Traffic Matrix Estimation for Low-loss Routing in Hybrid Networks

by George Manning Porter

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

Professor Randy H. Katz, Ph.D.
Research Advisor

(Date)

* * * * * * * * * * *

Professor Ion Stoica, Ph.D.
Reader

(Date)

Traffic Matrix Estimation for Low-loss Routing in Hybrid Networks

by

George Manning Porter

Abstract

Traffic Matrix Estimation for Low-loss Routing in Hybrid Networks

by

George Manning Porter

Master of Science in Computer Science

University of California at Berkeley

Professor Randy H. Katz, Ph.D., Research Advisor

Large enterprises connect their locations together by building a corporate network (sometimes called an *intranet*) out of private communication channels such as leased lines, Frame Relay and ATM links. Although these lines are dedicated to the company's traffic, and thus provide good quality of service, they are expensive and thus not always overprovisioned. 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 VPN does not offer the same accountability and predictability as an *intranet* does, since the Internet is not admistrated by a single provider.

To take advantage of the best of both worlds, we have developed the Delta Routing protocol, which allows nodes on a corporate network to communicate with each other using both private lines and Internet-based VPN tunnels. 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 based on local information only. It uses a novel congestion prediction mechanism called *Flow-based Delta Routing* to make the best use of the local information.

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

variety of scenarios.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

An enterprise's internal corporate network, called an Intranet, is becoming increasingly relied upon to provide business applications. These applications include e-mail, web access, video and audio teleconferencing, instant messaging, file service and bulk data backup and transfer systems. Additionally, corporations rely on Internet connectivity to do business by exchanging e-mail, providing web content and online customer account access, as well as other applications requiring that nodes in corporate networks have access to the public Internet. Because of its success, internal networks increasingly use "off-the-shelf" TCP/IP-based technology. Merging the administration of the internal and external networks simplifies network management and enables access from the internal network to the Internet. Providing good performance to internal network applications and ensuring that corporate traffic flows smoothly leads to Intranets being made up of leased, private lines from metro-area and long-haul telecom providers. These leased lines act as point-to-point links that connect an enterprise's geographic locations. To provide Internet connectivity to users of the internal network, some of the geographic locations of the enterprise have gateways to the Internet via Internet Service Providers (ISPs). While network traffic destined for the internal network stays on the leased private lines, traffic destined for the public Internet travels to one of these gateways, where it is passed on to a transit provider, and from there to the public Internet.

Provisioning Intranets becomes a balancing act between the need for large link capacities to support growing traffic demands while managing costs. Long-haul Intranet links are very expensive, and can often represent a significant percentage of an enterprise's IT budget. For this reason, Intranet links are sometimes just-provisioned or under-provisioned. In the event of some flash traffic, such as a widely viewed video flow, or a somewhat rarely executed bulk data transfer, the Intranet links can become congested, and packets can get lost. To prevent this, we propose a novel scheme in which excess and flash traffic that is about to cause congestion on the internal network can be temporarily diverted outside of the corporate network and through the Internet to its ultimate destination. By "overflowing" this excess traffic temporarily outside the internal network, it is possible to avoid dropping traffic as a result of temporary congestion events. In the case of flash traffic, this would represent the duration of the traffic surge. If the congestion is due to a stable increase in the traffic on the internal network, this scheme would help temporarily alleviate congestion until the internal network could be reprovisioned.

To make use of the public Internet to handle internal corporate traffic, we make use of Virtual Private Network (VPN) tunnels. These tunnels allow for the secure transmission of data from one part of the Internet to the other. In our proposed scheme, this would be from one location of our network to another. For example, if an enterprise was experiencing congestion on a link between San Francisco and Houston, some traffic could be sent through VPN tunnels over the Internet between those two cities, thereby ensuring that the excess traffic does not have to be dropped.

To make use of these VPN tunnels effectively, at least two problems must be addressed. First, when should traffic be diverted outside of the network? Second, to where should this diverted traffic be tunneled? In answer to the second question, it may not always be best to tunnel traffic directly to its ultimate destination. After all, Intranets use dedicated private links because of their reliability and performance benefits. In contrast, the public Internet is a shared resource that does not make any performance or reliability guarantees. For that reason, minimizing the time traffic spends on the Internet should give greater control to network planners and operators.

We propose two related protocols to address this tradeoff. The first is Delay-based Delta routing, a system developed at the Systems Research Center of Hewlett-Packard Labs, in Palo Alto, California. The second scheme is a Flow-based Delta Routing Protocol (FDR), developed by the author. Delay-based delta routing chooses VPN endpoints based on the minimization of perceived end-to-end delays, using only local information for VPN endpoint computations. Flow-based Delta routing uses a measurement system to predict where congestion exists in the network. Based on those predictions, VPN endpoints are selected that prevent traffic from traversing those congested points. Also considered are two other simple VPN endpoint selection schemes.

Chapter 2 presents the architecture of our system and a more detailed look at corporate enterprise networks. Chapter 4 presents an evaluation of the Delay-based and Flow-based Delta routing schemes on a set of different topologies and traffic patterns. Related work is outlined in Chapter 5.

Chapter 2

Architecture

The Delta Routing Protocol makes efficient use of hybrid corporate networks. Described in more detail below, a hybrid corporate network consists of nodes connected together via dedicated, leased lines. Additionally, a subset (possibly all) of the nodes have connectivity to the public Internet via Internet Service Providers (ISPs). These links to the Internet present an opportunity to make more efficient use of the dedicated leased lines (Intranet) by offloading some traffic from the Intranet to the public Internet during congestion events. In this way unexpected or excessive traffic on the Intranet "overflows" onto the Internet, reducing the congestion on the enterprise's network. Currently enterprise Intranet networks would drop such traffic according to router-specific drop policies (often Drop Tail).

A characteristic of private Intranets is that they are entirely dedicated to the enterprise that owns them. Because of that, their performance is very dependable and predictable. Incorporating network paths that transit the public Internet then poses a problem. Because of its shared nature, the Internet is unpredictable, often varying in end-to-end delay, loss rates, and availability. Some thought must be given to construct a mechanism that can make use of the Internet for transiting Intranet traffic. A simple and straightforward mechanism might be to include the n^2 Internet paths that exist between each Internet-reachable Intranet node in the Intranet routing protocol computa-

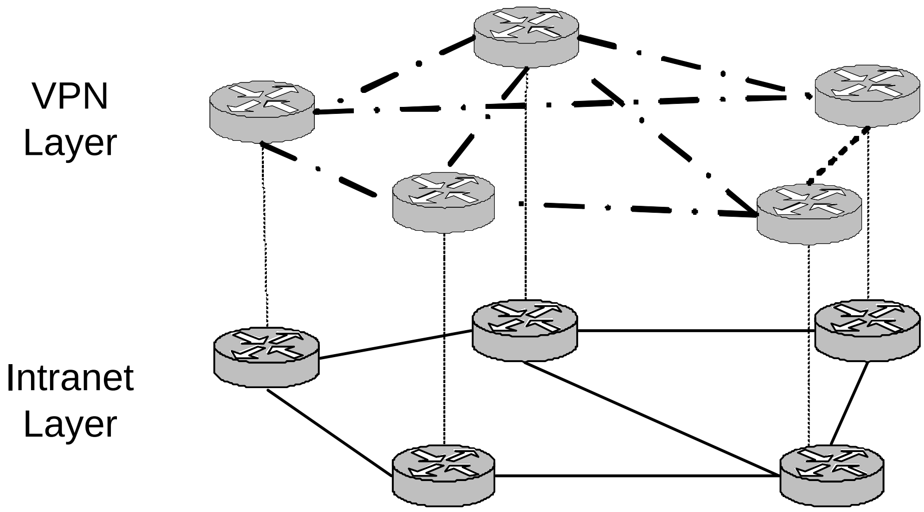

Figure 1: A Hybrid Corporate Network

tion (often OSPF[21] or RIP[14]). Unfortunately, such an arrangement would mean that wide-area Internet instabilities would disrupt an enterprise's internal network. A different protocol must be employed that separates route computations over the internal network from Internet-path selection. This protocol is Delta Routing, and is described in Section 2.2. It separates the control plane of the internal network from the public Internet, making computations using only local information. This behavior is important since network characteristics and traffic patterns can change rapidly, faster than information can propagate between distant nodes. Before we consider this proposal, we first look in greater detail at hybrid corporate networks.

2.1 Hybrid Public/Private Networks

A "private" enterprise network (also called an Intranet) is a collection of leased lines which span a set of locations (abstractly called "nodes"). These lines are dedicated to the enterprise, and have dependable and guaranteed performance characteristics, such as very low loss rates and fixed delays and bandwidths. All the enterprise's traffic flows over this network, including email, video, support applications and database traffic. Because they are dedicated to the company's network, these lines are very expensive, often constituting a significant portion of the cost of building an internal network. For this reason, they are often underprovisioned. In the event of flash traffic or a sudden new, large-bandwidth flow, congestion can develop and traffic can get lost. Assigning priorities to the different traffic flows can prevent important traffic from getting lost; however, the result might be total starvation to the lower-priority traffic.

In a typical corporate network, some of the locations will also have connectivity to the Internet via an ISP. This connectivity is provided 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 connections to the Internet through different ISP connections. The first is that users experience lower delays by using a local ISP connection. Secondly, traffic sent to a nearby ISP link does not have to travel as far, or at all, over the expensive, leased lines. This reduces the traffic load on the private network. Of course, any traffic destined for another node in the company's network must travel over the leased lines.

Pricing for Internet connectivity is usually structured as follows: First, the ISP charges a fixed price to connect with a certain type of link (such as a 1.5 Mbit/sec "T1" link, or a 45 Mbit/sec "OC-3" link). Secondly, a monthly fee is charged based on usage, in Mbits/sec. The charge per Megabit varies among providers and over time, but is in the range of hundreds of dollars per month. What is most important to consider is the fact that bit-per-bit, it is much cheaper to send traffic through an ISP than over the private, corporate network. Also, we assume that the Internet bandwidth between two nodes is large, since large enterprises often connect to the Internet through large ISPs, which are often overprovisioned.

An prominent networking technology, called a Virtual Private Network (VPN), can be used to connect two private networks together over the public Internet in a secure manner. A VPN works by having a private network encrypt traffic destined for a distant network and sending that encrypted traffic over a tunnel through the Internet. When that traffic arrives at the distant network, it is decrypted and forwarded along to its destination. VPNs are often used as a lightweight, low-cost mechanism to provide network connectivity and access to internal resources to a machine outside of a company's firewall. A typical use of this technology would be allowing employees to work from home over a broadband connection, yet have full access to their company's network. One of the biggest problems of using VPN tunnels to connect two networks together is that due to the shared nature of the Internet, it is not possible to guarantee strict delays or bandwidth to the VPN tunnel. Furthermore, there are no guarantees as to the availability or uptime of these links. Even within one communication session, it is possible to experience a wide variability of end-to-end delays and bandwidths.

We assume that the leased, physical network has better performance characteristics than VPN-tunnels through the public Internet. We assume that this holds both during high-congestion periods of the Internet (during the day), but also at night, and when the Internet has a relatively low residual congestion rate. Adapting the Delta Routing protocol for use as a general purpose routing protocol is considered future work. Such a new protocol might be able to supply a corporate network with higher performance paths between endpoints than the standard leased network. This behavior is in the spirit of overlay networks, which are able to find desirable paths between endpoints despite a lack of such paths at the network layer.

The Delta Routing system is designed to make use of the fact that corporate networks have two ways of communicating between nodes: the dedicated, but underprovisioned private network, as well as the cheaper, but more unpredictable public Internet. By carefully utilizing these links, it is possible to provide higher performance and lower loss to traffic on the private network, which still provides low loss and as little delay as possible to traffic that flows over the VPN links.

2.2 The Delta Routing Protocol

The Delta Routing protocol is based on the idea that when congestion is detected in a physical private network (PPN), packets should be forwarded through tunnels around the congestion, rather than simply dropped. Diverting traffic in this way leads to lower packet drop rates and often lower end-to-end delays (as a result of reducing overall queuing delay and time packets spend in buffers). There are costs associated with doing this, including increasing packet reordering at the receiver, possibly subjecting diverted flows to higher end-to-end delays, and subjecting diverted flows to increased exposure to wide-area network dynamics. In this work we primarily consider the tradeoff between packet drop rate and end-to-end delay. Applications are highly affected by these two metrics, as well as packet reorder and round-trip time stability (in the case of TCP flows). Although not considered in this work, a mechanism must be employed to pin TCP flows to certain paths for a reasonable amount of time before changing to a new path. Only in this way will it be possible to avoid triggering TCP's congestion control mechanisms. A mechanism that provides stability to TCP flows, yet is able to make use of better paths if available is the Dial Forwarding mechanism, described in [15]. In this work we target datagram-based traffic sources that can tolerate some packet reorder. Such applications include video, audio, voice-over-ip (VoIP), and some bulk-file transfer protocols.

First we describe Delay-based Delta Routing, which is a protocol developed at the Hewlett-Packard Systems Research Center (SRC) (Palo Alto, CA). Next, we will describe Flow-based Delta Routing, which uses an alternative measurement-based approach to predict congestion, rather than simply responding to congestion events at the point they occur. This scheme is based on a measurement infrastructure in which nodes individually compute a network traffic matrix (TM) based on primarily local information. A traffic matrix is an n by n matrix in which the (i, j)th element represents the amount of flow originating at node i that is destined for node j. Using that matrix, each node predicts which links are congested and chooses VPN endpoints that avoid that congestion.

2.2.1 Delay-based Delta Routing

The Delay-based Delta Routing protocol works as follows: Each node in the network has two choices for sending traffic to some destination. The first choice is via the PPN, which is the normal method used in corporate Intranets. The second method is to forward traffic over a VPN tunnel to the destination, using the node's local ISP. When using the VPN tunnel option it is not necessarily best to send the traffic all the way to its ultimate destination (the "last hop"). Rather, it might be better to send the traffic to a node that is closer to the destination than the sender. From that intermediate point the traffic can travel over the PPN to its ultimate destination.

The Delta Routing Protocol chooses this intermediate node according to the algorithm presented in Algorithm 1. This metric that this algorithm uses in choosing VPN endpoints is the predicted total end-to-end delay from the point Delta routing is initiated to its final destination. Note that as described, this algorithm is based entirely on local information. There are several quantities that should be mentioned:

- $QueueDelay(i)$ is the current queueing delay at node i. This quantity is treated as an instantaneous variable, and is known only by i.
- $TunnelDelay(j)$ is the amount of time it would take a packet to reach node j through a VPN tunnel. This quantity is estimated by periodically sending measurement packets to other nodes in the network and measuring the time taken to reach the destination. The frequency of message exchanges dictates to a certain extent the accuracy of this estimation.
- $IntranetDelay(j, i)$ is the propagation delay between nodes j and i on the Intranet. This amount is obtained from the underlying routing protocol, which we are assuming is a link-state routing protocol such as OSPF[21], and knowledge of the initial network setup.

At each node the packet forwarding mechanism is simple. Initially forwarding packets to the PPN is preferred. Whenever the queue to the PPN becomes full, the node sends those packets across the VPN tunnel to the prechosen destination. By not dropping the traffic in the queue, but rather forwarding it to the VPN, packet loss is avoided. However, that traffic possibly undergoes more delay than it would have otherwise. The ability to make this tradeoff between loss and delay is one of the benefits of this scheme. As described previously, altering a flow's path on a packet-by-packet basis has a very detrimental effect on TCP traffic [1].

Algorithm 1 DelayBasedTunnelEndpoints()

{This algorithm runs on each node, and establishes the next hop on the VPN network for each destination}

target[] $\Leftarrow$ NEW Array {target[i] will be the next hop for destination i}

for $i = 0$ to $numNodes()$ **do**

  if $i = myAddr()$ **then**

    continue

  end if

  curDelay $\Leftarrow \infty$

  curTarget $\Leftarrow myAddr()$

  {We consider each node in turn as a candidate next hop}

  for $j = 0$ to $numNodes()$ **do**

    lc $\Leftarrow$ QueueDelay($myAddr()$) + TunnelDelay(j) + IntranetDelay(j,i)

    if $lc < curDelay$ **then**

      {Node j is the newest candidate least-delay next hop for destination i}

      curDelay $\Leftarrow lc$

      curTarget $\Leftarrow j$

    end if

  end for

  {curTarget is the new VPN next hop for destination i}

  target[i] $\Leftarrow$ curTarget

end for

One might ask why a single protocol such as OSPF[21] is not run over both the PPN and the collection of n^2 VPN tunnels to determine the appropriate forwarding paths at each node. The reason is that with such a setup routes internal to the corporate network would be affected by transient Internet dynamics, which would lead to bad convergence times and excessive routing table changes. We are separating the control plane of the internal network, whose characteristics change relatively slowly, with the control plane of the dynamic Internet, which is much more unpredictable.

One might also wonder why a node i only takes into account its own queueing delay, and does not consider the delays at other nodes. To take such factors into account, it would be necessary to periodically exchange router queue lengths among the nodes. Given that the router queues grow and shrink very quickly, by the time any such advertisements reach their destinations the information they carry would be stale and out of date. For this reason, only the local queuing delay is taken in account when choosing VPN endpoints. Since each node in the network is performing Delta routing, if a node makes a bad forwarding decision and sends traffic to an overloaded node, that overloaded node will be able in turn to overflow traffic onto a VPN tunnel to its ultimate destination. Although this would add additional delay to the forwarded traffic, it would prevent that traffic from getting dropped.

A deficiency of the Delay-based Delta Routing protocol is that the metric that it minimizes is the predicted end-to-end delay. While this metric is important for efficient routing, it might lead to selecting a VPN endpoint that is not optimal. One example of this behavior can occur in a network with a slow bottleneck VPN link. This might occur if the ultimate Intranet destination is in Europe or Asia. If the bottleneck transatlantic (or transpacific) link becomes congested, Delay-based Delta routing might exacerbate the problem by sending additional traffic to the head of that link. This traffic would likely get dropped at that point (or be diverted yet again into another VPN tunnel). Another mechanism, and the contribution of this work, is a Flow-based routing protocol that attempts to avoid this problem using a combination of traffic-matrix estimation and congestion prediction.

2.2.2 Flow-based Delta Routing

With the Delay-based Delta Routing protocol, a node can detect if traffic that is about to be forwarded to its neighbor is about to be dropped due to congestion. In that case, it can forward that traffic through an alternate path, namely a VPN tunnel. In this scheme, traffic is diverted at the point of congestion. In many networks there is an aggregation of traffic at certain bottleneck points, such as trans-Atlantic or satellite links. When choosing a VPN endpoint for traffic destined to a particular destination, it can be beneficial to know where these points are, and send the traffic past all bottleneck points. In this way unnecessary congestion can be avoided.

This is the idea behind Flow-based Delta routing (FDR). If we assume that each node in the network knows which links are suffering congestion, we can choose as VPN endpoints nodes that are immediately past these congestion points. The motivation behind doing this is the desire to keep Intranet traffic on the enterprise's network as much as possible, for reasons of increased control, greater reliability, and in general, better performance as compared to the public Internet. To choose VPN endpoints past points of congestion we must know where the congestion is. If each node had an up-to-date TM, it could combine the TM with its knowledge of the Intranet topology (as learned by the internal routing protocol, such as OSPF[21]) to locate congested links. Call such a function f. The congestion at time t_0, denoted as $cong_{t_0}$ is given by

$$cong_{t_0} = f(TOPO_{t_0}, TM_{t_0}) \tag{2.1}$$

where $TOPO_t$ is the topology (including all link bandwidths) at time t and TM_t is the traffic matrix at time t. The internal routing protocol establishes $TOPO_t$ at each of the nodes, but what about TM_t? Finding this is more problematic. The task now is to provide nodes with a mechanism to learn the TM, which is constantly changing. One point to note is that Flow-based Delta routing requires a type of global information, namely long-running averages of node-to-node traffic. These averages are periodically exchanged among nodes. Their use is to supply missing information to nodes that cannot measure certain flow rates. In contrast to these long-running averages, a varient of Delay-based Delta routing that took into consideration remote queue lengths (in addition to local queue lengths), would require a form of global information that changes much more rapidly than the long-running averages that Flow-based Delta routing uses. During congestion events, queue lengths can change at a rate exceeding any rate that they could be communicated among distant nodes.

Traffic Matrix Estimation

Since the traffic patterns in the network are constantly changing, it isn't possible to keep up-to-date copies of the traffic matrix at each node. Instead, we have each node *predict* the traffic matrix, and base its calculation of $cong_{t_0}$ on that prediction. Our prediction scheme is very simple, and is described below.

We assume that multi-path routing is not enabled on the network. The reason for this assumption will soon be made clear. If a node i measures traffic flowing through it from source s to destination d, and there is no multi-path routing, then i sees *all* the traffic from s to d. For i to build a proper traffic matrix, however, it needs to know the rates of all flows, not just those flows that travel through i. Another way of thinking about this is to realize that for a given node n, the traffic matrix at n, TM_n, has some elements that n can directly measure, and some elements that n cannot measure. For the elements that n can measure, n simply uses those measurements. In order to make those measurements, each node keeps for each source and destination pair a count of bytes transiting the node. Periodically (each second, in our scheme), rate counts are computed from the measured byte counts. For the other nodes in which this measurement scheme cannot be applied, n relies on relatively long-running averages that are periodically exchanged between nodes. In this way, each node learns the rates of flow between each node to every other node. New measurements can be combined with earlier learned advertisements with an exponentially decaying average. This is how the traffic matrix is estimated at each node, from these learned advertisements. Alternatively, network engineers might statically provision a network based on a calculated or expected traffic matrix. If this static information is available, it could be used as well.

Congestion Prediction

Using this estimated traffic matrix, and the topology learned from the routing protocol, we can estimate congestion throughout the network and choose VPN endpoints that avoid these. We show such an algorithm as Algorithm 2. The product of this algorithm is a mapping between the set of links and the "congestion ratio". If this number is greater than 1, then the link is congested. If it is less than 1, then in the steady state the link has spare capacity. A congestion ratio of exactly 1 means that the link has exactly enough bandwidth to transit its traffic.

After predicting congestion in the network, we choose as a VPN endpoint an intermediate node that is on the Intranet path between the source and the destination and immediately past any predicted congestion points. The primary reason for choosing VPN endpoints on the original Intranet path is to avoid routing loops.

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. If the frequency of traffic measurement updates is high (every few minutes), then this will work very well for congestion caused by the introduction of a high-bandwidth and long-lived flow, for example a multi-media stream or bulk-data backup operation. Given that multimedia streams are forgiving of increases in delay due to their ability to prebuffer data, they should not be hurt by the congestion prediction-based VPN endpoint selection. Additionally, bulk operations like off-line data backups will not be hurt either. For such a backup, the lack of packet drops will often be more helpful than an increase in end-to-end delay. The conditions in which this scheme will not work are ones in which the aggregate location-to-location traffic patterns change at a frequency faster than the traffic measurement messages are exchanged or can be detected. This would be characteristic of short, bursty traffic flows between locations.

In the following chapter, we evaluate the performance of both the delay-based and flow-based Delta protocols on several different topologies under different traffic workloads. Additionally, we also consider as a baseline algorithm the **ppnonly** protocol. **ppnonly** does not make use of the

Algorithm 2 Predict Congestion

{TOPO is an adjacency matrix representing the topology, TM is the Traffic Matrix}

{We begin by calculating the amount of traffic on each individual link}

linkStress ⇐ NEW Matrix {linkStress$_i, j$ will represent the traffic from i to j}

for $i = 0$ to $numNodes()$ **do**

  for $j = 0$ to $numNodes()$ **do**

    midpoint ⇐ i

    while midpoint ≠ j **do**

      nexthop ⇐ $TOPO(midpoint, j)$

      {for each network link, we add the contribution of traffic from node i to node j}

      $linkStress(midpoint, nexthop) \Leftarrow linkStress(midpoint, nexthop) + TM_{i,j}$

      midpoint ⇐ nexthop

    end while

  end for

end for

{LINKS is the set of links in the Intranet, BW represents the bandwidth of each link}

congRatio ⇐ NEW Set

{By dividing the amount of flow per link by its bandwidth, we get the congestion ratio}

for all $l \in LINKS$ **do**

  congRatio(l) ⇐ $\frac{linkStress(l)}{BW(l)}$

end for

Algorithm 3 FlowBasedTunnelEndpoint()

```
{TOPO is an adjacency matrix representing the topology}
{congRatio is the predicted congestion on each link}
{vpnEndpoint[i] will represent the next hop for destination i}
vpnEndpoint ⇐ NEW Array
for i = 0 to numNodes() do
  if myAddr() == i then
    continue
  end if
  {initially we will send the traffic directly to the Intranet}
  bestHop ⇐ myAddr()
  midpoint ⇐ myAddr()
  {we search for a point past any predicted congestion}
  while midpoint ≠ i do
    nexhop ⇐ TOPO(midpoint, i)
    if congRatio(midpoint, nexthop) > 1.0 then
      {there is congestion on the link (midpoint, nexthop), so we set the target past that link}
      bestHop ⇐ nexthop
    end if
    midpoint ⇐ nexthop
  end while
  {vpnEndpoint[i] is a point past any predicted congestion to node i}
  vpnEndpoint[i] ⇐ bestHop
end for
```

Internet at all, and forwards traffic only on the Intranet according to the internal routing protocol. This represents a standard corporate Intranet (that is not a hybrid network). Additionally, we also consider two simple VPN endpoint selection algorithms: **nexthop** and **lasthop**. The **nexthop** algorithm always chooses as a VPN endpoint the next hop along the Intranet path to the destination. The **lasthop** algorithm always sends diverted traffic directly to the ultimate destination.

The contribution of this work is a dynamic protocol for improving the quality of corporate networks. This improvement is based on reducing congestion on leased Intranet lines by forwarding this traffic outside the leased network and onto the public Internet. The Delta Routing protocol is able to choose the path that such diverted traffic takes using a combination of local information and long-term, infrequently exchanged global information.

2.2.3 An Example of Delta Routing

To motivate the need for a dynamic VPN endpoint selection algorithm, we consider an example based on the topology presented in Figure 2. We consider a traffic flow destined for node 5 that starts at node 1. Because of cross-traffic, the links between nodes 1 and 2, and nodes 3 and 4, are congested. Now consider that node 5 is located behind a trans-Atlantic link, and because of that, paths on the public Internet to node 5 have a higher than average delay, and high loss rate (due to high drop rates due to congested public peering points). Given this scenario, let us consider each of the above mentioned VPN endpoint selection algorithms. In each case, we consider the choice that node 1 will make in regards to choosing a VPN endpoint for traffic destined for node 5.

The first two algorithms, **nexthop** and **lasthop**, are static algorithms. By choosing node 2 (in the case of **nexthop**), traffic destined for node 5 will undergo additional congestion when it reaches node 3. At that point, that traffic will be diverted once again into the Internet to node 4, at which point it will proceed to node 5. With the **lasthop** algorithm, traffic entering the Internet at node 1 will be forwarded directly to node 5. While this avoids additional congestion at node 3, this forwarded traffic might suffer needlessly high drop rates at public peering points at the entrance to

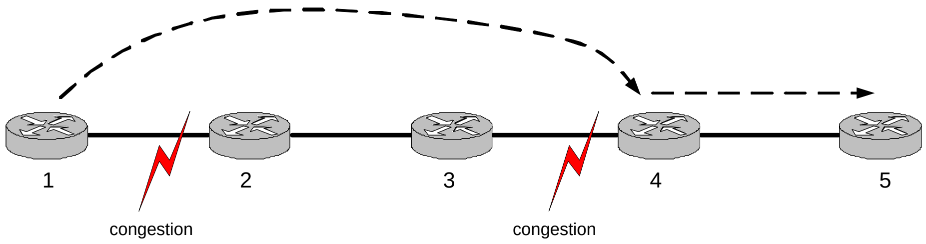

Figure 2: An example scenario

the trans-Atlantic link. Sending the traffic over the Intranet from node 4 to node 5 would be able to avoid those losses and provide better performance for the forwarded traffic.

Let us now consider the two dynamic algorithms. Delay-based Delta Routing would choose as a VPN endpoint the node that minimizes the end-to-end delay of forwarded traffic. Depending on tunnel delays, this could be node 2, 3, 4, or 5. In the event that it chooses nodes 2 or 3, the traffic would be subjected to additional congestion at node 3. In the event that it chooses node 5, it would subject the traffic to elevated drop and delay rates in a manner identical to **lasthop**. The Flow-based Delta routing protocol would not consider choosing nodes 2 or 3 as VPN endpoints, due to the congestion at node 3. As implemented, it would choose node 4 as the VPN endpoint, which means that the traffic would not be subjected to the public peering point on the path to node 5. Rather, the traffic would transit to node 5 over the guaranteed performance Intranet link between nodes 4 and 5. In this case, Flow-based Delta routing is able to make this decision if the cross traffic between nodes 3 and 4 persists for a time longer than the periodic advertisement rate that Flow-based Delta routing uses to establish TMs at each node. This would be the case if the cross traffic were due to a long-running video stream, for example.

This example, while simple, shows that a dynamic VPN endpoint selection algorithm is important to properly choose paths on Hybrid networks. Furthermore, it demonstrates that taking into consideration some global information can benefit path selection for forwarded traffic.

Chapter 3

Methodology

3.1 Methodology

Sending traffic through a Hybrid network consists of two processes: routing and forwarding. Routing consists of each node choosing for each possible destination a next hop both on the physical, private network, as well as the virtual, VPN-based n^2 network. The internal routing protocol such as OSPF[21] is responsible for choosing next hops on the physical network. The Delta Routing protocols (Delay-based and Flow-based) dynamically perform VPN endpoint selection at each node for each possible destination. These VPN selections are the set of next hops for the virtual network. As implemented, VPN endpoint selection occurs about once a second, and is calculated irrespective of any packets arriving at the node. Thus, the next hop to a particular destination is not calculated on a per-packet basis, but rather periodically at a certain interval.

While the routing function of a Hybrid network is responsible for choosing appropriate next hops on both the physical and virtual networks, the forwarding function is responsible for choosing which path is taken for a given packet. The forwarding function is performed on a packet-by-packet basis. For Flow-based Delta Routing, packets are always sent to the physical network as long as the queue to the physical link has spare capacity. Only when that queue is full are packets sent over the VPN tunnel chosen by the routing layer. The Delay-based Delta protocol can operate in two modes. As with Flow-based Delta Routing, the Delta-based protocol will forward traffic to the VPN network whenever the physical path is congested. The Delay-based protocol differs from the Flow-based protocol because it can also send packets to VPN tunnels if it calculates that doing so would reduce the packet's end-to-end delay (including queuing delay). In the experiments that follow, we consider both forwarding mechanisms. On the Linear Topology and the 43-node topology the Delay-based Delta routing protocol forwards traffic to the VPN tunnel whenever it calculates that doing so will reduce the end-to-end delay. On the algorithm-specific topologies, both the Delay-based and the Flow-based Delta routing protocols forward traffic to the VPN tunnels only when the physical queue is full.

To evaluate the Delta routing protocols, we implemented them in the *ns-2* simulator. We evaluate their performance, as well as the performance of two "static" algorithms on a set of six topologies. Unlike the Delta Routing protocols, which choose VPN endpoints based on network measurements, the static algorithms always make the same endpoint selections. The first static algorithm is the **lasthop** algorithm, which always forwards traffic directly to the destination. In constrast, the **nexthop** algorithm always chooses as its VPN endpoint the next hop on the physical network. The first topology we utilize is a simple linear topology. The next four experiments are performed on *application-antagonistic* topologies. Each of these topologies are specifically designed so that one of the algorithms considered will perform poorly on it. For example, on the algorithm-antagonistic topology for the **lasthop** algorithm, all VPN tunnels to the final hop in the network are much higher than for other nodes. For the Nexthop topology, there are several congested links in a row. Thus, the **nexthop** algorithm will forward traffic from one congested point to another. The reason for choosing these topologies is to see how all of the protocols respond under what would be considered the worst-case behavior of at least one of the protocols. The aim is to see if any of the protocols perform reasonably over many of these topologies.

Evaluating the Delta Routing protocol on a large, realistic topology would require a large number of Internet-connected nodes also connected via a set of dedicated lines. To do this, we simulate a network that is based on the Planet-lab[29] network. Planet-lab is a collection of over one hundred nodes distributed to 43 locations around the world. We construct a 43-node network in which node-to-node latencies are based on measurements taken from the Planet-lab network. To model the physical, dedicated network, we cluster the 43 nodes into a set of regions by geographic proximity (for example, nodes in Northern California, or nodes in England). For each region, we connect the nodes with relatively large links (20 or 45 Mbits/second) with relatively small delay. The regions are connected together with smaller, higher-delay links. This type of network is modeled as a transit-stub topology, and mimics the type of network that a corporate enterprise might construct to connect their locations together. We run the four different algorithms (as well as an experiment in which we do not make use of the VPN tunnels at all) over this large 43-node network.

The evaluation section that follows documents the results of these experiments, and shows the factors that affect the performance of various VPN endpoint selection algorithms.

Chapter 4

Evaluation

4.1 Experimental Framework

To test the efficacy of Flow-based Delta Routing (FDR), we constructed six different hybrid topologies and ran five different VPN-endpoint selection algorithms on them. The first two algorithms are **FDR** and the “**regular**” Delta routing protocol. The next two algorithms are **nexthop** and **lasthop**. Under the **nexthop** protocol, when congestion is detected at a point (when its link’s queue fills up), traffic is forwarded over a VPN tunnel to the next hop on the Intranet. In this case, traffic skips over the congested link and continues on its original path. Under the **lasthop** algorithm, traffic is sent through a VPN tunnel directly to the destination. Lastly, we also compare these results to the **ppnonly** algorithm, which never diverts traffic through VPN tunnels. **ppnonly** is thus a baseline that mimics a standard corporate network.

We run each of these algorithms on six different topologies. The first is a mostly linear topology that simulates one of the nodes being behind a set of bottleneck links. This might be the case if a corporate network has one or more locations in Asia or Europe. Next, we test the algorithms on a set of four “micro-benchmark” topologies. These topologies were hand-designed so that each of them should interact poorly with one of the algorithms. For example, the Lasthop topology is designed so that VPN tunnels heading to the destination have a very large delay compared to the

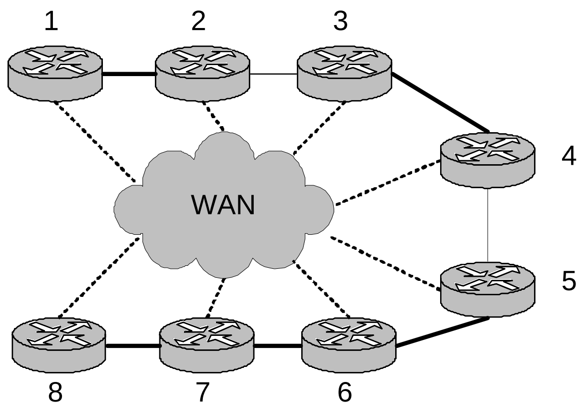

Figure 3: Linear topology with bottleneck links

other nodes. Continuing the above mentioned example, this would mimic a node in Asia or Europe connected to the Internet over a relatively slow link. The other three algorithm-antagonistic topologies target the **nexthop**, **regular**, and **FDR** topologies. 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.

Finally, we run the five different algorithms on a large, forty-three node topology based on the Planet-Lab network [29]. The details of this network are outlined in Section 4.4.

4.2 Linear Topology with Endhost Bottleneck Links

To begin our examination of Delay-based and Flow-based Delta Routing protocols, let us consider a simple topology consisting of eight nodes connected to each other in a linear fashion (Figure 3). In this example all links except for (2,3) and (4,5) have a capacity of 1.55Mbps and a delay of 10ms. Link (2,3) has a capacity of 1.2Mbps and link (4,5) has a capacity of 1.0Mbps. Additionally, there is a 50 packet queue before each link. In this example, there are three "background" traffic flows on the network which represent cross-traffic. Specifically, there is a 950Kbps constant-bitrate (CBR) UDP flow between nodes 4 and 5, a 1.1Mbps CBR flow between nodes 2

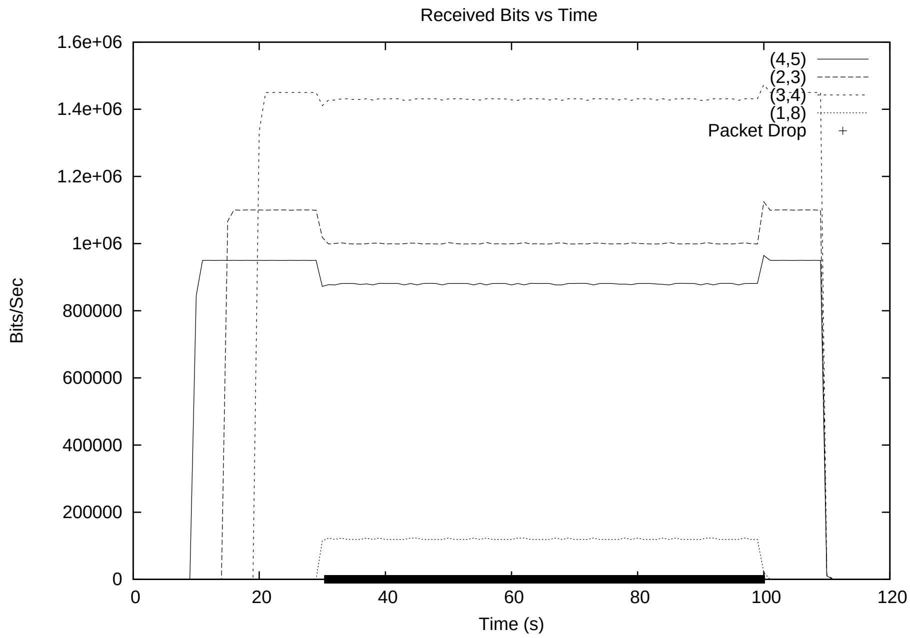

Figure 4: Packet reception rate under the **ppnonly** routing scheme

and 3, and a 1.45Mbps CBR UDP flow between nodes 3 and 4.

Thirty seconds into the simulation, a new flow is introduced. This new flow represents an unexpected flash traffic flow, for example a VoIP phone call, or a new video conference stream. This model of a flash traffic flow being introduced to a background set of flows will carry through all of the following examples. In the case of this linear topology, the newly introduced flow is a 200Kbps flow from node 1 to node 8. Figure 4 illustrates the reception rate of packets at the destination under the normal **ppnonly** routing scheme. At time $t = 30$, the newly added flow causes the aggregate rate of flow to exceed the capacity of the path between node 1 and node 8. Because of this, there is a large burst of packet loss during the time that the new flow is active. During this time, some of the traffic arrives at the destination, but there is a significant amount of packet loss. Once the flash traffic subsides, then the other flows return to their normal reception rate and the packet drop event ends.

Now consider Figure 5, which is a network running the Flow-based Delta routing scheme.

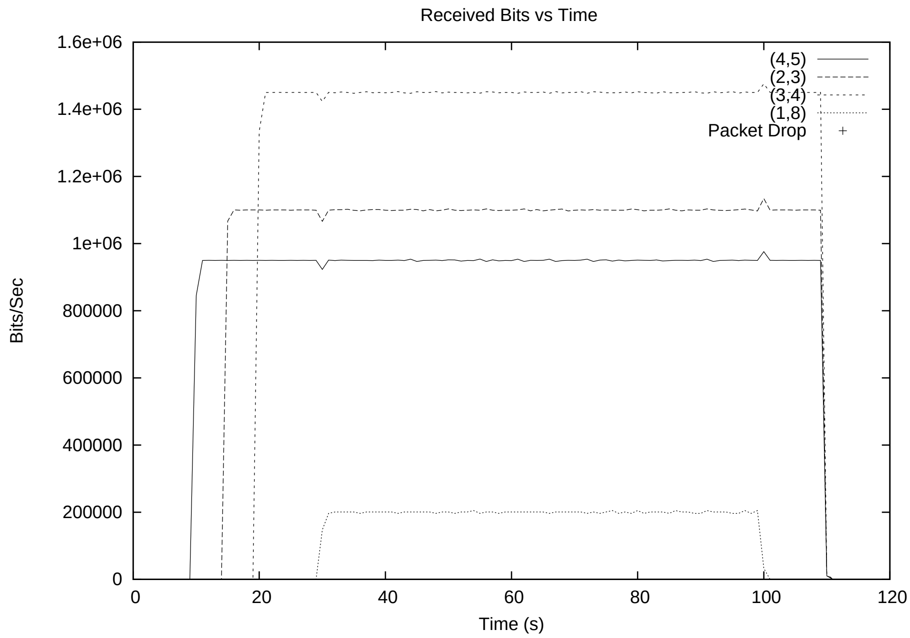

Figure 5: Packet reception rate under the **FDR** routing scheme

As before, at $t = 30$ seconds, a new 200Kbps flow is introduced to the system. Nodes along the path are monitoring the traffic flowing through them, and periodically they use these measurements to predict congestion and respond to directly measured congestion events. As the graph shows, this causes congestion in the network path. Instead of dropping the packets, though, the nodes in this scheme send those packets through VPN tunnels to a point further in the network path (each according to its algorithm). The short-term benefit of this is that these tunneled packets have another chance to arrive at the ultimate destination. The long-term concern is twofold: first, whether they will be subjected to further tunneling due to additional congested links; and second, whether the VPN endpoint is chosen as to minimize the end-to-end path. The first concern will be addressed by considering the overall packet drop rate. If packets are tunneled to a point in the network that is undergoing a congestion event, then they will either be retunneled, which will increase their end-to-end latency, or the distant node will exceed its ability to tunnel packets by congesting its ISP link. In that case, the packet will be dropped. The packet drop rate for each of the six topologies is presented in Table 1. In the case of the linear topology, only the **ppnonly** scheme suffers packet drops.

By examining Figures 6 and 7, we can see that on the linear topology, each of the algorithms (except **lasthop**) perform roughly as well as each other. The reason that **lasthop** does so poorly in this case is due to the very high delay incured by node 8's ISP. For each of the other algorithms, VPN endpoints are selected past the point of congestion, but before the final destination. For this reason, the packets are on the Intranet from node 7 to node 8, rather than on the much slower VPN tunnel to node 8. Figure 7 shows that except for the **lasthop** algorithm, each of the algorithms cause about the same maximum amount of end-to-end delay. Some of the topologies presented below do not share this behavior.

4.3 Algorithm-specific Topologies

We now consider a set of four topologies that were originally designed to interact poorly with each of the four dynamic VPN endpoint selection algorithms. For example, in the Nexthop topology, two consecutive links suffer from congestion. We would expect that upon detecting the first congested link, traffic will be forwarded over a VPN tunnel to the next hop. Unfortunately, this link also suffers from congestion, and so some traffic must be forwarded to the following link, and so forth. Each of these protocols is presented below, and the packet drop rate and mean and max end-to-end delays are presented.

Each of the Algorithm-specific topologies are based on the network described by Figure 8. 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).

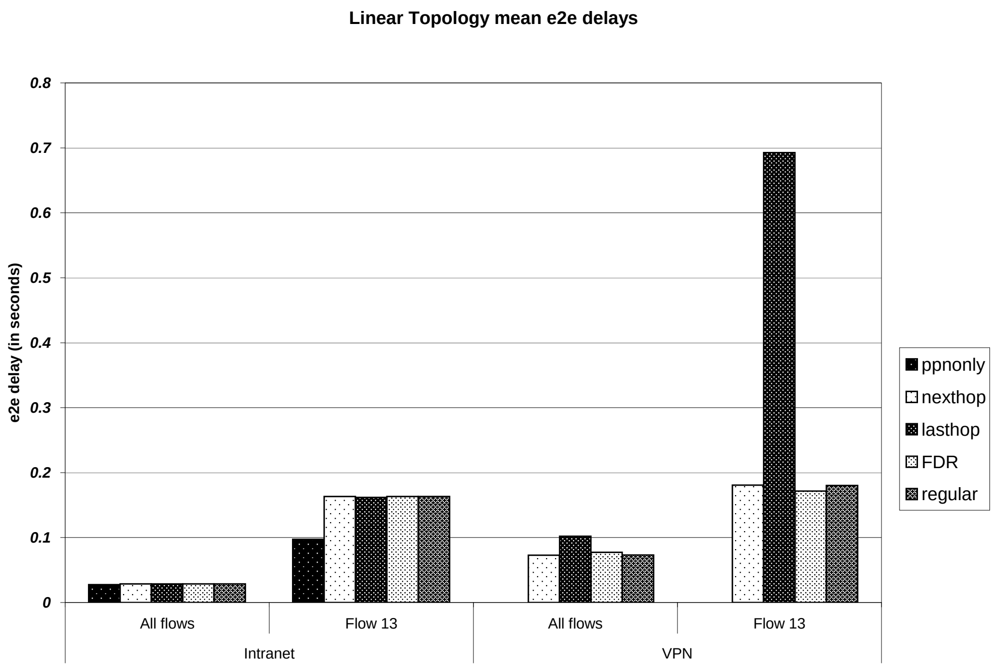

Figure 6: Mean e2e delays on the linear topology with bottleneck links

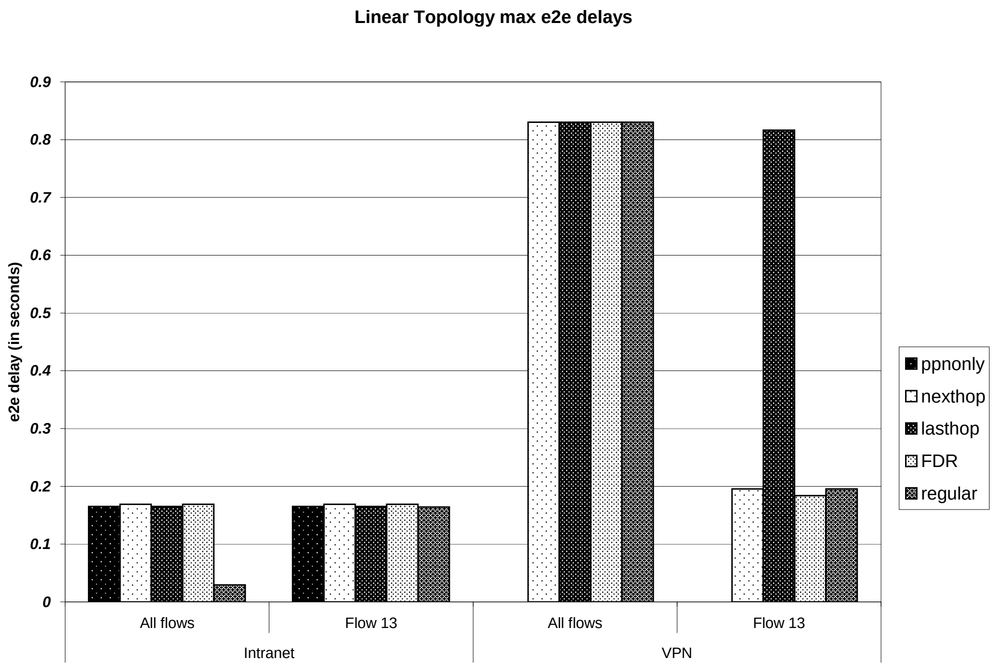

Figure 7: Max e2e delays on the linear topology with bottleneck links

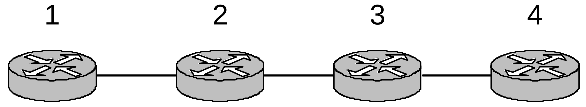

Figure 8: Algorithm-antagonistic topology

4.3.1 Lasthop Topology

The Lasthop topology is very simple, and shares some common traits with the Linear topology, presented above. 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 Intranet latency, in the Lasthop topology VPN tunnel delays to node 4 are several times longer.

As Table 1 indicates, none of the algorithms considered suffer from packet loss. In fact, only the *ppnonly* scheme dropped packets. The **FDR** algorithm 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. Figures 9 and 10 show that the **lasthop** algorithm suffers from the greatest end-to-end packet delay, followed by the **FDR**, **regular**, and **nexthop**, which all performed about as well as each other.

4.3.2 Nexthop Topology

The Nexthop topology is characterized by a consecutive set of congested links (specifically (1,2) and (2,3)) which interact poorly with the **nexthop** algorithm. Because **nexthop** always chooses the next hop as its VPN endpoint, packets are destined to suffer congestion multiple times (at least twice in this case). In looking at the packet drop rate given in Table 1, it can be seen that the static **ppnonly** scheme suffers the most packet drops, as expected. The other schemes are able to successfully avoid losing packets.

On this topology we see that the **nexthop** algorithm performs the worst since traffic must often traverse multiple VPN tunnels. Next is the **regular** algorithm. The two algorithms that performed the best were **lasthop** and **FDR**, which each made similar VPN endpoint selections. This demonstrates an behavior that will be brought out further in the evaluation, namely that because the **FDR** algorithm is able to choose VPN endpoints based on congestion in the network, it is often able to outperform both the **nexthop** and **lasthop** algorithms. On a highly congested topology **FDR** acts

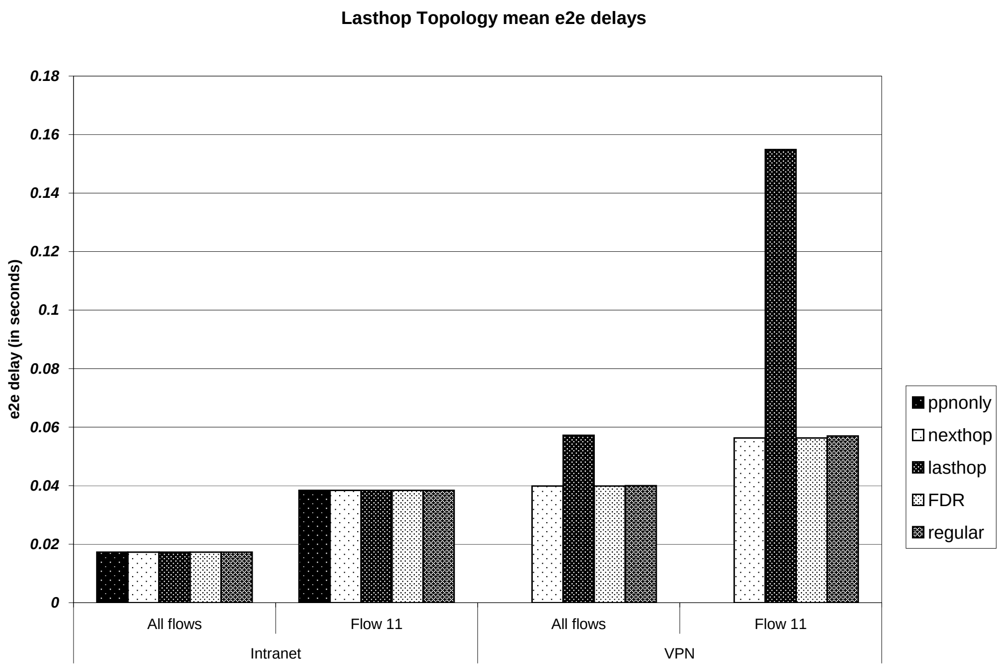

Figure 9: Mean e2e delays on the lasthop topology with bottleneck links

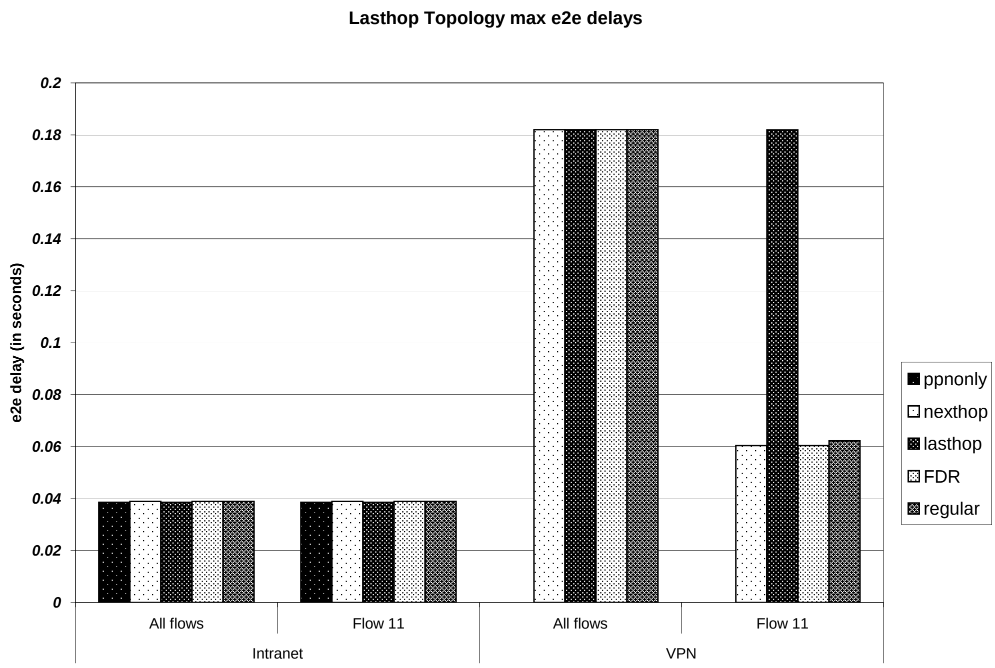

Figure 10: Max e2e delays on the lasthop topology with bottleneck links

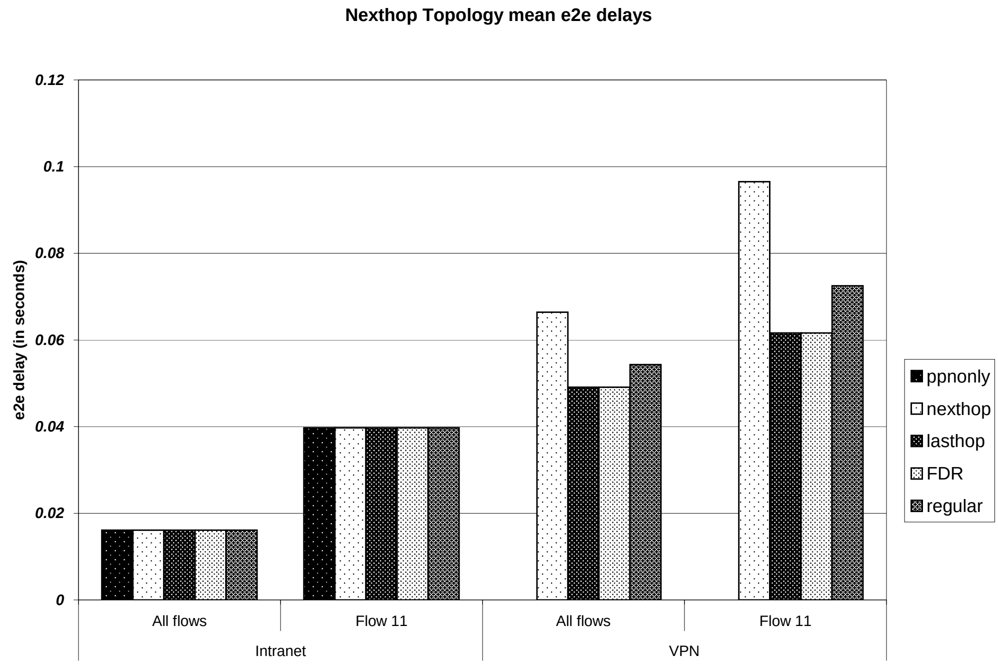

Figure 11: Mean e2e delays on the nexthop topology with bottleneck links

just like **lasthop**.

4.3.3 FDR Topology

In the FDR topology, there are two background traffic flows: one from node 1 to node 2, and one from node 2 to node 3. At time $t = 5s$, a traffic flow is introduced from node 1 to node 4. Unlike previous simulations, the **FDR** algorithm is not made aware of the flow between node 2 and node 3. 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

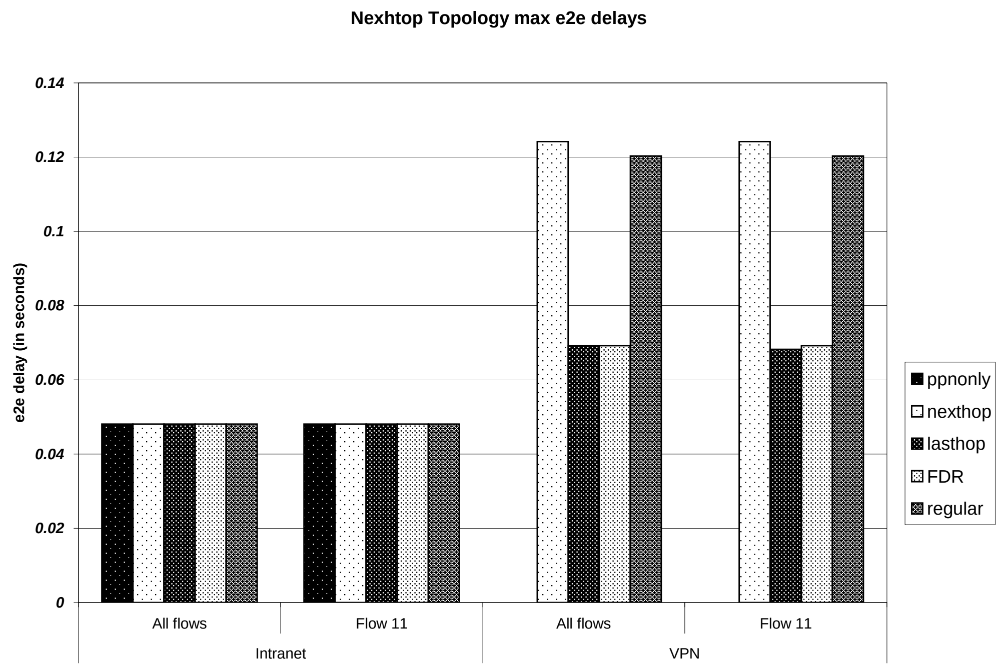

Figure 12: Max e2e delays on the nexthop topology with bottleneck links

as frequently or more frequently than the periodic messages could track, then this situation will develop. The motivation for choosing a periodic exchange rate in the order of a few minutes is that such a rate would ensure that long running flows (video, for example), that persist for longer than the exchange rate, would be compensated for in our scheme.

Figures 13 and 14 show that the **FDR** algorithm performed slightly worse than **lasthop** and **regular**. Because of the sequence of congested links, the **nexthop** algorithm performed poorly as expected. A situation that would hurt **FDR**'s performance in a greater way would be any topology in which 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 this small topology this effect is not very pronounced, but in a larger topology with larger end-to-end delays (and to a smaller extent larger variations in end-to-end delays) the difference in performance between **FDR** and other two algorithms that outperformed it would be greater. Again, this "blindness' to the additional congestion points would cause poor VPN endpoint selections to be made until the long-running TM averages could be updated to include the additional flows.

4.3.4 Regular Topology

The Regular topology 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 the **regular** algorithm 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.

Figures 15 and 16 show that in contrast to what was expected, **regular** performed rather well. The mean end-to-end delay for all tunneled traffic was very slighly lower for the **regular** algorithm, and the end-to-end delay for the measured flow (Flow 11) was somewhat higher. In this case **FDR** made similar selections as **lasthop**, and so the two performed similarly. In general, **regular** will perform much worse than other VPN endpoint selection algorithms whenever it chooses

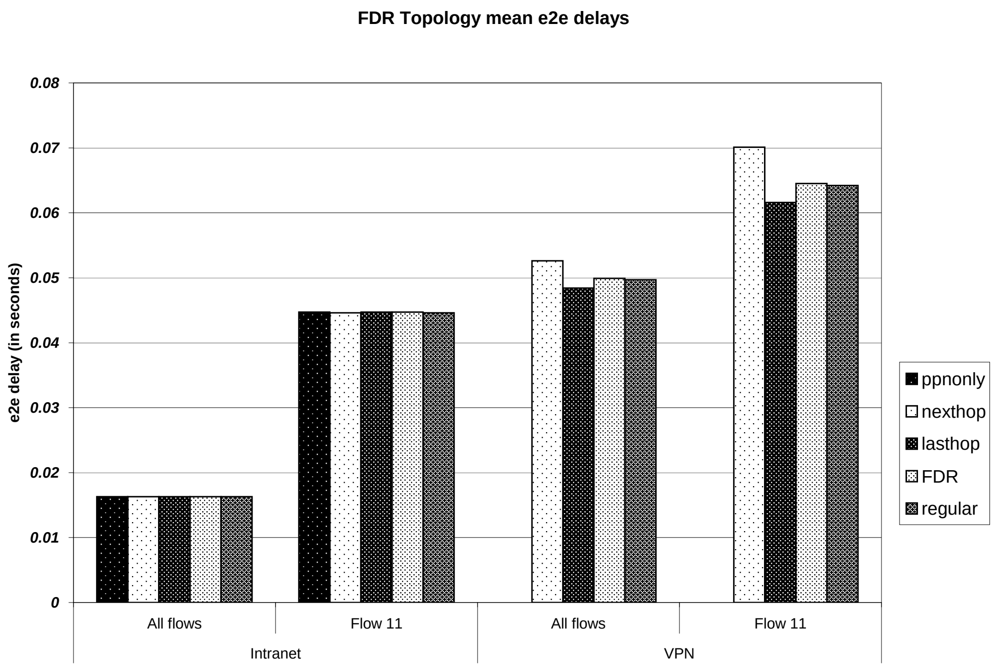

Figure 13: Mean e2e delays on the FDR topology with bottleneck links

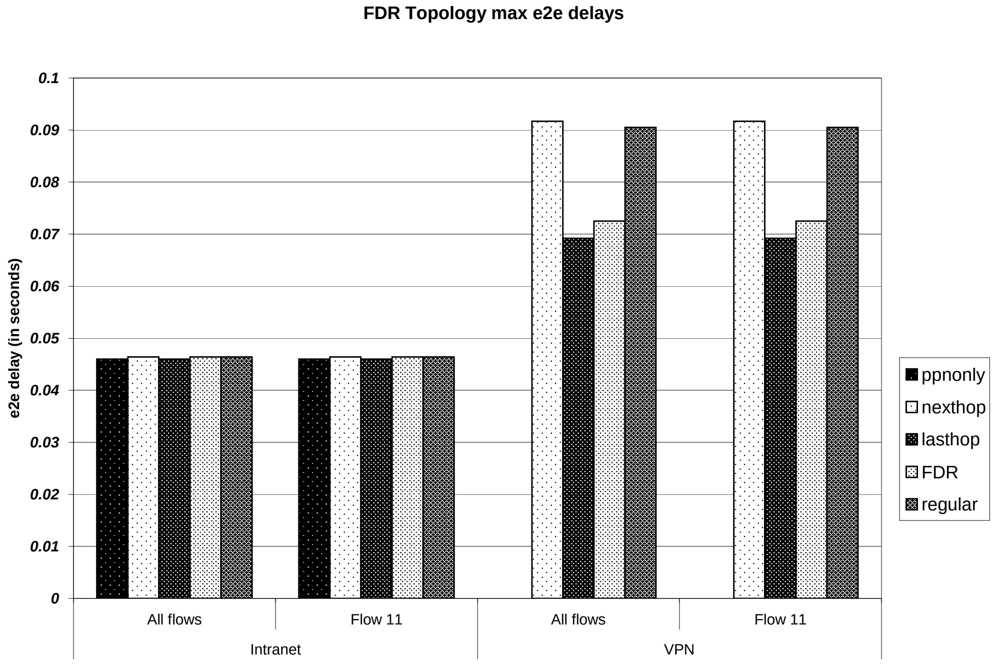

Figure 14: Max e2e delays on the FDR topology with bottleneck links

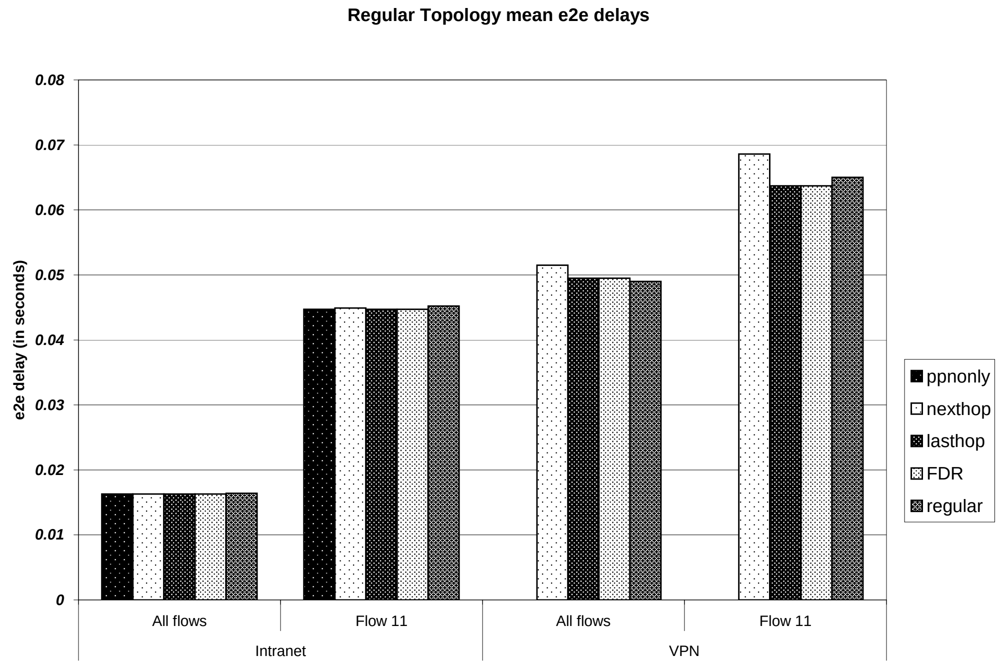

Figure 15: Mean e2e delays on the regular topology with bottleneck links

to send traffic to a node that is undergoing congestion. Since its choices are made using only local information, it can not include the remote node's congestion status in its computation. For this reason, it might choose a node that will force the forwarded traffic to be forwarded yet again upon arriving at the endpoint.

4.4 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,

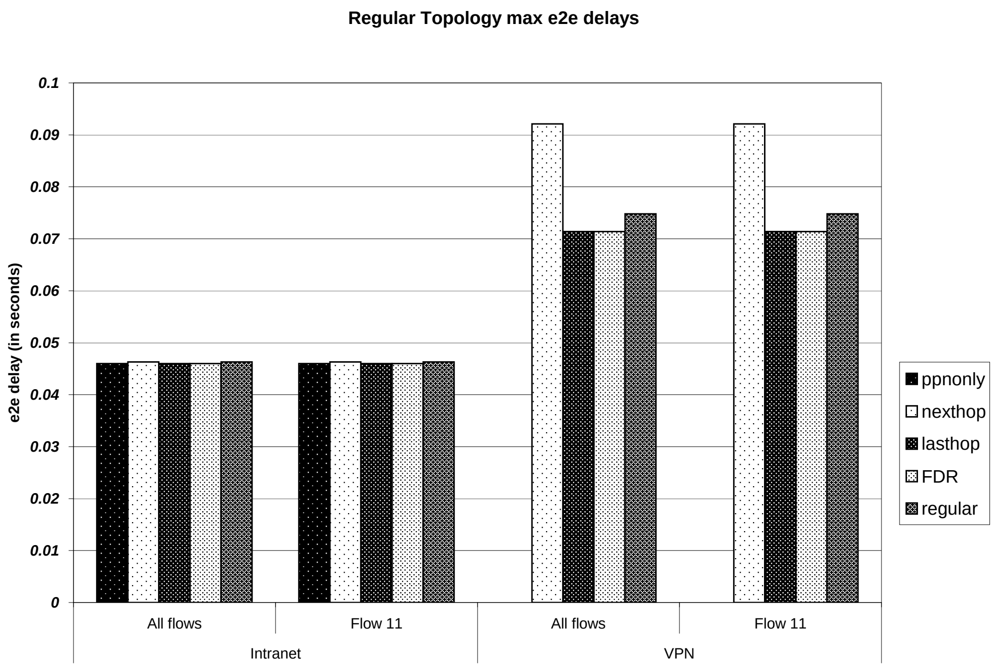

Figure 16: Max e2e delays on the regular topology with bottleneck links

hybrid, corporate 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 Planet-lab network [29]. Planet-lab is a collection of over one hundred machines distributed in forty-three different locations around the world. It was designed primarily to investigate overlay networks and distributed systems. Given that a hybrid corporate network consists of a set of n^2 Internet-based VPN tunnels, as well as a set of dedicated leased lines, we can use node-to-node delay measurements between each of the Planet-lab nodes as the values of the VPN tunnel delays. In our final topology, we use "all-pairs" measurements taken from the Planet-lab 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 an Intranet network that connects nodes geographically distributed just as the Planet-lab 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. The exact topology used, and the distribution of nodes to regions, is available from [6].

On top of this 43-node network there are a set of traffic flows, again simulating "normal" background traffic. As before, a new flow is introduced to the simulation that causes congestion. In this case, 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). As Figures 17 and 18 show, the end-to-end performance of the various algorithms is about even, with **regular** doing slightly better than the other algorithms. This might be somewhat expected, given that a variety of the conditions presented in the microbenchmark topologies above exist in the larger network. The main metric in which the protocols differ is in the packet drop rate. As expected, **ppnonly** suffers from the most packet drop events, usually at the ingress to a slow trans-Atlantic link. Both **nexthop** and **regular** suffer from a relatively low amount of loss compared to **nexthop**

algorithm	*Topology*					
	lasthop	regular	nexthop	FDR	linear	43-node
ppnonly	9740	17091	16737	17091	24214	260229
nexthop	0	0	0	0	0	119848
lasthop	0	0	0	0	0	37903
FDR	0	0	0	0	0	109511
regular	0	0	0	0	0	67541

Table 1: Packet drop events

and **FDR**, which exhibit a packet drop rate just in the middle of the set of algorithms considered. The **nexthop** algorithm chooses VPN endpoints very close to a current point of congestion, and as such, packets are often exposed to more than one congestion event between the original source and ultimate destination. The **FDR** protocol's relatively high drop rate is due to its choice of VPN endpoints. In several cases, **FDR** chooses a VPN endpoint that does ultimately result in additional congestion for slow bottleneck links. Based on this experiment, as well as the above mentioned "micro benchmarks", it is our conclusion that the **FDR** scheme is ideal in topologies where a congested link is followed by a relatively quiescent network segment. Furthermore, **FDR** is able to mimic both the **nexthop** as well as **lasthop** algorithms in topologies that are either heavily congested or very lightly congested (as needed). Delay-based Delta routing also performs well overall, especially when it utilizes a forwarding mechanism that is responsive to end-to-end delay as well as local queue overflows. Static VPN-endpoint selection algorithms do not perform well when applied to a heterogeneous set of topologies. Lastly, we found that utilizing VPN tunnels can dramatically decrease the packet drop rate in a network while providing good end-to-end performance.

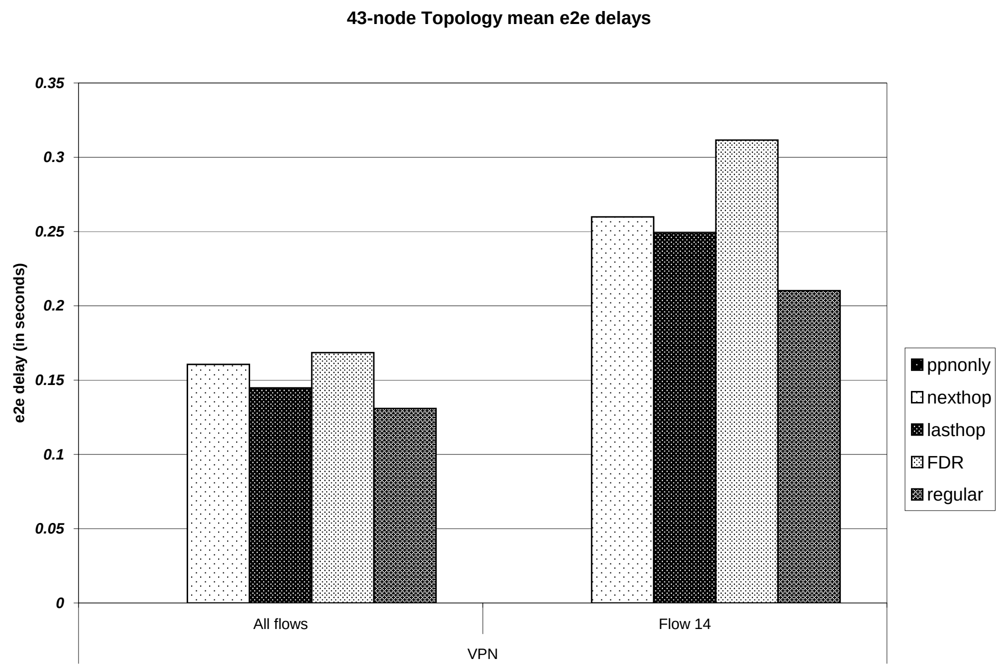

Figure 17: Mean e2e VPN delays on the 43-node topology with bottleneck links

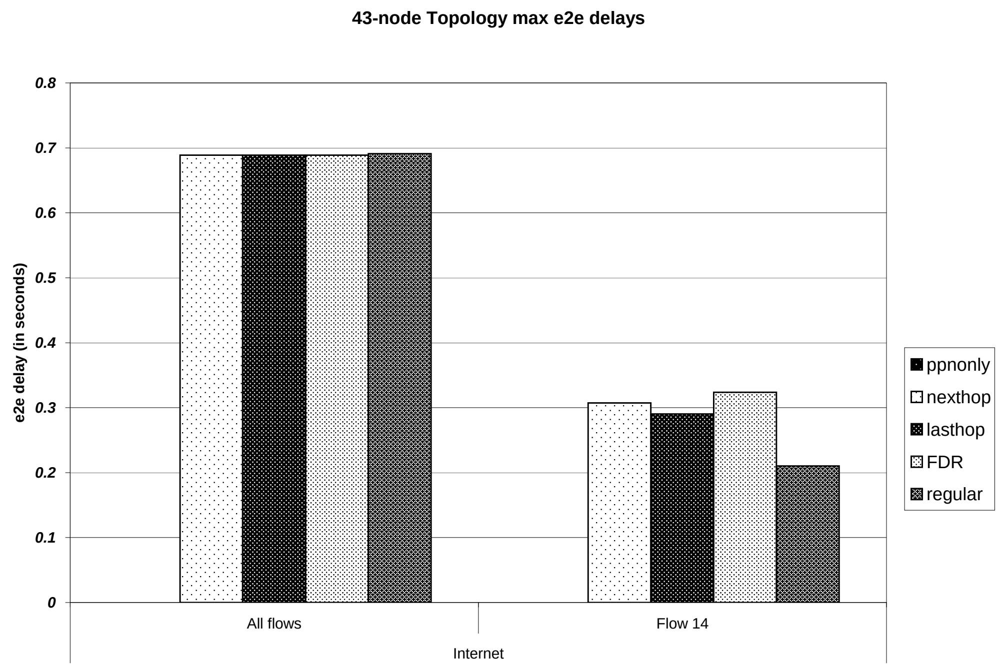

Figure 18: Max e2e VPN delays on the 43-node topology with bottleneck links

4.5 Summary of Results

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 dynamic 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.

Chapter 5

Related Work

Most current network routing protocols make path selection decisions using Dijkstra's Shortest Path Algorithm [7]. The mechanism used to propogate edge weights necessary for Dijkstra's algorithm can vary. Open Shortest Path First (OSPF) [21] is a link-state protocol in which edge weights adjacent to a node are broadcast throughout the network. The Routing Information Protocol (RIP) [14] is a distance-vector based protocol in which nodes send link information only to their neighbors. While the above protocols primarily optimize network paths to reduce end-to-end latency, there are other inter-Autonomous System (AS) protocols that are designed to additionally provide Quality of Service (QoS) as well.

One such protocol based on cell-switching is Asynchronous Transfer Mode (ATM) [12]. ATM is designed to natively support QoS mechanisms which could be used to prevent congestion in the network. Multi-Protocol Label Switching (MPLS) [8] [5] is a mechanism to enable IP-based networks to support more sophisticated per-flow path selection. One of MPLS's uses is assigning traffic flows to orthogonal paths to reduce congestion. This practice is often referred to as "Traffic Engineering".

Quality of Service-based routing and networking has been extensively studied. In the wide area, Integrated Services (IntServ) [3] [2] [16] [11], have been proposed to provide strict guarantees to traffic flows. Differentiated Services (DS) [2] [4] [30] [36] have also been proposed to allow network providers to classify flows into groups and subject each group to possibly different network characteristics and priorities. There have also been studies into ways to engineer network traffic to support QoS [9] [33] [37].

The use of Traffic Matrices for network provisioning is becoming increasingly common. The practice of estimating a network's traffic matrix from observed measurements is known as *Network Tomography*, and several research efforts have explored this subject. While the mechanism used in this work to estimate traffic matrices is very simple, more sophisticated techiniques are presented in [10], [13], [34], and [35]. Furthermore, the study of wide-area dynamics sheds some light into the stability and availability of Internet paths [32] [31].

Chapter 6

Conclusion

In this paper we present an architecture for improving the performance and reliability of corporate enterprise networks by making use of their connectivity to the public Internet via ISPs. These Hybrid networks can be made more efficient by appropriate selection of VPN endpoints. This paper presents two main mechanisms to perform this selection: the Delay-based and Flow-based Delta routing protocols. The Delay-based scheme worked well in a variety of topologies and traffic patterns. The Flow-based scheme also performed well, and was able to avoid forwarding traffic to points in the network undergoing congestion. We found that these two dynamic routing schemes outperformed static schemes over a variety of topologies. Making use of VPN tunnels to divert congestion traffic off of corporate Intranets dramatically reduces lost packets compared to using only the internal links. This reduction in packet drop rates will lead to better network application performance and behavior.

Acknowledgements

I would like to extend my deepest gratitude to my research advisor, Randy H. Katz, for his suppport during my graduate studies. Futhermore, I would like to thank Dr. Minwen Ji and the Systems Research Center at HP Labs for supporting this work as part of their internship program.

I could not have made it to this point, or to U.C. Berkeley, without the help and support of Prof. J Strother Moore and Prof. Robert S. Boyer (UT Austin), Karl Lehenbauer and Ellyn Jones (NeoSoft, Inc), and of course, my family, to which I owe so much.

Bibliography

[1] M. Allman. RFC 2581: TCP congestion control, Apr. 1999.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss. RFC 2475: An architecture for differentiated services, December 1998. Status: PROPOSED STANDARD.

[3] R. Braden, D. Clark, and S. Shenker. RFC 1633: Integrated services in the Internet architecture: an overview, June 1994. Status: INFORMATIONAL.

[4] E. Crawley, R. Nair, B. Rajagopalan, and H. Sandick. RFC 2386: A framework for QoS-based routing in the Internet, August 1998. Status: INFORMATIONAL.

[5] D. Awduche et al. Requirements for traffic engineering over mpls. Technical report, Internet-draft, draft-ietf-mpls-traffic-01.txt, Oct 1998.

[6] Simulation data and software. http://www.cs.berkeley.edu/g̃porter/delta/.

[7] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[8] E. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. Technical report, Internet-draft, draft-ietf-mpls-arch-01.txt, Mar 1998.

[9] Anwar Elwalid, Cheng Jin, Steven H. Low, and Indra Widjaja. MATE: MPLS adaptive traffic engineering. In *INFOCOM*, pages 1300–1309, 2001.

[10] A. Feldmann et al. Deriving traffic demands for operationsal ip networks: Methodology and experience. *IEEE/ACM Transactions on Networking*, June 2001.

[11] G. Apostolopoulos et al. Quality of service based routing: A performance perspective. In *ACM SIGCOMM '98*, pages 17–28, Vancouver, Canada, Aug. 1998.

[12] D. Ginsburg. *ATM: Solutions for Enterprise Internetworking*. Addison-Wesley, Reading, MA, 1996.

[13] O. Goldschmidt. ISP backbone traffic interference methods to support traffic engineering. In *Internet Statistics and Metrics Analysis (ISMA) Workshop*, San Diego, CA, Dec 2000.

[14] C. L. Hedrick. RFC 1058: Routing information protocol, June 1988. Updated by RFC1388, RFC1723 [17, 18]. Status: HISTORIC.

[15] Minwen Ji. Dial-controlled hash: Reducing path oscillation on multipath networks. Technical Report HPL-2003-98, HP Labs, May 2003.

[16] Q. Ma. *QoS Routing in the Integrated Services Networks*. PhD thesis, CMU-CS-98-138, Jan. 1998.

[17] G. Malkin. RFC 1388: RIP version 2 carrying additional information, January 1993. Obsoleted by RFC1723, RFC2453 [18, 19]. Updates RFC1058 [14]. Updated by RFC2453 [19], STD0056 [20]. Status: PROPOSED STANDARD.

[18] G. Malkin. RFC 1723: RIP version 2 — carrying additional information, November 1994. Obsoletes RFC1388 [17]. Obsoleted by RFC2453 [19]. Updates RFC1058 [14]. Updated by RFC2453 [19], STD0056 [20]. Status: DRAFT STANDARD.

[19] G. Malkin. RFC 2453: RIP version 2, November 1998. See also STD0056 [20]. Obsoletes RFC1388, RFC1723 [17, 18]. Status: STANDARD.

[20] G. Malkin. STD 56: RIP version 2, November 1998. See also RFC2453 [19]. Obsoletes RFC1723 [18].

[21] J. Moy. RFC 1131: OSPF specification, October 1989. Obsoleted by RFC1247 [24]. Status: PROPOSED STANDARD.

[22] J. Moy. RFC 1245: OSPF protocol analysis, July 1991. See also RFC1247, RFC1246 [24, 23]. Status: INFORMATIONAL.

[23] J. Moy. RFC 1246: Experience with the OSPF protocol, July 1991. See also RFC1247, RFC1245 [24, 22]. Status: INFORMATIONAL.

[24] J. Moy. RFC 1247: OSPF version 2, July 1991. See also RFC1246, RFC1245 [23, 22]. Obsoleted by RFC1583 [25]. Obsoletes RFC1131 [21]. Status: DRAFT STANDARD.

[25] J. Moy. RFC 1583: OSPF version 2, March 1994. Obsoleted by RFC2178 [26]. Obsoletes RFC1247 [24]. Status: DRAFT STANDARD.

[26] J. Moy. RFC 2178: OSPF version 2, July 1997. Obsoleted by RFC2328 [27]. Obsoletes RFC1583 [25]. Status: DRAFT STANDARD.

[27] J. Moy. RFC 2328: OSPF version 2, April 1998. See also STD0054 [28]. Obsoletes RFC2178 [26]. Status: STANDARD.

[28] J. Moy. STD 54: OSPF version 2, April 1998. See also RFC2328 [27].

[29] The Planet-Lab Network. http://www.planet-lab.org/.

[30] K. Nichols, S. Blake, F. Baker, and D. Black. RFC 2474: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 headers, December 1998.

[31] V. Paxson. End-to-End Routing Behavior in the Internet. *IEEE/ACM Transactions on Networking*, 5(5):601–615, Oct. 1997.

[32] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, Univ. of California - Berkeley, 1997.

[33] R. Guerin et al. Qos routing mechanisms and ospf extensions. Technical report, Internet-draft, draft-guerin-QoS-routing-ospf-03.txt, Jan 1998.

[34] C. Tebaldi and M. West. Bayesian inference of network traffic using link count data. *J. of the American Statistical Association*, pages 557–573, June 1998.

[35] Y. Vardi. Network tomography: Estimating source-destination traffic intensities from link data. *J. of the American Statistical Association*, pages 365–377, 1996.

[36] Y. Bernet et al. A framework for use of rsvp with diff-serv networks. Technical report, Internet-draft, draft-ietf-diffserv-rsvp-00.txt, June 1998.

[37] Z. Zhang et al. Qos extensions to ospf. Technical report, Internet-draft, draft-zhang-qps-ospf-01.txt, Sept 1997.