Designing the Network Layer for End-Host Traffic Engineering

Chris Erway *John Jannotti*
Brown University
{cce,jj}@cs.brown.edu

Abstract

Traffic engineering (TE) is used to control the distribution of network load across the links and routers of a network. TE techniques usually focus on multiplexing traffic across competing paths connecting a source and destination. There are two significant barriers to traffic engineering on the Internet. First, it is difficult to employ interdomain traffic engineering (across multiple autonomous systems) because BGP forces a router to select a single best route per prefix. Second, intradomain TE is complicated because the paths of individual flows can not be changed arbitrarily without negatively affecting high-layer protocols, particularly TCP.

These disparate difficulties can both be addressed by exposing a portion of the TE decision process to end-hosts. We propose a *tag* mechanism that allows end-hosts to influence, but not control, the paths chosen by the network for its packets. End-hosts will provide explicit opportunities for TE by tagging packets with opaque identifiers. Only packets with the same tag must be delivered along the same path. The tag mechanism will also allow multiple ASs to coordinate to provide interdomain traffic engineering.

Our analysis of Internet provider/peering topologies shows that there are multiple paths available between the majority of ASes that do not violate AS transit policies, and that tags would expose these paths to end-host traffic engineering.

1 Introduction

Internet routing occurs at two scales. Within a single domain, a routing protocols such as OSPF selects paths from any source to any destination. These paths consist of routers and the links that connect them. At a larger scale, BGP selects paths that consist of entire networks and the link between them. At both scales, sources and destinations may be connected by several potential paths but existing routing techniques favor the use of a single path. This preference complicates any attempt to balance network load across available resources.

Intradomain protocols such as OSPF are capable of finding multiple routes with identical costs, but best practices require that IP packets from the same flow be forwarded along a single path. Multiplexing packets from a single flow across multiple paths would lead to out-of-order arrivals and unpredictable latencies, which complicates congestion control, particularly for TCP.

BGP, the standard interdomain routing protocol, is a path-vector protocol that determines viable, rather than optimal, paths. An individual domain or autonomous system (AS) obtains advertisements from its neighbors describing the path from that neighbor to a given destination. Using a combination of local policies and heuristics such as minimizing the length of the AS path to the destination, one of the advertised paths is selected. Packets for that destination will be forwarded to the first AS of the chosen path. In addition, the AS may (again, depending on policies) advertise that path, after appending itself, to its neighbors.

BGP finds a single path from an AS to any given destination. Although an AS is presented with many possible advertisements, it must select exactly one such path in order to avoid cycles. An AS prevents cycles by refusing any advertisement that contains itself in the advertised path. The implicit assumption is that the advertising AS will forward along the path it advertises. If it might select from several possibilities, one of those possibilities might cycle back to the AS receiving the advertisement.

Intradomain TE Despite these difficulties, traffic engineering has emerged in an effort to exploit path diversity within an AS to balance network load. Balanced load minimizes queueing delay at a given utilization. The AS's *traffic matrix*, which describes how the flow of traffic coming in from each neighbor AS is divided among each outgoing neighbor, is analyzed to determine the optimal usage of (multiple) paths from each ingress to each egress. However, routing must depart from this optimal solution, in order to avoid poor interactions with end-host flow-control. Traffic must be migrated entire flows at a time so that any given flow follows a single path.

Current TE work includes OSPF-TE [6], MATE [4], and TeXCP [9]. In each case, the goal is to obtain multiple paths betwen points and determine traffic weightings for the use of each path to best balance load. An open issue is the difficulty of migrating traffic according to the changing weights, without shifting existing flows. Flowlets [15] are an attempt to migrate flows when their migration will not cause confusion in TCP end-points.

Interdomain TE At the interdomain level, TE is very primitive. A simple interdomain TE example is the attempt to balance load across a multihomed AS's multiple upstream links. Unfortunately, network operators must resort to crude techniques involving manipulating BGP path attributes [13, 5] in an effort to express their priorities across

these links. These intentions are often not easily achievable given the coarseness of these methods (*i.e.* prefixes, AS path lengths).

An alternative approach to interdomain TE is to open "virtual peering" tunneled connections between specific multi-homed peering points[12]. These connections, once agreed upon by the source and destination AS, can help to smooth out traffic imbalances for multi-homed stubs.

End-host TE An alternative approach is to place more control in the hands of end-hosts. By allowing end-hosts to select from multiple paths, the end-hosts will make self-interested choices that optimize the network behavior as a whole. For example, a flow may have four viable paths, of which two are congested. The end-host will naturally prefer to send traffic over the uncongested paths, balancing load.

IP Source Routing [1] allows end-hosts to specify the paths of their packets. However, source routing requires that end-hosts possess network topology information, and is rarely turned on in commercial network as it is perceived to represent a security threat and to offer end-hosts a mechanism to violate AS transit policies.

BANANAS [11] describes a framework for intra- and inter-domain multipath routing on the Internet by adding "PathIDs" to packets. These PathIDs specify source routes, either by fixed-length hash or variable-length link identifiers. Using this scheme, upon receiving a packet a router matches the PathID to a table of available routes, selects that route, and replaces the PathID with the route selector for the next router. This kind of explicit source routing requires the sender to have global knowledge of available paths in order to compute a PathID, which is similar to MPLS for intra-AS routing, but impractical across ASs for the same security reasons as IP source routing.

Overlays such as RON [2] and Akamai's content distribution network [17] represent another way that traffic may be redirected from the default path chosen by BGP. In these systems, the end-hosts participate in a routing mesh that allows them to determine the best path through some set of overlay routers. These systems do not take advantage of multiple routes simultaneously.

Contributions This paper introduces the concept of *tags*, which allow end systems to influence routing decisions without violating routing policy or obtaining topology information. Routers select from multiple forwarding paths based on a packet's tag and flow identifier. A host achieves path diversity, with its associated reliability/performance benefits, without resorting to specifying complete source routes. Yet tags do not force routers into any particular routing decision. Routers select from a set of paths that they have determined to be acceptable, according to local policies and routing algorithms.

Section 2 explains the tag mechanism. Section 3 describes how the tag mechanism can be used on the Internet, either natively or through overlays. Section 4 shows simulation data backing our claim that multiple paths are available, and accessible using the tags mechanism. Section 5 discusses open issues and concludes.

2 End-host Tags

Previous attempts to take advantage of multiple paths have fallen at two ends of a spectrum. At one end, source routing (whether IP or overlay) have required that end-hosts make routing decisions, and supplied end-hosts with the extra power required to enforce their decisions. At the other end, intradomain TE and hash-based load balancing schemes [3] have kept end-hosts completely oblivious to multi-path routing by forcing packets of individual flows to follow a single path from source to destination.

We propose a compromise interface based on opaque *tags* which are supplied by end-hosts to provide extra flexibility to the network during routing. Today, end hosts exert an implied demand when they transfer packets in a single flow. Networks have been designed to route a flow along a single path in order to provide stability that is useful for congestion control. Tags allow end-hosts to explicitly exert this control only when necessary. Within a flow, only packets with the same tag must be forwarded on the same path.

Tagged Forwarding A router, upon seeing a tag, hashes the connection identifier and the tag together to select between several potential routes. In order to influence the likelihood of certain paths, a larger set of hash values might be assigned to certain preferred paths. The set of potential paths is determined by the router according to local routing protocols and policies. We describe how conventional routing protocols should be extended to return multiple paths in Section 3.

We assume that intradomain routing protocols will be configured to provide multiple routes with the same (optimal) cost, and that interdomain routing protocols will be configured to provide either (1) all paths that survive local filtering policies, weighted by additional attributes, or (2) only those those paths that survive local filtering and have equivalent additional attributes, such as BGP's `local_pref` and AS hop-count length.

Figure 1 demonstrates the tagged forwarding algorithm. A router, R receives a packet from S, destined for D, and tagged t. R computes $H = hash(S, D, t)$, and looks up the routes associated with D. R's routing table favors that path through A. When H is 0, 1, 2, 5, or 7, the path through A is chosen, else the path through B. Subsequent packets from the same flow, with the same tag will be routed in the same way. Packets with another tag may be forwarded along a different path.

Preventing Cycles Depending on the routing protocol in use, tagged forwarding risks the creation of cycles. In particular, when an AS forwards to a neighbor other than the neighbor it has advertised as its next hop for a particular destination, that AS (or a subsequent AS) might eventually route the packet back to the AS that has forwarded "non-

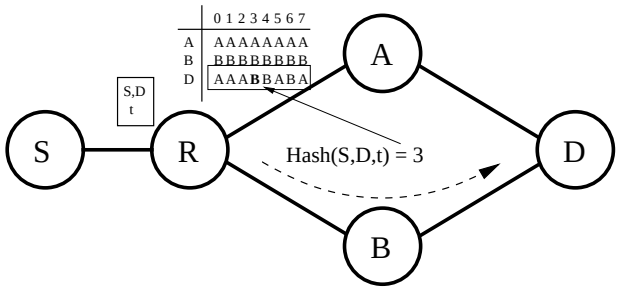

Figure 1: Router R considers forwarding a packet from S to D, tagged t. R considers the set of next-hops shown in the routing table entry for D. Since $hash(S, D, t) = 3$, the path through D is chosen.

optimally". Figure 2 demonstrates the problem in detail. Although Figure 2 is a simple example in which it is easy to prevent the cycle (don't forward packets back from where they came, for example), more complicated topologies preserve the problem without a simple soultion.

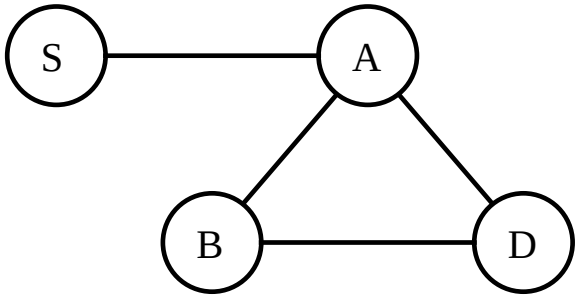

Figure 2: A potential routing cycle created by the confluence of BGP's policy mechanism and tagged forwarding. A and B have both advertised their path to D to each other. A might decide, for local policy reasons, to forward some tagged packets for D through B, similarly B may forward some tagged packets through A. If they chose the same tags, the packets will be forwarded back and forth.

To avoid the possibility of cyclic forwarding, and ensure that packets make progress, we limit the number of times that a packet may be forwarded to a next-hop other than the default. Tagged packets include a *flexcount* which reresents the number of times a packet may be routed flexibly—that is, along a non-default path. In Figure 2, if S's packets have a flexcount of one, A might forward a packet through B, but B will use default routing to forward the packet on to D.

Existing routing protocols must already ensure that the path determined by default next-hops is cycle free. Using this rule, packets may iterate around cycles only a finite number of times. In practice, we expect that a single non-default forwarding step offers sufficient flexibility to exploit most available paths. Section 4.3 confirms this expectation.

Tag selection In order to take advantage of multiple paths, end-hosts must tag the packets in their flows. Of course, if the packets of a TCP flow are arbitrarily tagged, congestion detection would be falsely triggered, just as it might be by TE that routes a single flow over several paths.

Several possibilities address this dilemma. First, there has been recent work in multipath TCPs [20, 18]. These are TCPs that explicitly take advantage of multiple paths if they are available. Fortunately, the multipath model considered in these works is exactly that provided by tags. The individual paths are not known to be distinct, but they are consistent. In effect, these TCPs perform independent congestion control along each path, with some additional features to decrease unfairness to single-path TCP implementations. These multipath TCP implementations would simply tag packets intended for each path with a unique tag.

A second possibility is end-host supplied flowlet tagging. Flowlets [15] are bursts of packets within a TCP flow. It has been proposed that intradomain TE can take advantage of the lulls between flowlets to switch a flow to another path without causing reordering at the destination. Doing so requires detecting these lulls in the network. However, the TCP sender is in the best position to detect these lulls, and could switch tags after a lull to explicitly allow a route change. Flowlet labeling does not exploit simultaneous paths for a single flow.

When tagging packets to allow the use of multiple paths, the end-host must decide on the number of unique tags to use. Each additional tag offers the possibility (but not guarentee) of another path, but each tag might require independent congestion control which may slow the flow's growth or waste resources, particularly when the new tag does not actually offer a new path.

Our analysis of an inferred AS topology [16] leads us to conclude that a small constant number of tags, perhaps 4-8, would allow sufficient flexibility to find most available paths at the AS level. Section 4.2 describes our analysis. A more careful end-host might traceroute to the destination and base the number of tags on the path length of the default path to the destination, or try several tags and discontinue the use of tags that appear to follow the same path (based on tagged traceroutes, or statistical analysis).

3 Routing

In order take advantage of the freedom offered by tagged packets, routers must obtain multiple routes to potential destinations. Most routing protocols can be easily adapted to obtain extra routes. Link-state protocols like OSPF distribute global topology information that can be searched for extra routes. Distance-vector (or path-vector) protocols provide routers with multiple advertisements from neighbors, each representing a different potential path. In both cases, care must be taken to avoid cycles.

As described in Section 2, cycles can occur when a router selects a non-default forwarding path, and that next-hop has any potential path that may return the packet to the first router.

3.1 Intradomain Routing

The most common intradomain routing protocols are OSPF and RIP. Though OSPF is a link-state protocol and RIP is distance-vector, they share the property that paths are selected by an optimality criterion (generally hop count), rather than through arbitrary policy. This property allows multipath forwarding to avoid cycles. If multipath routers select only from routes that are equally optimal, each hop decreases the distance (or cost) to the destination. Therfore no cycles are possible.

Alternatively, if an AS would like to configure some tags to follow suboptimal paths, MPLS can be used to label the packet and prevent further processing within the AS by other multipath routers. MPLS allows a single router to select an entire *path* rather than a next-hop.

In our design, the tag is a 16-bit IP option added to each packet. An IP option allows incremental deployment, since unrecognized options can simply be ignored. The performance impact of IP option processing on oblivious hosts has studied and reported as minimal [14].

3.2 Interdomain Routing

BGP cannot use either trick to avoid cycles. There is no optimality criteria, and no mechanism to choose entire paths at once. Figure 2 demonstrated how an AS that forwards on a path other than the path it has advertised can cause forwarding cycles.

To break these cycles, if a BGP router advertises a path for a given destination, then it must follow the path it has announced, or decrement a small integer, called *flexcount* that accompanies a packet's tag. Once flexcount reaches zero, the packet must be forwarded along the paths that BGP has advertised. A router is only constrained if it has advertised a route, so stubs need not decrement flexcount.

Flexcount is so named because it represents the number of times a packet may be flexibility routed along a non-default path. With a flexcount of n, a packet may be routed in a cycle at most n times before it is forwarded along the default path which is known to be cycle-free.

Flexcount may reduce the number of paths that may be followed by a packet between two points. Our experiments, described in Section 4.3, demonstrate that most paths can be found with a flexcount of one. For example, when ASes have 8 potential paths between them, a flexcount of one suffices find and average of 85% of those paths. Using a flexcount of two ensures that almost every path is available.

Although we envision tags as a mechanism that enables end-hosts to implement traffic engineering, there is room for network administrators to exert some pressure to favor certain paths. Recall that Figure 1 should how a router might prefer certain paths by overweighting the number of hash buckets containing the chosen path. We envision BGP routers that weight buckets based on a flexible combination of of local_pref, AS path length, and MED attributes associated with path advertsiements. Then choices will lay the groundwork for a reasonably distribution of traffic using random tags, and then end-hosts will overweight those tags that perform well for them.

3.3 Overlay Routing

Although we have investigated the effectiveness of deploying multipath routing in all Internet routers, and shown that multipath routing can be deployed incrementally, a more realistic deployment scenario uses an overlay network to provide multiple paths. Routing overlays such as RON [2], One-Hop Source Routing [8], and Akamai's SureRoute [17] have proven the ability of overlay networks to exploit path diversity for reliability purposes, and to respond quicker to failures than BGP.

These past results leverage the fact that an overlay node has access to more underlay characteristics than BGP, e.g. delay, loss and bandwidth history, bottleneck link capacity estimation [10], and AS/router path traceroute information. However, past overlay routing designs have primarily focused on finding a single failover route, rather than on maintaining and ranking a set of well-performing multiple paths to a destination.

We have begun work on an overlay router designed to provide multiple paths for packet-tagging transport layers such as [18]. Extending on previous monitoring and routing overlays, nodes also discover network topology information through probing, and use it to optimize multi-path routing, e.g. to prefer disjointedness in the underlay access paths. The main challenge lies in maintaining an overlay topology that is both network- and topology-aware.

4 Evaluation

The potential success of the tags interface can be evaluated in two ways. First, how many paths exist between arbitrary end-hosts on the Internet? This provides an upper bound on the path diversity that can be taken advantage of by tags. Next, what percentage of those paths can actually be found by tagged packets using a conservative flexcount? In order to avoid wasted network resources, we advocate a flexcount of one, which allows at most one cycle.

4.1 Simulation environment

To answer these questions we simulate the Internet AS graph with C-BGP [19], using a methodology similar to [12]. As our focus is solely on interdomain paths, each AS appears in our model as a single BGP router advertising a single prefix. Our AS topology, from [16], contains roughly 14,400 stub and 2400 transit ASs, and was collected from multiple views of BGP routing tables in February 2004.

The AS topology comprises more than 37,000 links, each annotated with an inferred business relationship: either *customer-provider* or *peer-to-peer*. These relationships define the selective export rules [7] in place at each AS, which control the direction of route advertisements (and thus traffic flow). These relationships also define the

`local_pref` value each BGP router assigns to routes it learns of: routes are weighted to prefer, in decreasing order, paths to customers, peers, and providers.

4.2 Available path diversity

To provide an idea of the amount of "hidden" path diversity that exists between ASs, we examine available paths between 9954 randomly-chosen pairs of stub ASs, accepting and counting only paths "just as good" as the BGP path. These are the paths with at most the same AS hop length as the BGP path. By "available", we mean each AS along the path has received a route advertisement from the next AS for the destination prefix; these routes exist in the router's Adj-RIB-In tables, but were not selected by the BGP decision process. This implies that the route is acceptable, but was beaten by another path, possibly by an arbitrary tie-breaking decision.

Additionally, we require that links traversed between ASs in a path must have the same (or better) `local_pref` value as the BGP path. This excludes unrealistic scenarios, such as preferring sending to a paid provider rather than to a peer or customer. Taken together, these constraints produce a conservative estimate of path diversity: relaxing the hop length requirement would yield many more acceptable paths. A more generous evaluation might weight longer AS paths to receive some traffic.

Figure 3 shows the cumulative distribution of the number of available paths found per AS pair. It shows that about 60% of AS pairs have at least two acceptable paths between them, while approximately 25% of pairs have at least four possible paths. Figure 4 shows how multihoming adds path diversity. Examining pairs of multihomed ASes, Figure 4 shows that among pairs of dual-homed ASes, about 65% of pairs have at least two equally good paths between them. The similarity of available paths among between single-homed and dual-homed ASes demonstrates the conservative nature of our evaluation. Of course, between two dual-homed ASes, there *must* be at least four different paths (each AS chooses one of its two homes). However, in our evaluation we assume that routers will spread tagged packets only among equally good AS paths—those with the same hop-length and `local_pref`.

4.3 Captured path diversity

The flexcount mechanism may prevent certain paths from being discovered by tagged packets. First, we consider how many of the paths from the previous experiments could be found using a flexcount of one. This means that packets are forwarded to the non-default next hop at most one time by a non-stub AS. Stub ASes may forward to a non-default next-hop with decrementing flexcount because they have not advertised their default choice to a neighboring AS. No cycles are possible. We then compare this result to a flexcount of two, where two non-default choices are possible.

Figure 5 shows, for each number of potential paths between two ASes, the number paths that a flexcount=1

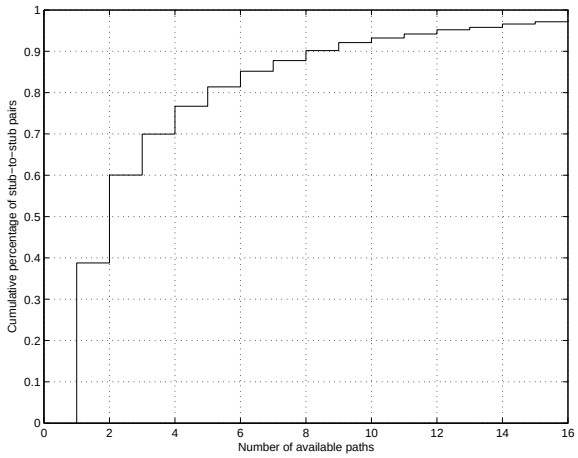

Figure 3: CDF of path diversity between 9954 random stub AS pairs. 60% of pairs have at least two paths; 10% have more than eight paths.

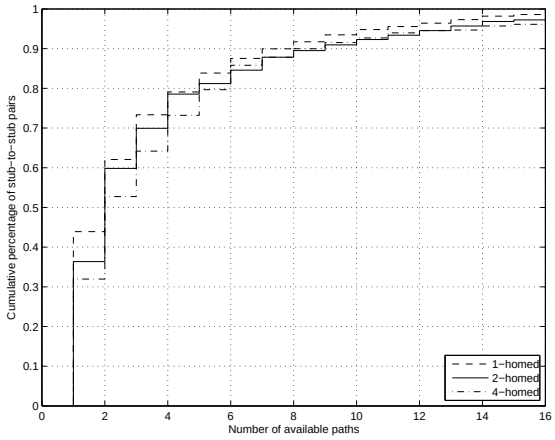

Figure 4: Available path diversity with multihoming. 65% of dual-homed pairs have at least two good paths.

packet could be forwarded along. For those AS pairs with five paths available between them, a flexcount of one was sufficient to find an average of 92% of those paths. When ten paths were available, 83% were accesible. The remaining results show that even for large numbers of paths, a flex count of one is enough to find between 70-90% of available paths.

Using a flexcount of two shows considerable flexibility; nearly all of the routes considered in our last experiment are attainable. This is because, for our experiment, we are limited to paths which are the same length of the BGP path. Since most AS paths on the Internet are typically short (our tests found average BGP path length to be about 5), two non-default choices are plenty.

5 Conclusion

Tags are a compromise between source routing interfaces that place complete control in the hands of end-hosts, and the traditional IP interface which forces the network to

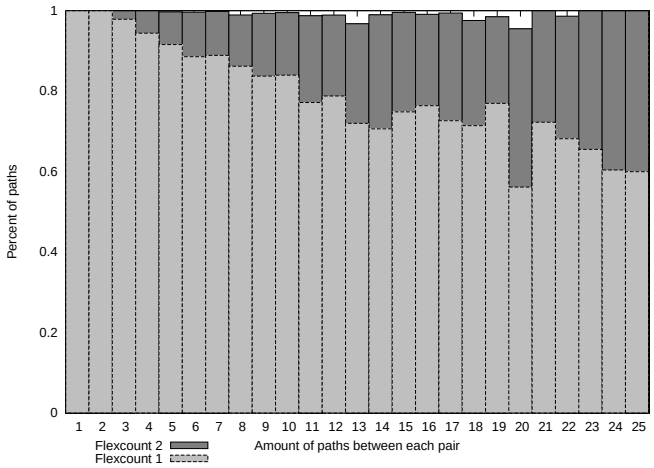

Figure 5: Captured path diversity for flexcount one and two.

make assumptions about the end-host's requirements. Using tags, end-hosts can obtain deterministic paths when needed, or let the network take advantage of mulitple paths when possible.

An analysis of an AS-level Internet topology shows that there are ample opportunities for multipath forwarding, even when BGP policies are taken into account. Furthermore, we have shown that tags are sufficient for exposing those opportunities.

References

[1] Internet Protocol. RFC 791, Internet Engineering Task Force, September 1981. `http://www.ietf.org/rfc/rfc0791.txt`.

[2] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proc. of the 18th ACM Symposium on Operating Systems Principles*, pages 131–145, October 2001.

[3] Zhiruo Cao, Zheng Wang, and Ellen W. Zegura. Performance of hashing-based schemes for internet load balancing. In *INFOCOM*, pages 332–341, 2000.

[4] Anwar Elwalid, Cheng Jin, Steven Low, and Indra Widjaja. Mate: multipath adaptive traffic engineering. *Comput. Networks*, 40(6):695–709, 2002.

[5] Nick Feamster, Jay Borkenhagen, and Jennifer Rexford. Guidelines for interdomain traffic engineering. *SIGCOMM Comput. Commun. Rev.*, 33(5):19–30, 2003.

[6] Bernard Fortz and Mikkel Thorup. Optimizing ospf weights in a changing world. *IEEE JSAC*, 2002.

[7] Lixin Gao. On inferring autonomous system relationships in the internet. *IEEE/ACM Trans. Netw.*, 9(6):733–745, 2001.

[8] Krishna P. Gummadi, Harsha V. Madhyastha, Steven D. Gribble, Henry M. Levy, and David Wetherall. Improving the reliability of internet paths with one-hop source routing. In *Proc. 6nd Symposium on Operating Systems Design and Implementat ion*, pages 183–198, San Francisco, CA, December 2004.

[9] Srikanth Kandula, Dina Katabi, Bruce Davie, and Anna Charny. Walking the Tightrope: Responsive Yet Stable Traffic Engineering. In *Proc. ACM SIGCOMM Conference*, August 2005.

[10] Sachin Katti, Dina Katabi, Charles Blake, Eddie Kohler, and Jacob Strauss. Multiq: automated detection of multiple bottleneck capacities along a path. In *IMC '04: Proceedings of the 4th ACM SIGCOMM conference on Internet measurement*, pages 245–250, New York, NY, USA, 2004. ACM Press.

[11] H. Tahilramani Kaur, S. Kalyanaraman, A. Weiss, S. Kanwar, and A. Gandhi. Bananas: an evolutionary framework for explicit and multipath routing in the internet. *SIGCOMM Comput. Commun. Rev.*, 33(4):277–288, 2003.

[12] B. Quoitin and O. Bonaventure. A cooperative approach to interdomain traffic engineering. In *1st Conference on Next Generation Internet Networks Traffic Engineering (NGI 2005)*, Rome, Italy, April 18-20th 2005.

[13] Bruno Quoitin, Steve Uhlig, Cristel Pelsser, Louis Swinnen, and Olivier Bonaventure. Interdomain traffic engineering with bgp. *IEEE Communications Magazine*, may 2003.

[14] Mattia Rossi and Michael Welzl. On the impact of ip option processing. Preprint-Reihe des Fachbereichs Mathematik - Informatik.

[15] Shan Sinha, Srikanth Kandula, and Dina Katabi. Harnessing TCPs Burstiness using Flowlet Switching. In *Proc. 3rd Workshop on Hot Topics in Networking (HotNets-III)*, November 2004.

[16] Lakshminarayanan Subramanian, Sharad Agarwal, Jennifer Rexford, and Randy H. Katz. Characterizing the internet hierarchy from multiple vantage points. In *Proc. IEEE INFOCOM*, New York, NY, June 2002.

[17] Akamai Technologies. Akarouting (sureroute). `http://www.akamai.com/`.

[18] Ronald Henry Tse. TCP fairness in multipath transport protocols, 2006.

[19] Steve Uhlig and Olivier Bonaventure. Designing bgp-based outbound traffic engineering techniques for stub ases. *SIGCOMM Comput. Commun. Rev.*, 34(5):89–106, 2004.

[20] Ming Zhang, Junwen Lai, Arvind Krishnamurthy, Larry Peterson, and Randolph Wang. A transport layer approach for improving end-to-end performance and robustness using redundant paths. In *Proc. USENIX 2004 Annual Technical Conference*, pages 99–112, June 2004.