Delta Routing: Improving the Price-Performance of Hybrid Private Networks

G. Porter
Department of Electrical Engineering and Computer Science
University of California, Berkeley
Berkeley, CA
USA
gporter@cs.berkeley.edu

M. Ji
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA
USA
Minwen.Ji@hp.com

Abstract

Large enterprises connect their locations together by building a corporate network out of private communication channels such as leased lines, Frame Relay and ATM links, called physical private networks or PPNs. PPNs provide good quality of service, but they are expensive. On the other hand, Internet-based Virtual Private Networks (VPNs) can provide speedy deployment of multisite corporate networks at a small fraction of the cost of private lines. However, Internet-based VPNs do not offer the same accountability and predictability as PPNs do, since the Internet is not admistrated by a single provider.

In order to build a hybrid network that is both reliable and affordable, we have developed the Delta Routing protocol, which allows nodes on a corporate network to communicate with each other using both PPN and Internet-based VPN. The Delta routing protocol works with the existing routing protocol on the PPN and allows each node to determine the best routes on the hybrid network using local information only.

We have simulated and compared the Delta routing protocol and the alternatives using the ns-2 simulator. The results show that the Delta routing protocol outperforms the alternatives in a variety of scenarios.

Keywords

integrated control and management, converged networks, route control, alternate path routing, virtual private networks

1. Introduction

An increasing number of companies are relying on private networks for business communications, which include emails, web accesses, database accesses, streaming media, file sharing and file transferring. These private networks protect the traffic from security attacks by external parties, such as eavesdropping and tampering. The amount of traffic on private networks is expected to increase as companies move more and more business activities online and as new applications (e.g., streaming media applications) develop and mature.

Traditional private networks are built from physically private communication channels, such as leased lines, Frame Relay and ATM links, collectively called *Physical Private Networks* or PPNs hereafter. A PPN is dedicated to the company's traffic, and has

dependable and guaranteed performance characteristics, such as low loss rate and predictable delay and bandwidth. On the other hand, a PPN is expensive. The company must pay a substantial amount of money for a certain capacity, whether that capacity is used or not. The charge per Mbps varies among providers and over time, but is in the range of hundreds of dollars per month. Due to limited IT budgets, PPN resources are not always overprovisioned. In the event of flash traffic or sudden large-bandwidth flows, congestion can develop and packets can be dropped.

Not all traffic on private networks has the same requirement on reliability, delay and bandwidth. For example, on the pre-merger Compaq's corporate network, only 30% of the traffic is generated by mission-critical mission-critical applications, including financial and manufacturing applications, applications, while the rest is generated by applications that are more tolerant of jitters, such as emails and web accesses. Intuitively, while the critical applications should stay on expensive, high-quality PPNs, non-critical applications can take advantage of cheaper but lower-quality alternatives.

In a typical corporate network, some locations have connectivity (or a *gateway*) to the public Internet via a local Internet Service Provider (ISP). This connectivity is present for many reasons, including providing employees with quick access to resources and services on the Internet. At least two benefits are obtained by having multiple gateways to the Internet on a corporate network. First, users experience lower delays by using a local ISP connection. Second, traffic sent to a nearby gateway does not have to travel as far, or at all, over expensive private networks. Additionally, in the last few years, IP-tunneling and Internet security techniques have boosted the development of Internet-based *Virtual Private Networks* (VPNs). An Internet-based VPN typically consists of secure IP tunnels established between sites on the Internet, which offer security services such as encryption and authentication [7]. Internet-based VPNs cost significantly less than a PPN. Although the connectivity to the local ISP has a monthly cost based on bandwidth, the connectivity beyond the local ISP is essentially free. Once a site is connected to the Internet, it will be able to establish a VPN tunnel with any other site on the Internet. An important difference in the pricing models of VPN and private lines is that VPN is charged per site, rather than per link or per mile. For example, a 10-site full mesh VPN costs approximately 7% of the price for leased lines and 29% of the price for Frame Relay [15].

Unfortunately, Internet-based VPNs do not offer the same quality of service as PPNs do. Due to the shared nature of the Internet, it is not possible to guarantee a certain loss rate, delay or bandwidth to a VPN tunnel. Furthermore, there are no guarantees as to the availability or uptime of links on the Internet. Even within one communication session, it is possible to experience a wide variability of end-to-end delays and bottleneck bandwidths. Therefore, the current use of VPNs is largely limited to connecting small offices, home offices or partners to regional sites or headquaters. The majority of the corporate traffic, e.g. site-to-site traffic, still goes through expensive PPNs.

We propose a combination of PPN and VPN links called a *hybrid private network* (HPN) with a price-performance that benefits from the best of both worlds, i.e., the accountability and predictability and PPN and the low cost of VPN. Intuitively, by carefully utilizing the combination of PPN and VPN, it is possible to provide higher performance

and lower loss to traffic on the private network at a fraction of the cost for upgrading or overprovisioning private lines.

Traffic can be routed on the PPN as long as its capacity allows, and excess traffic can be routed to the VPN when the PPN is congested. By doing this, we can avoid dropping traffic in the PPN and hence can trade delay or delay variation for packet loss. There is also plenty of opportunity for differentiated services on the HPN: mission critical traffic can be given higher priority for the PPN, while other traffic can be routed to the VPN when the PPN is busy.

2. Strawman's approaches

Given that an HPN consists of the existing PPN plus a mesh of VPN tunnels between the nodes, how should routes be computed on the new network? There are several naive approaches:

Uniform routing : run a single instance of an (existing) routing protocol, such as OSPF, BGP or alternate path routing [14], on the entire HPN. In this setup, transient Internet dynamics may result in excessive routing table changes or bad convergence times on the HPN, which would negatively affect the otherwise stable physical links. This approach might also generate a large amount of routing traffic if the VPN consists of a full mesh of tunnels between all nodes on the PPN.

Lasthop routing : run an existing routing protocol on the PPN only; when it is necessary to route traffic to the VPN, always route it to the tunnel ending at the destination. This approach may not choose the best route available on the HPN since it does not take advantage of any links on the PPN [1].

Nexthop routing : run an existing routing protocol on the PPN only; when it is necessary to route traffic to the VPN, always route it to the tunnel ending at the next hop on the original PPN route. This approach attempts to minimize the use of the VPN. However, if the next hop is also congested (which could well be the case due to the correlation of congestion events on the neighboring links), the traffic will be routed to the VPN again. This may result in a poor overall route to the destination.

3. Design

The main contribution of this paper is the design and evaluation of a new routing protocol for the HPN called the *delta routing protocol*. Like the last-hop and next-hop routing protocols, the delta routing protocol leaves the routes on the PPN intact and each node computes the routes on the HPN based on local information only (i.e., without messages from other nodes). In addition, the delta routing protocol strives to find the best possible, loop-free routes on the HPN, using the information from the existing routing protocol on the PPN and a congestion prediction mechanism called traffic matrix. A *traffic matrix* is a data structure that each node uses to record or estimate the load on each link in the PPN, and to predict when and where congestion events occur. The delta routing protocol can be deployed in a VPN-enabled edge router or VPN appliance that connects corporate sites to both private networks and the Internet.

Because the delta routing protocol is responsive to network conditions, it is able to adjust to network congestion and path conditions in a way that *nexthop* and *lasthop* cannot. The advantage of using delta over simpler, static schemes is that delta is able to perform well over a range of network conditions. In contrast, the static schemes perform well in some situations, but poorly in others.

3.1 Assumptions and definitions

We assume that a VPN tunnel can be established between any two Internet-accessible nodes. Links or tunnels that are directly attached to a node are called *local* links or tunnels to that node, others are *remote*. Accordingly, information about a local link or tunnel, such as queue length and congestion condition, is called *local* information, while information on a remote link or tunnel is called *remote* information. Traffic that goes through a node is called *local traffic*, while traffic that does not go through a node is called *remote traffic*.

The average delay of a VPN tunnel can be measured by its two end points relatively easily by observing packet headers or by sending explicit ping packets. Even if the clocks are not synchronized, nodes can still send ping packets to each other to measure a round-trip tunnel delay and estimate the one-way delay based on that, assuming reasonable symmetry of the tunnel delays. Even in cases where paths are not symmetric, we expect that the selection of paths in the wide-area will largely be unaffected.

We also assume that the internal routing information of a protocol, such as the recent *link-state packets* (LSPs) in OSPF, can be made available to other routing protocols on the same node. This seems reasonable since the edge routers and VPN endpoints themselves would need to be modified to implement the Delta routing protocol.

3.2 Delta routing

The delta routing protocol is based on the idea that when congestion is detected in PPN, packets should be forwarded through VPN tunnels around the congestion, rather than simply dropped. Diverting traffic in this way leads to lower packet drop rates. There are costs associated with doing this, including increasing path oscillation for diverted flows, subjecting diverted flows to higher delays and jitters. This paper is focused on minimizing the end-to-end path delays on the HPN. Reducing path oscillation is the topic of a seperate paper [6].

Loop-free, per-hop routes

Delta routing seperates the control plane of the PPN, whose characteristics change relatively slowly, with the control plane of the VPN, which is much more unpredictable. It works as follows. An instance of an existing routing protocol, such as OSPF, is run on the PPN as if the VPN tunnels do not exist. We call this instance of routing *base routing* and the routes it computes *base routes*. An instance of the delta routing algorithm runs on each node and has the access to the topology information maintained by the local base routing protocol. Each delta routing instance computes the best route on the HPN to each destination *independently*. That is, delta routing uses local information only or does not require routing traffic between nodes. The local information used for delta routing includes the topology of the PPN and the measured characteristics of the local tunnels.

A route on the HPN from a node S to a destination D (called a *delta route*

or HPNRoute(S, D)) consists of a tunnel VPNTunnel(S, N) and the base route PPNRoute(N, D) computed by the base routing, where N is a node *closer* (i.e., having a shorter path cost) than S to the destination D on the PPN, including D itself. This construction of delta routes restricts the next hops to a subset of the nodes on the HPN, which is intended to prevent loops from being formed. As long as the base routes are free of loops, a packet forwarded to a delta route will never come back to a node that it has already visited, because every time it is forwarded (either on the PPN or to a delta route), it always moves closer to the destination.

Algorithm 1 ComputeDeltaRoutes()

```
S = LocalAddress
for D = 1 to n do
  if D = S then
    continue
  HPNDelay[D] ⇐ ∞
  NextHop[D] ⇐ ∅
  for N = 0 to n do
    if PPNDelay[N, D] ≥ PPNDelay[S, D] then
      continue
    delay ⇐ VPNDelay[N] + PPNDelay[N, D]
    if delay < HPNDelay[D] then
      HPNDelay[D] ⇐ delay
      NextHop[D] ⇐ {N}
    else
      if delay = HPNDelay[D] then
        HPNDelay[D] ⇐ delay
        NextHop[D] ⇐ {NextHop[D], N}
```

The routine for computing delta routes is presented in Algorithm 1. It consists of a minimization function that chooses a VPN endpoint (i.e., NextHop[D]) for each destination D such that the expected end-to-end delay of the delta route (i.e., HPNDelay[D]) is minimized. The n noces on the network are numbered 1 through n. VPNDelay[N] is the measured VPN tunnel delay from node S to node N. PPNDelay[N, D] is the delay between nodes N and D on the PPN, and is obtained from the base routing protocol. For example, PPNDelay(N, D) can be calculated as the sum of the propagation latency of the links on the base route from N to D plus estimated queuing delays.

Routing traffic reduction

In order to construct a delta route with all tunnels as candidates, it would be necessary to periodically exchange the measured delays of all tunnels among the nodes. Given that the Internet characteristics change rapidly, by the time a routing packet reaches its destination, the information it carries could already be out of date. For this reason, only the local tunnel is taken into account when a delta route is constructed.

Existing routing protocols, such as distance-vector and link-state protocols, use both local and remote information for computing routes, and remote information is propagated

Metrics	How to Obtain	Update Interval	Triggered/Affected Operations
Physical link latency	Statically set in base routing protocol	Hours or on demand	Base route computation
Remote link utilization	Estimated with local traffic or broadcast from other nodes	Minutes to hours	Delta route computation
Virtual tunnel delay	Periodically measured	Seconds to minutes	Delta route computation
Local link utilization	Instantaneous queue length examined	Per packet	Per-packet or per-flow path selection

Table 1 Metrics and the operations in delta routing that are triggered or affected by the changes in metrics.

between nodes by repeated exchanges [4] or reliable flooding [9]. For example, if OSPF is run over an HPN that consists of n nodes, m physical links and $n(n-1)/2$ VPN tunnels, then each route update packet will be transmitted over all non-redundant links, i.e., m physical links and $n(n-1)/2 - m$ tunnels. In delta routing, each LSP is transmitted over the m physical links only, which substantially reduces the amount of routing traffic.

Per-packet forwarding

Each node in the HPN has two routes to each destination, the base route and the delta route. Initially, forwarding packets to the base route is preferred. Whenever the queue to the base route becomes full, the node sends excess packets to the delta route. The diverted traffic might undergo more delay than it would have otherwise. The ability to make this tradeoff between loss and delay is one of the benefits of delta routing. One fact to note is that altering a flow's path on a per-packet basis may have a detrimental effect on TCP traffic. This problem is addressed using a dial-controlled hashing scheme, which is discussed in a seperate paper [6].

3.3 Congestion Prediction Matrix

With the delta routing protocol, a node can detect the congestion on its local links by examining the packet queue lengths, and can forward traffic to a delta route when the congestion is detected. However, the delta routes are computed with the optimistic assumption that there will be no congestion on the remote physical links that are closer to the destination. Therefore, a packet travelling on a delta route might encounter another congested physical link, and hence be forwarded back to the VPN. This would cause the packet to be encrypted/decrypted and transmitted on some Internet links multiple times, which unneededly consumes bandwidths and processing cycles and increases end-to-end

delays. This could also cause the actual route on which the packet travels to the destination to be suboptimal. In an extreme case, a packet could travel from the source to the destimation by a sequence of next-hop tunnels, while it could get to the destination by a single last-hop tunnel if the congestion on the remote links were known to each node.

Therefore, we introduce a prediction mechansim that allows each node in the HPN to identify the remote links that will be congested, so that traffic can be diverted to bypass as many congested links as possible. To predict congestion on any remote links, we must know two things: the network's traffic matrix, and the topology of the network. In a traffic matrix, the element $M_{i,j}$ represents the amount of flow that originates at nodes i and is destined for node j. If at each node we had such a data structure, we could use the topology and routing tables learned from the base routing protocol to estimate the load on each link in the PPN. Links whose estimated loads are close to exceeding their capacity would be identified as congested links.

Since the traffic patterns in the network are constantly changing, it is not possible to keep up-to-date copies of the traffic matrix at each node. Instead, we have each node measure its local traffic, record the short-term average rates in its traffic matrix and broadcast the long-term averge rates to other nodes periodically (e.g., once an hour). Local queue lengths and packet counts are updated each time a packet arrives on an interface, and flow rates are calculated from these measurements once every second. Each node records the broadcast rates in its matrix for its remote traffic.

Using the traffic matrix and the topology learned from the base routing protocol, we can estimate congestion throughout the network and choose VPN endpoints that avoid these congestion points. Specifically, at node S, the next hop N on the HPN to a destination D is chosen to meet the following constraints:

1. N is closer than node S to the destination D on the PPN; and
2. No links on the route PPNRoute(N, D) are congested according to the traffic matrix at node S; and
3. HPNDelay$[N]$ is minimized among the intermediate nodes that meet the constraints above.

The traffic matrix at a node can help the node identify congested remote links if the congestion is caused (in part) by some of its local traffic. In response to such a congestion, the node can avoid routing other local traffic to the identified links on its delta routes.

We expect that this measurement plane and prediction scheme will work well in environments where the aggregate location-to-location traffic is not highly bursty. We also expect that delta routing with traffic matrix will benefit traffic that is tolerant of delay variation but not tolerant of packet loss, such as media streams and data backup traffic.

Path metrics

The cost of a path in delta routing is structured with four metrics: latency of physical links (obtained from the base routing protocol and rarely changed), delay of local tunnels (changed frequently and measured periodically), load on remote links (estimated with local traffic or broadcast periodically from other nodes), and load on local links (measured with local traffic on a per-packet basis). Based on their frequency of updates, the changes in the four metrics will trigger different parts of the overall routing on the HPN, namely

base route computation, delta route computation, and per-packet forwarding decision. Table 1 summarizes the metrics and their roles in delta routing.

4. Evaluation

4.1 Experimental Framework

To test the efficacy of delta routing, we constructed five different hybrid topologies and ran five different VPN-endpoint selection algorithms on them. The first two algorithms are *delta* and *delta+TM* (i.e., *delta* with congestion prediction). The next two algorithms are *nexthop* and *lasthop*. Lastly, we also compare these results to that of *ppnonly*, which never diverts traffic through VPN tunnels. *ppnonly* is a baseline that mimics a standard corporate network.

We tested the algorithms on a set of four algorithm-antagonistic topologies. Each topology is hand-designed to interact poorly with one of the five algorithms. The motivation for running the algorithms on these topologies is to see if one particular algorithm performs well on a majority of the different scenarios, and to see the effect that topology plays on the various algorithms. Next, we ran the five different algorithms on a large, forty-three node topology based on the PlanetLab network [10]. The details of this network are outlined in Section 4.3. To simulate each of these algorithms on a hybrid network, we used the $ns-2$ network simulator[5].

4.2 Algorithm-Antagonistic Topologies

We now consider a set of four topologies that were originally designed to frustrate each of the four dynamic VPN endpoint selection algorithms. Each of the algorithm-antagonistic topologies is based on a network of four nodes in a linear topology, numbered 1, 2, 3, and 4 (respectively). In this network the there is always congestion on the link between nodes 1 and 2. Furthermore, there is always a flow of traffic between nodes 1 and 4. This traffic flow is the flow that we measure in each case (the other flows are considered background or cross traffic).

Lasthop Topology

In the *lasthop* topology VPN tunnels destined to the final hop (node 4) suffer greater delay than packets destined to any other node. In fact, while the VPN tunnel delays from node to node grow roughly linearly with the PPN latency, in the *lasthop* topology VPN tunnel delays to node 4 are several times longer. This would mimic a node in Asia or Europe connected to the Internet backbone through slow public links, but connected to another node on the corporate network over a faster private link.

As Table 2 indicates, the dynamic algorithms considered suffer from very low packet loss rates (caused by a residual packet drop rate of 5% on the VPN tunnels). In fact, only *ppnonly* dropped packets. *delta+TM* chose for most of its forwarded traffic a point immediately before the last hop as its VPN endpoint. As discussed below, this lowers its mean end-to-end packet delays.

Nexthop Topology

The *nexthop* topology is characterized by a consecutive set of congested links which interact poorly with the *nexthop* algorithm. On this topology we see that *nexthop* performs

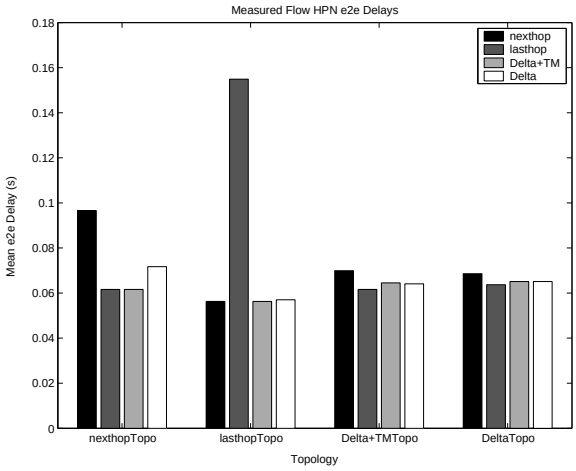


Figure 1: HPN e2e delays on algorithm-antagonistic topologies

the worst since traffic must often traverse multiple VPN tunnels. The two algorithms that performed the best were *lasthop* and *delta+TM*, which made similar VPN endpoint selections. This demonstrates a behavior that will be brought out further in the evaluation, namely that because *delta+TM* is able to choose VPN endpoints based on congestion in the network, it is often able to perform at least as well as both *nexthop* and *lasthop*.

Delta+TM Topology

A situation that would hurt *delta+TM*'s performance would be one where there are undetected congestion points *past* the point at which VPN endpoint selections are being made, which would cause packets to be reforwarded through additional VPN tunnels. In the *delta+TM* topology, there are two background traffic flows. At time $t = 5s$, a traffic flow is introduced. *delta+TM* on node 1 is not aware of this flow. This behavior would occur if new flows were added to the network at a rate higher than the rate that long-term TM measurements could be deduced. If periodic traffic measurements are exchanged at a rate of once every few minutes, and large traffic flows begin and end either as frequently or more frequently than the periodic messages could track, then this situation will develop.

Like *nexthop*, *delta+TM* at node 1 selects node 2 as the VPN tunnel endpoint. Unlike *nexthop*, which routes traffic from node 2 to node 3 through a VPN tunnel again, *delta+TM* selects a better tunnel (i.e., the one from node 2 to node 4) this time, resulting in better performance than *nexthop*. *delta+TM* performed slightly worse than *lasthop* and *delta* on the *delta+TM* topology. In a larger topology with larger end-to-end delays, the difference in performance between *delta+TM* and other two algorithms that outperformed it would be greater.

The "blindness' of *delta+TM* to the additional congestion points would cause poor VPN endpoint selections until the long-running TM averages could be updated to include the additional flows. On the other hand, *delta+TM* is able to correct a mistake made upstream and degrades gracefully in case of undetected congestion.

Delta Topology
delta will perform poorly whenever it chooses to send traffic to a node that is undergoing congestion. Since its choices are made using only local information, it cannot include the remote node's congestion condition in its computation. For this reason, it might choose a node that will force the tunneled traffic to be tunneled yet again upon arriving at the endpoint. The delta topology in our simulation consists of two background flows: one from node 1 to node 2, and another from node 3 to node 4. The VPN tunnel delay between node 1 and node 2 was made much lower than in the other topologies, which was intended to cause *delta* to choose as its VPN endpoint node 2. Since the link between node 3 and node 4 is congested, this would induce additional delay and packet tunneling at node 3.

Figure 1 shows that in contrast to what was expected, *delta* performed rather well on the *delta* topology. The mean end-to-end delay for all tunneled traffic was slighly lower for *delta*, and the end-to-end delay for the measured flow was somewhat higher. In this case *delta+TM* made the same selection as *lasthop*, and so the two performed similarly. Although *delta* causes some packets to be forwarded to the VPN twice, once from node 1 to 2 and again from node 3 to 4, it still benefits from the shorter delay on the first tunnel and yields a good end-to-end delay.

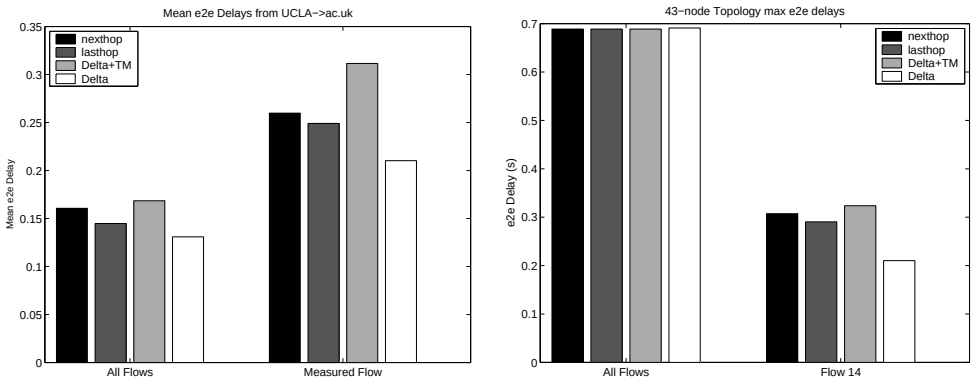
4.3 43-node Topology

The above mentioned topologies are all relatively simple in nature, and contain only a few nodes. To better test our algorithms in a more realistic setting, we need to model a large, complex network. Due to the difficulties in obtaining the detailed topology information of large corporate networks, we are synthetically building such a network based loosely on the PlanetLab network [10]. PlanetLab is a collection of over one hundred machines distributed in forty-three different locations around the world. Given that a hybrid network consists of a set of $n(n-1)/2$ VPN tunnels, as well as a set of dedicated leased lines, we can use node-to-node delay measurements between each of the PlanetLab nodes as the values of the VPN tunnel delays. In our final topology, we use "all-pairs" measurements taken from the PlanetLab network to model the 43-by-43 VPN tunnels that connect the nodes in our network.

But what about the leased lines? To realistically model a PPN that connects nodes geographically distributed just as the PlanetLab nodes, we proceeded in two steps: First, we grouped geographically proximate nodes together into a set of about eight regions. Within these regions the nodes are rather well connected, usually with links similar in performance and capacity to OC-3 fiber. Between the regions there are smaller connections, representing more expensive long-haul and trans-continental links. These links have a lower capacity, and a higher propagation delay.

On top of this 43-node network there are a set of traffic flows, again simulating "normal" background traffic. A number of new flows are introduced to the simulation and cause congestion. For example, the congestion occurs on a trans-Atlantic link, specifically from the US East coast and England (which in this topology is a primary conduit for US/European traffic flow).

Figure 2 shows that *delta* outperforms *nexthop* by 25% and *lasthop* by 19% in the mean end-to-end delay of the measured flow. It shows that *delta* outperforms *nexthop* by



(a) Mean delays

(b) Max delays

Figure 2: HPN delays on the 43-node topology with bottleneck links

44% and **lasthop** by 38% in the maximum end-to-end delay of the measured flow. These results validate our design for the **delta** routing protocol: in a large, complex network like planet-lab, the fixed selection algorithms such as *nexthop* and *lasthop* would unlikely excel since they are not able to adapt to the topology details and load changes.

The other metric in which the protocols differ is in the packet drop rate, shown in Table 2 As expected, *ppnonly* suffers from the most packet drop events, usually at the ingress to a slow trans-Atlantic link. *Delta+TM* has the second highest amount of loss and *nexthop* has the lowest amount of loss. In this example the ingress VPN links are 1.5 Mbits/s, while the Intranet link is 5 Mbits/s. This is a consequence of limited bandwidth on the Internet links. *nexthop* often causes packets to be forwarded to the VPN multiple times; therefore, it distributes load over the Internet links more evenly than the other algorithms. We expect that, if more bandwidth is purchased on the Internet links (which is very likely in practice due to its low price), the advantage of *nexthop* will not be sustained. *Delta+TM* has relatively high drop rate because multiple nodes (e.g., nodes along the same route) often predict the same congestion points with their traffic matrixes, and hence route traffic to the same node beyond those congestion points, causing the Internet link at that node to be congested. This also explains why *delta+TM* has unexpectedly higher delays, which mainly attribute to queuing delays at the congested Internet links. This result helps to reveal a weakness in the traffic matrix scheme, i.e., many nodes make similar decisions on load distribution and consequently overload the most popular target links. As future work, we might consider randomly choosing a subset of candidate nodes as a way to avoid this type of synchronicity.

The comparison between these two measured flows demonstrates that the Delta routing protocol is able to adaptively make trade-offs between packet drop rate and end-to-end packet delay. Providing for a dynamic algorithm to make measurement-based path selections will be able to respond to wide-area Internet dynamics. Providing for lower packet

	Topology					
algorithm	lasthop-antagonistic	delta-antagonistic	nexthop-antagonistic	delta+TM-antagonistic	linear	43-node
ppnonly	1750	8510	8749	8510	1358	4438
nexthop	101	466	873	0	0	1878
lasthop	101	416	443	0	0	3030
deltaTM	101	452	443	0	0	3105
delta	101	452	586	0	0	2991

Table 2 Packet drop events in the measured flows on physical links and links to ISP.

loss rates and the possibility of additionally lowering end-to-end packet delays will allow for more reliable and efficient corporate networks.

5. Related Work

While most routing protocols in production today assume that there is a single "best" path between any pair of nodes and that all traffic between them should use it, a lot of research has been done on *multipath routing*, based on the belief that better performance can be obtained by splitting the traffic over several paths for the purpose of load balancing the traffic [2] [14] [8] [16] [13] [12]. Unlike delta routing, existing multipath routing protocols treat all links in the network as equal-quality ones. In the asymptotic approximation approach [12], traffic is routed over all paths whose length are no more than $1 + \beta$ times of that of the shortest path. In the shortest path first with emergency exits approach [14], neighbors are recursively queried for an exit when the shortest path to a destination is congested and the resulting exit path may not be the shortest path and may not be free of loops.

The performance of minimum-delay routing algorithms heavily depends on the way in which end-to-end delay or cost of a path is estimated. RIP and OSPF in practice use a preconfigured constant cost per link. ARPANET was used to test a number of dynamically calculated costs, such as queue length, queuing delay + transmission time + latency, and link utilization. Some research has been done on *marginal delay* as the link cost [2] [13]. However, the proof of minimal delay and other properties of most known routing algorithms are based on the assumption about constant bandwidth and latency of the links, which are not true for VPN tunnels on the Internet. Therefore, most existing methods of estimating path costs are not directly applicable to the hybrid network we study.

Because Delta Routing with congestion prediction is a form of QoS-based routing, we must ensure that it is stable in the face of incorrect or inaccurate predictions. Delta's separation of the VPN-vs-Intranet control planes helps to avoid introducing instability to the PPN when VPN tunnels' characteristics change. Secondly, Delta's ability to divert traffic from the PPN to the VPN at points of congestion mean that even if a bad prediction is made at one point in the network, that mistake can be corrected later with a minimum

of packet drops. Considerable work has been done on examining these issues in a variety of settings. One proposal [11] separates long-lived from short-lived TCP flows and only applies QoS routing to the long-lived flows. The effect of basing routing decisions on inaccurate information has been examined in [3].

It has been shown effective to use long-term, end-to-end information for route computation and short-term, local information for per-packet or per-flow path selection [8] [13]. In delta routing, metrics and route computation are further categorized to be PPN related and VPN related. In particular, frequently changing tunnel delays and traffic rates are used for delta route computation, while relatively stable link latency is used for base route computation.

6. Conclusion

We have designed a delta routing protocol for hybrid private networks. The protocol has the following desirable properties: it preserves route stability on the physical private network in the face of Internet dynamics; it computes the best possible, loop-free delta routes on the hybrid network by leveraging existing routing protocols and locally available information.

We have evaluated the delta routing protocol in comparison with alternatives (i.e., fixed selections of VPN tunnels) with various simulations. The results show that the delta routing protocol is able to automatically adapt to many network topologies and traffic patterns, and hence perform better than or as good as the fixed selection policies. In the planet-lab topology, the delta routing protocol outperforms the fixed schemes by up to 25% in mean delay and by up to 44% in maximum delay. Such a routing protocol would help us to achieve the goal of building a hybrid network: to effectively reduce packet drops while maintaining reasonable end-to-end delays.

Acknowledgments

We would like to thank Dennis Fetterly, John Grillo, Valerie King, Tom Rodeheffer and Li Zhang for valuable input to the delta routing project.

References

[1] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, Octobor 2001.

[2] C. G. Cassandras, M V. Abidi, and D. Towsley. Distributed routing with on-line marginal delay estimation. *IEEE Transactions on Communications*, Vol. 18, March 1990.

[3] Roch Guerin and Ariel Orda. Qos-based routing in networks with inaccurate information: Theory and algorithms. In *INFOCOM (1)*, pages 75–83, 1997.

[4] C. Hedrick. Routing information protocol. Technical Report RFC-1058, IETF, April 1988.

[5] http://www.isi.edu/nsnam/ns/. Network simulator (ns), version 2.

[6] Minwen Ji. Dial-controlled hash: Reducing path oscillation in multipath networks. In *Proceedings of the International Conference on Computer Communications and Networks (ICCCN), also as HP Labs techincal report HPL-2003-98*, October 2003.

[7] S. Kent. Security architecture for the internet protocol. Technical Report RFC-2401, IETF, November 1998.

[8] D. Kourkouzelis. *Multipath Routing Using Diffusing Computations*. M.S. Thesis. University of California, Santa Cruz, March 1997.

[9] J. Moy. Ospf version 2. Technical Report RFC-2328, IETF, April 1998.

[10] The PlanetLab Network. http://www.planet-lab.org/.

[11] A. Shaikh, J. Rexford, and K. Shin. Load-sensitive routing of long-lived ip fbws. In *Proceedings of ACM SIGCOMM*, August 1999.

[12] X. Su and G. de Veciana. Dynamic multi-path routing: Asymptotic approximation and simulations. In *Proceedings of ACM SIGMETRICS*, June 2001.

[13] S. Vutukury and J. J. Garcia-Luna-Aceves. A simple approximation to minimum-delay routing. In *Proceedings of ACM SIGCOMM*, August, 1999.

[14] Z. Wang and J. Crowcroft. Shortest path fi rst with emergency exits. In *Proceedings of ACM SIGCOMM*, August 1990.

[15] Mark Winther. Using the internet for the corporate virtual private network: Overview of costs, fbxibility, and management. Technical Report 15829, International Data Corporation, March 1998.

[16] W. T. Zaumen and J. J. Garcia-Luna-Aceves. Loop-free multipath routing using generalized diffusing computations. In *Proceedings of IEEE INFOCOM*, March 1998.