UFO: A Resilient Layered Routing Architecture

Yaping Zhu *, Jennifer Rexford *, Andy Bavier *, Nick Feamster †
* Princeton University † Georgia Tech

Abstract

Conventional wisdom has held that routing protocols cannot achieve both scalability and high availability. Despite scaling rather well, today's Internet routing system does not react quickly to changing network conditions. Overlay networks, on the other hand, can respond quickly to changing network conditions, but their reliance on aggressive probing does not scale to large topologies. This paper presents an Underlay Fused with Overlays (UFO), which achieves the best of both worlds by providing router support for overlay routing mechanisms. UFO comprises two changes to existing routers: (i) data-plane support for deflecting packets at the behest of the overlay and (ii) explicit notification about significant changes in network conditions. We discuss how these two mechanisms facilitate the scalable deployment of overlay routing mechanisms at the network layer.

1 Introduction

Today's "one size fits all" routing system scales well, but at the expense of availability. Routers do not respond automatically to congestion, forcing traffic to traverse overloaded paths even when better paths exist. Transient disruptions during routing-protocol convergence [16, 29] degrade the performance of interactive applications like Voice over IP (VoIP) [3, 15, 19] and online gaming. In addition, today's routing protocols do not satisfy the diverse availability requirements of different applications. For example, some applications may prefer high-throughput paths while others prefer low loss and low delay, but today's routing protocols route all traffic over the same set of paths. In this paper, we propose a routing architecture that scales well and provides high availability for diverse applications by combining ideas from conventional routing protocols and overlay networks.

Our architecture is motivated by the success of overlay networks at providing reliable communication services, albeit at limited scales. For example, a Resilient Overlay Network (RON) [1] uses a small collection of hosts to form a topology where each overlay link traverses one or more hops in the underlying network. RON reacts more quickly than the underlying network to changes in network conditions; unfortunately, it does not scale to a large number of hosts, and the resulting paths can sometimes be inefficient. Traversing intermediate hosts adds delay and consumes bandwidth and processing resources. Quickly masking failures requires monitoring the underlying paths with frequent probes, which does not scale to large overlay topologies.

This paper argues that, to scale well *and* provide high availability, the routing architecture should consist of multiple layers. Overlay routing should be viewed as part of the routing system itself, not as "just another application". The key questions that a multi-layer routing system must address concern: (1) What *functions* should be placed at each layer of the routing system? and (2) What *interfaces* should these layers expose to one another? To explore these questions, we present a two-level routing architecture that supports high availability by reacting quickly to failures by borrowing some design principles and mechanisms from overlay networks. We believe that the lower layer, the "underlay", should support the following two functions:

- *Direct control over forwarding table entries.* We believe that routers should provide data-plane support for forwarding packets over "overlay" links, at the behest of the "overlay" control plane. This enables flexible control over the end-to-end paths, without the efficiency drawbacks (e.g., higher latency and wasted bandwidth) of traditional overlay networks.
- *Explicit notification about changing network conditions.* We believe that routers should notify the overlay about critical changes in the properties of an overlay link. Explicit notifications improve the efficiency of reactive routing at the "overlay" layer without compromising scalability. This raises important questions concerning who to notify about what kinds of events, and how to design a scalable notification system.

We call our system "underlay fused with overlays" (UFO) to emphasize the *cross-layer* nature of our design. In practice, we envision two main deployment scenarios. First, ISPs could view the two-layer design as an effective way to offer highly-available communication services to their customers. Second, ISPs could host third-party overlay services, offering them extra support for packet forwarding and explicit notification, as an extension of their server hosting business.

The remainder of the paper is structured as follows Section 2 describes how routers can host overlays and perform customized packet forwarding on high-speed links. Then, Section 3 discusses how the routers generate notification messages when overlay links fail or become heavily congested. Section 4 describes enhancements to UFO for better scalability, followed by a discussion of related work in Section 5. Section 6 concludes with a discussion of the economic incentives for deploying UFO.

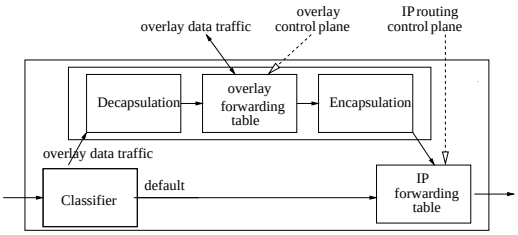


Figure 1: Line cards provide support for overlay forwarding.

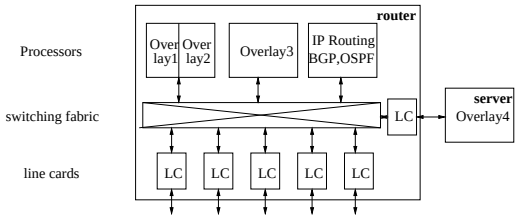


Figure 2: The router itself hosts the overlay control plane.

2 Efficient Overlay Forwarding

UFO proposes that routers provide *line-card support* for forwarding packets on overlay links and expose a standard interface to allow a *separate control plane* (either running as an overlay or as a remote server) to install forwarding table entries directly on the routers. This function circumvents some forwarding inefficiencies that today's overlay routing incurs: specifically, it not only speeds up overlay packet encapsulation and decapsulation but also reduces traffic going both inbound and outbound to reach the overlay servers, which are traditionally located at the network edge.

2.1 Overlay Forwarding on Line Cards

Overlay servers use tunnels to communicate with each other and participating end hosts[1]. A receiving server decapsulates the data packet, performs a lookup in an overlay forwarding table, and reencapsulates the packet with the address of the next tunnel end point in the overlay path. Instead of terminating at separate overlay servers, the tunnel could terminate at the router itself: the router's line cards could recognize encapsulated packets based on an outer header, such as a destination address and port number. The line card can terminate multiple overlay links, perhaps belonging to different overlay networks, using different (address, port) pairs.

After decapsulating the packet, the router indexes into an overlay forwarding table populated by the overlay control plane, as shown in Figure 1. Since each overlay network has its own forwarding-table entries, the line card must include the overlay identifier (e.g., the port number) when indexing the forwarding table. Based on lookup results, the line card reencapsulates the packet and forwards it toward the next overlay node. The line card performs a second lookup in the IP forwarding table to direct the packet to the outgoing link en route to the next overlay node. This data-plane pipeline can be easily implemented at high speed, e.g., on line cards with network processors or custom logic.

2.2 Hosting the Overlay Control Plane

The contents of the two forwarding tables come from different sources. As in a traditional router, the IP forwarding table is generated by the routing software by combining information from various routing protocols (e.g., OSPF and BGP), but the entries in the overlay forwarding table are generated by overlay networks. Each overlay can add, remove, and modify its own forwarding-table entries but does not have permission to modify the entries belonging to other overlays. To avoid exhausting the memory, the line card may limit the number of entries per overlay.

A natural question arises about *where* the overlay control plane should run. The overlay control plane could conceivably run on a separate set of servers, as illustrated by Overlay 4 in Figure 2. In this model, only the data-plane functions (i.e., "in network" packet reflection) moves to the routers, while the control-plane function remains separate. Because the data packets are reflected "in network", the overlay server would not need to exchange much traffic with the routers, minimizing the bandwidth and processing requirements for the overlay server. In addition, the overlay server may be relatively cheap, compared to router, making this an inexpensive solution. However, allowing the overlay server to communicate with the router over the network introduces some security challenges, such as the need to prevent unauthorized access and denial-of-service attacks. In addition, the control and data planes may fail independently, introducing additional challenges.

Instead, running the control-plane software directly on the router allows for fast updates of the forwarding tables, and avoids extra loss and delay for control packets to reach the overlay control plane. Moreover, today's high-end routers already have the ability to host overlay control planes. Routers typically consist of line cards and processors connected by a switching fabric. Most packets travel directly from one line card to another, though some packets (e.g., OSPF and BGP messages) are sent or received by one of the processors. In addition to conventional IP routing, the processors in the router could run overlay control planes; router virtualization [20], the process by which one physical router acts as several logical routers, makes this even easier. For example, in Figure 2, router could host overlay control plane on dedicated processors or dedicated processor cores (Overlay 3), or virtual machines with a dedicated share of system resources (Overlay 1 and Overlay 2). Each approach provides resource isolation between the overlay control planes, while also protecting the resources needed by the IP control plane.

3 Scalable Overlay Monitoring

To improve scalability, UFO provides explicit notification about events that affect the overlay links (e.g., reachability

[1]End hosts can employ a variety of techniques for "opting in" to the overlay service [13, 17, 18].

failures, congestion). Different types of overlays may require different types of notifications: for example, an overlay designed for video might be more concerned with jitter than one defined for bulk file transfer. We design the notification mechanism to preserve overlay link abstractions: overlays are deployed without knowledge of underlying topology, and underlay can send notifications to overlays about overlay links without exposing individual underlay link properties. Routers store state about the overlay links which travel through them, which is done by explicit registration. In this section, we describe how UFO registers an overlay link and sends explicit notifications in response to various network events. We also describe how overlays can recover without explicit notification from the underlay.

3.1 Registration of Overlay Links

An overlay node registers a unidirectional overlay link by sending a registration message to the router hosting the remote overlay node, including the event types of interest. The routers along the underlay path see this message and store the (source, destination, event) pair, along with information about the next hop in the path. In Figure 3, the overlay node S registers the overlay links S-D1 (following the underlying path S-R1-R2-R3-D1) and S-D2 (following the underlying path S-R1-R2-R4-D2); similarly, D1 registers D1-S and D2 registers D2-S. Suppose the R2-R3 link fails, R2 can notify S that the overlay link S-D1 has failed, and R3 can notify D1 that the overlay link D1-S has failed. Separate registrations for each unidirectional overlay link ensures registration is correct when paths are asymmetric.

To ensure correct operation even when paths change, registration messages are stored as soft state: the overlay nodes periodically refresh the registration and the routers discard stale registrations. Moreover, each registration packet carries a version number that the routers store and include in notification packets. This ensures that the overlay nodes can safely ignore notification packets with out-of-date version numbers, to avoid erroneously reacting to events along an old forwarding path.

When processing a registration message, a router verifies that the two overlay nodes are allowed to register an overlay link. To prevent denial-of-service attacks and unauthorized use of the service, the routers must verify that the registration packet is authorized. Verification may involve consulting a separate server or a locally cached list of valid participating nodes. The ISP could also prevent source-address spoofing by network ingress filtering. Another option for authentication is to have registration packets carry a capability. In this way, access to registration and notification service will be efficiently controlled by light-weight authentication cookies, such as those found in L2TPv3.

3.2 Notification of Network Events

The router must notify registered overlay nodes about various kinds of network events which would affect the performance of an overlay link, including physical failure of routers of links, heavy congestion, or lost reachability due to policy changes or session failures. To generate the notifications in timely fashion, the router must have efficient access to the registration information. The router could maintain a single data structure on the control plane processor, and consult the table to generate notifications upon learning about a network event affecting a particular outgoing link. Alternatively, each line card could store just the registration entries that correspond to the outgoing link. This approach allows line cards to create and update table entries as part of data-plane processing of the registration packets and to automatically send notification packets when the link experiences a failure or heavy congestion.

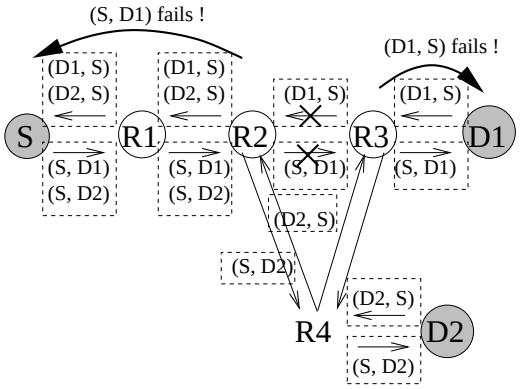


Figure 3: Overlays receive explicit notifications about faults.

Only the router that is directly incident to the offending underlay link sends a notification: for example, in Figure 3, in the event of a problem on the R2-R3 link, only R2 would send a notification to S. Some downstream failures might cause R1 to switch to a new underlay path to reach D1 or D2, or cause the overlay link S-D1 or S-D2 to no longer traverse R1, but since registration information is soft state, R1 would eventually delete any out-of-date registration information. Similarly, if a second event (e.g., such as a failure of the R1-R2 link) happens in the meantime, R1 would send an unnecessary notification to S, but the stale version number would allow S to safely discard the information.

Upon detecting a network event, the underlay node must determine who to notify by identifying the affected overlay links. In the preceding example, R2 must determine which overlay links (among S-D1, S-D2, D1-S and D2-S) are affected by the failure of the R2-R3 link. To do so, R2 matches the next-hop of each overlay destination with the failed output interface: because the overlay path S-D1 uses R2-R3 to reach the destination D1, R2 will notify the S that S-D1 is faulty. Similarly, R3 notifies D1 that (D1, S) is faulty. Moreover, R2 periodically sends notification messages to each affected overlay node until it receives an acknowledgment.

3.3 Lazy Re-registration of Overlay Links

After learning about a problem with an overlay link, the overlay control plane can quickly reroute (at the overlay level) to

circumvent the problem. Yet, an important question remains about how the overlay learns that the overlay link is performing well again. One seemingly natural approach would be for the underlay to send another notification message when the overlay link has recovered. However, determining that the underlay *path* (between two overlay nodes) has recovered is difficult, since no one router has sufficient visibility into the path-level performance. For example, even if a failed link recovers, the incident router may not know that all of the other routers have converged to a new, stable path. In addition, some of the routers on the new path may not have any registration state for the overlay link, if they were not on old path that received the registration packet. As such, UFO does *not* have the underlay send recovery messages to the overlays.

Fortunately, fast recovery of overlay links is not as important as fast notification of reachability or performance problems. Overlay networks typically have a rich (logical) topology, with numerous alternate paths to circumvent a problematic overlay link. As such, we rely on the overlay nodes to re-register the overlay link at some time in the future. The overlay node could be conservative and wait for several seconds (or minutes) for the underlying path to reconverge before attempting to re-register the overlay link. This keeps the design simple without compromising the high availability of the overlay routing layer.

4 Enhancing the Scalability of UFO

In this subsection, we describe the how to use IP multicast to improve the scalability of monitoring the overlay links. Then, we discuss how nesting of overlay links can offer additional scalability.

4.1 IP Multicast for Overlay Link Monitoring

The overhead of processing registration packets and generating notification packets increases with the number of overlay links. For example, Figure 4(a) shows the registration state on each underlay link after four overlay sources (S1 to S4) register overlay links to overlay destination D. The router R1 must process four registration packets (as well as periodic refreshes to maintain the soft state) and keep state about the four overlay links. If the R1-D link fails, R1 must send notification packets to each of the four overlay sources, as shown by the four arrows from R1 to R2. More generally, the storage, bandwidth, and processing resources for both registration and notification grow in proportion to the number of overlay links that traverse a given underlay link. These overheads could grow quite significant in large deployments.

Fortunately, UFO can capitalize on ideas from multicast protocols to improve the efficiency of registration and notification. Using a multicast tree rooted at the overlay destination ensures that each registration and notification packet traverses a link only once. The overlay sources join the multicast group to register for notifications about changes in reachability to the overlay destination. For example, in Figure 4(b), the four overlay sources join a multicast group for overlay links terminating at D. Suppose S1 joins the group first. Then, S1's registration message (i.e., join request) to D would traverse the path R3-R2-R1-D, and each node along the way would keep track of the outgoing link to forward packets sent to the multicast group. Later, when S2 joins the multicast group, router R3 grafts link R3-S2 into the multicast tree, but does not need to forward the registration request further. This process reduces the storage, bandwidth, and processing overhead for registration requests.

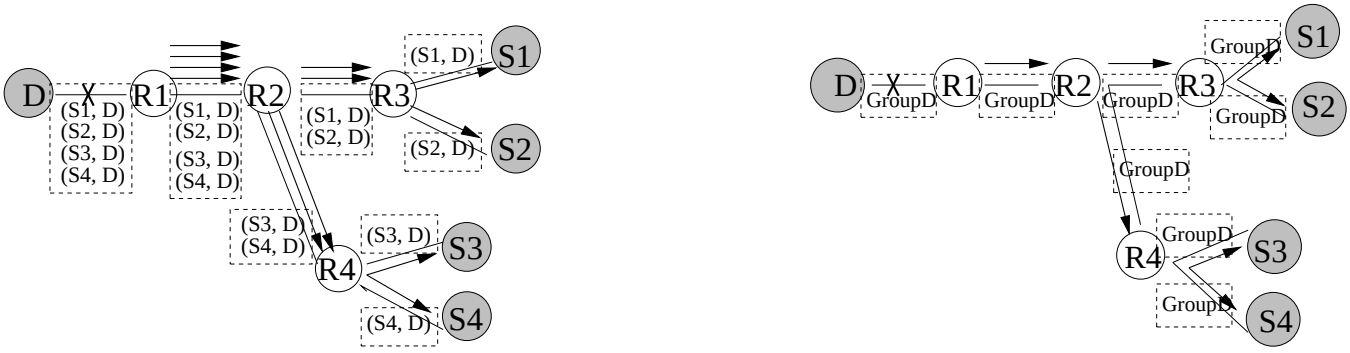
The multicast tree also reduces the overhead of disseminating notification packets. In particular, the notification packet will traverse the subtree rooted at the router that detected the network event. Suppose an the link R1-D experiences some fault. Upon detecting the event, R1 directs a notification packet to all downstream members in the multicast tree. Only one copy of the notification packet would traverse the downstream links, as shown by the arrows in Figure 4(b), using significantly less bandwidth than the unicast scenario shown (Figure 4(a)). Similarly, if an event occurs on the R3-R2 link, R3 would send a notification packet downstream to S1 and S2. Thus, a single multicast tree allows a node to send notifications that affect any overlay links that it terminates.

Many multicast protocols exist, with different complexity. Since UFO has relatively simple requirements, we envision using Protocol Independent Multicast (PIM) [9]; PIM "sparse mode" is well suited for UFO since the number of overlay nodes is small, relative to the number of underlying routers and links. UFO can also capitalize existing data-plane support for multicast forwarding in IP routers. However, multicast forwarding typically requires a multicast group address. UFO could conceivably assign a multicast group address to each overlay destination, at the expense of administrative overhead and consuming a large part of the multicast "class D" addresses. Instead, we envision that the data-plane of the routers would treat registration packets (destined to D) as implicit multicast join messages. Similarly, the router could treat notification messages as implicit multicast packets destined to the downstream members of the multicast group.

Multicast enables UFO's registration and notification mechanisms to scale to a large number of overlay nodes. In the unicast design, a fully-connected overlay with n nodes would have n^2 overlay links, each requiring registration state for one or more underlay links. In the multicast design, the routers need only to participate in at most n multicast groups. In fact, multiple overlay destinations (participating in different overlay networks) could conceivably *share* a single multicast group, if they were hosted on the same router. Suppose a second overlay node E runs on the same router as D. Then, notification messages could be distributed over a single multicast tree rooted at the hosting router. Then, the number of multicast groups would be limited to the number of routers that host overlay nodes.

4.2 Nesting of Overlay Links

In recent years, various kinds of tunneling techniques (such as MPLS or IP-in-IP) has been deployed to decouple the



(a) Unicast

(b) Multicast

Figure 4: Comparison of unicast and multicast notification

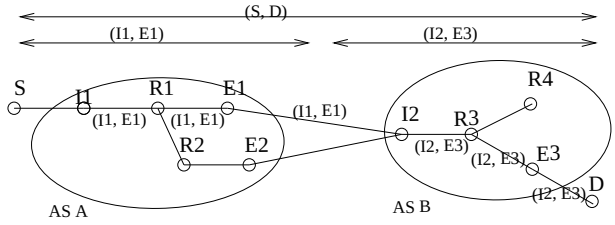


Figure 5: Nested registration and notification

routing control over inter-domain egress route selection and intra-domain next-hop route selection. For instance, in Figure 5, there are two overlay nodes S and D, whose overlay link traverses through the underlay path S-I1-R1-E1-I2-R3-E3-D in two autonomous systems AS A and AS B. Suppose AS A has two egress routers E1 and E2 to reach AS B, and the ingress router I1 switches to use E2 instead of E1 for some policy reason, then overlay link must be re-registered corresponding to change in egress route selection, and I1 needs to determine among all the (source, destination) pairs registered, which overlay link would be affected by this egress route change. In this subsection, we describe the solution to this problem by nested registration.

As shown in Figure 5, while registration packet enters AS A at ingress router I1, instead of registering for overlay link (S, D), I1 could register the overlay link between itself and the current egress E1 (I1, E1). Moreover, I1 also has to remember that (I1, E1) is a nested registration for (S, D). Under such circumstance, if the egress route selection changes to E2, I1 will notify S that (S, D) changes, since (I1, E1) is a nested overlay link for (S, D). Similarly, AS B could also register (I2, E3) at ingress router I2 for overlay link (S, D). In this way, the registration of (S, D) in composed of two nested registration of (I1, E1) and (I2, E3) in two autonomous systems, as illustrated in Figure 5.

Nested registration has several advantages, including: it solves the problem of re-registration while egress route selection is decoupled from next-hop route choice. Moreover, the same ingress/egress pair might be shared among different overlay pairs, which improves the scalability of UFO design.

5 Related Work

Detour recognized the potential benefits in redirecting traffic along overlay paths [6]; Resilient Overlay Networks (RON) [1] subsequently built a system based on this idea to improving the availability by quickly detecting paths with high latency or packet loss and rerouting around them; RON nodes probe aggressively to maintain information about alternate paths, which limits its scalability. By augmenting the underlay to provide some native support for overlay functions, UFO circumvents some of these scaling problems. Overlays also can make routing decisions tailored for different applications such as Content Distribution Networks [26], Multicast [5, 27], network level proximity [10, 30], and Quality of Service [25]. UFO preserves the cusomizability of overlays while eradicating many of the scaling and efficiency problems.

Jannotti proposes path reflection to reduce overlay forwarding inefficiency by allowing end hosts to request short-circuit packet routing and duplication in nearby routers [12]; UFO's line-card support for overlay forwarding implicitly pushes overlay reflection points inside the network. Nakao and Chen propose optimizations to reduce overlay measurement overhead by probing along most disjoint underlay paths, with inference or knowledge of the underlay network topology [4, 21]; UFO's explicit notification support pushes this monitoring functionality into all routers to improve scalability. Subsequent work on overlays has involved performance studies [7, 11], systems for managing of overlay deployments [13, 17, 18] and examinations of interactions between underlays and overlays [24].

UFO relates to many recent proposals that rely on network

virtualization to provide support for multiple routing systems operating in parallel on the same physical infrastructure [17]. Cabo [8] proposes an architecture with diversified infrastructure providers and service providers. Recent research on testbed support for virtualization [2, 22, 23, 28, 14] an initial platform for deploying UFO.

6 Conclusion

Routing faces a tension between high availability and scale. This paper presents a routing architecture that provides two new functions—direct control over forwarding table entries and explicit notification about changing network conditions overlays—to provide some of the benefits of overlay routing at the IP layer without incurring the same scaling limitations.

Although UFO requires additional router support, we believe that these changes to routers are relatively modest and, further, that ISPs nonetheless have sufficient incentive to augment their routers to provide this support. First, many ISPs already run overlay nodes of their own, (e.g., voice over IP (VoIP) gateways and IPTV servers). Second, the ISP could upgrade a small fraction of its routers to provide overlay forwarding support. Finally, providing explicit feedback about the performance of overlay links allows ISPs to offer better service to their overlay customers, giving them a competitive edge over non-participating ISPs.

This paper has presented an initial design for UFO, but we hope that, as we progress, it will enable overlay-based services to operate more scalably and efficiently. As a first-of-its-kind routing architecture that embraces routing as inherently multi-later, we believe that it will also generally offers important insights about how to design routing systems that are both reactive and scalable.

References

[1] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. *Proc. ACM SOSP*, 2001.

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

[3] C. Boutremans, G. Iannaccone, and C. Diot. Impact of link failures on VoIP performance. *Proc. NOSSDAV Workshop*, 2002.

[4] Y. Chen, D. Bindel, H. Song, and R. H. Katz. An algebraic approach to practical and scalable overlay network monitoring. *Proc. ACM SIGCOMM*, 2004.

[5] Y. Chu, A. Ganjam, T. S. E. Ng, S. Rao, K. Sripanidkulchai, J. Zhan, and H. Zhang. Early experience with an Internet broadcast system. *Proc. USENIX Symposium on Internet Technologies and Systems*, 2004.

[6] A. Collins. The Detour Framework for Packet Rerouting. Master's thesis, University of Washington, 1998.

[7] N. Feamster, D. Andersen, H. Balakrishnan, and M. F. Kaashoek. Measuring the effects of Internet path faults on reactive routing. *Proc. ACM SIGMETRICS*, 2003.

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

[9] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas. Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). `ftp://ftp.rfc-editor.org/in-notes/rfc4601.txt`, August 2006.

[10] M. Freedman, K. Lakshminarayanan, and D. Mazieres. OASIS: Anycast for any service. *Proc. USENIX/ACM NSDI*, 2006.

[11] K. P. Gummadi and H. V. Madhyastha. Improving the reliability of Internet paths with one-hop source routing. *Proc. OSDI*, 2004.

[12] J. Jannotti. *Network Layer Suppport for Overlay Networks*. PhD thesis, Massachusetts Institute of Technology, 2002.

[13] D. Joseph, J. Kannan, A. Kubota, K. Lakshminarayanan, I. Stoica, and K. Wehrle. OCALA: An architecture for supporting legacy applications over overlays. *Proc. USENIX/ACM NSDI*, 2006.

[14] S. Karlin and L. Peterson. VERA: An extensible router architecture. *Computer Networks*, 2002.

[15] N. Kushman, S. Kandula, and D. Katabi. Can you hear me now?! it must be BGP. *ACM Computer Communication Review*, 2007.

[16] C. Labovitz, A. Ahuja, A. Bose, and F. Jahanian. Delayed Internet routing convergence. *Proc. ACM SIGCOMM*, 2000.

[17] K. Lakshminarayanan, I. Stoica, S. Shenker, and J. Rexford. Routing as a service. Technical report, University of California, Berkeley, 2006.

[18] H. V. Madhyastha, A. Venkataramani, A. Krishnamurthy, and T. Anderson. Oasis: An overlay-aware network stack. *SIGOPS*, 2006.

[19] A. Markopoulou, F. Tobagi, and M. Karam. Assessment of VoIP quality over Internet backbones. *Proc. IEEE INFOCOM*, 2002.

[20] D. McPherson et al. Core network design and vendor prophecies. *NANOG 25*, 2003.

[21] A. Nakao, L. Peterson, and A. Bavier. A routing underlay for overlay networks. *Proc. ACM SIGCOMM*, August 2003.

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

[23] L. Peterson, A. Bavier, M. E. Fiuczynski, and S. Muir. Experiences building Planetlab. *Proc. OSDI*, 2006.

[24] S. Seetharaman, V. Hilt, M. Hofmann, and M. Ammar. Preemptive strategies to improve routing performance of native and overlay layers. *Proc. IEEE INFOCOM*, 2007.

[25] L. Subramanian, I. Stoica, H. Balakrishnan, and R. H. Katz. Overqos: An overlay based architecture for enhancing Internet QoS. *Proc. USENIX/ACM NSDI*, 2004.

[26] SureRoute. Sureroute. `http://www.akamai.com/dl/feature_sheets/fs_edgesuite_sureroute.pdf`.

[27] K. Svetz, N. Randall, and Y. Lepage. Mbone: Multicasting tomorrow's internet. *IDG Books Worldwide*, 1996.

[28] J. S. Turner. A proposed architecture for the GENI backbone platform. *ANCS*, 2006.

[29] F. Wang, Z. M. Mao, J. Wang, and L. Gao. A measurement study on the impact of routing events on end-to-end Internet path performance. *Proc. ACM SIGCOMM*, 2006.

[30] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A lightweight network location service without virtual coordinates. *Proc. ACM SIGCOMM*, 2005.