Routing as a Service

Karthik Lakshminarayanan Ion Stoica Scott Shenker
University of California, Berkeley

Abstract

Typically routing is either scalable but inflexible, such as current Internet routing, or flexible but unscalable, such as source routing with with per-flow route discovery. In this paper we argue that to achieve both flexibility and scalability, customized routing should be offered as a service *by third-party providers. The logical separation of routing from forwarding allows different route selection mechanisms to coexist and to evolve over time as routing requirements change.*

1 Introduction

The recent renaissance of Internet routing research (see [5, 6, 21, 30] for a sampling of the recent literature) has produced a bevy of new designs that alleviate one or more of the problems plaguing current routing protocols [21, 31]. In this paper we do not propose a particular new routing protocol but instead address the *structure* of routing. Currently, routing is limited to the default route offered by the infrastructure. Here we argue that to offer more routing flexibility to flows without sacrificing scalability, customized routing should be offered as a *service* by third-party providers. This approach would enable routing to constantly evolve to meet new application and performance requirements as they arise, without requiring the global adoption of a new routing protocol.

By *customized routing*, we mean that the routing requirements are idiosyncratic to individual flows, and may consider particular policy constraints or unusual routing metrics (not just QoS parameters[1]) or some combination thereof. Our goal is to provide customized routing as a way of enhancing Internet routing, not replacing it.

There has long been a tension between flexibility and scalability in routing algorithms. The one-size-fits-all approach of current routing is scalable because the overhead of route computation is amortized across all flows. While this approach is generally adequate, there have been recurrent efforts to allow end users to tailor their routes to better meet their performance and policy requirements. These efforts at customized routing, which were intended to augment rather than replace default routing, scaled poorly because typically the route computation was done separately for each flow. Source routing is the canonical example where individual sources independently explored potential routes. Thus, while source routing can accommodate specialized needs, it does not provide a scalable routing alternative for any sizable fraction of Internet traffic.

In this paper we seek to achieve both scalability and flexibility. We do so by leveraging two recent trends in routing research. The first is that of overlays. Work such as Detour [26], RON [3], and others have shown that one need not control every router in order to have a substantial impact on the routes packets take; controlling a much smaller set of *overlay* nodes is sufficient for meeting many routing goals. Overlays are not a solution to customized routing, because overlays typically offer a single default route instead of catering to flows' individual needs. Thus, while overlays can better the Internet's routing algorithm along one or more dimensions, such as performance [26] or responsiveness [3], they are not the solution to our problem. Instead, we use overlays coupled with *extensible* routing, rather than embedded routing, to provide the desired customization.

Making routing extensible relies on the second trend, which is to separate the computation of routes from the actual forwarding actions that implement those routes [14]. Forwarding, in which a router consults a routing table to choose an outgoing interface for a given packet, obviously must occur on the router itself. However, there is no inherent reason why the computation of the routing table must also occur on the router; the routing table could be computed elsewhere and transmitted to the router. For example, an ISP or enterprise could perform centralized route computation and distribute them to all routers in their domain. The recent RCP proposal [14] makes an eloquent case for this physical separation, describing a basic design and the associated challenges of such an approach.

Here we extend the separation to make routing and forwarding not only *physically* separate but also *logically* separate. That is, the route computations need not be performed by the same organization that operates the routers. In particular, we advocate that the route computations be performed by third parties (which we call Routing Service Providers, or RSPs) that are outside the routing infrastructure. Thus, our proposal is simply that customized routing be provided as a *service*. Our main point here is that routing be logically separated from forwarding; an ISP can also provide RSP service much like an ISP provides email service today in the Internet (though they are logically separate).

The *Routing as Service* (RAS) approach allows for both scalability[2] and flexibility. The fact that routes are provided by *services* allows the sharing of route exploration overhead between all customers of that service, making RAS scalable. The fact that these services are deployed by *third-parties* means that the route computation is not part of the fixed infrastructure but instead is provided by a diverse and competitive market. The route choices can thus be many in number

[1] Proposals for computing a few different paths based on different routing metrics, such as maximizing bandwidth, are merely limited extensions of default routing rather than truly customized routing.

[2] Here, we only try to argue qualitatively that our architecture allows for scalability; as we note later developing scalable mechanisms for particular sub-issues is still a daunting challenge.

and varied in nature, and can change over time as different requirements emerge. In this way, route computation can become a flexible and evolvable *tussle space* [11] where multiple RSPs can safely compete without endangering the stability of the core infrastructure.

But route computation is only half the story. To implement these routes, they must be installed in routers willing to accept them. We imagine that there is a coherent forwarding infrastructure (FI) that accepts computed routes. We envision an FI that is an overlay on the underlying Internet. We discuss later why ISPs might be willing to create such an FI, but we acknowledge that this will be a major hurdle to overcome.

The RAS approach has other challenges and disadvantages. Using RAS incurs extra overhead and perhaps expense. Moreover, it isn't clear that these explicitly constructed routes can be very robust during network stress. However, flows can always resort to default routing if the customized route isn't adequate; thus, there is a lower bound on how badly flows can suffer under RAS. We explore the challenges with the RAS approach in Section 2.2. In Section 3, we describe a prototype implementation of a FI along with a simple distributed RSP that addresses some of the challenges.

The goal of the RAS approach is to scalably support customized routing. While we believe this is an important goal, we also believe it is relevant only to a small fraction of Internet traffic. We assume that most flows will be adequately served by the Internet's default routing. However, we imagine that there are now, and will continue to be, flows that for one reason or another (e.g., higher reliability, better per-packet quality of service, policy constraints, etc.) desire customized routes. The key assumption we make is that this population is large enough so that unscalable methods like source routing aren't viable; we don't assume, nor does the RAS approach require, that this population comprises the bulk of Internet traffic. Thus, the role of RSPs is not to displace basic Internet routing, but to enhance it. It is crucial that default routes will still be implemented by the infrastructure; only flows requiring specialized routes will use an RSP.

2 Overview and Challenges

In this section we first present the basic RAS architecture and then discuss the challenges in realizing this architecture.

2.1 Overview

The basic architecture, as shown in Figure 1, consists of three entities: the forwarding infrastructure (FI), a collection of routing service providers (RSPs), and the population of end-hosts. The FI can take many forms: it could be a set of specialized routers run by ISPs, or a set of commodity overlay nodes operated either by the ISPs, or some other entity (*e.g.*, PlanetLab). In order to be effective, FI nodes must be well-provisioned, well-connected, and well-managed, but otherwise the RAS approach does not dictate the nature of the FI nodes (except as specified below).

The FI nodes are interconnected to form a routing overlay network. The *virtual links* of this overlay are logical unidirectional paths along which packets are forwarded via the underlying default routing protocol. Furthermore, these nodes export a forwarding primitive that allows computed paths to be used. This primitive could be as simple as accepting source routes [25], in which no state is kept in the FI and all routes are carried in the packets. Alternatively, the primitive could be an explicit route establishment interface (*e.g.*, MPLS tunnels [24]) and, in this case, the route is stored inside the FI.[3]

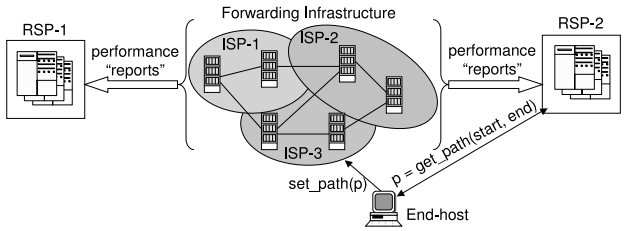

Figure 1: The main components of the RAS architecture: the forwarding infrastructure (FI), one or more routing service providers (RSPs), and end-hosts.

Route computation is provided by third-party entities (RSPs), using network information gathered from the FI. This network information may be provided by the FI itself based on internal measurements, or the product of external measurements conducted by the RSPs (either individually or jointly).

End-hosts that desire special routing query the appropriate RSPs to obtain paths with the desired characteristics.[4] The RSP returns a path (a sequence of FI nodes) and the end-host then uses this path for forwarding packets along the FI. Flows that do not require special routing simply use the default Internet routing. We thus think that the bandwidth that the ISPs allocate to the virtual links of the forwarding infrastructure will be a fraction of the total bandwidth of the physical links.

It is an open question how the inter-domain policies of the ISPs would interact and influence the routing policies of the FI. In the simplest scenario, a virtual link once established in the FI can be used by any RSP for computing routes over the FI. One could also envision a more complex system (such as Platypus [23]) where only routes that conform to the ISP policies are allowed to exist in the FI. In the rest of the paper, however, we assume that each RSP knows beforehand the virtual links it can use for selecting the routes.

2.2 Challenges

While the basic RAS architecture seems simple, it is not without significant challenges. These challenges can be separated into two categories: those that arise from the physical separation of forwarding from routing, and those that arise from the logical separation. We now discuss each in turn. While we have partial solutions (and directions) to some of these challenges, addressing the rest forms our research agenda.

[3] The reason why RAS employs an overlay, rather than just the native routers, is the hurdle of implementing this forwarding primitive. Since controlling routing at a small subset of nodes appears to give very good routing performance [26], given default routing in between those nodes, we choose to implement the forwarding primitive at a reduced set of nodes.

[4] Defining a flexible querying API (which need not be uniform across all RSPs) will be an important enabler of RAS, but we don't address that issue in this paper.

2.2.1 Physical Separation

The physical separation between routing and forwarding was discussed in [14], which addressed the problem in the context of computing default routes for a medium scale network (ISP or large enterprise). Here we address a much larger network and, more importantly, are concerned with customized routes rather than default routes. Thus, we focus on a somewhat (but not completely) different set of challenges.

Scalability. The size of the Internet poses a daunting challenge to any RSP (though the fact that RAS uses an overlay, and not the full set of native routers, reduces the size of the problem). The actual computation of the route is relatively tractable, but obtaining the information needed as input to the computation is far more problematic. In order to make informed route choices, an RSP needs an accurate network map with reasonably up-to-date and accurate estimates of the relevant network characteristics (such as connectivity information, delay, loss rate, bandwidth).

With only physical separation between routing and forwarding, the network infrastructure could take the necessary measurements. However, at very large network scales, the rate of incoming network information would overwhelm a single RSP node. To fully scale, there must be a way to distribute the route computation, so that each node need only maintain a map of a workable portion of the network. We discuss such an approach in the following section.

The other aspect of scaling is the number of flows requesting routes. Separate route requests are parallelizable and hence can be handled by replicating the RSP nodes (and by having multiple RSPs).

Of course, there are many tricks one might use to accommodate scale, such as caching recent route computations, adaptive measurement dissemination (update measurement information only when it has changed significantly), and incremental path computations, that we do not discuss here.

Route Insertion. The physical separation necessitates a method for communicating a path to the network. Source routing is one approach, and it has the benefit of not requiring the network to store state. But source routing incurs significant per-packet overhead (both in bandwidth and in processing), and isn't suitable for multicast.

The alternative is to install state in the network. The host could either insert local forwarding state (like MPLS) at each FI node along the path, or use a higher level state-establishment protocol (like RSVP) to establish the state. Note that while the host knows the entire path, the state installed in each FI node only describes the next-hop of the flow's packets. We assume that the combination of source-destination, and a flow identifier (if available) will be used to identify the packets intended for this custom path. Installing state in the network removes the per-packet overhead, and handles single source multicast.

Robustness and Responsiveness. In cases where there is end-to-end feedback about the state of the flow, the source is probably in a better position than the network (or the RSP) to determine if a path has failed (the source might not know why or where the path failed, but it can detect degraded performance). Upon detection, the source can resort to default routing, send another request to the RSP, or even proactively setup backup paths in the FI.

However, it isn't clear what should happen if the flow doesn't have end-to-end feedback (such as in some large multicast flows). If the RSP gets information indicating that a link has seriously degraded, it might alert the sources using paths traversing the degraded link, but an RSP doesn't know how long flows will stay active so it isn't clear who the RSP should contact. It appears that flows using customized routes should either do their own end-to-end performance checking, or be prepared for undetected outages.

The rate at which measurements are updated is the limiting factor in providing routes with currently adequate performance. If there are portions of the network that are extremely flaky, where the performance regularly changes on time scales shorter than the measurement interval, RSPs could either avoid such areas or indicate, when returning a path, that their confidence in the path is low.

2.2.2 Logical Separation

The logical separation of routing and forwarding removes the natural trust between the FI and the entity that computes the route. It also removes the natural coordination between the various entities computing the routes, because now there are several competing route providers.

DoS Attacks. All mechanisms for route computation and route insertion are driven by RSPs and end-hosts respectively. Since both are untrusted (or partially trusted) entities, the FI must protect itself from traffic attacks arising from malicious route insertions. Simple cryptographic puzzles can protect the FI nodes from state-based attacks [22, 13].

A malicious route insertion can create a loop (MPLS-like state insertion algorithms are vulnerable to this, but source routing is not), thereby consuming network resources until the TTL has expired. Thus, the FI should be able to detect loops, or only deal with RSPs that it trusts.[5]

While state insertion raises the possibility of traffic *hijacking* (whereby one host injects a route for another host's traffic), a simple challenge-response can make sure that the request came from the appropriate source (or at least another host on that subnet; the challenge can't prevent a same-subnet spoofing attack).

Traffic Engineering. The underlying ISPs carefully tune their routing protocols to produce the desired traffic patterns. However, the RSPs choose overlay routes to best serve their customers, without regard to the wishes of the ISP. Moreover, there can be many RSPs computing their routes in parallel without any coordination. This could lead to wild traffic oscillations as RSPs individually react (and collectively overreact) to the latest traffic measurements. Since the RSPs are only semi-trusted, the FI should also protect itself from RSPs

[5] When the host inserts the path, it can do so with a path description digitally signed by the RSP, so that the FI nodes can be assured that the path came from a trusted source.

that maliciously select routes to cause congestion on selected links. The impact that the decisions of the RSPs has on the rest of the Internet can be reduced by the ISPs by restricting the bandwidth devoted to the FI. Studying the interactions between multiple, perhaps non-cooperating, RSPs forms an important part of our future research.

Trust and Cooperation. There is (at least) one fundamental question that we have so far ducked: why would ISPs ever participate in the forwarding infrastructure and give up some control over routing? The fact is that currently overlays are a thorn in the side of ISPs, both as a business competitor — they reap profits that the ISPs would like for themselves — and as a traffic source. Overlays make traffic engineering hard because they are large sources of traffic that can unpredictably change *en masse*. ISPs currently have very little leverage over overlays (such as Akamai), and the problems are likely to worsen over time. This conflict of interests has been noted by many, including [23].

RAS is a compromise between the interests of overlay providers and ISPs. The ISPs, through their management of the FI, can presumably recoup some of the profits that go to the overlay operators. In return, the ISPs would make life easier for overlay operators (or at least those that choose to follow this path) by turning them into RSPs; they would no longer have to manage the network, they would only have to provide correct routes.[6] Similarly, the ISPs might provide good traffic measurements to the RSPs as a way of helping them provide good service, since the ISPs recoup some of the resulting revenue.

3 Prototype Implementation

The previous discussion of the architecture was quite general. In order to understand the RAS approach better, we implemented a particular instantiation. This prototype is presented here only as a proof-of-concept; we expect further refinement as we gain experience. While this instantiation partially answers some questions such as security and scalability, many topics such as trust, interaction between multiple RSPs and other aspects of scalability are left for future investigation.

3.1 Forwarding Infrastructure: Path Setup

Based on our earlier discussion (see Section 2.2), implementing an MPLS-like forwarding primitive is straightforward. In a simple realization, end-hosts would construct the state on the FI nodes hop by hop. That is, each infrastructure node would know the next hop for a particular *(source, destination, flow-label)* combination. The flow-label here is a flat identifier that represents a flow corresponding to the *(source, destination)* pair.[7] For multicast communication, receivers insert the paths, which are subject to a challenge-response protocol to protect themselves from unwanted subscription attacks.

[6] Some overlay networks such as Akamai deliver services beside pure packet delivery. The RAS approach could support such services by directing traffic to the appropriate service-level nodes. We don't dwell on this here, but the RAS approach is, in this respect, quite similar to that of $i3$ [27] and related proposals [4, 28].

[7] Source and destination information is not needed if the flow-label is chosen uniquely.

This simple primitive is limited to unicast and receiver-driven single-source multicast and, as such, is quite conventional. In order to gain experience with a more flexible forwarding primitive (and to understand how much we can stretch the generality of RAS), we also leveraged recent research in forwarding overlays and experimented with an infrastructure that allows end-hosts to insert *delegates* in the path of packets. As observed in the proposal for a layered naming architecture [4, 28] and in the $i3$ design [27], delegation allows a more general set of communication primitives, including anycast, architecturally coherent middleboxes, and service composition.

In particular, packets are not sent to the IP addresses of the logical destination, but instead to that of an intermediary who has been delegated to process the packet on behalf of the destination; the intermediary may choose, as part of that processing, to forward the packet to the end-point, or it might drop it (in the case of a firewall), or it might send the packet to other intermediaries (in the case of service composition). In this architecture, both senders and receivers have control over the path packets take along the infrastructure. We also used the cryptographic operations proposed in [18] to protect end-hosts (and the infrastructure) from DoS attacks such as formation of loops.

The question of what the actual FI primitive should be is still open; we however believe that experience gained with deployed implementations will guide us in answering it.

3.2 Monitoring the Network

Infrastructure measurements. The FI nodes monitor the performance of the virtual links, export this information to RSPs. Currently, we monitor the delay, loss-rate and the available bandwidth of the virtual links. In an ISP deployment, the FI nodes might export coarse forwarding rules that might indirectly reflect the preference or the policy constraints of the ISPs. We also plan to investigate adaptive techniques for updating the network information in the future.

Indirect measurements. While we expect that direct measurements performed by the infrastructure would be preferred due to efficiency considerations, we also have explored the alternative strategy of RSPs performing indirect measurements over the FI (without additional help from the FI). Owing to lack of space, we now give a brief outline of one simple example: an RSP node N measures the RTT between two FI nodes F_1 and F_2. To do so, R sends one packet along the path $R_1 = (N, F_1, N)$, and another along the path $R_2 = (N, F_1, F_2, F_1, N)$. The RTT between F_1 and F_2 can be computed as the difference between the RTTs of the packets sent along paths R_2 and R_1. In a technical report [19], we have presented (and evaluated) techniques for measuring some commonly used metrics (delay, loss rate and bandwidth) in more detail.

3.3 Routing Service

In response to requests from end-hosts, the RSP returns application-specific paths between any two nodes in the in-

frastructure.[8] For this, the RSP maintains a performance map of the infrastructure (using either its own external measurements or internal measurements from the FI).

In the case of an infrastructure with no more than a few hundred nodes, the RSP can maintain the map of the entire infrastructure at a single server. To handle a large number of requests, the RSP can simply replicate this server. However, as the FI becomes larger, a single server can no longer maintain the map of the entire infrastructure.

In our initial realization, we used a simple graph partitioning technique to divide the network graph among multiple sets of replicated RSP nodes spread through the network. Each RSP node maintains information about the virtual links between all the nodes that it is responsible for, and a subset of the virtual links that have one node that it is responsible for. More precisely, given any two RSP nodes N and N', the RSP ensures the existence of at least k pairs of nodes $n \in N$ and $n' \in N'$ such that there is a virtual link from n to n'. Note that in this design, it is possible to compute a path between any two nodes n_1 and n_2 by contacting no more than two RSP nodes, one that is responsible for the domain of n_1, and another responsible for the domain of n_2. A redundancy factor of k is chosen to ensure that there are multiple paths between two domains.

The quality of the paths computed by the RSP is determined by the partition of the graph. We used METIS [16], a serial graph partitioning tool, to divide the graph. Initial experimental results (reported in [19]) show that while our partition method works reasonably well in the case of a small infrastructure, there is room for improvement with large network sizes.

3.4 End-hosts

Our current RSP query interface is simple: end-hosts specify the start and end points of the route and the metric to optimize. We assume that the end-hosts know of appropriate start and end FI nodes; discovering the appropriate FI nodes could be done, but we don't discuss it here. Designing a flexible API for end-hosts to query the RSPs is an important challenge. In the future, we plan to leverage the power of a high level declarative query language as proposed in [20].

3.5 Implementation Status

We have implemented the FI and a distributed RSP that runs on multiple nodes, each node monitoring a portion of the infrastructure. In our implementation, we used flow labels that were 128-bits long. We have deployed the initial version of our implementation over the PlanetLab testbed. Our microbenchmarks on a Pentium 2.4GHz machine indicate that the FI can support a high packet forwarding rate ($47,600$ 1KB packets/second) and has a small processing overhead for insertion of one hop of a route (of 2.5 μs). Our initial experiments with a distributed RSP instrumented for the delay metric running over a small FI of 100 PlanetLab nodes was able to perform well (found better paths than the Internet in 40% of the cases) by monitoring a small fraction of the network using the simple graph partitioning technique. See [19] for more details.

[8] A more complex RSP can also determine the entire end-to-end path, including end-hosts.

4 Related Work

The paper touches upon two particular issues that have separately received substantial attention in the literature. We discuss each of these in turn.

Separating control and data planes. The separation of data and control planes is a recurring theme in the literature, with the aforementioned RCP [14] being the latest example. RCP proposes having a separate routing control platform that performs the task of route computation on behalf of IP routers. Earlier examples include the Generalized Switch Management Protocol (GSMP) [12], in which external switch controllers establish and maintain paths in an ATM, frame relay and/or MPLS switches. The IETF ForCES [17] working group proposes separating forwarding and control elements for IP forwarding devices in a small area system. The Bandwidth Broker [7] is a centralized entity that computes routes based on QoS requirements for an entire domain. Akamai [8] implements route computation as a separate component, though the details of the patent are not public. Our proposal merely follows the lead established by these earlier works, but extends it to the case where the routing and forwarding are handled by *different* logical entities.

Routing control. There is a long list of proposals that give end-hosts more control over routing in the network. The Nimrod [9] architecture proposed computation of routes by the clients of the network, and gave mechanisms for distribution of network maps. Clark *et al.* [11] have argued for giving end-hosts more control over routing as a way of promoting competition among ISPs. In this context, Yang [30] has proposed a solution that allows both senders and receivers to choose routes at the AS level. TRIAD [10] has proposed a name-based routing scheme where end-hosts specify the path across multiple address space domains. Some of these proposals only discuss mechanisms for host-control over routing and do not address the problem of how hosts gather the information to intelligently choose those routes. Among the proposals that address the problem of choice, none focuses on achieving scalability by sharing network information except Nimrod, which only deals with network topology, and not finer-grained and more dynamic network measurements.

Several companies such as Sockeye [2] and RouteScience [1] develop products for multi-homed organizations to pick their last hop ISP. However, these operate only at the last-hop at a granularity of organizations, not applications.

Platypus [23] is a recent proposal that allows the construction of routes that conforms to a certain set of policies of the infrastructure. We can leverage ideas in Platypus to ensure that RSPs compute paths conforming to the ISP requirements. However, we defer a detailed study of different policy enforcement mechanisms (and their tradeoffs) to the future.

5 Conclusion

In this paper, we proposed a new routing paradigm: *Routing as Service*, wherein specialized route computation is performed by third-party entities outside the forwarding infrastructure. This approach allows great flexibility in route customization while remaining reasonably scalable. More generally, it allows customized routing to evolve as needs change without any change to the infrastructure. The RAS poses several challenges that must be addressed before it can be considered viable, and we have discussed various ways in which those challenges can be met.

Moreover, we have implemented and deployed a proof-of-concept prototype of the RAS architecture. We plan to make this prototype a long-running service on PlanetLab and add an RSP that can provide some attractive routing functionality (such as that provided by RON). Additionally, we hope to extend ideas presented in [15] to allow unmodified legacy applications to use the routing service.

While routing has been our subject here, a more general issue is at stake: how does one build an evolvable architecture? PlanetLab is one possible model, in which multiple services are deployed over a single multiplexed infrastructure. Such an approach increases evolvability by lowering the deployment barrier to any particular service. This is a familiar model: Web hosting services are merely more centralized versions of physical multiplexing.

In contrast, the RAS approach multiplexes at a much higher-level control plane. Rather than giving access to a virtual machine, the RAS approach calls for the forwarding infrastructure to support a higher-level control interface that other entities can use. As such, it bears similarity to the Active Network [29] paradigm, but with a more limited interface. We have less experience with such extensible control of an infrastructure. It poses not only technical challenges, that we have tried to explore here, but also economic ones. It may seem economically unlikely that ISPs will relinquish such control (over a portion of their bandwidth) to third-parties. Yet, this may be the only workable control-sharing compromise in the ongoing tug-of-war between overlays and ISPs.

References

[1] RouteScience. `http://www.routescience.com`.

[2] Sockeye Networks. `http://www.sockeye.com`.

[3] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient Overlay Networks. In *Proc. SOSP*, 2001.

[4] H. Balakrishnan, K. Lakshminarayanan, S. Ratnasamy, S. Shenker, I. Stoica, and M. Walfish. A Layered Naming Architecture for the Internet. In *Proc. ACM SIGCOMM*, 2004.

[5] A. Basu, A. Lin, and S. Ramanathan. Routing Using Potentials: A Dynamic Traffic-Aware Routing Algorithm. In *Proc. ACM SIGCOMM*, 2003.

[6] A. Basu, C.-H. L. Ong, A. Rasala, F. B. Shepherd, and G. Wilfong. Route Oscillations in I-BGP with Route Reflection. In *Proc. ACM SIGCOMM*, 2002.

[7] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Service. RFC 2475, 1998.

[8] C. F. Borenstein, G. L. Miller, S. B. Rao, and T. K. Canfield. Optimal Route Selection in a Content Delivery Network. *US Patent*, Sept. 2002. #WO02071242.

[9] I. Castineyra, N. Chiappa, and M. Steenstrup. The Nimrod Routing Architecture. RFC 1992, 1996.

[10] D. R. Cheriton and M. Gritter. TRIAD: A New Next Generation Internet Architecture, Mar. 2000. `http://www-dsg.stanford.edu/triad/triad.ps.gz`.

[11] D. D. Clark, J. Wroclawski, K. R. Sollins, and R. Braden. Tussle in Cyberspace: Defining Tomorrow's Internet. In *Proc. of ACM SIGCOMM*, 2002.

[12] A. Doria, F. Hellstrand, K. Sundell, and T. Worster. General Switch Management Protocol (GSMP) V3. RFC 3292, 2002.

[13] C. Dwork and M. Naor. Pricing via Processing or Combatting Junk Mail. In E. Brickell, editor, *Advances in Cryptology — CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*, pages 139–147. International Association for Cryptologic Research, Springer-Verlag, 1993.

[14] N. Feamster, H. Balakrishnan, J. Rexford, A. Shaikh, and K. van der Merwe. The Case for Separating Routing from Routers. In *Proc. FDNA*, 2004.

[15] J. Kannan, A. Kubota, K. Lakshminarayanan, I. Stoica, and K. Wehrle. Supporting Legacy Applications over i3. Technical report, UCB, 2004.

[16] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. In *SIAM Journal on Scientific Comp.*, 1995.

[17] H. Khosravi and T. Anderson. Requirements for Separation of IP Control and Forwarding. RFC 3654, Nov. 2003.

[18] K. Lakshminarayanan, D. Adkins, A. Perrig, and I. Stoica. Towards a Secure Indirection Infrastructure. In *Proc. ACM PODC (Brief Announcement)*, 2004.

[19] K. Lakshminarayanan, I. Stoica, and S. Shenker. Routing as a Service. Technical Report UCB-CS-04-1327, UC Berkeley, 2004.

[20] B. T. Loo, R. Huebsch, J. M. Hellerstein, T. Roscoe, and I. Stoica. Analyzing P2P Overlays with Recursive Queries. Technical Report IRB-TR-03-045, Intel Research, Nov. 2003.

[21] Z. M. Mao, R. Govindan, G. Varghese, and R. Katz. Route Flap Damping Exacerbates Internet Routing Convergence. In *Proc. ACM SIGCOMM*, 2002.

[22] R. Merkle. Secure Communication Over Insecure Channels. *Commun. ACM*, 21(4):294–299, Apr. 1978.

[23] B. Raghavan and A. C. Snoeren. A System for Authenticated Policy-Compliant Routing. In *Proc. SIGCOMM*, 2004.

[24] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol Label Switching Architecture. RFC 3031, Jan. 2001.

[25] J. Saltzer. Source Routing for Campus-wide Internet Transport. In *IFIP Working Group 6.4 Workshop on Local Area Networks*, 1980.

[26] 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. Technical Report TR-98-10-05, 1998.

[27] I. Stoica, D. Adkins, S. Zhuang, S. Shenker, and S. Surana. Internet Indirection Infrastructure. In *SIGCOMM*, 2002.

[28] M. Walfish, J. Stribling, M. Krohn, H. Balakrishnan, R. Morris, and S. Shenker. Middleboxes No Longer Considered Harmful. To appear in OSDI, 2004.

[29] D. Wetherall. Active Network Vision and Reality: Lessons from a Capsule-based System. In *Proc. of SOSP*, 1999.

[30] X. Yang. NIRA: A New Internet Routing Architecture. In *Proc FDNA*, 2003.

[31] D. Zhu, M. Gritter, and D. Cheriton. Feedback-based Routing. In *Proc Hotnets-I*, 2002.