Autonomous-System Interfaces

Katerina Argyraki, Petros Maniatis, Timothy Roscoe
EPF Lausanne, Intel Research Berkeley, ETH Zürich

Abstract

The current Internet is characterized by a growing tension between the “core” (the Internet service providers) and the ”edge” (the operators of edge networks and distributed applications). Much of this tension concerns path visibility and control – where traffic goes (route control), where traffic comes from (path identification and filtering), and what happened in between (monitoring and accountability). We argue that this conflict harms both the core and the edge and that, to resolve it, we have to expose the Autonomous System (AS) as a first-class Internet object. This would map the functional structure of the Internet (the granularity at which edge systems can observe and control their traffic) to the organizational one (a graph of ASes). We argue that providing a well-defined interface between core and edge ASes offers significant benefits to both of them.

1 Introduction

Today we see a growing tension in the Internet between the ISPs and the operators of edge networks and distributed applications. On the one hand, end systems increasingly resort to overlay networks (content distribution networks, P2P systems, etc.) to overcome perceived limitations of the current Internet. Edge-controlled overlays are used, for example, to provide probabilistic guarantees of bandwidth, delay, or loss; to provide content addressing; or, in some cases, to explicitly “bypass ISP control” [25]. On the other hand, ISPs have come to believe that at least some of these efforts impede their ability to capture value from the traffic they forward. A prime example is the ongoing debate over “net neutrality.” A Google search on “Skype Verso” reveals a rich set of commentary on the developing ISP war against “undesirable traffic,” defined as “network traffic that takes up large amounts of bandwidth without generating any revenue for the carrier, such as Skype calls” [3].

Underlying this tension is a difference of views as to what constitutes the “service” provided by the network. The edge believes it to be an undifferentiated best-effort service that ships bits regardless of application or load, paid for by individual end systems. The “middle” believes it to be an integral component of Internet applications, which should share proportionately in the revenue these applications generate. In these unfortunate circumstances, we argue it is impossible to objectively define what is meant by terms like “availability” or “service”: for example, if the network is running at 25% utilization, but blocks Skype calls, is it available?

Can ISPs regain some control over the use of their networks, to capture revenue by differentiating their service and thereby provide better performance for edge-controlled networking? This would seem preferable to the currently-favored strategy of reacting by executing business agreements with some large application providers while blocking others, a course of action that seems likely to result in further “walled gardens” of large content and network providers.

In this paper, we argue that a major technical obstacle to this is the somewhat ambiguous position of the Autonomous System (AS) in the Internet. ASes map roughly to the commercial, organizational structure of the Internet: AS boundaries are revenue boundaries, where financial settlement occurs for carrying traffic. Yet an AS does not quite manage to be a first-class object in the Internet. To end systems, the Internet is a black box, which they poke with various probing tools in an effort to reverse-engineer its structure and localize its failures; even when they are successful, the visibility they get is at the level of routers, and mapping router addresses to AS numbers is an error-prone task [24]. The antagonism between ISPs and end systems is hardly surprising given that the latter’s operational units (overlay networks) have almost no way to communicate with the former’s (ASes).

Consequently, we examine a simple idea that has been implicitly present or hinted to in many recent research proposals: to make the AS a first-class, visible Internet object. The goal is to map the functional structure of the Internet (the granularity at which end systems can observe and control their traffic) to the organizational one (a graph of ASes). We argue that providing a well-defined interface between ASes and end systems offers significant benefits to both: ISPs can offer functionality which is useful to the end systems, and extract value from it.

The rest of the paper is structured as follows. We first examine the issues that have led to the deployment of edge-controlled overlays today. We then present a basic AS interface that addresses these issues, and discuss how it would make life easier both for end systems and ISPs. We stay away from implementation details, as the goal of this paper is to put forward and motivate the idea of an explicit AS interface; but we do present, before concluding, a few thoughts on how such an interface might be implemented in a way that does not disturb the current ISP operational model, while leaving room for evolution in the future.

2 Antagonisms over Path Control

A major part of networking research has traditionally centered on finding the right balance of *path control* between the edge (end hosts and edge networks) and the core (the ISPs). More specifically:

Route control mechanisms enable a sender to influence the end-to-end path of its outgoing traffic. This was the aim of the loose source route (LSR) IP option, however, forwarding overheads (due to processing off the fast path) and security concerns hindered its adoption; as a result, the Internet evolved to be opaque to the routing preferences of traffic sources. Recently, interest in highly-available routing and wide-area media streaming has led to developments in hardware-friendly source routing [12] and fast verification of source-route compliance with ISP policy [26]. Unfortunately, so far, such technical advancements have come with little adoption incentive for ISPs, who operate their networks on the assumption that they control the routes of the packets they forward. The response by network users has been application-layer overlays, by which end systems form their own P2P networks and direct their traffic through a specified sequence of edge peers (e.g., RON [4]). In return, ISPs are increasingly offering their own QoS contracts and treating the corresponding traffic preferentially to overlay traffic.

Path identification and filtering mechanisms enable a receiver to control what traffic uses its resources by (i) identifying the path followed by incoming traffic and (ii) filtering traffic according to that path. The record route (RR) IP option was aimed at path identification, but, sharing the same limitations with LSR, it was rarely adopted; as a result, the Internet turned equally opaque to the ability of receivers to identify the path followed by incoming traffic. The need for highly-available services in the face of denial-of-service (DoS) attacks has recently led to research in hardware-friendly path identification [27,31] and complementary mechanisms for traffic filtering according to sender or path [5, 6]. As with route control, the need to change routers' fast-path functionality has not been accompanied by clear incentives for adoption by the ISPs. Users have again turned to edge-based solutions that enable them to direct incoming traffic through specialized filtering nodes without ISP involvement [2]. In response, ISPs are increasingly offering custom DoS-protection contracts, promising to detect and block undesired traffic before it reaches their customers.

Path monitoring mechanisms enable a pair of end systems to assess the quality of their communication path and potentially switch to an alternative in case of failure. Offering such capabilities to end systems made little sense in the absence of path control of outgoing traffic or path observability of incoming traffic; hence, the Internet evolved with no official monitoring mechanism. However, failure accountability (who dropped what when) is now increasingly important not only for better provider choice, but also to assign liability – recent economic analysis argues that without verifiable service level agreements (SLAs) on the path, innovation and, consequently, service differentiation are impossible [19]. Technical proposals for domain path monitoring (e.g., packet obituaries [7]) face the usual criticism: ISPs have no reason to concede such information to end systems. Unsurprisingly, researchers turn to overlay-based solutions, i.e., probing between multiple communicating end points and combining the results using network tomography [13]. A potential ISP response (imminent, some argue) is to render their networks opaque to end-to-end probes or manipulate them to misreport their performance – some carriers already disable TTL decrementing for intra-POP links to prevent topology discovery.

In all three cases, the antagonism is marked by the ISPs threatening penalization at the underlay on the one hand, and the end systems attempting to "bypass ISP control" [25] on the other. In the next section, we attempt to answer the following question: Is there a mutually beneficial way in which ISPs can be adequately enticed to cede to end systems control of where outgoing traffic goes, where incoming traffic comes from, and where traffic gets lost or delayed?

3 ASes as First-class Objects

3.1 The Right Granularity

A common trend in all three research areas is that they started out with a router-level view of the Internet, but are more or less converging to an AS-level view. Consider, for instance, route control: traditional LSR enabled a sender to specify a sequence of routers that should forward a packet; recent solutions enable a sender to specify a sequence of *border* routers [12] or domains [33]. Path identification and filtering solutions have evolved the same way: they started from packet marking by individual routers [27, 31] and router-by-router propagation of filters [23], and are moving towards marking and filtering of packets at domain boundaries [34].

Network-monitoring solutions have evolved the least in this aspect. Traceroute lets a sender generate traffic intended to trigger predictable ICMP responses from encountered routers, which can be aggregated to infer router or path quality. Similarly, much monitoring research employs black-box probing to infer router-level internal state or structure [13]. Recently, it was suggested that, if the end systems are to receive any feedback on the state of the Internet, it would be sufficient and practical to offer such feedback at the granularity of an AS [7], i.e., tell which AS, not which router, is dropping or delaying packets.

Plausible explanations of this trend exist from the point of view of both the ISPs and the end systems. ISPs are increasingly reluctant to allow visibility into their internal structure and state (not to mention control over the latter) to the outside world; the same techniques that enable probing of their internals at router-level granularity can be used to direct traffic to under-provisioned corners of their domain, leaving them vulnerable to attacks and to their competitors' marketing literature. Security at least via obscurity of their internals is a common practice that ISPs are not ready to abandon.

From the end-system point of view, router-level visibility of the Internet and router-level control of end-to-end communications, though seductive with an excess of information, can be a scalability nightmare. For example, a path-identification solution must solve a very different problem when dealing with AS-granularity paths (with their average AS Internet diameter of under 10 from about 30,000 ASes) versus dealing with router-granularity paths (at lengths longer by an order of magnitude and hundreds or thousands of routers per AS). Similar arguments hold for "Internet health services" that must monitor thousands of AS-to-AS links versus millions of router-to-router links. In certain cases, router-level information may even be useless to the edges: when monitoring end-to-end paths for accountability, what matters to them is not which particular router dropped or delayed a packet, but which administrative entity is responsible – perhaps liable – for the failure.

A natural design then is to map the functional structure of the Internet (the granularity at which route control, path identification, and monitoring can be applied) to the organizational structure of the Internet (the granularity at which these functions are meaningful); this structure is a graph of ASes whose vested interests push them to remain opaque. However, the Internet was not designed this way, leading much research (and practice) to resort to emulating this functional structure by poking into ISP internals, for example, using traceroute (a router-level solution) to figure out which ASes are losing or delaying packets. Even when they attempt to respect AS boundaries, such solutions make choices in the end systems' own terms, thereby remaining brittle and vulnerable to internal AS reorganizations.

We argue instead that an explicit interface exported by ASes to end systems can resolve this conflict by giving the end systems useful path control, while respecting ISP privacy and benefiting ISP business – experience has shown that any solution without these two traits is bound to fail. In particular, we suggest an interface that enables end systems to observe and control exactly what ISPs expose even today by necessity: given a certain packet, its forwarding (or dropping) to another AS. At that granularity, the interface enables opaque flexibility "below" (which the ISP can exercise by changing internal technology, management, topology, policy, or by service differentiation) while leaving room for attracting customer choice "above."

3.2 A Basic AS Interface

We now specify a minimal AS interface that enables end systems to monitor, forward, and filter their traffic at AS granularity, while treating each AS as a black box. An AS offers its interface via its *checkpoints*, a set of virtual, publicly addressable nodes that represent explicit points of observation and control exported by the AS. At one extreme, an AS may export a checkpoint for each of its external links with neighboring ASes – essentially, providing visibility to its inter-AS links; on the other extreme, it may export a single checkpoint that represents the entire AS as one "dimensionless" vertex in the Internet's topology.

$report(aggregate, attribute)$
$forward(aggregate, nextHop)$
$mark(aggregate, offset, attribute)$
$drop(aggregate, lastHop)$

Table 1: A basic AS interface. An $aggregate$ consists of a bit mask m and a comparison bit field c; a packet p belongs to a certain traffic aggregate when, viewed as a bit string, it satisfies (p BITWISE-AND $m = c$). The acceptable $attribute$ values are specified in Table 2.

To use the interface, an edge AS adds a path-control module to its management platform, which discovers available checkpoints, starting with a pre-configured list of those of its immediate neighboring ASes. This module can submit requests to the checkpoints of other ASes; upon receiving a request, a checkpoint verifies the requester's credentials and potentially configures its AS's data-path to perform the requested operation – provided it is consistent with local policies. Note that the intended clients of the interface are not individual end nodes; to exercise path control, end nodes send their requests to their local AS's path-control module, which performs any necessary authentication and access control, and interacts with the checkpoints of other ASes.

Table 1 summarizes the available interface primitives; the object of each primitive is an *aggregate* describing a set of packets (see caption) and the "topic" is a traffic $attribute$ (from the list in Table 2). With the $report$ primitive, a source AS asks from a transit AS to send it information on how an outgoing traffic aggregate is forwarded. For instance, a source AS can ask the checkpoint of its ISP to report the $nextHop$ for a certain traffic aggregate, then ask the same thing from the reported $nextHop$ and so on, and construct, in this manner, a checkpoint sequence that maps to the AS-level path followed by the specified aggregate. The source AS can then use the $forward$ primitive to request a routing change for the aggregate. Similarly, with the $mark$ primitive, a destination AS asks from a transit AS to mark an *incoming* traffic aggregate with path information – in particular, the $lastHop$ that forwards the aggregate to the reporting AS. The destination AS can then use the $drop$ primitive to ask from an AS to drop an unwanted incoming aggregate. So, $report$ reveals outgoing path information that can be leveraged with $forward$, while $mark$ reveals incoming path information that can be leveraged with $drop$.

A checkpoint determines the validity of a request based on its origin: a $report$ or $forward$ request is valid when coming from the edge AS that sourced the specified traffic aggregate; a $mark$ or $drop$ request is valid when coming from the edge AS that received the specified traffic aggregate. Clearly, there are other security and deployment issues – more than we can fit in this paper; we defer such issues to future technical proposals and focus, instead, on *why* end systems and ISPs would use or deploy our interface, next.

Attribute	Description	Request
$pktsIn$	Number of $aggregate$ packets that entered the reporter.	$report$
$pktsOut$	Number of $aggregate$ packets that exited the reporter.	$report$
$entryTime$	Average entry time of $aggregate$ packets into the reporter.	$report$
$exitTime$	Average exit time of $aggregate$ packets from reporter.	$report$
$lastHop$	The previous checkpoint that observed $aggregate$.	$report$, $mark$
$nextHop$	The next checkpoint to which $aggregate$ was forwarded.	$report$, $mark$

Table 2: The $attribute$ values that can be specified with $report$ or $mark$ requests. $entryTime$ and $exitTime$ represent absolute wall-clock time.

3.3 Uses and Incentives

We now discuss how the interface presented in Section 3.2 would help define and improve network availability for end systems, while benefiting the participating ISPs.

Accountability: Today, when packets get lost or delayed, there is no way of identifying the culprit – it could be any of the ISPs on the path or the destination network. To compensate for this lack of accountability, end-system operators typically resort to probing tools like traceroute; these treat the Internet like a black box and poke it with different probes in a (not always successful) effort to reverse-engineer its structure and localize its failures.

The proposed AS interface enables an alternative approach: each source AS can explicitly ask from transit ASes to report the loss and average delay they introduce in its traffic. Of course there are challenges in building such an accountability framework – detecting lying ASes, preventing abuse by malicious nodes, dealing with unsynchronized checkpoint clocks, to name a few; for a review of these challenges and a way to address them, we refer the reader to our technical report [8].

For end systems, this approach is better than probing, because (i) it provides accurate statistics on the actual traffic (not just probes), and (ii) reveals which ASes are accountable for each failure without requiring any mapping from router IP addresses to AS numbers [24]. Most importantly, it allows each end system to define a maximum acceptable loss and average delay per transit AS and compute each transit AS's availability with respect to the end-system's own traffic aggregates. A source can leverage this information to make the best use of whatever route control it has available (through multi-homing or overlays), and also verify whether its ISP is honoring their SLA. A recent economic study shows that such verifiable SLAs are the only way to ensure competition and innovation in the Internet [19].

The benefit for ISPs is less intuitive – the first question that comes to mind is, why would ISPs ever want to subject themselves to accurate evaluation? The answer is: to escape the inaccurate and intrusive evaluation to which they are already subjected today. Traceroute may provide less information than end systems want, but, as far as ISPs are concerned, it gives away too much: their router-level structure and internal routing policy. Yet ISPs cannot simply turn traceroute off, because customers have come to expect it. If an ISP's routers stop responding to traceroute probes, its customers assume that the ISP is malfunctioning; there are even reports of customers using traceroute logs as evidence that their ISP violated their SLA to claim compensation. The only way out for ISPs is to offer an alternative, which provides end systems with the information they want without exposing internal ISP structure and policy. Exporting an interface that provides statistics at AS granularity meets both goals.

ISP-friendly route control: Today, ISPs' ability to fetch revenues is directly linked to their routing policies. We do not advocate that end systems dictate or even discover these policies; instead, we propose that ISPs expose multiple (policy-compliant) options, from which end systems are allowed to choose. E.g., Skype nodes can be divided in "simple" and "super-" nodes; the latter are nodes with public IP addresses and enough resources to act as proxies to the simple nodes [9]. By allowing source networks to explicitly request forwarding of their Skype traffic through different next hops, an AS can essentially deploy multiple super-nodes – all of them routing consistently with its policies.

End systems can only win by using such a service: they can choose highly available routes (consistent with transit-AS policies) and reduce the route flaps and path inflation associated with diverging overlay and underlay topologies [16, 29]. For ISPs, providing this service means ceding some explicit route control to end systems; in exchange, they avoid the implicit route control already exercised by overlays today, which successfully bypasses ISP routing policies [28] and defeats their ability to provision their infrastructure via computation of traffic matrices [17].

Traceback: The basic idea behind most traceback solutions is simple and elegant: each participating router marks the packets it forwards with an identifier; the receiver of a packet processes the sequence of identifiers and reconstructs the path followed by the packet. The problem is the implementation: the IP header does not provide sufficient room for path identifiers, leading researchers to invent intelligent marking schemes that fit trace information in unused IP header fields [14, 31]. The catch is that these "compressed" identifiers either do not reveal the full path followed by each packet or distribute this information in multiple packets, making path-based packet filtering impossible.

The *mark* primitive of Table 1 simplifies the implementation of traceback – and potentially any proposal that requires marking space, like network capabilities [5, 32, 34] – by removing the need for compression. When asked by a receiver, senders can reserve traceback space in their packets in advance; the required size of this space can be determined similarly to MTU size. At each transit AS, the sender's packets are classified based on their traceback header and consequently marked with path information at an appropriate offset within the scratch space. Senders that send packets to the receiver with no space for traceback can be treated with lower priority.

Filtering services: In response to the recent rise of denial of service, ISPs have started to offer pro-active DoS protection: they monitor the traffic addressed to their customers (edge networks or other ISPs) and drop suspicious traffic before it consumes customer resources. The catch is that ISPs cannot always tell unwanted from legitimate traffic (the distinction can be application-specific) nor selectively block traffic from each attack source (there can be tens or hundreds of thousands of attack sources); under heavy attacks, their only option may be to pull the plug on the targeted customer.

The proposed AS interface enables an alternative approach, already implied or advocated in research proposals [6, 23]: If the path followed by unwanted traffic is known (see *Traceback* paragraph above), the receiving AS can explicitly ask from the source AS that generates the unwanted traffic (or a transit AS that carries it) to stop forwarding this traffic. The receiving AS can then treat the (legitimate) traffic from cooperating ASes preferentially, while rate-limiting traffic from non-cooperating ASes.

There are two well known arguments for why this reactive approach improves the availability of a server that is under attack compared to pro-active ISP protection. First, it leads to less collateral damage, because the target is in a better position to distinguish unwanted from legitimate traffic than its ISP. Second, it can handle more attack sources, because unwanted traffic is blocked as close as possible to its sources, where more filtering resources are available per attack source.

The benefit for participating ISPs is also related to availability – the availability of their own service as experienced by their customers. Given the nature of legitimate TCP flows, which back off in the face of packet loss, rate-limiting a mix of legitimate and attack traffic can practically drive legitimate throughput to zero [18]. When an AS hosts misbehaving clients that flood, say, eBay's link with unwanted traffic, eBay's network can at best rate-limit all traffic from that AS; this essentially penalizes the AS's legitimate clients, which can no longer connect to eBay. However, if the AS blocks its own misbehaving clients, then there is no reason for eBay to rate-limit all its traffic, which means that its legitimate clients maintain their connectivity to eBay throughout the attack. So, exporting a filtering interface gives ASes more control over *their own* connectivity: they can choose whether it is worth blocking traffic from misbehaving clients, based on how much that will benefit the connectivity of their legitimate clients.

4 Implementation and Deployment

We have argued for an AS interface that enables end systems to explicitly request monitoring, forwarding, marking and filtering of their traffic. The question might legitimately be raised as to whether implementing such an interface is practical for ISPs, particularly given the complexity involved in the myriad of BGP policies in use today.

Nevertheless, we are optimistic that a more flexible routing platform supporting user-supplied policies is feasible, and indeed desirable to an ISP in terms of management overhead. Our optimism is based on recent developments in different areas of networking research, which point the way to the system architecture, underlying theoretical model, and implementation technology for such a scheme.

We first observe that the datapath part of the problem, namely packet classification, marking or monitoring and policy routing at line rate, is largely a solved problem in modern networking hardware [1]. Moreover, recent research has shown how to enhance the datapath with configurable line-rate packet processing [11].

Architecturally, we favor the use of an explicit control plane reminiscent of a *routing platform* [10, 30], i.e., a logically centralized (though physically distributed) service exporting the interface we propose to clients. This control plane consists of the checkpoints mentioned in Section 3.2.

Theoretically, while current route control platforms tend to focus explicitly on BGP, we believe that a routing model based on the algebras of *metarouting* [15] offers a more sound and comprehensive theoretical basis for new routing/filtering systems. More formalized models of routing are easier to reason about automatically, and are thus more amenable to incorporating user-supplied policies in a way that is safe for the carrier (and verifiably so).

Finally, we argue that the implementation of such an AS routing/filtering service would benefit greatly from the use of declarative logic languages. Such languages have been previously demonstrated to cover a large fraction of Internet routing protocols [22] as well as overlay networks [21]. For example, a simple link-state routing protocol can be expressed in a handful of rules and automatically translated into an executable specification that generates the same messages that a hand-coded protocol would generate [20]. Declarativity is a time-honored tradition, especially for policy and contract specifications, where the means are unknown or unknowable, but the ends are declared in detail.

Besides their conciseness and high-level, formal nature, the relationship between logic languages and database queries is an appealing characteristic for implementing our black-box AS interface. Declarative routing systems treat the set of routing tables in a network as a *database view* over the distributed state of the network, also represented as database relations. The routing process is therefore one of distributed view maintenance, performed by evaluating continuous distributed queries. This data-centric view naturally integrates user policy state (as well as resource discovery and system management) in a single implementation framework.

We also observe in passing that declarative languages point to a way to evolve the basic strawman interface we presented in Section 3.2. In the future, we speculate that operators will allow users to specify a much richer range of routing behaviors for their traffic, by accepting declarative descriptions of such policies instead of the primitive behaviors we have dealt with so far. In fact, declarative languages, especially those based on Datalog, have a fairly extensive literature on static verification of important properties of programs, including termination and convergence [20]. Furthermore, this opens up the possibility for ASes to innovate in the functionality they offer without requiring protocol extensions to be agreed in advance by all operators, as long as they can be expressed in the language.

5 The Way Forward

As the growing pains of the Internet lead researchers to bold revolutionary redesigns of everything and at the same time to broad band-aids to the current proven but ailing network, this paper proposes a step in between: the explicit though incremental institution of an AS interface as a basic extensibility and observability building block. This has the potential to reconcile the need of the edge for disruptive new services with the need for a compatible competitive playing field among ISPs. We have argued that much of the ambitious, evolutionary research on route control, filtering and monitoring services can be accommodated by such an interface and yet be amenable to plausible pricing and control by the ISPs. We have discussed a strawman design for this interface and a few thoughts on how to implement it.

Many interesting and important research questions open up. First, though we have argued for declarative interfaces and implementations thereof, to give ISPs internal independence from their customers' expectations, the particular choice of language can vary in terms of readability, expressiveness, and intuitiveness. Second, small variations in the granularity of the interface may highlight important trade-offs: whereas a checkpoint as the building block exposes multiple paths for the same AS-to-AS route, an AS as a larger building block may be an easier and perhaps cheaper abstraction for applications to exploit programmatically. Third, it may be possible to revisit the broad results of the OpenArch/OpenSig communities in resource reservation, isolation of routing extensions, etc., as flexible tools towards implementing the AS interface, under the covers. Finally, beyond our qualitative argument for the mutual benefit to ISPs and the edge of a controllable AS interface, a rigorous economic argument extending the results of [19] to this compartmentalization would be a valuable next step.

References

[1] LSI Network Processors. http://www.lsi.com/networking_home/networking_products/network_processors/.

[2] Prolexic. http://www.prolexic.com.

[3] Verso Technologies blocks Skype. http://blog.tmcnet.com/blog/tom-keating/skype/verso-technologies-blocks-skype.asp.

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

[5] T. Anderson, T. Roscoe, and D. Wetherall. Preventing Internet Denial-of-Service with Capabilities. In *HotNets*, 2003.

[6] K. Argyraki and D. R. Cheriton. Active Internet Traffic Filtering: Real-time Response To Denial-of-service Attacks. In *USENIX*, 2005.

[7] K. Argyraki, P. Maniatis, D. Cheriton, and S. Shenker. Providing Packet Obituaries. In *ACM HotNets*, 2004.

[8] K. Argyraki, P. Maniatis, O. Irzak, and S. Shenker. An Accountability Interface for the Internet. Technical report, EPFL, 2007.

[9] S. A. Baset and H. Schulzrinne. An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol. In *INFOCOM*, 2006.

[10] M. Caesar, D. Caldwell, N. Feamster, J. Rexford, A. Shaikh, and K. van der Merwe. Design and Implementation of a Routing Control Platform. In *USENIX NSDI*, 2005.

[11] K. L. Calvert, J. Griffioen, and S. Wen. Lightweight Network Support for Scalable End-to-end Services. In *SIGCOMM*, 2002.

[12] D. R. Cheriton and M. Gritter. TRIAD: A Scalable Deployable NAT-based Internet Architecture. Technical report, Stanford University, 2000.

[13] M. Coates, A. O. Hero, R. Nowak, and B. Yu. Internet Tomography. 19(3), 2002.

[14] D. Dean, M. Franklin, and A. Stubblefield. An Algebraic Approach to IP Traceback. *Information and System Security*, 5(2), 2002.

[15] T. G. Griffin and J. L. Sobrinho. Metarouting. In *SIGCOMM*, 2005.

[16] R. Keralapura, C.-N. Chuah, N. Taft, and G. Iannaccone. Can Coexisting Overlays Inadvertently Step on Each Other? In *ICNP*, 2005.

[17] R. Keralapura, N. Taft, C.-N. Chuah, and G. Iannaccone. Can ISPs take the heat from Overlay Networks? In *HotNets*, 2004.

[18] A. Kuzmanovic and E. W. Knightly. Low-Rate TCP-Targeted Denial of Service Attacks. In *SIGCOMM*, 2003.

[19] P. Laskowski and J. Chuang. Network Monitors and Contracting Systems. In *ACM SIGCOMM*, 2006.

[20] B. T. Loo, T. Condie, M. Garofalakis, D. E. Gay, J. M. Hellerstein, P. Maniatis, R. Ramakrishnan, T. Roscoe, and I. Stoica. Declarative Networking: Language, Execution and Optimization. In *SIGMOD*, 2006.

[21] B. T. Loo, T. Condie, J. M. Hellerstein, P. Maniatis, T. Roscoe, and I. Stoica. Implementing Declarative Overlays. In *SOSP*, 2005.

[22] B. T. Loo, J. M. Hellerstein, I. Stoica, and R. Ramakrishnan. Declarative Routing: Extensible Routing with Declarative Queries. In *SIGCOMM*, 2005.

[23] R. Mahajan, S. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Schenker. Controlling High Bandwidth Aggregates in the Network. *ACM CCR*, 32(3), 2002.

[24] Z. M. Mao, J. Rexford, J. Wang, and R. H. Katz. Towards an Accurate AS-Level Traceroute Tool. In *ACM SIGCOMM*, 2003.

[25] L. Peterson, S. Shenker, and J. Turner. Overcoming the Internet Impasse through Virtualization. In *HotNets*, 2004.

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

[27] S. Savage, D. Wetherall, A. Karlin, and T. Anderson. Practical Network Support for IP Traceback. In *SIGCOMM*, 2000.

[28] S. Seetharaman and M. Ammar. Characterizing and Mitigating Inter-domain Policy Violations in Overlay Routes. In *ICNP*, 2006.

[29] S. Seetharaman and M. Ammar. On the Interaction between Dynamic Routing in the Native and Overlay Layers. In *INFOCOM*, 2006.

[30] P. Verkaik, D. Pei, T. Scholl, A. Shaikh, A. C. Snoeren, and J. E. van der Merwe. Wresting Control from BGP: Scalable, Fine-grained Route Control. In *USENIX*, 2007.

[31] A. Yaar, A. Perrig, and D. Song. Pi: A Path Identification Mechanism to Defend against DDoS Attacks. In *IEEE Symposium on Security and Privacy*, 2003.

[32] A. Yaar, A. Perrig, and D. Song. SIFF: A Stateless Internet Flow Filter to Mitigate DDoS Flooding Attacks. In *IEEE Symposium on Security and Privacy*, 2004.

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

[34] X. Yang, D. Wetherall, and T. Anderson. A DoS-limiting Architecture. In *SIGCOMM*, 2005.