Pathlet Routing

P. Brighten Godfrey, Scott Shenker, and Ion Stoica
{pbg,shenker,istoica}@cs.berkeley.edu
University of California, Berkeley

Abstract

Source-controlled multipath routing can be highly beneficial to both sources and to network providers. For a source, the flexibility to choose among multiple paths can improve reliability and path quality. To a network provider, source-controlled routing could be a salable service. Unfortunately, the Internet's interdomain routing protocol, BGP, offers no multipath routing mechanism. Several proposals offer multiple paths, but are limited in the paths they can expose.

This paper introduces a new scheme, pathlet routing, in which networks advertise fragments of end-to-end paths from which a source can assemble an end-to-end route. Pathlet routing is a flexible mechanism that, we show, can emulate a number of existing routing protocols, including BGP and unrestricted source routing. It also enables a new type of routing policy, local transit (LT) policies, which allows networks to control the portions of routes which transit across them, while giving a large amount of flexibility to sources. Finally, we show that LT policies have much better scalability than BGP.

1 Introduction

Multipath routing, in which a packet's source selects its path from among multiple options, would be highly beneficial to both end hosts and network providers on the Internet.

For an end host, multipath routing is a solution to two important deficiencies of the Border Gateway Protocol (BGP) [12]: poor reliability [1, 7, 9] and suboptimal path quality, in terms of metrics such as latency, throughput, or loss rate [1, 13]. Both reliability and path quality could be improved via multipath routing. End-hosts (or perhaps edge routers) can observe end-to-end failures quickly and react by switching to a different path, and can observe end-to-end path quality in real time and make decisions appropriate to each application. Greater routing flexibility may bring other benefits as well, such as enabling competition and encouraging "tussles" between different parties to be resolved within the protocol [5].

For a network provider, multipath routing represents a new service that can be sold to customers who desire the benefits described above. In fact, commercial route control products exist today which dynamically select paths based on availability, performance, and cost for multi-homed edge networks [3]; exposing more flexibility in route selection would improve the effectiveness of such products.

Unfortunately BGP has very limited *policy expressiveness*: it greatly constrains the routing policies that a network owner can encode—despite its position as the dominant policy-aware routing protocol! For example, consider a very permissive policy in which a network allows any possible route involving it to be used. Even if a network decided to adopt this policy, perhaps because it had been paid sufficiently, it could not be expressed in BGP.

Several proposals [16, 17] give networks the ability to offer multiple paths, but we argue they are still relatively limited. For example, MIRO uses BGP routes by default, with negotiation between autonomous systems for each additional path; offering too many paths thus involves a prohibitively large amount of state. NIRA [17] allows networks to offer any valley-free path, but *only* valley-free paths, thus making it in that respect more limited than BGP. It also requires assumptions about the network topology.

Can we design a protocol which has rich policy expressiveness, thus allowing network operators to offer a service of greater routing flexibility and hence greater reliability and path quality?

This paper addresses that question with a novel scheme called **pathlet routing**. In pathlet routing, networks advertise *pathlets*—fragments of end-to-end paths—along which they are willing to route. A sender concatenates its selection of pathlets into a full end-to-end source route, which is specified in each packet. Pathlet routing is a simple generalization of both path vector routing and source routing, in terms of the end-to-end paths it can allow and disallow. If each pathlet is a full end-to-end route, the scheme is equivalent to path vector routing. If the pathlets are short, one-hop fragments corresponding to links, then senders can use any of the exponentially large number of paths in the network, as in unrestricted source routing.

Pathlet routing has significant advantages over BGP, including (1) highly expressive policies, and in particular, (2) enabling a new type of routing policy which would offer dramatic improvements in router scalability and in route flexibility for senders. In more detail, we evaluate pathlet routing as follows:

- To demonstrate its policy expressiveness, we show that pathlet routing can efficiently emulate unrestricted source routing, path vector routing (BGP), and two recent multipath routing proposals, MIRO [16] and NIRA [17]. On the other hand there exist protocols like FBR [18] which pathlet routing cannot emulate.

- We show that pathlet routing enables a new class of policies, *local transit (LT) policies*, that allow networks to control the portions of routes which transit across their own networks. Subject to these restrictions, LT policies expose the full flexibility of the Internet's autonomous system (AS)-level routes to sources. The exponentially large set of paths dramatically increases route flexibility relative to BGP and many other policy-aware routing protocols.
- We argue that pathlet routing with LT policies has much better scalability than BGP and MIRO where it matters most—forwarding plane memory usage—and in other scalability metrics is comparable to or better than BGP.

The remainder of the paper proceeds as follows. In Sec. 2, we introduce our pathlet routing mechanism. We evaluate its policy expressiveness in Sec. 3 by comparison with other protocols. Finally, we introduce and evaluate LT policies in Sec. 4.

2 Pathlet routing

This section defines pathlet routing, beginning with its building blocks of vnodes and pathlets, and continuing with the control and data planes.

2.1 Building blocks

Pathlet routing is built upon virtual nodes, or **vnodes**, which are arbitrary abstract nodes created by an AS to represent the structure of routes within its own network. This virtualization enables flexible control. A vnode might variously represent an entire AS, a presence in a geographic area, a physical router, a type of service within a router, or other granularities that we will demonstrate later. There can be multiple routers for each vnode, and multiple vnodes at each router.

Vnodes need to be associated with physical routers only at peering points between ASes, where neighboring routers announce their vnodes to each other. For other vnodes, the mapping to physical routers is not exposed in the protocol.

Finally but importantly, a vnode can be tagged with a destination, such as an IP prefix.

A **pathlet** represents a sequence of vnodes along which the announcing AS x is willing to route. The first vnode should be in x's own network, but this may be followed by vnodes in x or in other ASes as the pathlet may continue outside x's network.

A pathlet announcement consists of the following information: (1) The path that packets using this pathlet will traverse, given as a sequence of vnodes. Vnode identifiers are local to each AS, so the path lists a pair $(AS, vnode)$ to globally identify each hop. (2) A forwarding identifier sequence (**FIDseq**): a sequence of forwarding identifiers (**FIDs**) to be placed in the packet to instruct the first vnode to use this pathlet.

The first entry of the FIDseq is a FID that uniquely identifies pathlet p within the first vnode in p. Importantly, it need not be globally unique like the identifiers in IP source routing, or even unique within an AS. The result is very compact FIDs; for example a vnode handling 256 or fewer unique pathlets could use one-byte FIDs. The remaining FIDs in the FIDseq, if any, identify other pathlets that are used to effect forwarding along this pathlet. (We will see examples in §3.)

2.2 Control plane

Pathlet construction. A router r announces to each neighbor r' its AS number and a vnode identifier v, indicating that every packet sent from r' to r will be interpreted as being directed to vnode v. Thus, initially, a router can construct pathlets that include its own vnodes and those of its neighbors' peering points. After learning other ASes' pathlets, it can concatenate multiple pathlets to produce new pathlets spanning multiple ASes.

Pathlet dissemination. Any pair of routers, regardless of physical location, may open a control plane connection to disseminate pathlets. Presumably this will be done *at least* by physically adjacent routers. Disseminating information in distributed systems generally can be described as either "push" or "pull", and we will find it useful to include both of these fundamental communication patterns. Intuitively, pushing state is useful at least for bootstrapping, while pulling allows additional state to be created on demand.

First, a router may **push** some subset of the pathlets it knows, according to a local export policy. For example, in several cases we will use a gossiping policy, where each router pushes to its neighbors all the pathlets it has constructed or learned. Second, a router may **pull** pathlets by requesting certain pathlets from a router, such as those relevant to a specified destination. We will use this pull dissemination pattern to emulate both MIRO and NIRA.

2.3 Data plane

Route selection. Once a router has learned a set of pathlets, it can select from among these a route for each packet. The schemes by which routers learn dynamic path quality and availability and select routes are decoupled from the pathlet routing protocol. Separating these components, as in [17, 18] but unlike BGP, gives room for a wide range of solutions to coexist, such as each network operating a route monitoring service for its users [17], commercial route selection products [3], individual sources probing paths, or a global "Internet weathermap" service.

However, it is likely useful to include a basic availability-monitoring protocol. In the rest of the paper, including the scalability evaluation in Section 4, we will assume the following. We run a global link-state protocol disseminating the state of all links between adjacent vnodes (where adjacency is defined by advertised pathlets). A router keeps an *active* *set* of pathlets, all of whose links are currently available. To find a path to a destination, it can then run any shortest-path algorithm on a graph whose edges are pathlets in the active set.

Note it would also be possible to withdraw and re-advertise pathlets in response to failure and recovery, instead of using a link state protocol. This may incur more overhead if many pathlets use a single link between vnodes.

Packet forwarding. The data structure routers use for forwarding is as follows. For each vnode at a router, there is an exact-match table that maps each valid FID f to two pieces of information: (1) the FIDseq of the pathlet corresponding to f, and (2) a *next hop rule* to send the packet to the next pathlet routing-aware hop or its destination. Examples include transferring the packet to another vnode at the same router; sending it on a certain outgoing interface; or tunneling it across an intradomain routing protocol like OSPF to the IP address of an egress point.

We now describe the forwarding procedure. A packet contains a sequence of FIDs $(f_1, f_2, \ldots, f_n)$ of the pathlets forming the route to its destination. Initially, this is set by the sender to the selected route. When a packet is sent from router r' to r, it is interpreted as arriving at a specific vnode v at r (which r declared to r' in the control plane). Router r checks for f_1 in the forwarding table for vnode v. If no entry is found, the packet is malformed and is dropped. Otherwise, the table maps f_1 to the FIDseq $(f'_1, f'_2, \ldots, f'_k)$ for the pathlet, and a next hop rule. The router verifies that the FIDseq is a prefix of the FIDs in the packet, i.e., $f_i = f'_i$ for $i \in \{1, \ldots, k\}$, to ensure that the route is legitimate for this pathlet, dropping the packet if the match fails. Otherwise, the router pops the first FID off of the route and forwards the packet according to its next hop rule.

In the next section, we will see several examples of the protocol in action.

3 Comparison with other protocols

A key goal for pathlet routing is that it enables ASes to express a broad range of routing policies. In this section, we give evidence for this policy expressiveness by showing that pathlet routing can emulate several existing protocols: unrestricted source routing (USR), BGP, MIRO, and NIRA. By "emulate" we mean that pathlet routing can match the same set of end-to-end allowed and prohibited paths. Note that these protocols have substantially different forwarding architectures, but all are special cases of pathlet routing. Finally, we compare pathlet routing and Feedback Based Routing, neither of which can emulate the other.

These sections also serve to illustrate the mechanisms of pathlet routing described in Section 2, to which end we give a fairly detailed description of pathlet routing's emulation of USR and BGP.

Unrestricted source routing (USR) at the AS level disseminates the entire topology globally, and lets a source AS use any path in the graph by putting a sequence of ASes in each packet.

At a high level, pathlet routing can emulate USR by using one pathlet for each directed link. We give a more detailed description using the AS-level topology in Fig. 1. Each AS has a vnode representing it, which for convenience we will name a, b, c, d, e, each tagged with a destination (IP prefix). Each AS discovers its neighbors' vnodes, and creates pathlets to them. We will write pathlets in the form $(P : F)$ where P is the path of vnodes and F is the FIDseq to be installed in the packet to direct it along P. Then node b, for example, creates pathlets $(b, a : 1)$, $(b, c : 2)$, and $(b, d : 3)$. Here the FIDseqs have just a single entry because each pathlet is just one hop. The FIDs themselves are arbitrary identifiers unique to each vnode. Suppose the other ASes also announce pathlets for each outgoing link and each terminal vnode, such as $(a, b : 1)$, $(c, d : 2)$. All these pathlets are gossiped globally.

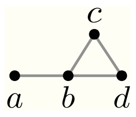

Figure 1: Example AS-level network topology.

We can now send a packet at AS a with the route $(1, 2, 2)$ to use the path (a, b, c, d). At vnode a the router looks up index 1 in its forwarding table, finding the pathlet $(a, b : 1)$ and the appropriate outgoing interface along which to forward the packet. It pops off the first FID and sends the packet with route $(2, 2)$ to node b. This process repeats until the packet reaches its destination d.

BGP. Pathlet routing can emulate BGP's routing policies using pathlets which extend all the way to a destination. We give a simple example, again using the topology of Fig. 1. Suppose that in BGP, d advertises a destination IP prefix; d exports routes only to c but all other ASes export all routes; and in the BGP decision process, all ASes select the shortest of their available routes.

We emulate this in pathlet routing as follows. To allow selective route exporting, each AS has one vnode per neighbor, e.g. d's vnodes are d_b and d_c, as well as a terminal vnode $d_\bullet$ tagged with its IP prefix(es). It creates a pathlet $(d_c, d_\bullet : 1)$, but no pathlet from d_b to $d_\bullet$: any packet arriving from b must therefore be invalid and hence will be dropped. The other pathlets created are $(c_b, d_c, d_\bullet : 7, 1)$, $(b_a, c_b, d_c, d_\bullet : 4, 7, 1)$. Here the FIDseq has multiple entries, unlike the USR example above. (Note again that the FIDs $4, 7, 1$ are arbitrary.)

A packet can now be sent from AS a to AS b with route $(4, 7, 1)$ to use the path $(b_a, c_b, d_c, d_\bullet)$. The vnode b_a looks up index 4 in its forwarding table, verifies that the associated pathlet's path is a prefix of the packet's path, and forwards the packet to c_b with route $(7, 1)$—which, in turn, forwards it to d_c with route (1), which forwards it to $d_\bullet$ with the empty route $()$, where it is delivered.

MIRO [16] is a multipath extension of BGP. In addition to using BGP's paths, a MIRO router r_1 can contact any other MIRO router r_2 and request alternate routes to a given destination d, potentially including preferences regarding which alternate routes are returned. Router r_2 responds with some subset P of the alternate routes that r_2 has available to d, from which r_1 picks one route $p \in D$. Finally, a tunnel is constructed: r_1 can send a packet along an existing path to some IP address specified by r_2, which forwards them to p.

MIRO's tunneling is easy to emulate using source routing over pathlets, by placing two pathlets in a packet's route: one representing the tunnel, and a second representing the remainder of the alternate route. To obtain the alternate-route pathlet, a pathlet router can contact any other and pull routes to a specified destination, similar to MIRO. We omit the details.

Pathlet routing can emulate MIRO; is the converse true? MIRO can represent any possible end-to-end path with the appropriate tunnels. But each allowed end-to-end route is constructed and represented explicitly, so there are network topologies for which MIRO would require $\Theta(n!)$ state to represent all possible routes in a graph of n nodes. Thus MIRO cannot scalably emulate pathlet routing (or USR).

NIRA [17] offers more choice to sources than BGP, while simultaneously reducing control plane state. NIRA supports routing along so-called *valley-free* paths, which consist of an "up" portion along provider links, then potentially a peering link, and finally a "down" portion along customer links. Each AS learns all of its "up" routes and publishes them at a well-known Name-to-Route Lookup Service (NRLS). A source constructs the first half of a path by choosing a route from its own up-graph, and the second half from the reverse of a route in the destination's up-graph, which it obtains by querying the NRLS.

Pathlet routing can emulate NIRA's routing policy, including its compact routing state. We again use the "pull" mode of obtaining pathlets in place of the NRLS. We concatenate appropriately constructed pathlets representing the up route, a short route across the core, and the down route. We omit details due to space constraints.

MIRO and BGP cannot scalably emulate NIRA because NIRA can compactly represent a very large number of paths by allowing any up-route to be paired with any down-route. On the other hand, NIRA cannot emulate USR, MIRO, BGP, or pathlet routing since it is limited to valley-free routes.

Feedback Based Routing (FBR) [18] is an example of a protocol which is incomparable with pathlet routing in the sense that neither protocol can emulate the other. FBR is source routing at the AS level, with each link tagged with an access control rule, which either whitelists or blacklists a packet based on its source or destination IP address. Pathlet routing cannot emulate FBR for two reasons. First, a pathlet router can decide whether to accept a packet based only on its immediately previous hop and on its remaining hops—not based on the full end-to-end path including the source. Second, FBR has both blacklisting and whitelisting, while pathlet routing effectively has only whitelisting, meaning FBR can represent some policies more efficiently.

However, FBR cannot emulate pathlet routing, either. For example, controlling access based on source and destination address ignores intermediate hops which can be taken into account by pathlet routing.

Zhu et al [18] suggested that the access control rules could be more complex than source/destination address matching. Similarly, it is possible that pathlet routing could be extended to include matches on the full end-to-end path and blacklisting; this may be an interesting area of future research.

4 LOCAL TRANSIT POLICIES

In the previous section, we saw that pathlet routing can efficiently express a wide variety of routing policies, emulating a number of past schemes. In this section we discuss a new class of policies enabled by pathlet routing, **local transit (LT) policies**. LT policies allow networks to control what is arguably the most important aspect of routing policy: the portions of routes which transit across their own networks.

We first define LT policies (§4.1) and argue that LT policies are useful (§4.2) and offer a large amount of route flexibility to sources (§4.3). We then show how LT policies can be implemented in pathlet routing (§4.4) and that this implementation has much better scalability than BGP (§4.5).

4.1 Definition

We define **local transit policies** as those policies in which a network x's willingness to carry traffic following some path across its network depends only on the portion of the path that crosses x's network. In other words, under an LT policy the permissibility and cost of some path, according to x, is a function only of the ingress and egress point of the path in x. Note that LT policies are independent of x's preferences on the paths taken by traffic that x sends, which may be arbitrary.

Consider Fig. 1. If AS b follows an LT policy and allows the path (a,b,c), then it must also allow the path (a,b,c,d), but possibly not (a,b,d) or (c,b,a) which have different ingress or egress points. Essentially, in LT policies, pathlets do not extend beyond a network's ingress/egress points.

4.2 Capturing policies and costs

We argue that LT policies represent an important class of routing policy, with two points. First, the *direct* costs to a network of carrying traffic is incurred between its ingress and egress points; for example, the path a packet follows before it arrives at an AS x and after it leaves x do not affect the congestion on x's network. Second, a special case of LT policies—namely valley-free routes [6]—is a common route export policy in BGP today. Valley-free routes can be defined as follows: each neighbor is labeled as a customer, provider, or peer; a BGP route is exported to a neighbor x if and only if either the next hop of the route is a customer or x is customer. This is a function of the ingress and egress point, and hence is an LT policy.

Of course, valley-freeness defines which routes are allowed, but not which ones are preferred. In BGP, this is handled by picking the single most preferred route for each destination as the only route. This brings us to a key challenge: providing the incentive for a transit AS to offer more than the single end-to-end path which is most convenient for that AS.

We envision two ways this incentive could be provided. The first is simply payment between ASes for a multipath routing service which does not discriminate the cost of each path—much as today's transit services have a fixed rate regardless of a packet's destination.

Second, a more discriminatory service could differentiate prices based on the path used. In pathlet routing, it would be easy to annotate each pathlet with a cost. This does not effect a monetary payment, but it does permit the communication of prices to sources, so that payment can happen via some mechanism external to the protocol. Designing such a payment mechanism is outside the scope of this paper. However, note that measurements indicate that ASes' routing preferences are based on next-hops (i.e., egress point) for 98% of IP prefixes [15]. Thus, once a payment mechanism is in place, it would be possible to represent those preferences as LT policies.

It is also possible that ASes would be reluctant to use LT policies because their policies become more explicitly visible. But high level routing policies such as business relationships between neighbors can be inferred from publicly available data from BGP routers today. [14]

4.3 Enabling route flexibility

LT policies can provide a dramatic amount of flexibility for sources—potentially, any AS-level path—because the networks' pathlets can be concatenated in an exponentially large number of ways. But this flexibility is limited by the specific LT policies that are chosen.

For example, networks could limit routes to being valley-free. In this case there would still be vastly more flexibility than BGP, but no more than in NIRA. Unlike NIRA, pathlet routing would not be limited to Internet-like topologies. Also, handling exceptions to policies is more difficult in NIRA: a special source routing option is required in the data plane and in the control plane, no mechanism for discovering non-valley-free routes is provided. In pathlet routing, if for example an AS wished to provide transit service between two peers or two providers, this would simply involve advertising two additional pathlets (one in each direction).

However, much more flexibility than valley-free routes may be available. Some incentive for this flexibility exists: for example, a path which is not the cheapest may be worthwhile for real-time applications, or when cheaper paths have failed. Whether ASes choose to expose these routes depends on business decisions and payment mechanisms, but pathlet routing makes it feasible at a technical level.

4.4 Implementation

LT policies are easy to implement in pathlet routing: for each ingress x and egress y for which an AS allows routing from x to y, it announces a pathlet (x,y), in addition to pathlets

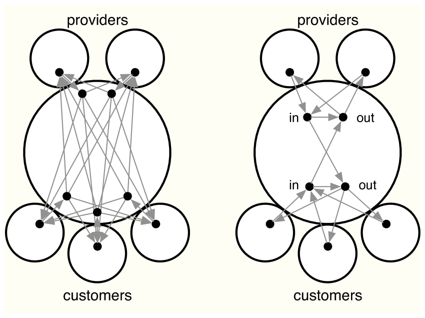

Figure 2: vnodes and pathlets for a full LT policy (left) and a class-based LT policy (right) in a single AS with two providers and three customers.

that terminate at its own IP prefixes. However, this results in d^2 pathlets for a network with d neighbors. Thus, for large networks, it may be more appropriate to use what we call *class-based LT policies*, in which each neighbor is assigned to a class (such as a geographical region, or business relationship) represented by ingress and egress vnodes, and we use full LT policies between only these class vnodes. These two options are depicted in Fig. 2.

To the best of our knowledge, other policy-aware routing proposals cannot efficiently implement the same set of possible paths that is exposed by LT policies. BGP cannot represent multiple paths per neighbor; MIRO must set up each end-to-end path explicitly, resulting in exponentially more state; NIRA represents only valley-free routes; and FBR examines a packet's source and destination IP address, which does not include information about the intermediate hops, such as a transit from peer to customer vs. peer to provider.

4.5 Scalability

We evaluate the scalability of LT policies, which is similar to that of unrestricted source routing. It has been suggested [16] that source routing schemes may not scale since each router must have current knowledge of the entire network. On the contrary, we argue that pathlet routing with LT policies in fact has much better scalability than BGP (and, hence, MIRO [16]) where it matters most—forwarding plane memory usage—and in other scalability metrics is comparable to or better than BGP.

In this evaluation, we assume class-based LT policies with three classes representing the customer, provider, and peer business relationships. (Our conclusions would be substantially similar with full LT policies on all ASes except for the very high degree (top 1%) ASes, or with other class-based LT policies with a limited number of classes.) Following the pattern in Fig. 2 which depicts two classes, using three classes results in $6+d$ pathlets created by each AS with d neighbors, plus 3 pathlets to link each class to a destination vnode with associated IP prefixes. In fact, this may overestimate the number of pathlets, e.g. if the provider→provider pathlet is omitted in order to disallow transit between providers.

We produce numerical results by analyzing an AS-level topology of the Internet generated by CAIDA [4] and data from APNIC on global IP prefix allocation [2], both from August 18, 2008.

Forwarding plane memory. Because it has to operate at high speeds and often uses SRAM rather than commodity DRAM, memory that stores a router's Forwarding Information Base (FIB) is arguably more constrained and expensive than other resources in a router [10]. LT policies dramatically reduce the necessary size of the FIB relative to BGP. Using class-based LT policies as described above in this topology results in a maximum of $2,317$ and a mean of only 6.1 pathlets to represent an AS. This is also an upper bound on the number of pathlets per router assuming at least one router per AS. In comparison, BGP FIBs would need to store entries for $266,073$ IP prefixes.

Control plane memory. In the AS-level measured topology, there are a total of $157,454$ pathlets; tagging vnodes with IP prefixes brings the total to $423,527$ entries. In comparison, the RIB in a BGP router with d neighbors advertising a route to every prefix would contain $266,073 \cdot d$ entries, which is already worse than pathlet routing for $d \geq 2$ and can become problematic in practice for larger d [8].

Control plane messaging. Here we employ simple analysis. Consider the effect of a single link failure in a AS-level topology of n nodes, mean degree d, and mean path length ℓ. In pathlet routing, a standard gossiping protocol (§2.3) results in a link state update being sent once along every edge in the graph, i.e., $dn/2$ messages.

In BGP, the number of updates is *at least* the number N of source-destination pairs that were using the failed link. Suppose a random link fails, and let ℓ_{st} be the length of a path $s \rightarrow t$. Then we have

$$\begin{aligned} E[N] &= \sum_{s,t} \Pr[\text{failed link } \in \text{ path } s \rightarrow t] \\ &= \sum_{s,t} \frac{\ell_{st}}{nd/2} \\ &= \frac{2(n-1)\ell}{d}. \end{aligned}$$

In the Internet AS-level topology, we roughly have $\ell \approx d \approx 4$, making the messaging cost for both protocols close to $2n$. Moreover, BGP's messaging cost would likely be substantially higher than this lower bound for two reasons. First, because BGP operates at the IP prefix level instead of the AS level, it has in effect about $9n$ destinations [2] rather than n. Second, BGP's path exploration can result in much more than one message per source-destination pair with a failed link. [11]

References

[1] David G. Andersen, Hari Balakrishnan, M. Frans Kaashoek, and Robert Morris. Resilient overlay networks. In *Proc. 18th ACM SOSP*, October 2001.

[2] Routing table report. http://thyme.apnic.net/ap-data/2008/08/18/0400/mail-global.

[3] Avaya. Converged network analyzer. http://www.avaya.com/master-usa/en-us/resource/assets/whitepapers/ef-lb2687.pdf.

[4] CAIDA AS ranking. http://as-rank.caida.org/.

[5] David Clark, John Wroclawski, Karen Sollins, and Robert Braden. Tussle in cyberspace: defining tomorrow's Internet. In *SIGCOMM*, 2002.

[6] L. Gao and J. Rexford. Stable Internet routing without global coordination. *IEEE/ACM Transactions on Networking*, 9(6):681–692, December 2001.

[7] 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. OSDI*, 2004.

[8] Elliott Karpilovsky and Jennifer Rexford. Using forgetful routing to control BGP table size. In *CoNEXT*, 2006.

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

[10] D. Meyer, L. Zhang, and K. Fall. Report from the iab workshop on routing and addressing. In *RFC2439*, September 2007.

[11] Ricardo Oliveira, Beichuan Zhang, Dan Pei, Rafit Izhak-Ratzin, and Lixia Zhang. Quantifying path exploration in the Internet. In *Proc. Internet Measurement Conference*, October 2006.

[12] Y. Rekhter, T. Li, and S. Hares. A border gateway protocol 4 (bgp-4). In *RFC4271*, January 2006.

[13] Stefan Savage, Thomas Anderson, Amit Aggarwal, David Becker, Neal Cardwell, Andy Collins, Eric Hoffman, John Snell, Amin Vahdat, Geoff Voelker, and John Zahorjan. Detour: Informed Internet routing and transport. In *IEEE Micro*, January 1999.

[14] L. Subramanian, S. Agarwal, J. Rexford, and R. Katz. Characterizing the internet hierarchy from multiple vantage points. In *IEEE INFOCOM*, 2002.

[15] F. Wang and L. Gao. On inferring and characterizing internet routing policies. In *IMC*, 2003.

[16] Wen Xu and Jennifer Rexford. MIRO: Multi-path Interdomain ROuting. In *SIGCOMM*, 2006.

[17] Xiaowei Yang, David Clark, and Arthur Berger. NIRA: a new inter-domain routing architecture. *IEEE/ACM Transactions on Networking*, 15(4):775–788, 2007.

[18] Dapeng Zhu, Mark Gritter, and David Cheriton. Feedback based routing. *Computer Communication Review (CCR)*, 33(1):71–76, 2003.