentiate the virtual networks in different layers, we call the virtual network that a connectivity provider operates a *connectivity VN*, and the virtual network that a service provider operates a *service VN*.

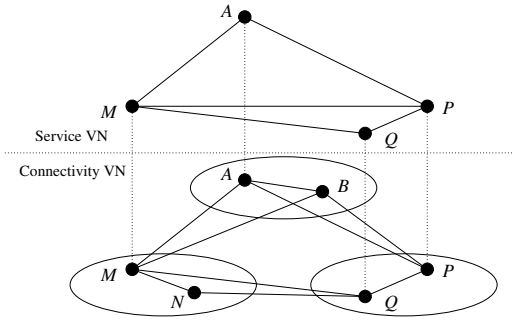

Figure 2: The infrastructure providers (ovals), the connectivity virtual network, and a service virtual network

A service provider may request that its virtual network have certain reliability and performance guarantees. For example, a service provider carrying Voice-over-IP (VoIP) and gaming traffic may prefer a network with low latency and low loss; a service provider that carries large files may request paths with high throughput. A connectivity provider can obtain links requested by the service providers with the desired properties directly from the infrastructure providers, but sometimes it may not be possible or economical to do so. Alternatively, the connectivity provider can set up tunnels and backup paths or use multipath routing to "create" virtual links with the required properties.

2.2 Virtual Links From Infrastructure Layer

The connectivity layer builds a virtual network from the virtual links and nodes it obtains from the infrastructure layer. The virtual links can be within one infrastructure provider, or span multiple of them, and may or may not come with certain performance guarantees. The connectivity layer can monitor these links' performance to enforce accountability.

Virtual links from one domain. A single infrastructure provider can provide a virtual link in several ways. 1) A virtual link can be an optical circuit, realized by wavelength-division or time-division multiplexing. Such virtual links are limited by the optical fiber footprint. 2) The widely used Multi-Protocol Label Switching (MPLS) can provide virtual links (tunnels) between two nodes that are not directly connected. 3) IP-in-IP encapsulation provides tunnels with best-effort service. The first two methods can provide links with protection and recovery mechanisms [3], as well as quality of service (QoS) and bandwidth guarantees [4, 5].

Virtual links across domains. Virtual links can be provided across multiple infrastructure networks through collaboration of infrastructure providers. One option is to signal MPLS paths across multiple domains [6], and online

path computation can find paths to meet certain bandwidth, latency, cost, or QoS constraints. Alternatively, neighboring infrastructure providers can negotiate to establish virtual links that span their networks, and coordinate to switch the virtual links to alternate paths in response to failures and congestion [7].

Accountability. Accountability is established between the infrastructure layer and the connectivity layer on whether the virtual nodes and virtual links meet their performance requirements. For a virtual link spanning two infrastructure providers, both of the providers are accountable and responsible for the link's performance. To prevent the infrastructure provider from having persistent performance degradation on the virtual links, the connectivity layer can use probes and deploy path-quality monitoring protocols [8, 9] to monitor whether the loss rate and delay of the virtual links exceed their thresholds. This way, even if the performance is allowed to degrade due to occasional network failures or congestion, the infrastructure provider will be held responsible if the overall performance of the virtual links does not meet expectations. Note that since the connectivity layer only needs to verify the long-term performance of the infrastructure layer, it does not need to collect measurements data and react in real time. Thus the overhead is much less than real-time performance monitoring.

Besides the recovery mechanism and QoS support of the virtual links, the connectivity layer may choose to receive more support from the infrastructure layer. For example, instead of simply monitoring performance on the virtual links in the connectivity VN, the connectivity layer can subscribe to the infrastructure layer to receive notification about changes in network conditions (such as routing failure or network congestions) [10]. Multiple connectivity VNs can also share the same monitoring infrastructure at the infrastructure layer, which helps improve the efficiency of monitoring.

2.3 Customized Service Virtual Networks

With the resources it obtained from the infrastructure providers, a connectivity provider can build customized virtual networks for the service providers. The connectivity layer can offer service VNs with protection and recovery mechanism on the virtual links. It can also offer virtual networks with customized routing to select paths with QoS constraints, using reactive routing protocols in spirit to RON [11, 12]. Services may require special transmission of packets at the connectivity layer, such as multi-paths delivery, packet scheduling, or transmitting duplicate packets. The connectivity layer can also help service providers to perform load balancing and traffic engineering in their service VN.

The connectivity provider has a range of options to provide service virtual networks with performance guarantees: in one extreme, it can obtain virtual links with the desired performance from the infrastructure providers, and just stitch them together; in the other extreme, it may obtain virtual links with no performance guarantees, and run its own protocols to create virtual links with the desired performance. Links from the infrastructure providers with performance guarantees are likely more expensive, while running its own protocols to provide such guarantees incurs management overhead. Depending on the situation, the connectivity provider can strike a balance between the two, and operate somewhere between the two extremes.

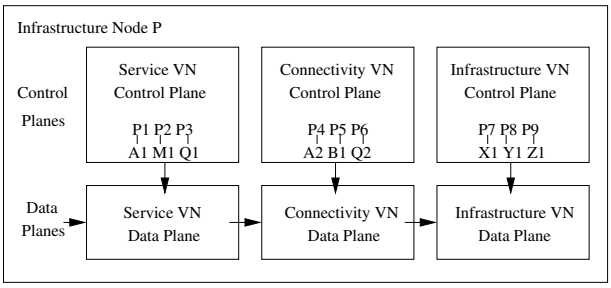

Figure 3: Systems Architecture of Infrastructure Node P

3. INFRASTRUCTURE NODE DESIGN

The infrastructure layer consists of networks operated by individual infrastructure providers, and its basic functionality is to host connectivity virtual networks over its physical nodes and links. The infrastructure layer is responsible for hosting the control-plane software, populating the routing table to the forwarding plane, and providing virtual links to direct packets between virtual nodes.

In this section, we focus on the node architecture of the infrastructure layer. We first describe how to instantiate virtual networks at different layers on the infrastructure nodes. Then, we present the run-time support for the control planes and data planes.

3.1 Instantiating Virtual Networks

A connectivity or service virtual network is composed of a network topology, a control plane to compute routes, and a data plane to forward packets. They can be implemented inside an infrastructure node by virtualization techniques [13, 14, 15]. The control plane is a process running inside a virtual machine. Inside the virtual machine, interfaces are configured as ends of virtual links according to the topology. Figure 3 shows the architecture of infrastructure node P from Figure 2. Three virtual machines are running in parallel for the service VN, the connectivity VN, and the infrastructure network respectively. The infrastructure layer establishes virtual links on behalf of the connectivity layer, and associates them to the interfaces that the connectivity layer sees. For instance, in Figure 3, node P connects to three virtual links in the connectivity VN: $P4$-$A2$, $P5$-$B1$ and $P6$-$Q2$. Thus, the infrastructure layer allocates three interfaces, $P4$ to $P6$, in the virtual machine of connectivity VN, and associates them to the interfaces of $A2$, $B1$, and $Q2$, on the other end of the virtual links in the connectiv-

ity layer. Similarly, the connectivity layer establishes virtual links on behalf of the service layer, and associates them to the interfaces that the service layer sees. In the virtual machine of the infrastructure network, interfaces $P7$ to $P9$ are associated with the physical interfaces on node P, which are directly connected to the neighboring physical interfaces $X1$, $Y1$, and $Z1$, on nodes not shown in Figure 2.

Note that the virtual links in the service VN may not map directly to the virtual links in the connectivity VN. For example, the virtual link P-M in the service VN may map to the virtual links P-Q-M as the primary path and P-A-M as the backup path in the connectivity VN. Moreover, not all infrastructure nodes contain all of the three layers. For instance, in Figure 2, the nodes in the infrastructure providers that do not host connectivity layer nodes are omitted, and nodes B and N only host the infrastructure layer and the connectivity layer, and are not visible to the service VN. Thus, the infrastructure provider does not need all the capabilities on every node inside its network, but only on a subset that it uses to host the connectivity VNs and service VNs.

The infrastructure layer is responsible for resource allocation and admission control of the connectivity VN. Similarily, the connectivity layer is responsible for resource allocation and admission control of the service VN. For example, a connectivity provider which hosts two service VNs may own 60% of the CPU on the infrastructure node. The connectivity provider may decide to give 40% of its share of the CPU to each service VN. In this case, both service VNs get 24% of the total CPU on that infrastructure node. Traffic shaper and queueing mechanisms may be employed by the infrastructure provider to achieve the desired bandwidth allocation among the many connectivity VNs and service VNs.

3.2 Run-time Support for Virtual Nodes

Inside the virtual machines of different layers, the corresponding provider can run any control-plane protocols. For example, a infrastructure provider can run routing protocols like OSPF and BGP, and a connectivity provider can run a reactive routing protocol to build a reliable virtual network. The virtual machines provide isolated address space and resources for each control-plane process. Routing tables computed by the control planes are populated in run time at the corresponding data plane, as illustrated by the vertical arrows in Figure 3.

Figure 3 illustrates how the data planes of different layers are connected in an infrastructure node. Data packets may have to pass through multiple data-plane stages, one for each layer that exists in the node. Packets are passed between the layers through encapsulation and decapsulation. When the node receives a packet, the packet is decapsulated and demultiplexed to the top layer VN that the packet belongs to. Then in order to find the interface where the packet should leave the node, the node performs lookup from the top layer down, encapsulating the packet along the way. Thus, the data plane in the infrastructure node should provide hierarchical forwarding tables and packet lookup, as well as packet encapsulation and decapsulation. Note that during the different stages of packet processing, the data packets can be stored in memory, and only packet header and control signals need to be passed between the data plane elements.

4. CASE STUDY: IPTV DELIVERY

In this section, we use IPTV as a case study to illustrate the effectiveness of building network services based on the Cabernet architecture. We present the idea of running wide-area IP multicast over a reliable virtual network to have efficient IPTV data delivery, and describe the control and data planes running at different layers in Cabernet. We then compare our design with existing IPTV deployment.

4.1 Multicast over Reliable Virtual Network

Wide-area deployment of IPTV service is especially challenging: multimedia distribution, especially broadcast TV distribution, is characterized by high bandwidth requirement and tight latency and loss constraints, even under failure conditions. Therefore, network design for IPTV service must meet the challenges of both reliability and good quality of service. In terms of reliability, the challenge is to minimize disruption after failure, including miminizing routing convergence time and minimizing the churn in the multicast distribution tree. As for quality of service, the challenge is to adapt to changes in network conditions such as congestion, and select routes to meet certain latency, loss rate, and bandwidth requirements.

In Cabernet, we can provide a reliable wide-area virtual network at the connectivity layer, so that service providers can simply run IP multicast to distribute content in its own service VN. This design makes the job of the service provider simple. Upon failures in the infrastructure, links are rerouted in the connectivity layer or below, transparent to the service layer. The connectivity layer can dynamically adjust to changes in network conditions (such as failures and congestion) to ensure the QoS of the virtual links in the service layer. Thus, a service provider can simply build a multicast tree in its VN, and expect no churns and good QoS.

A service provider can run any multicast protocol (e.g., PIM) with complete control over its service VN, which resembles a single network domain. Routes are computed to map multicast group addresses to interfaces in the service VN. The connectivity layer has various options to build a reliable service VN, as discussed in Section 2.3.

Next, we describe how the data planes run in different layers in Cabernet. As illustrated in Figure 3, data packets go through three stages in the data plane during lookup: first, the multicast group address is mapped to one or multiple nexthop interfaces in the service VN and packets are duplicated and encapsulated correspondingly; the service-layer interface is then mapped to a nexthop interface at the connectivity layer and packets are encapsulated correspondingly; finally, the connectivity-layer interface is mapped to a

physical interface in the infrastructure layer and packets are sent to the nexthop router.

The connectivity layer is responsible for handling failures and congestion in the infrastructure layer. For example, suppose in Figure 2, the performance of the connectivity-layer virtual link P-Q degrades due to congestion. The connectivity-layer node P detects this, and since the link is shared by two service-layer virtual links P-Q and P-M, node P decides to reroute link P-M through node A. The connectivity-layer control plane updates the outgoing interface of link P-M, and data packets will be routed accordingly. After the change, the congestion on connectivity-layer link P-Q is alleviated, and both service-layer links P-Q and P-M traverse non-congested paths. This change is transparent to the service layer, and no actions are required from the service layer.

Although the three layers in Cabernet can operate independently, in practice, cross-layer optimization can improve the efficiency of utilizing network resource. For instance, multicast in the service or connectivity layer may result in multiple copies of the same packet traversing a physical link, because multiple virtual links go through the same physical link. To reduce bandwidth consumption, an infrastructure provider can run multicast protocols within its own domain, and provide that as a feature to the connectivity layer. Providing additional supports like this is a way for infrastructure providers to compete with each other. Similarly, a connectivity provider can run multicast in a service virtual network on behalf of the service provider, and connectivity providers can compete with each other based on the services they provide.

4.2 Comparison to Today's IPTV Deployments

We now compare our design to existing IPTV deployments by network providers and service providers respectively.

Network providers have deployed private networks to offer cable-TV-like services [1, 2]. A common key design element of these networks is the use of a single IP multicast tree within a single network domain, which is efficient in terms of bandwidth usage. For some failure scenarios, especially single link failures, reliability can be achieved by reroute through pre-computed backup paths. However, building the private network infrastructure can be expensive, and the service is still limited to the footprint of a single ISP. Failure scenarios which are not computed in advance can cause churns in the multicast tree and cause congestion. In comparison, Cabernet can provide a large geographical footprint spanning multiple infrastructure providers, and achieve reliability and good quality of service transparent to the service virtual network.

Service providers have used their existing Content Distribution Network infrastructure to provide live-streaming services for their customers [16]. Service providers typically deploy servers widely, and obtain bandwidth and connectivity from multiple ISPs. Since only best-effort packet delivery is available in today's IP networks, application-layer mechanisms for circumventing network performance and reliability problems are used. There are at least three problems with this approach: 1) It is inefficient to monitor performance degradation from the application layer, due to limited visibility into the networks. 2) The service providers have to deploy a large number of servers in different geographical locations, in order to reroute through disjoint paths after failures, which is infeasible for small service providers. 3) Since IP multicast in wide-area networks is largely unavailable, application-layer multicast is commonly used, causing inefficient use of network and server resources. In contrast, in the Cabernet architecture, the connectivity layer has complete visibility and control over its own virtual network, and therefore can build service virtual networks that meet the reliability and performance requirements. In addition, multicast can be provided by the lower layers to the service layer. Thus, Cabernet lowers the barrier of entry for small service providers.

Many Peer-to-Peer (P2P) systems [17, 18] have been used for live streaming. P2P has the advantage of obtaining inherent scalability and geographical footprint from the participating peers. But P2P suffers from poor performance [19], because it relies on the best-effort routing in IP networks, it is limited by the peers' low uplink capacities, and the unpredictable peer dynamics causes peer churns. In comparison, Cabernet can provide high-quality IPTV service with great reliability.

5. RELATED WORK

Several proposals have argued that the current Internet is at an impasse, because new architectures cannot be deployed, or even adequately evaluated. The architectural "pluralists" [20, 21, 22, 23] use virtualization as a tool to introduce multiple (possibly competing) designs. Along these lines, Cabernet is most similar to the CABO architecture [23], which consists of infrastructure providers (who manage the physical infrastructure) and service providers (who deploy network protocols and offer end-to-end services). We take this argument one step further by advocating a middle layer that (i) forms business relationships with the infrastructure and service providers and (ii) runs customized protocols and mechanisms to offer service providers virtual networks with the necessary performance and reliability. The three-layer architecture in Cabernet can also be viewed as an incarnation of the virtual layers and meta-protocol abstractions presented in the Recursive Network Architecture [24].

Cabernet also relates to PlanetLab [25], a global experimental platform that pools resources from different research institutions to support experimental research on distributed systems. Like Cabernet, PlanetLab has a separation between infrastructure (i.e., servers and upstream connectivity provided by the participating institutions) and service providers (i.e., the many "slices" that run on top of PlanetLab), as well

as a middle layer (i.e., the PlanetLab management software) that forms relationships with both parties. Yet, Cabernet takes this approach further by having a middle layer that (i) constructs and runs a virtual *network*, including virtual links and packet-forwarding logic, and (ii) presents a separate virtual network to each service provider.

Cabernet is similar to "routing as a service" (ROSE) [26], which offers flexible routing control by having third-party providers compute end-to-end routes on behalf of their customers. In comparison, the connectivity layer in Cabernet not only computes routes on behalf of customers, but also (i) constructs and runs its own virtual network spanning multiple infrastructure providers and (ii) hosts virtual networks on behalf of multiple service providers. In contrast to ROSE, both the connectivity providers and the service providers have both control-plane and data-plane logic *running directly on the network elements*.

6. CONCLUSION

In summary, Cabernet presents a virtual network abstraction for the deployment of new network services. For future work, we are exploring more network services, and how Cabernet can effectively support their wide-area deployment. Example services of interest include VoIP, online gaming, and security [27].

Currently, Cabernet only supports IPv4 packet forwarding in its data planes. Network service may need dedicated hardware support for enabling different addressing schemes and special packet modification, since it is expensive to implement all these functions in software. We are exploring ways for the infrastructure node to provide configurable and programmable hardware [14, 28], leaving the flexibility for the connectivity layer to implement more customized functions for services.

7. REFERENCES

[1] R. Doverspike, G. Li, K. Oikonomou, K. K. Ramakrishnan, and D. Wang, "IP backbone design for multimedia distribution: Architecture and performance," in *Proc. IEEE INFOCOM*, 2007.

[2] D. Wang, G. Li, and R. Doverspike, "IGP weight setting in multimedia IP networks," in *Proc. IEEE INFOCOM*, 2007.

[3] J.-P. Vasseur, M. Pickavet, and P. Demeester, *Network Recovery: Protection and Restoration of Optical, SONET-SDH, and MPLS*. Morgan Kaufmann, 2004.

[4] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation Protocol (RSVP)." RFC 2205, September 1997.

[5] W. Liu, H. Karaoglu, A. Gupta, M. Yuksel, and K. Kar, "Edge-to-edge bailout forward contracts for single-domain internet services," in *Proc. International Workshop on Quality of Service*, 2008.

[6] A. Farrel, J.-P. Vasseur, and J. Ash, "A Path Computation Element (PCE)-Based Architecture." RFC 4655, August 2006.

[7] R. Mahajan, D. Wetherall, and T. Anderson, "Negotiation-based routing between neighboring ISPs," in *Proc. Networked Systems Design and Implementation*, 2005.

[8] S. Goldberg, D. Xiao, E. Tromer, B. Barak, and J. Rexford, "Path-quality monitoring in the presence of adversaries," in *Proc. ACM SIGMETRICS*, 2008.

[9] I. Avramopoulos and J. Rexford, "Stealth Probing: Efficient Data-Plane Security for IP Routing," in *Proc. USENIX Annual Technical Conference*, 2006.

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

[11] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. Symposium on Operating System Principles*, 2001.

[12] "Sureroute." `http://www.akamai.com/dl/feature_sheets/fs_edgesuite_sureroute.pdf`.

[13] A. Bavier, N. Feamster, M. Huang, L. Peterson, and J. Rexford, "In VINI veritas: Realistic and controlled network experimentation," in *Proc. ACM SIGCOMM*, September 2006.

[14] J. S. Turner, "A proposed architecture for the GENI backbone platform," in *Proc. Architectures for Networking and Communications Systems*, 2006.

[15] "Juniper Networks: Intelligent Logical Router Service." `http://www.juniper.net/solutions/literature/white_papers/200097.pdf`.

[16] L. Kontothanassis., R. Sitaraman, J. Wein, D. Hong, R. Kleinberg., B. Mancuso, D. Shaw, and D. Stodolsky, "A transport layer for live streaming in a content delivery network," *Proceedings of the IEEE*, vol. 92, no. 9, pp. 1408–1419, 2004.

[17] PPLive. http://www.pplive.com.

[18] PPStream. http://www.ppstream.com.

[19] W.-P. Yiu., X. Jin, and S.-H. Chan, "Challenges and approaches in large-scale P2P media streaming," *IEEE Multimedia*, vol. 14, no. 2, pp. 50–59, 2007.

[20] L. Peterson, T. Anderson, D. Culler, and T. Roscoe, "A blueprint for introducing disruptive technology into the Internet," in *Proc. HotNets*, 2002.

[21] T. Anderson, L. Peterson, S. Shenker, and J. Turner, "Overcoming the Internet impasse through virtualization," *IEEE Computer Magazine*, vol. 38, pp. 34–41, April 2005.

[22] J. Turner and D. Taylor, "Diversifying the Internet," in *Proc. IEEE GLOBECOM*, 2005.

[23] N. Feamster, L. Gao, and J. Rexford, "How to lease the Internet in your spare time," *ACM Computer Communication Review*, January 2006.

[24] J. Touch and V. Pingali, "The RNA Mataprotocol," in *IEEE ICCCN (Future Internet Architectures and Protocols track)*, 2008.

[25] L. Peterson, A. Bavier, M. E. Fiuczynski, and S. Muir, "Experiences building PlanetLab," in *Proc. Operating System Design and Implementation*, 2006.

[26] K. Lakshminarayanan, I. Stoica, and S. Shenker, "Routing as a service," Tech. Rep. CSD-04-1327, University of California, Berkeley, January 2004.

[27] A. Keromytis, V. Misra, and D. Rubenstein, "SOS: Secure Overlay Services," in *Proc. ACM SIGCOMM*, 2002.

[28] J. Turner and et al, "Supercharging PlanetLab: High performance, multi-application, overlay network platform," in *Proc. ACM SIGCOMM*, 2007.