Towards Internet-wide Multipath Routing

Jiayue He and Jennifer Rexford
Princeton University, USA
Email: {jhe, jrex}@princeton.edu

***Abstract*— The Internet would be more efficient and robust if routers could flexibly divide traffic over multiple paths. Often, having one or two extra paths is enough for significant gains in security, performance, and reliability. However, support for Internet-wide multipath routing faces two significant barriers. First, multipath routing could impose significant computational and storage overheads in a network the size of the Internet. Second, the independent networks that comprise the Internet will not relinquish control over the flow of traffic without appropriate incentives. In this paper, we survey flexible multipath routing techniques which are both scalable and incentive compatible. Techniques covered include: multihoming, tagging, tunneling, and extensions to existing Internet routing protocols.**

Keywords: Internet, multipath routing, multihoming, tunneling, scalability.

I. Introduction

Most currently deployed routing protocols select only a single path for the traffic between each source-destination pair. In this paper, we explore techniques that allow a *flexible* division of traffic over multiple paths. That is, we argue that an end host or edge network should have access to multiple paths through the Internet, and direct control over which traffic traverses each path. Application requirements dictate the granularity of division, *e.g.*, by IP address blocks (i.e., IP prefixes), destination host, a Transmission Control Protocol (TCP) flow, or a single packet.

A. Motivation for Flexible Multipath Routing

Flexible Internet-wide multipath routing would offer many benefits, including the following:

- *Customizing to application performance requirements:* Different applications have different needs. If multiple paths exist, VoIP and online-gaming traffic can use a low-delay path, while file-sharing traffic uses a high-throughput path. In addition, an application can access more bandwidth by using multiple paths simultaneously.
- *Improving end-to-end reliability:* If multiple paths exist, traffic can switch quickly to an alternate path when a link or router fails. Similarly, if an adversary drops packets along a path, the traffic could be moved to an alternate path to circumvent the adversary [1]. This is particularly useful if disjoint paths are available.
- *Avoiding congested paths:* When multiple paths are available, traffic can move to an alternate path to circumvent congestion. Despite problems with routing oscillation in the early ARPANET, recent work has shown how to dynamically split traffic over multiple paths in a stable fashion [2]. In fact, by just having two paths and flexible splitting between them, protocols can be easily tuned to efficiently utilize network resources.

Past work indicates that the Internet's network-layer topology has significant underlying path diversity.[1] Each network is a collection of routers and links under the control of one entity, such as an *Internet Service Provider* (ISP) that offers connectivity to other networks or a *stub network* that just provides connectivity to its own users and services. In this paper, we explore how to give stub networks greater end-to-end path diversity. Extra end-to-end paths may arise because a stub network is connected to multiple ISPs, individual ISPs have intradomain path diversity, or ISPs connect to each other in multiple locations. In fact, a measurement study of a large ISP found that almost 90% of Point-of-Presence (PoP) pairs have at least four link-disjoint paths between them [4]. Another study showed that, although Internet traffic traverses a single path, 30% to 80% of the time, an alternate path with lower loss or smaller delay exists [3].

B. Challenges: Scalability and Incentives

Unfortunately much of the existing path diversity in today's Internet is never exploited. The scalability challenges of multipath routing is one of the reasons. Multipath routing would introduce extra overhead in both the *control plane* and *data plane* of the routers. In the control plane, routers exchange information and compute the forwarding tables that the data plane then uses to direct incoming packets to outgoing links. Multipath routing would increase the overhead in both the control and data planes:

- *Control-plane overhead:* First, exchanging the extra topology or path information required for multipath routing would consume extra bandwidth and processing resources. Second, storage overhead at each router would grow with the number of paths. Third, computing multiple paths would require more computational power.
- *Data-plane overhead:* Forwarding traffic on different paths requires the data packets to carry an extra header or label. In addition, forwarding tables need extra entries for each destination, thus consuming more memory; in addition, this data-plane memory is expensive, due to the need to forward packets at high speed.

The ultimate flexibility would be for sources to see the entire Internet-wide topology and utilize *any* path to each destination. This would create a large scaling problem, however, since

[1] In this paper we focus on the path diversity at the network layer. There is a large body of research on path diversity at the physical layer which is important for reliability and security, but the shared risks at the physical layer are not visible to the IP routing system.

the Internet has more than 25,000 networks and many more paths. Even if the scalability challenges were surmountable, accessing *all* paths would require many (or even all) networks to cooperate, which may be unrealistic. Instead, it is more likely for ISPs to allow other networks to select from a small set of paths, under a specific business agreement. Since business models in the Internet today are *bilateral*, multipath solutions based on cooperation between pairs of networks are much more likely to succeed than solutions that require widespread cooperation between many (sometimes competing) networks. Fortunately, multipath routing solutions that limit the number of additional paths and the coordination between different networks are aligned with both goals—scalability and business incentives. As such, in this paper, we focus on solutions where stub networks select amongst a small set of paths provided by a limited number of bilateral agreements, rather than techniques that require a stub network to compute and signal a complete, end-to-end path.

This survey focuses on multipath routing schemes with low overhead and minimal cooperation between networks. The sections progress from deployed techniques to proposed solutions that are easily deployable, to techniques that rely on new business models. We start by reviewing how Internet routing works today in Section II, with an eye towards the limitations of the existing routing system. End-to-end multipath routing relies on two key capabilities: discovering extra end-to-end paths and directing packets over them. Section III covers a range of solutions for flexible forwarding such as tunneling and tagging, for directing packets to different paths, while Sections IV and V describe control-plane extensions that enable networks to learn additional paths. In particular, Section IV discusses techniques for a single network to achieve multipath routing, without requiring cooperation from other networks. The impact of a single network on end-to-end path performance is limited, however, and more end-to-end paths would be available if networks cooperated. Section V discusses techniques which only require cooperation between a pair of networks. Finally, we conclude in Section VI.

II. Internet Routing Today

In this section, we introduce the key routing protocols used in today's Internet. Routers use the Border Gateway Protocol (BGP) to exchange reachability information with neighboring networks. BGP is a *path-vector* protocol, where routing decisions are made based on local policies. Inside a network, routers communicate using an Interior Gateway Protocol (IGP). Most ISPs run *link-state* protocols that perform shortest-path routing based on configurable link weights. The link weights in IGP and the policies in BGP are configured by human operators to satisfy business objectives.

A. Interdomain: Path-Vector Protocol and Multihoming

Figure 1a represents a network-level topology, where each cloud is a network and each link represents a physical connection, as well as the existence of a business relationship between two networks. In a path-vector protocol, the *entire routing path* is exchanged between neighbors. Edge routers in each network learn multiple paths to reach a particular destination and store all of them in a routing table. From the list of paths, a router then applies a set of policies to select a *single active route*. A router optionally advertises the active route to each neighboring network, depending on the business relationship. Using a path-vector protocol allows BGP to support flexible local policies that give each network control over its incoming and outgoing traffic. For example, a stub network, like network D in Figure 1a, would not advertise routes learned from B to C (and vice versa) because D does not wish to carry transit traffic between the two neighbors.

Today's BGP has two limitations as a single-path protocol. First, since only the active path is advertised, customer networks are prevented from seeing alternate paths, including ones they might prefer. Second, by using only the active path, a network does not have fine-grained control, and can only balance traffic over multiple paths at the IP address block (i.e., prefix) level. Extending BGP to a multipath protocol, however, requires alignment of economic incentives between networks. The economic incentives are likely to grow stronger in the future as the demand for performance and robustness increase, and customers are willing to pay for value-added Internet services. Today, two networks usually have a customer-provider relationship or a peering relationship. In a peering relationship, two networks could mutually provide additional paths to each other without any economic exchange, similar to how they carry traffic on peering links for free today. In a customer-provider relationship, the provider could offer additional paths to its customers as a value-added service.

One such example is multihoming, where a stub network pays to connect to more than one ISP. The use of multihoming has seen a dramatic increase in recent years for two main reasons. First, as more enterprises rely heavily on the Internet for their business transactions, having a second provider is important to survive a failure of the other provider. Second, multihoming can be used to drive down the cost of Internet access. For example, the multihomed network can use a cheap ISP for most traffic and an expensive but better ISP for performance-sensitive traffic. In Figure 1a, network D is multihomed to networks B and C. Despite having two upstream routes, network D can only balance load between the two at the prefix level, and only forwards traffic for each destination on a single path. So, while multi-homing provides additional paths to stub networks, fine-grained control remains elusive.

B. Intra-domain: Link-State Protocol

Unlike the interdomain case, each network has full control of its internal network. In addition, a network typically has just tens or hundreds of routers, much fewer than the 25,000 networks in the Internet. Inside a single network, each router is configured with a static integer weight on each of its outgoing links, as shown in Figure 1b. The routers flood the link weights throughout the network and compute shortest paths as the sum of the weights using Dijkstra's algorithm. Each router uses this information to construct a table that drives the forwarding of each IP packet to the next hop in its path to the destination.

Link-state protocols offer several advantages. First, routing is based only on a single link metric, *i.e.* link weights. Second, to reduce message-passing overhead, routers only disseminate information when the topology changes. Finally, by flooding the link-weight information, each router has a complete view of the topology and associated link weights.

On the other hand, even though each router can see the whole topology, the existing path diversity is under-exploited [4]. Even when alternative paths have been computed, packets towards a destination are often forwarded on a single path. *Equal-cost multipath* is a commonly deployed technique where the routers keep track of all *shortest* paths, and then evenly split amongst them. In Figure 1b, we see that router i has two shortest paths to reach router j. In today's IGPs, the traffic would be divided evenly between the two paths. Even this limited version of multipath routing is useful for fast reaction to failures. In fact, some operators tune the link weights to create equal-cost multipaths [5].

Multiple shortest paths enable the operator to balance load and react quickly to failures, but does not enable the operator customize paths for different applications. An existing option for operators to customize paths inside their own network is the Constrained Shortest Path First (CSPF) protocol, an extension of the shortest-path protocol. The path computed using CSPF is a shortest path fulfilling a set of constraints. A constraint could be minimum bandwidth required per link, end-to-end delay or maximum number of links traversed. CSPF can be useful for a range of applications, *e.g.*, picking a low-delay path for a VoIP call, but cannot pick paths based on dynamic constraints such as packet loss.

III. Towards Flexible Forwarding

The most prevalent forwarding mechanism in the Internet today is *destination-based hop-by-hop forwarding*. Each router forwards a packet to an outgoing link based on the destination address from the IP packet header and the corresponding longest-prefix match entry in the forwarding table. For example, in Figure 1b, a router will forward a packet destined for j, independent of where the packet came from. Destination-based hop-by-hop forwarding leads to small forwarding tables, but cannot realize flexible forwarding policies. For example, in Figure 1a, if network A wanted to reach D via (B, C), but B wanted to reach D directly, then A is forced to use path (A, B, D). Even when the forwarding table contains multiple next hops for the same destination, common practice would divide the traffic evenly amongst the multiple paths.

In this section, we describe alternative schemes which forward traffic over multiple paths. This is useful for customizing paths for different applications. In order to decide which path should carry a packet, an edge router or end host need to first classify a packet, and then map the packet to a corresponding path:

- *Packet classification:* Packets can be classified based on the requirements of the application. application may want low delay, high throughput, or a secure path. The application could be defined by a prefix, a destination, or a TCP flow (source and destination addresses and port numbers). Packets within the same flow are normally classified in the same way. One option is to mark the Type of Service (ToS) bits in the IP header, and later forward the packet using the same bits.
- *Mapping packets to paths:* The edge routers can measure (or infer) path properties, to determine which path is best-suited to each class of traffic. By examining the packet header, a packet can be mapped to an appropriate path. Designing a measurement infrastructure to monitor path performance is challenging. One reason is that measurements of path performance can be inherently inaccurate; for example, round-trip time estimation is a classic challenge. In addition, the inaccuracies can be even greater in a competitive environment where other networks may treat probing packets differently than data packets to make paths look more attractive than they are.

Both steps incur extra data-plane overhead. Though the overhead of marking packets and processing the marked packets is minimal, the measurement overhead associated with monitoring path performance can be significant, particularly if the measurements are fine-grained (e.g., the the destination prefix level).

If multiple paths are associated with a particular class of traffic, the router can send a fraction of the packets on each path, to balance load and circumvent congestion. In Section III-A, we survey existing techniques for forwarding packets on alternate paths. In Section III-B, we discuss the pros and cons of splitting traffic at different granularities. We focus on existing techniques (round-robin, hashing, and flow-cache), but also describe flowlet-cache, a promising technique that is yet to be deployed.

A. Forwarding on Alternate Paths

Tunneling is a widely available alternative to destination-based hop-by-hop forwarding that offers much more flexibility. At a high level, tunneling establishes a *logical* link between two routers (or hosts). Forwarding packets over a tunnel usually involves "pushing" a header (or label) at the tunnel ingress and "popping" the header (or label) at the tunnel egress, in a process called *encapsulation*. For example, in Figure 2, a packet going from B to F could be encapsulated to ensure it travels through E. At B, an extra header would be "pushed" on the packet to indicate E is the destination. Once the packet reaches router E, the extra header would be "popped" from the packet, then E would forward the packet to F hop-by-hop. Encapsulation can be implemented through IP-in-IP tunnels or MultiProtocol Label Switching (MPLS). MPLS is a label-based forwarding mechanism that encapsulates packets using labels. In either case, encapsulation requires packets to carry an extra label or an extra IP header. In the case of MPLS, each router also stores the label-based forwarding table, although a label-based look-up is simpler than matching the longest prefix of the destination address.

The path between the tunnel ingress to tunnel egress can depend on the underlying routing protocol, or the entire path can be specified explicitly. Encapsulation alone is often sufficient for most application needs such as directing a packet

to a particular egress point or through a particular network. When the path between tunnel endpoints only depends on the underlying protocol, the path adapts automatically when the topology changes. For example, in Figure 2, B could forward the packet towards E one hop at a time. This implies if the link from C to D fails, the encapsulated packets would *transparently* switch to another path. Still, by only specifying the endpoints of the tunnel, it is difficult to satisfy certain applications needs, *e.g.*, an end-to-end bandwidth requirement. So for those specialized applications, explicit routing is a useful alternative.

Explicit routing specifies every router (or network) along the path. The routers (or networks) along the path can be specified directly in the packet header or indirectly through a label in the packet header. One possibility is to implement explicit routing by specifying the whole router-level path with IP options. In Figure 2, if the path sequence (A, B, C, D, E, F) is an explicit path for certain packets traveling from A to F, then A would know to forward to router B based on the IP options in the packet header. An alternative is to implement explicit routing with MPLS as a combination of Constrained Shortest Path First (CSPF) and Resource Reservation Protocol (RSVP). CSPF selects the path using a variety of metrics, while RSVP is the signaling protocol used to set-up the path within a single network. RSVP establishes a hop-by-hop chain of labels to represent the path and it reserves bandwidth along the path by signaling in advance. At source end of the path, a label would be pushed onto the packet based on information from the packet header such as source address, destination address, and port numbers. Each intermediate router would do a label look-up to find the outgoing label and outgoing link. Compared to tunneling, explicit routing does impose more data-plane overhead (to swap the labels at each hop), though the overhead is manageable when the number of explicitly-routed paths is limited.

B. Flexible Splitting Amongst Multiple Paths

The network management system may wish to balance traffic between multiple paths to achieve certain traffic engineering objectives. For example, sending 40% of traffic on one path and 60% on another could lead to less congestion in the network. To achieve a splitting percentage determined by the network management system, traffic can be switched onto different paths using four major techniques: round-robin, hashing, flow cache, and flowlet cache [6]. Each technique strikes a different trade-off between overhead, splitting percentage accuracy, and the likelihood of packet reordering.

A weighted **round-robin** will switch traffic at the granularity of packets. Since packets are small in size, round-robin scheduling can achieve very accurate splitting percentages on a small timescale. Round-robin scheduling also adds very little extra overhead on today's forwarding functions. The downside is that since different paths between the same source-destination pair often have different delays, some packets which belong to the same TCP flow could arrive out-of-order. This is problematic as TCP considers out-of-order packet delivery as a sign of network congestion, and consequently, the TCP sender would slow down the transfer. If the paths have very similar delay, then weighted round-robin is a good choice due to its low overhead and accurate splitting percentages.

Hashing involves first dividing the hash space into weighted partitions corresponding to the outbound paths. Then packets are hashed based on their header information and forwarded on the corresponding path. A flow is defined by the following attributes in the packet header: source IP address, destination IP address, transport protocol, source port, and destination port. Hashing ensures in-order delivery of most packets since a flow is likely to be mapped to a specific path for its entire duration. On the other hand, since flows vary drastically in their sizes and rates, it is difficult to realize accurate splitting percentages. Finally, if splitting percentages change or a path fails, a flow is likely to be hashed onto a different path, possibly causing a few out-of-order packets during the transition.

The best way to avoid out-of-order packets is to implement a **flow cache**. A flow cache is a forwarding table that keeps track which path each active flow traverses. A flow cache ensures packets belonging to the same flow always follow the same path. Another advantage of flow caching over hashing is that when new flows arrive, they can be placed on any path, which leads to better control of dynamic splitting percentages, although the splitting percentages achieved are less accurate than in round-robin scheduling. The major drawback is that a high-speed link could easily carry tens of thousands concurrent flows [6], leading the flow cache to consume a significant amount of additional memory in the router.

It is possible to reduce data-plane overhead and improve splitting ratios by dividing traffic at the granularity of packet-bursts, using a **flowlet cache** [6]. If the time between two successive packets is larger than the maximum delay difference between the multiple paths, the second packet can be safely forwarded on any available path without the risk of packet reordering. A flowlet cache is typically much smaller than a flow cache, since there are significantly fewer active packet bursts than active flows [6]. In addition, flowlet switching always achieves within a few percent of the desired splitting percentage, without reordering any packets. Overall, flowlet cache would be the best choice for most applications, although it is not yet implemented in routers today.

IV. Multipath Routing by a Single Network

In this section, we present incrementally deployable techniques which can be adopted by a single network. Each ISP can exploit its internal path diversity, and a multihomed stub network can split traffic over multiple end-to-end paths.

A. Intradomain: Non-shortest Paths within an ISP

Each network can select its own IGP, allowing it to change the protocol without requiring cooperation from others. In link-state protocols, since link weights and topology information are already flooded to all routers, multipath routing does not incur extra dissemination overhead. One natural way to extend a link-state protocol is to compute the K-shortest paths rather than just the shortest path. This is cumbersome for several

reasons. To start with, computing the K-shortest paths is more computationally intensive (i.e., $O(N \log N + KN)$ for a network with N routers) than computing a single shortest path (i.e., $O(N \log N)$). The forwarding-table size would also grow with the increase in number of paths per destination. Perhaps the biggest overhead increase is in the data plane, where K tunnels need to be established between each source-destination pair. If each router does destination-based hop-by-hop forwarding, then there is no guarantee packets would travel on the K-shortest paths from source to destination. This is significantly more cumbersome than the current hop-by-hop forwarding.

Another approach is to run multiple instances of the link-state routing protocol [10]. Instead of having a *single* weight associated with each link, each link has a *vector* of weights. Each instance of the link-state protocol can just compute the shortest path and create a forwarding table for the corresponding topology. The vector of weights does not lead to the K shortest paths, but rather a shortest path for each of K sets of link weights. Each set of link weights can be tuned independently to customize the paths to different applications; for example, one set of weights could be tuned for high throughput and another for low delay. The link weights could even be specialized to handle different failure scenarios. In the control plane, if K routing instances run simultaneously, the control-plane overhead would be exactly K times as much as shortest-path routing. In the data plane, there are two ways to forward packets on the multiple topologies. The simpler (and more restrictive) way is for each packet to belong to a single topology [10]. Further benefits are possible when packets can switch between topologies based on network conditions [10].

An alternate approach to multipath routing is to forward traffic on *all* paths that make forward progress toward the destination [8], [9], based on a single set of link weights. Each router can make local forwarding decisions based on the cost of the shortest path through each of its neighbors [9]. Forwarding packets only to routers that have a shorter path to the destination guarantees that the path is loop-free [8]. To encourage the use of shorter paths, diminishing proportions of the traffic would be directed on the longer paths. For example, in Figure 1b, i has two outgoing links along shorter paths to j. Since these paths have costs 8 and 9, less traffic would be placed on the path with cost 9 [9]. Under this scheme, the path-computation costs are still $O(N \log N)$, since each router will just run Dijkstra's shortest-path algorithm. Compared to shortest-path routing, forwarding along the "downward" paths requires more entries in the forwarding tables. In addition, there will be slightly more data-plane overhead in order to implement the splitting percentages, as explained in Section III-B. Still, each router can make local forwarding decisions without the use of tunnels.

B. Interdomain: Fine-grained Splitting by a Multihomed Stub

So far, we have described how to exploit path diversity inside a single network. Next, we will examine how to exploit interdomain path diversity. Many routers learn multiple interdomain paths and could conceivably split traffic over them by installing multiple next-hops in the forwarding table. This is not done in practice due to the extra control-plane overhead. For ISP networks, edge routers would need to announce multiple paths to neighboring networks, and the neighboring networks would now need to store multiple paths. In addition, tunneling would be needed to direct packets on any non-default path, as explained in Section III.

In a stub network, however, edge routers do not need to propagate any of the learnt paths. In addition, packet classification is simpler for a stub network since the data rates tend to be lower and all packets originate from a single domain. Therefore, stub networks are natural places to deploy flexible splitting. Since applications are run at the edge, the stub network also has direct knowledge of the application requirements. Flexible splitting enables a network to place different classes of traffic onto different paths and balance load across multiple paths. Balancing load between multiple classes allows for efficient use of network resources and can avoid potential routing oscillations. Luckily, flexible path selection adds very little extra overhead on the data plane for a stub network, since choosing an outgoing link determines the entire path a packet will follow and no tunneling is required.

V. Cross-network Cooperation for Multiple Paths

When multiple networks cooperate, even more paths are available than when a network acts alone. In addition to scalability challenges, new business models must be put in place to enable inter-network cooperation, *e.g.*, charging for providing additional paths. In this section, we focus on proposed schemes which access additional paths with only limited cooperation between networks. Sources can encapsulate packets to direct the traffic through a *deflection point*—an end host or edge router that lies on an alternate path. This only requires a bilateral agreement between two parties. Sources can also deflect packets indirectly via *tagging*, where a few opaque bits in the packet header are used to indicate dissatisfaction with the current path. Tagging requires more networks to cooperate since routers need to be modified to forward packets based on the tags.

A. Encapsulation: Forwarding through a Deflection Point

Encapsulation can be used to explicitly force traffic onto an alternate path with better performance properties. A packet would be encapsulated to first arrive at the deflection point, then follow that deflection point's default path to the destination [1], [7]. Deflections can occur at the application layer or the network layer.

The easiest way to access another path is by deflecting through another end host, which does not require cooperation from or coordination between ISPs. First, an *overlay* or logical topology can be established between end hosts using tunnels [11]. Then each end host can measure the end-to-end performance properties of paths to a destination via other end hosts. If a path with better performance is found, packets can be deflected through another end host as seen in Figure 3. In addition to ease of deployment, application-layer deflections are attractive because they avoid advertisement of additional

paths. On the other hand, as the overlay grows in size, probing all paths through other end hosts imposes a significant amount of measurement overhead and does not scale beyond tens of end hosts [11]. In addition, sending traffic through other end hosts consumes edge link bandwidth and potentially incurs extra costs for the edge network.

A more scalable and efficient approach is for ISPs to provide alternative paths [1], [7]. As seen in Figure 3, the deflection point can be an edge router inside a network, rather than an end host. While this approach requires more cooperation from (and between) ISPs, it is still incrementally deployable. To ensure scalability, a network would only request an alternative path (perhaps with certain properties) from another network if it is unhappy with its default path. For example in Figure 3, the source could request an alternative path from its provider network A for reaching the destination D, network A can then choose to forward traffic on the alternative path (A, C, D), possibly for a price. Encapsulation would be used to deflect the packets through network C. The amount of control-plane overhead is directly proportional to the portion of networks unhappy with their default paths.

B. Tagging: Requesting an Alternate Path

An alternative to encapsulation is for end-hosts to simply *tag* their packets to request an alternate path [10], [8], without knowing the details of the path. A router forwards an incoming packet on the default path or an alternate path, based on the associated tag. Alternative paths inside an ISP can be constructed by one of the methods described in Section IV-A. Tagging without path visibility is effective when an end-to-end path is undesirable due to one particular segment of the path. For example, the path could contain a low capacity link, a high delay link, or a point of congestion. In these cases, routing around the problem link or router does not require direct knowledge of the route. By trying out a few tag values, the source network is likely to find a better path.

Tagging is quite scalable in the control plane since intermediate networks do not need to disseminate extra information or store network-level paths. An intermediate network can merely exploit the path diversity inside its own domain. There is little extra data-plane overhead, since the tag can use some rarely used bits in the existing IP header [8]. The extra data-plane overhead only comes from an ISP processing the received tag, and directing the packet onto an alternate path based on its tag value. Although tagging imposes less overhead than forwarding through a deflection point, it may require business relationships between the stub network and multiple ISPs. The incentives for honoring the tags are the most obvious in the context of hosts served by a single ISP.

VI. Conclusions

The ability to forward traffic on multiple paths would be useful for customizing paths for different applications, improving reliability, and balancing load. Yet Internet-wide multipath routing remains elusive, due to scalability and economic challenges. In this paper, we survey a variety of deployed and incrementally deployable techniques which achieve flexible multipath routing. Routers already have data-plane support for forwarding on alternate paths through tunneling: encapsulation and explicit routing, though such techniques should be used in moderation for scalability reasons. We examine existing techniques for fine-grained traffic division, then propose flowlet-cache as a more accurate and scalable alternative. To access more end-to-end paths, stub networks can continue the trend of multihoming and extend it to perform fine-grained load balancing.

Inside an ISP, multi-topology routing and forwarding on "downward" paths are both light-weight and easily deployable methods to leverage internal path diversity. Finally, we argue that deflecting packets at the network layer is a promising way to access more end-to-end paths with limited cooperation between networks, though new business models are needed to enable inter-network cooperation. We believe that more research could be done to better quantify the trade-off between overhead and performance for the more heavy-weight solutions, including end-to-end signaling techniques not surveyed in this paper. As technology advances, routers may become more capable of handling the overhead, making a wider range of solutions viable in practice. In addition, the economic incentives for providing value-added services will likely grow in the future and hopefully motivate the creation of new inter-network business models that enable Internet-wide multipath routing.

Acknowledgments

We would like to thank Dan Wendlandt, Vytautas Valancius, Rui Zhang, Yi Wang, Haakon Larsen, and Tian Lan for their feedback on earlier drafts of this paper. We also like to thank our shepherd Steve Uhlig and the anonymous reviewers for their insightful comments.

References

[1] D. Wendlandt, I. Avramopoulos, D. G. Andersen, and J. Rexford, "Don't secure routing protocols, secure data delivery," *in Proc. SIGCOMM Workshop on Hot Topics in Networking, November 2006.*

[2] S. Kandula, D. Katabi, B. Davie, and A. Charny, "Walking the tightrope: Responsive yet stable traffic engineering," *in Proc. ACM SIGCOMM, August 2005.*

[3] S. Savage, A. Collins, E. Hoffman, J. Snell, and T. Anderson, "The end-to-end effects of Internet path selection," *in Proc. ACM SIGCOMM, August 1999.*

[4] R. Teixeira, K. Marzullo, S. Savage, and G. M. Voelker, "Characterizing and measuring path diversity of Internet topologies," *in Proc. ACM SIGMETRICS, June 2003.*

[5] G. Iannaccone, C.-N. Chuah, S. Bhattacharyya, and C. Diot, "Feasibility of IP restoration in a tier-1 backbone," *IEEE Network Magazine, March 2004.*

[6] S. Kandula, D. Katabi, S. Sinha, and A. Berger, "Dynamic Load Balancing Without Packet Reordering," *in ACM SIGCOMM Computer Communication Review, June 2007.*

[7] W. Xu and J. Rexford, "MIRO: Multipath Interdomain Routing," *in Proc. ACM SIGCOMM, August 2006.*

[8] X. Yang and D. Wetherall, "Source selectable path diversity via routing deflections," *in Proc. ACM SIGCOMM, August 2006.*

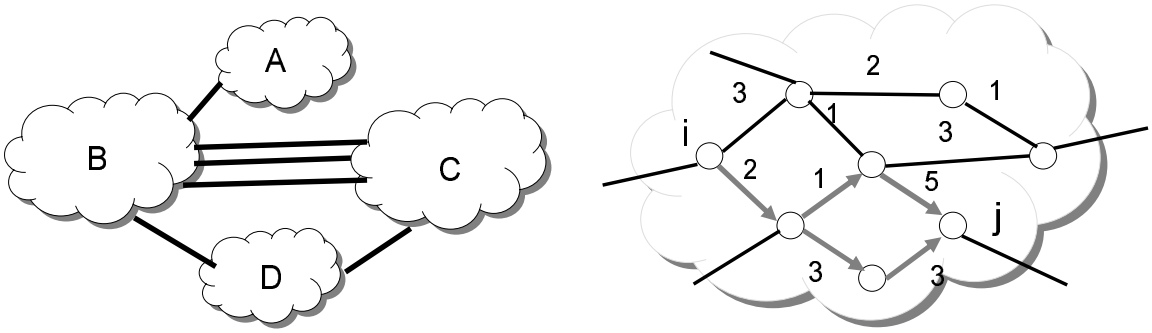
[9] D. Xu, M. Chiang, and J. Rexford, "DEFT: Distributed exponentially-weighted flow splitting," *in Proc. IEEE INFOCOM, May 2007.*

[10] M. Motiwala, N. Feamster, and S. Vempala, "Path splicing: Reliable connectivity with rapid recovery," *in Proc. SIGCOMM Workshop on Hot Topics in Networking, November 2007.*

[11] D. G. Andersen, H. Balakrishnan, M. F. Kaashoek, and R. Morris, "Resilient Overlay Networks," *in Proc. ACM SOSP, October 2001.*

Jiayue He received her B.A.Sc. (Hon.) in Engineering Science from University of Toronto in 2004. She is currently working towards her PhD degree at Princeton University. Her PhD is partially funded by the Gordon Wu Fellowship at Princeton University and the graduate fellowship from National Science and Engineering Research Council of Canada.

Jennifer Rexford is a Professor in the Computer Science department at Princeton University. From 1996-2004, she was a member of the Network Management and Performance department at AT&T Labs–Research. Jennifer serves on the CRA Board of Directors, the ACM Council, and the GENI Science Council. She received her BSE degree in Electrical Engineering from Princeton University in 1991, and her MSE and PhD degrees in Computer Science and Electrical Engineering from the University of Michigan in 1993 and 1996, respectively.



(a) Topology between 4 networks: B and C are ISPs A and D are stub networks

(b) Topology inside network C

Fig. 1. Sample inter-network topology, with a close-up on one network.

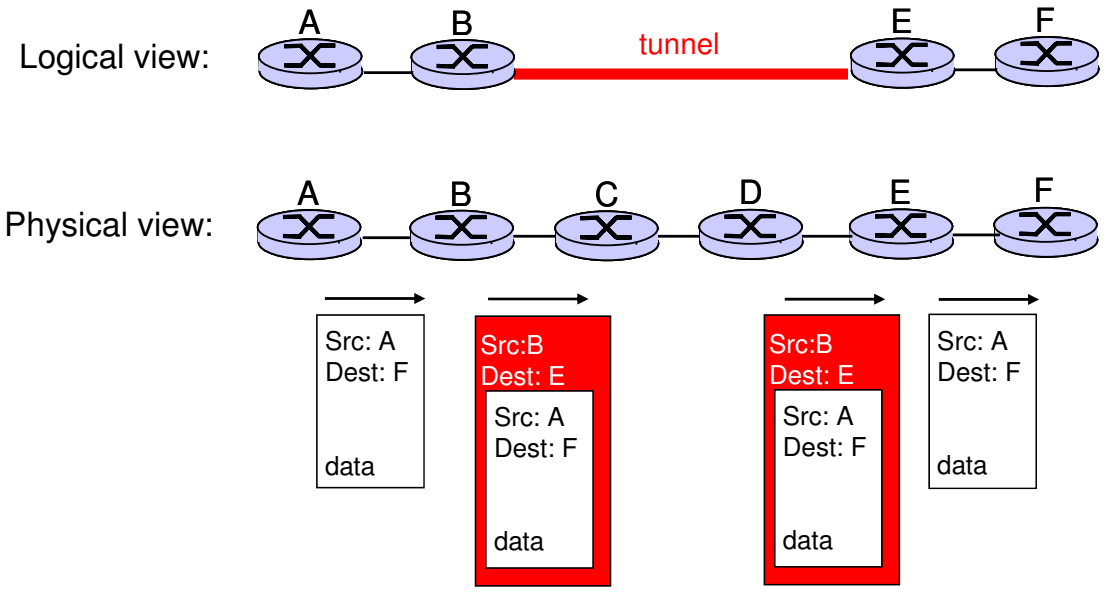


Fig. 2. Illustration of how a tunnel works.

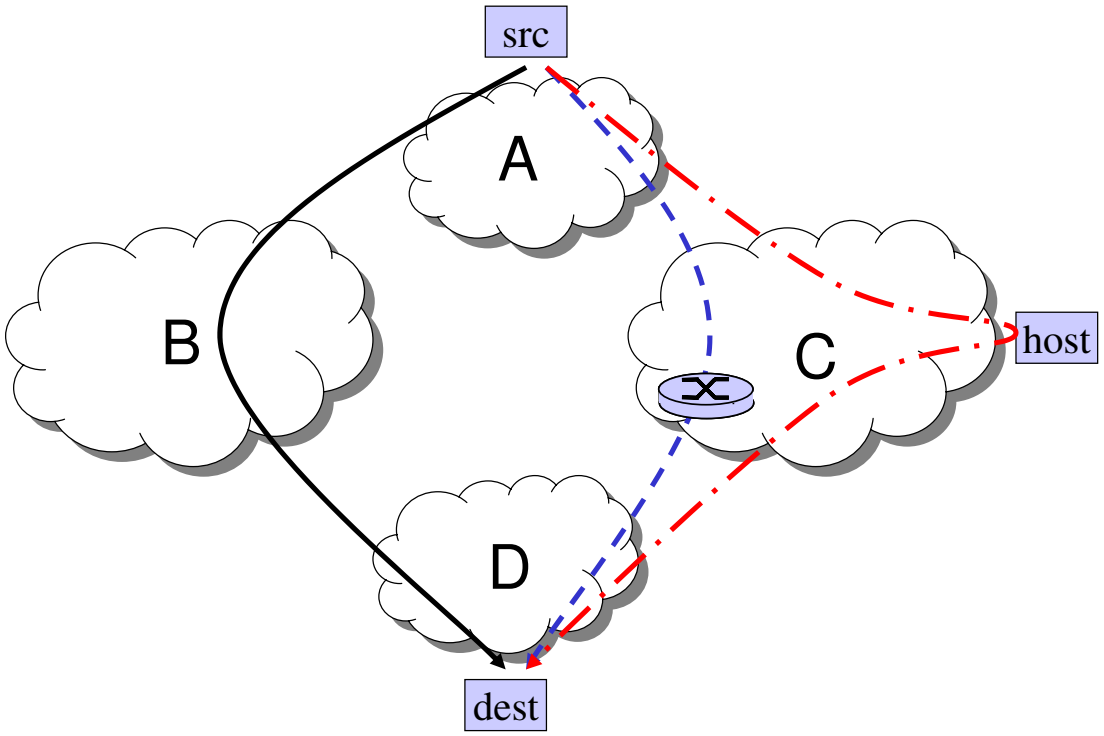


Fig. 3. The default path, shown in solid line is through network B. Deflection through network C is possible either with an overlay (dot-dash line) or through an ISP (dashed line).