Decoupling Policy from Mechanism in Internet Routing

Alex C. Snoeren and Barath Raghavan
University of California, San Diego
{snoeren,barath}@cs.ucsd.edu

Abstract

Routing is a black art in today's Internet. End users and ISPs alike have little control over how their packets are handled outside of their networks, stemming in part from limitations of the current wide-area routing protocol, BGP. We believe that many of these constraints are due to policy-based restrictions on route exportation. Separating forwarding policy from route discovery would allow users to select among the possibly many inter-AS paths available to them and enable ISPs to more effectively manage the end-to-end behavior of their customers' traffic.

As a concrete mechanism for enforcing forwarding policy, we propose the concept of a network capability that binds together a path request, an accountable resource principal, and an authorizing agent. Network capabilities are central to Platypus, a loose source routing protocol we are designing, which composes network capabilities authorized by multiple ISPs to construct alternative inter-AS routes that can be independently validated and accounted for on the fly.

1 Introduction

Traffic engineers, academic researchers, and professional Internet pundits alike are in agreement that today's Internet routing is not what it should be [10, 12, 18]. The literature is rife with suggestions of how to extend, replace, or circumvent [1, 3, 4, 18, 25] the existing wide-area routing protocol, BGP. Continued increase in Internet complexity, both in terms of the number of autonomous systems and the intricacy of their relationships does not augur well for the disappearance of the problems that prior approaches have attempted to address. Unfortunately, the tremendous amount of deployed infrastructure does not make it likely we will be able to start over from scratch anytime soon, either.

The fundamental function of wide-area routing protocols is data aggregation. In order to manage the scale and complexity of the problem, each autonomous system (AS) summarizes its own network in terms of the other ASes it can reach. Unfortunately, ASes selectively export this information, making decisions about which routes to advertise based upon local policy, thereby controlling whose traffic they will be asked to forward [15]. We believe this is an unnecessary and counter-productive restriction: no single decision will be best for all other ASes. In particular, certain local policies may prohibit desirable end-to-end policies that require the composition of multiple local policies. Because ISPs lack an effective mechanism for policy enforcement, they currently express their policies through selective route export; we argue that policy should be enforced entirely on the forwarding plane, enabling policy-neutral route export and adaptive, fine-grained, policy-aware route selection.

Fine-grained route selection has long been a holy grail of networking researchers. It is well known that many of the deficiencies of today's Internet could be solved with various forms of loose source routing (LSR). In fact, LSR is quite commonly used for intra-AS traffic engineering tasks under the guise of MPLS. Researchers have designed many global source-routing techniques over the past twenty years, some requiring router support [16, 21, 23, 24, 25, 26], and others entirely end-host based [22], each complete with their own mechanisms for encoding and forwarding. To the best of our knowledge, however, no proposals for inter-AS source routing have yet enjoyed wide-scale deployment save one: the original source routing IP options, which are almost universally disabled due to security concerns[1]. We believe that the unavailability or lack of deployment of these approaches stems from the omission of a key feature: a mechanism to enable accountability and composable authorization.

We believe that the fundamental problem with existing source-routing mechanisms is the inability of an AS to determine whether or not a packet has been authorized to transit its network. Currently, the implicit assumption is that a packet arrives at an AS only because the AS advertised a route to the packet's destination. Source-routed packets can be made to transit any AS, violating this precondition. Hence, ISPs install filters to prevent unauthorized traffic from entering their network [7]. Unfortunately, these filters must act strictly upon the information contained within the packet—source address,destination address, and perhaps protocol or type of service—and the current network location. None of these attributes are sufficient to fully convey authorization, forcing end hosts or ASes wishing to implement such functionality in today's Internet to use clumsy overlay tunneling techniques.

In this position paper, we argue for the development of *network capabilities*, which are functionally quite similar to their operating system brethren. Each capability contains a path request, a resource principal, and an authorization. By

[1] Although it is quite common—indeed, often required by AUPs—for LSR to be enabled on gateway routers for network diagnostics (*e.g.*, `traceroute -g`), its use for traffic forwarding is discouraged.

presenting a capability along with requests for network services, both end users and ISPs attest to their willingness to be held accountable for the service. (Whether or not payment is required of course depends on the business relationship between the entity requesting the service and the provider.) One of the key features of network capabilities is that they are transferable—an entity can pass capabilities on to others—and composable—one request may be accompanied by a set of capabilities.

By requiring a capability to authorize each hop in a source route, we show how LSR can be implemented in a straightforward, efficient, and accountable fashion in today's Internet. It is important to note that we are not arguing for the removal of traditional wide-area routing or its replacement with an entirely source-routed architecture; quite the contrary. We believe each AS should export all available routes—independent of local policy—including an intelligent (policy-compliant) default, but both end users and ISPs should be empowered to select among them, or, if necessary, construct their own.

2 The state of BGP

To motivate the need for separating policy enforcement from route advertisement and selection, we enumerate several deficiencies of BGP and describe how the coupling of policy and mechanism either creates or exacerbates the problem.

Poor reliability/stability. BGP is a notoriously difficult protocol to configure properly [13]. We believe a significant portion of this complexity stems from the need to simultaneously optimize for route efficiency and policy compliance. Part of the problem is that it is very hard to know beforehand what the right configuration might be [6] because policy goals cannot be directly mapped to configuration settings; instead, operators must adjust a number of overloaded parameter values (MED, LOCAL_PREF, and COMMUNITY being three of the most prominent) in hopes of coercing both their own internal network and adjacent ASes to select the desired routes. In fact, it's possible for local policy settings to guarantee that the routing configuration will diverge [9]. This issue could largely be avoided if ASes could simply export all possible routes, and determine whether or not to forward a particular packet (because it did or did not meet the AS's local policy constraints) when it arrived at a border router. We believe that a mechanism for explicit routing can free ASes to fully export routes as routing policy can be decoupled from route computation and distribution.

Poor reachability/performance. Even if network operators manage to get BGP configured properly, BGP is slow to adapt to changes in network topology [12]. In addition, BGP does not have all of the existing routes available to it; studies have shown that a large number of BGP outages can be avoided by overlay networks [3], implying serviceable routes do exist. While it is reasonable to assume many of the paths exploited by these overlays might not be policy compliant (because they transit stub networks), how many would be is an open question. We believe our approach could have two benefits: first, BGP may be able to converge to a working path more rapidly when provided with the full set of available routes; second, full route export could provide existing overlay techniques with more alternatives. Overlay networks are constrained to select among the set of paths they themselves can construct. We posit even greater redundancy exists in the network, but is being concealed by policy-based route export filters.

Poor accountability. There are few ways to determine where an Internet packet came from [19, 20], none of which are widely deployed. Further, even if a packet's source is identified, there is no easy or correct way to identity the accountable party—either for the path selection or the forwarding decision [14, 17]. Even if a packet is deemed admissible, the options for assigning appropriate resource principals are few. The obvious candidates are either the adjacent AS, or, should finer-grained accounting be desired, the source or destination IP.

While several router vendors now support class-based accounting, the lack of a mechanism for defining consistent, globally-meaningful classes makes this functionality difficult to utilize across ASes. Network capabilities provide an explicit, locally meaningful accounting principal at each point in the network. Accounting and/or rate control can be done on a per-flow or even a per-packet basis; recent results indicate such fine-grained accounting is entirely plausible even at high speed [5]. Our primary complaint is that the current mechanisms for determining the appropriate resource principal and authorizing agent are insufficient.

Poor flexibility. All of BGP's ills could be forgiven (if not forgotten) if motivated networks and end hosts could efficiently implement their own routing mechanisms. Unfortunately—but understandably—ISPs are unwilling to allow external entities to influence their operation in the absence of effective accounting and enforcement mechanisms. For a multi-homed AS, the inability to affect in-bound routing is particularly limiting: an AS can determine its own outgoing routes, and can limit the possible choices for the incoming routes, but it cannot, in general, control which incoming link traffic from a particular AS will arrive on.

Such restrictions are painfully ironic, as they prevent ISPs from realizing many of their own goals. For example, it is increasingly common for ASes to have complex business relationships [15], where an adjacent AS is not entirely customer or peer, but sometimes one or the other depending on the particular peering point. It is difficult to express such policies with BGP, let alone finer-grained ones like transit relationships for certain types of traffic, and peering relationships for others. Further, the mechanisms used to implement these policies typically operate on human time-scales, so there's little opportunity for dynamic reconfiguration in the face of failure or other network events.

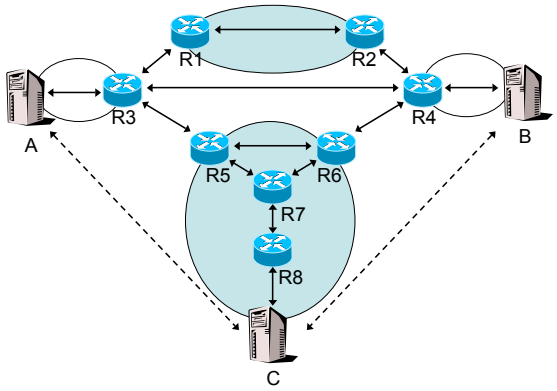

Figure 1: A simple network topology. Hosts A, B, and C all have different ISPs.

3 Sample applications

We are developing *Platypus*, an architecture for loose source routing. Platypus allows end hosts to construct arbitrary paths through the network using the network infrastructure itself and allows ISPs to implement sophisticated routing policies. While Platypus is general in principle, we envision it will be used primarily for AS-level source routing, as we expect few ASes would allow intra-AS source routing. Critical to our design is the integration of authorization and forwarding. A capability explicitly confers the necessary rights for a packet's source routing to be honored. In addition, Platypus makes it easy for ASes to track the usage of capabilities. We provide a few examples of how capabilities could be used to address common or interesting routing problems below.

3.1 Overlay construction

Consider the (partial) network topology shown in Figure 1. Nodes A, B, and C are all willing to transit traffic for each other in an overlay fashion. Let us assume for the moment that A and B wish to exchange traffic, but the default route $A \leftrightarrow R_3 \leftrightarrow R_4 \leftrightarrow B$ is unsatisfactory—perhaps because the link $R_3 \leftrightarrow R_4$ is congested or down. Using existing overlay technologies, A and B can use C as a transit point by tunneling their traffic directly to C. While effective at avoiding the misbehaving link, this route is clearly sub-optimal for all involved. In particular:

1. C is forced to forward each packet itself, consuming both last-hop bandwidth (in both directions) and processor resources. It would prefer that R_8 forward the traffic instead.

2. Any path $A \leftrightarrow R_3 \leftrightarrow R_5 \leftrightarrow R_7 \ldots R_7 \leftrightarrow R_6 \leftrightarrow R_4 \leftrightarrow B$ is also sub-optimal from the point of view of both A and B—they would likely prefer shorter, equivalent routes like $A \leftrightarrow R_3 \leftrightarrow R_5 \leftrightarrow R_6 \leftrightarrow R_4 \leftrightarrow B$.

3. The ISP owning R_5, R_6, R_7 and R_8 (and the links between them) would likely prefer not to transit the traffic even to R_7 unnecessarily.

4. If avoiding $R_3 \leftrightarrow R_4$ is the objective, an alternate route exists: $A \leftrightarrow R_3 \leftrightarrow R_1 \leftrightarrow R_2 \leftrightarrow R_4 \leftrightarrow B$. In the case where C's ISP also owns R_1 and R_2, C should be able to authorize use of the link $R_1 \leftrightarrow R_2$.

The first issue can be addressed if node C were able to request its upstream router to redirect packets from A to B, such as with the recently proposed *reflection* primitive [11]; C could ask R_8 to *reflect* packets from A to B. Unfortunately, C's ISP's now cannot implicitly limit C's bandwidth use by restricting C's last hop. C's ISP is now liable for transitting a potentially large amount of traffic and needs some way to account for this usage. The ISP would likely want to rate limit the flow at the router using a token-bucket type scheme.

The second issue can be only partially addressed using the reflection primitive recursively. If R_8 propagates the reflection up to R_7, the perceived path from A to B no longer traverses R_8 or C, but this form of path relaxation cannot avoid R_7, since R_7 is likely unaware that a better path exists between R_5 and R_6. In Platypus, however, C could provide A with a capability allowing it to source route through C's ISP—naming C as the resource principal—and C's ISP could intelligently route A's packets, addressing issues three and four.[2]

3.2 Intelligent multihoming

We describe two instances in which an AS could use network capabilities to its benefit in the context of multihoming.

Affecting in-bound traffic. Multihomed stub ASes often select multiple upstream providers and send different traffic through each depending on network conditions and destination. Unfortunately, a stub AS remains at the mercy of its upstream providers to control how incoming traffic arrives. Using AS-level source routing, however, an AS could advertise multiple routes to itself, along with associated capabilities. Just as with toll-free phone numbers, there are many instances in which a stub AS would be willing to be the resource principal responsible for incoming traffic if it can affect how that incoming traffic arrives.

Virtual multihoming. A stub AS with a single upstream connection is limited to the default routes of its provider. Without multihoming, the AS is incapable of selecting backbone providers to carry its traffic—it must use whichever backbone provided to it by its upstream AS. With Platypus, the stub AS could place capabilities on its out-bound traffic indicating which of its regional provider's upstream back-

[2] ISPs like C with multiple, disjoint transit paths could even publish multiple next hops, enabling C to provide A with capabilities specifying R_1 or R_5 in particular.

bones to use for particular traffic, in effect making the AS virtually multihomed.

3.3 Routing Accountability

Today's routing infrastructure depends a large part on the good behavior of ASes and the correct configuration of BGP. BGP makes it easy for malicious speakers to falsely announce routes for prefixes they do not own. Future traffic to and from a hijacked prefix cannot be easily differentiated from valid traffic by third parties. However, networks in possession of capabilities to affect routing decisions can benefit from not only the increased flexibility of such capabilities, but also from the verifiability of their packets and routes. Furthermore, this benefits transit providers, as they can now verify packets easily, allowing for convenient accounting and billing of distinct resource principals.

4 Network capabilities

In this section we discuss the issues that arise with one possible design of network capabilities and their use in Platypus. While network capabilities themselves are intended to be far more general, we have yet to examine how to integrate them with other systems.

4.1 Capability format and authentication

A network capability specifies three things: the service being requested, which, in Platypus, is expressed as a next hop through which to source route; the resource principal to be "charged" for the use of the service; and the authorizing agent, who is the entity approving the capability (and likely the one issuing it). Only the next-hop field must be globally significant, as capabilities need only be validated by the entity providing the service—in Platypus, validation occurs upon arrival at the next hop or the edge its AS. Hence, the resource principal and authorizing agent may have only local significance.

We prevent forgery of capabilities by using the "double MAC" trick [2]. We define a secret, $s = \text{MAC}_k(c)$, generated by computing a keyed message authentication code (MAC) over a capability, c. The key k is known only to the authorizing agent and routers that will validate capabilities protected with the key. To be used, a capability must be bound to an individual packet with a binding, $b = \text{MAC}_s(p)$, where p is the invariant [20] contents of the packet *except* the packet length and any other capabilities that may be attached. Both b and c are included in the packet using a simple encapsulation protocol.

Security is provided by the fact that not all parties have the information needed to create new capabilities, or create new bindings. Table 1 shows the information known to various parties. To generate a secret, a party must have the key, k. To generate a binding, a party must have the secret, s. The secret, however, does not reveal the key; hence, the secret can be transferred to third parties who can then use it to generate bindings for their own packets. Others, including those sniffing on the network, can see capabilities and their bindings, but lack the secret s required to generate valid bindings for other packets. To verify a packet's binding, a Platypus router recomputes the correct binding $b' = \text{MAC}_{\text{MAC}_k(c)}(p)$ and compares it to the binding b attached to the packet in question; if $b \neq b'$, the binding is invalid.

	Key k	Secret s	Capability c
Authorizing Agent	×	×	×
Platypus Router	×	·	×
Resource Principal		×	×
Trusted Third-Party		×	×
Others			×

Table 1: Capability knowledge hierarchy. Platypus routers are not explicitly given the secret, but regenerate it from the capability and the key.

Bindings intentionally do not consider packet length or any other capabilities that may be bound to the packet, allowing capabilities to be bound and removed from a packet in flight. Since the rest of a packet is fed into the MAC (including source and destination addresses), source-routed packets cannot be modified in any way. This unfortunately precludes both fragmentation and NAT. However, we do not consider the inability to fragment a significant limitation, as the sender should have a good idea of the path characteristics beforehand. The second limitation does seem somewhat more significant—we are currently considering alternative, NAT-friendly capability encodings.

4.2 Forwarding

In Platypus, an entity wishing to source route a flow must mark each packet with a capability for each AS or intermediate hop it wishes to specify. Intermediate hops are specified by IP address. We intend that the next hop IP is not necessarily a particular host or router in an AS, however, but instead a well-known address that falls within an AS's routable address space, and, thus, if used as a unicast packet destination, will cause the packet to reach some gateway of the AS in question.

The use of IP addresses as next-hop identifiers makes partial deployment of Platypus practical. Since Platypus-unaware ASes will route directly to the IP address of the next hop, a series of Platypus-capable end hosts can effectively provide the same service ideally provided within routers, but at the application level. Furthermore, this allows an AS to deploy a few Platypus-capable routers or hosts within its network before committing to a full deployment.

In Platypus, multiple capabilities can be bound to each packet. We expect that ASes or similar entities issue secrets for a set of capabilities to some subset of their users. These users can freely transfer their secrets to others, allowing third

parties to use their contracted services. In some cases, an end host will want to bind capabilities to a packet, but our architecture explicitly supports the inclusion of capabilities that are bound in flight by a middle box such as a NAT or gateway router to authorize requests as they exit an AS, allowing for specialized default routes.

Let us assume for the moment that Platypus is implemented entirely at border routers, although this need not be the case. When a source-routed packet arrives at an AS entry point (either from a customer or at a border with another AS), the gateway router inspects the destination address to see if it is a next-hop address in this AS. If so, it checks the capability binding, and, if the binding is valid, replaces the packet's destination with the next hop from the next capability in the source route list (or final destination, if no capabilities are left) and forwards the packet on. If the binding is invalid, the packet is dropped.

Platypus is not intended for intra-AS routing for two reasons. First, we argue that the benefits of source routing in the wide-area stem almost exclusively from its impact on the particular last mile links, AS peering points, and transit ASes used in a route. Preliminary measurements indicate that the benefit of specifying routes inside a particular (well-engineered) AS is often negligible. Second, and perhaps more importantly, traffic engineering is substantially easier when flows are relatively stable; hence, ISPs are unlikely to allow external entities to arbitrarily alter their internal traffic patterns.

5 Research challenges

There are a number of remaining issues that must be addressed to enable implementation and wide-spread deployment of Platypus. We are actively working on several key obstacles and describe our initial approaches below.

5.1 Capability distribution

We expect that each resource principal has a pre-existing relationship with the authorization agent(s) issuing capabilities on its behalf; therefore, mechanisms for mutual authentication already exist. The remaining challenge is in actually distributing new capabilities and secrets from authorization agent to resource principal. Once a secret is obtained, it may be used until it or the associated authorization key becomes invalid. To re-key, an authorization agent must not only update its keys at its associated routers, but must also send new secrets to every resource principal (who, presumably, will in turn further delegate them). We assume that the number of routers is not too large (even tier-one ISPs have only a few thousand routers) and that ISPs have pre-existing secure mechanisms of communicating with them.

Re-keying becomes problematic as the number of resource principals grows large or the re-keying rate increases. Broadcast encryption [8] may ease this distribution difficulty. Broadcast encryption allows a single broadcast over an insecure channel to be decipherable by only a pre-chosen subset of the receivers. For example, secrets could be posted to a web page. A more challenging problem is raised by resource principals that suspect that their secrets have been compromised or exposed, and wish to have their capabilities revoked and reissued. Due to the expected difference in frequency between authorization agent re-keying and individual capability revocation, it is desirable to have a far more lightweight mechanism for the latter. Such techniques are a subject of ongoing research.

5.2 Performance

We thus far have described Platypus in its idealized form, enabling per-packet path selection, verification, and accounting, but simultaneously requiring per-packet computation. We hypothesize, however, that such fine-grained flexibility is generally not required. For example, per-flow route selection is likely to be sufficient in most cases. In such instances, a trade-off exists between router state and computation: an RSVP-like flow establishment procedure could allow route requests to be validated once per flow, increasing state but significantly decreasing computation. We are actively exploring such a mechanism. An attractive feature of such an approach is its ability to off-load initial verification and flow set-up from routers to dedicated Platypus nodes.

In our current (completely unoptimized) UNIX prototype, HMAC-MD5-based capability verification takes on the order of 2 μsec per packet on a 2.8-GHz Intel P4—sufficient to handle a few 100-Mbps Ethernet links. We are optimistic that the order-of-magnitude improvement required to handle an OC-48 link can be achieved through optimization and the use of a more efficient MAC. We recognize, however, that hardware-assisted verification may be required for OC-192 line rates and above. Without such hardware, probabilistic verification schemes could be employed at resource-constrained routers. That is, Platypus routers could verify a packet's capability binding with some probability $p \ll 1$, reducing the computational overhead while, when deployed intelligently, continuing to verify the capability bindings of high-traffic principals. It is likely that ISPs are most interested in such principals; to ensure that their capabilities are both verified and accounted for, Platypus routers could utilize one of the many schemes to track high-frequency elements in a stream [5].

5.3 Accounting

The number of authorization keys that must be stored at a router scales with the number of associated authorization agents in an AS—that is, the number of entities that can issue capabilities authorizing the use of a particular router (or set of routers) in a source route. This number may be small—perhaps even just one. While mechanisms for supporting fine-grained accounting are outside the scope of this paper, since resource principals are meaningful only in the context of a particular authorization agent, an AS is also in full control of the number of distinct resource principals it

wishes to account for. It seems straightforward to employ a hierarchical resource-principal naming scheme in which an ISP need only account for a number of resource principals on the order of the number of its peers.

Within an ISP, routers can maintain per-principal accounts for payment and billing purposes. A straightforward approach would simply maintain a count of all packets for each resource principal at Platypus routers. However, since replay attacks are sometimes possible with our current design, an adversary could cause a resource principal to be billed an arbitrary amount by replaying a captured packet bearing a capability naming the resource principal.

A natural countermeasure is to track packets that traverse a router in an efficient manner and only count each distinct packet once within some time interval. A Bloom filter easily allows for tracking of packets in such a way. However, this presents a problem, as Bloom filters fill up over time, causing a high false positive rate. Ideally, we could use a windowed approach that would allow the contents the Bloom filter to decay over time.

6 Summary and status

In this paper we introduced the concept of network capabilities, an abstract mechanism for explicitly identifying both resource and authorization principals. We then presented Platypus, a loose source routing mechanism that uses network capabilities to enable local policymaking while giving both end hosts and ISPs the freedom to specify arbitrary routes. While capabilities themselves are not new, we believe their application to wide-area Internet routing is novel, and may free routing protocols from dealing with issues of policy. Instead, routing protocols can compute and distribute multiple routes, while policy can be enforced effectively and efficiently on the forwarding path. We are implementing prototype Platypus functionality in UNIX-based software routers and are studying its effectiveness on the RON test-bed.

References

[1] AGARWAL, S., CHUAH, C.-N., AND KATZ, R. H. OPCA: Robust interdomain policy routing and traffic control. In *Proc. IEEE OPENARCH* (New York, New York, June 2002).

[2] AGUILERA, M. K., JI, M., LILLIBRIDGE, M., MACCORMICK, J., OERTLI, E., ANDERSEN, D. G., BURROWS, M., MANN, T., AND THEKKATH, C. A. Block-level security for network-attached disks. In *Proc. USENIX Conference on File and Storage Technologies* (San Francisco, California, Apr. 2003).

[3] ANDERSEN, D. G., BALAKRISHNAN, H., KAASHOEK, M. F., AND MORRIS, R. T. Resilient overlay networks. In *Proceedings of the 18th ACM Symposium on Operating System Principles (SOSP)* (Banff, Canada, Oct. 2001), pp. 131–145.

[4] CLARK, D. D., WROCLAWSKI, J., SOLLINS, K. R., AND BRADEN, R. Tussle in cyberspace: Defining tomorrow's Internet. In *Proceedings of the ACM SIGCOMM Conference* (Pittsburgh, Pennsylvania, Aug. 2002), pp. 346–356.

[5] ESTAN, C., AND VARGHESE, G. New directions in traffic measurement and accounting. In *Proceedings of the ACM SIGCOMM Conference* (Pittsburgh, Pennsylvania, Aug. 2002), pp. 323–336.

[6] FEAMSTER, N., AND REXFORD, J. Network-wide BGP route prediction for traffic engineering. In *Proc. ITCOM* (Boston, Massachusetts, July 2002).

[7] FERGUSON, P., AND SENIE, D. Network ingress filtering: Defeating denial of service attacks which employ IP source address spoofing. RFC 2267, Internet Engineering Task Force, Jan. 1998.

[8] FIAT, A., AND NAOR, M. Broadcast encryption. In *Proc. CRYPTO* (Santa Barbara, California, Aug. 1993), pp. 480–491.

[9] GRIFFIN, T. G., AND WILFONG, G. An analysis of BGP convergence properties. In *Proceedings of the ACM SIGCOMM Conference* (Cambridge, Massachusetts, Sept. 1999), pp. 277–288.

[10] HUSTON, G. Commentary on inter-domain routing in the Internet. RFC 3221, Internet Engineering Task Force, Dec. 2001.

[11] JANNOTTI, J. Network layer support for overlay networks. In *Proc. IEEE OPENARCH* (New York, New York, June 2002), pp. 3–13.

[12] LABOVITZ, C., AHUJA, A., BOSE, A., AND JAHANIAN, F. Delayed Internet routing convergence. *IEEE/ACM Transactions on Networking 9*, 3 (June 2001), 293–306.

[13] MAHAJAN, R., WETHERALL, D., AND ANDERSON, T. Understanding BGP misconfiguration. In *Proceedings of the ACM SIGCOMM Conference* (Pittsburgh, Pennsylvania, Aug. 2002), pp. 3–16.

[14] MEDINA, A., TAFT, N., SALAMATIAN, K., BHATTACHARYYA, S., AND DIOT, C. Traffic matrix estimation: Existing techniques and new directions. In *Proceedings of the ACM SIGCOMM Conference* (Pittsburgh, Pennsylvania, Aug. 2002), pp. 161–174.

[15] NORTON, W. B. Internet service providers and peering. In *Proceedings of NANOG 19* (Albuquerque, New Mexico, June 2000).

[16] POSTEL, J. Internet Protocol. RFC 791, Internet Engineering Task Force, Sept. 1981.

[17] ROUGHAN, M., THORUP, M., AND ZHANG, Y. Traffic engineering with estimated traffic matrices. In *Proceedings of the USENIX/ACM Internet Measurement Conference* (Miami, Florida, Oct. 2003), pp. 248–258.

[18] SAVAGE, S., COLLINS, A., HOFFMAN, E., SNELL, J., AND ANDERSON, T. The end-to-end effects of Internet path selection. In *Proceedings of the ACM SIGCOMM Conference* (Cambridge, Massachusetts, Sept. 1999), pp. 289–299.

[19] SAVAGE, S., WETHERALL, D., KARLIN, A., AND ANDERSON, T. Network support for IP traceback. *IEEE/ACM Transactions on Networking 9*, 3 (June 2001).

[20] SNOEREN, A. C., PARTRIDGE, C., SANCHEZ, L. A., JONES, C. E., TCHAKOUNTIO, F., SCHWARTZ, B., KENT, S. T., AND STRAYER, W. T. Single-packet IP traceback. *IEEE/ACM Transactions on Networking 10*, 6 (Dec. 2002), 721–734.

[21] STEENSTRUP, M. An architecture for inter-domain policy routing. RFC 1478, Internet Engineering Task Force, June 1993.

[22] STOICA, I., ADINGS, D., ZHUANG, S., SHENKER, S., AND SURANA, S. Internet indirection infrastructure. In *Proceedings of the ACM SIGCOMM Conference* (Pittsburgh, Pennsylvania, Aug. 2002), pp. 73–88.

[23] STOICA, I., AND ZHANG, H. LIRA: An approach for service differentiation in the Internet. In *Proc. NOSSDAV* (June 1998), pp. 115–128.

[24] TAHILRAMANI KAUR, H., , WEISS, A., KANWAR, S., KALYANARAMAN, S., AND GANDHI, A. BANANAS: An evolutionary framework for explicit and multipath routing in the internet. In *Proc. ACM SIGCOMM Workshop on Future Directions in Network Architecture* (Karlsruhe, Germany, Aug. 2003), pp. 277–288.

[25] YANG, X. NIRA: A new Internet routing architecture. In *Proc. ACM SIGCOMM Workshop on Future Directions in Network Architecture* (Karlsruhe, Germany, Aug. 2003), pp. 301–312.

[26] ZHU, D., GRITTER, M., AND CHERITON, D. R. Feedback based routing. In *Proc. Workshop on Hot Topics in Networks* (Princeton, New Jersey, Oct. 2002), pp. 71–76.