

# Scalable VPN Routing via Relaying

Changhoon Kim  
Princeton University  
chkim@cs.princeton.edu

Alexandre Gerber  
AT&T Labs–Research  
gerber@research.att.com

Carsten Lund  
AT&T Labs–Research  
lund@research.att.com

Dan Pei  
AT&T Labs–Research  
peidan@research.att.com

Subhabrata Sen  
AT&T Labs–Research  
sen@research.att.com

## ABSTRACT

Enterprise customers are increasingly adopting MPLS (Multiprotocol Label Switching) VPN (Virtual Private Network) service that offers direct any-to-any reachability among the customer sites via a provider network. Unfortunately this direct reachability model makes the service provider's routing tables grow very large as the number of VPNs and the number of routes per customer increase. As a result, router memory in the provider's network has become a key bottleneck in provisioning new customers. This paper proposes *Relaying*, a scalable VPN routing architecture that the provider can implement simply by modifying the configuration of routers in the provider network, without requiring changes to the router hardware and software. Relaying substantially reduces the memory footprint of VPNs by choosing a small number of hub routers in each VPN that maintain full reachability information, and by allowing non-hub routers to reach other routers through a hub. Deploying Relaying in practice, however, poses a challenging optimization problem that involves minimizing router memory usage by having as few hubs as possible, while limiting the additional latency due to indirect delivery via a hub. We first investigate the fundamental tension between the two objectives and then develop algorithms to solve the optimization problem by leveraging some unique properties of VPNs, such as sparsity of traffic matrices and spatial locality of customer sites. Extensive evaluations using real traffic matrices, routing configurations, and VPN topologies demonstrate that Relaying is very promising and can reduce routing-table usage by up to 90%, while increasing the additional distances traversed by traffic by only a few hundred miles, and the backbone bandwidth usage by less than 10%.

## Categories and Subject Descriptors

C.2.3 [Computer-Communication Network]: Network Operations; C.4 [Performance of Systems]: Design Studies

## General Terms

Management, Measurement, Performance, Reliability

## Keywords

VPN, Routing, Optimization, Measurement

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGMETRICS'08, June 2–6, 2008, Annapolis, Maryland, USA.  
Copyright 2008 ACM 978-1-60558-005-0/08/06 ...\$5.00.

## 1. INTRODUCTION

VPN service allows for enterprise customers to interconnect their sites via dedicated, secure tunnels that are established over a provider network. Among various VPN architectures, layer 3 MPLS VPN [14] offers direct any-to-any reachability among all sites of a customer, without requiring the customer to maintain full-mesh tunnels between each pair of sites. This any-to-any reachability model makes each customer VPN highly scalable and cost-efficient, leading to the growth of the MPLS VPN service at an unprecedentedly rapid pace. According to the market researcher IDC, the MPLS VPN market was worth \$16.4 billion in 2006 and is still growing fast [8]. By 2010, it is expected that nearly all medium-sized and large businesses in the United States will have MPLS VPNs in place.

The any-any reachability model of MPLS VPNs imposes a heavy cost on the providers' router memory resources. Each provider edge (PE) router in a VPN *provider network* (see, e.g., Figure 1a) is connected to one or more different customer sites, and each customer edge (CE) router in a site announces its own address blocks (i.e., routes) to the PE router it is connected to. To enable *direct* any-to-any reachability over the provider network, for each VPN, each PE router advertises all routes it received from the CE routers that are directly connected to it, to all other PEs in the same VPN. Then, the other PEs keep those routes in their VPN routing tables for later packet delivery. Thus, the VPN routing tables in PE routers grow very fast as the number of customers (i.e., VPNs) and the number of routes per customer increase. As a result, router memory space required for storing VPN routing tables has become a key bottleneck in provisioning new customers.

We give a simple example to illustrate how critical the memory management problem is. Consider a PE with a network interface card with OC-12 (622 Mbps) bandwidth that can be channelized into 336 T1 (1.544 Mbps) ports - this is a very common interface card configuration for PEs. This interface can serve up to 336 different customer sites. It is not unusual that a large company has hundreds or even thousands of sites. For instance, a large convenience store chain in the U.S. has 7,200 stores. Now, suppose the PE in question serves one retail store of the chain via one of the T1 ports. Since each of the 7,200 stores announces at least two routes (one for the site, and the other for the link connecting the site and the backbone), that single PE has to maintain at least 14,400 routes just to maintain any-any connectivity to all sites in this customer's VPN. On the other hand, a router's network interface has a limited amount of memory that is specifically designed for fast address look-up. Today's state-of-the-art interface card can store at most 1 million routes, and a mid-level interface card popularly used for PEs can hold at most 200 to 300K routes. Obviously, using 7.2%

(14,400/200K) of the total memory for a single site that accounts for only at most 0.3% of the total capacity (1 out of 336 T1 ports) leads to very low utilization; having only 14 customers that are similar to the convenience store can use up the entire interface card memory, while 322 other ports are still available. Even if interface cards with larger amounts of memory become available in the future, since the port-density of interfaces also grows, this resource utilization gap remains.

Fortunately, in reality, every customer site typically does not communicate with every other site in the VPN. This is driven by a number of factors including *i*) most networking applications today are predominantly client-server applications, and the servers (e.g., database, mail, file servers, etc.) are located at a few central locations for a customer, and *ii*) enterprise communications typically follow corporate structures and hierarchies. In fact, a measurement study based on traffic volumes in a large VPN provider’s backbone shows that traffic matrices (i.e., matrices of traffic volumes between each pair of PEs) in VPNs are typically very sparse, and have a clear hub-and-spoke communication pattern [12, 11]. We also observed similar patterns by analyzing our own flow-level traffic traces. Hence, PE routers nowadays install more routes than they actually need, perhaps much more than they frequently need.

This sparse communication behavior of VPNs motivates a router interface memory saving approach that *installs only a smaller number of routes* at a PE, while still *maintains any-to-any connectivity between customer sites*. In this paper, we propose *Relaying*, a scalable VPN routing architecture. Relaying substantially reduces the memory footprint of VPNs by *selecting a small number of hub PEs that maintain full reachability information, and by allowing non-hub PEs to reach other routers only through the hubs*. To be useful in practice, however, Relaying needs to satisfy the following requirements:

- **Bounded penalty:** The performance penalty associated with indirect delivery (i.e., detouring through a hub) should be properly restricted, so that the service quality perceived by customers does not get noticeably deteriorated and that the workload posed on the provider’s network does not significantly increase either. Specifically, both *i*) additional latency between communicating pairs of PEs, and *ii*) the increase of load on the provider network should be insignificant on average and be strictly bounded within the values specified in SLAs (Service Level Agreements) in the worst case.
- **Deployability:** The solution should be immediately deployable, work in the context of existing routing protocols, require no changes to router hardware and software, and be transparent to customers.

To bound the performance penalty and to reduce the memory footprint of routing tables at the same time, we need to choose a *small* number of hub PEs out of all PEs, where the hub PEs originate or receive *most* traffic within the VPN. Specifically, we formulate this requirement as the following optimization problem. For each VPN whose traffic matrices, topology, and indirection constraints (e.g., maximum additional latency, or total latency) are given, *select as small a number of hubs as possible, such that the total number of routes installed at all PEs is minimized, while the constraints on indirect routing are not violated*. Note that, unlike conventional routing studies that typically limit overall stretch (i.e., the ratio between the length of the actual path used for delivery and the length of the shortest path), we instead bound the additional (or total) latency of each *individual* path. This is because an overall stretch is often not quite useful in directly quantifying the performance impact on applications along each path, and hence hard to be derived from SLAs. Moreover, most applications are rather

tolerant to the small increase of latency, but the perceived quality of those applications drastically drops beyond a certain threshold which can be very well specified by an absolute maximum latency value, rather than a ratio (i.e., stretch).

To solve this optimization problem, we first explore the fundamental trade-off relationship between the number of hubs and the cost due to the relayed delivery. Then, we propose algorithms that can strictly limit the increase of individual path lengths and can reduce the number of hubs at the same time. Our algorithms exploit some unique properties of VPNs, such as sparse traffic matrices and spatial locality of customer sites. We then perform extensive evaluations using real traffic matrices, route advertisement configuration data, and network topologies of hundreds of VPNs at a large provider. The results show that Relaying can reduce routing table sizes by up to 90%. The cost for this large saving is the increase of individual communication’s unidirectional latency only by at most 2 to 3 ms (i.e., the increase of each path’s length by up to a few hundred miles), and the increase of backbone resource utilization by less than 10%. Moreover, even when we assume a full any-to-any conversation pattern in each VPN, rather than the sparse patterns that are monitored during a measurement period, our algorithms can save more than 60% of memory for moderate penalties.

This paper has four contributions: *i*) We propose Relaying, a new routing architecture for MPLS VPNs that substantially reduces memory usage of routing tables; *ii*) we formulate an optimization problem of determining a hub set, and assigning hubs to the remaining PEs in a VPN; *iii*) we develop practical algorithms to solve the hub selection problem; and *iv*) we extensively evaluate the proposed architecture and algorithms with real traffic traces, routing configuration, and topologies from hundreds of operational VPNs.

The rest of the paper is organized as follows. Section 2 explains the problem background and introduces desirable properties that a solution should have. Section 3 presents our measurement results that motivate Relaying, and Section 4 describes our Relaying approach. Section 5 investigates the solution space with a baseline Relaying scheme. Sections 6 and 7 formulate problems of finding practical Relaying configuration, propose algorithms to solve the problems, and evaluate the solutions. We discuss the implementation and deployment issues in Section 8, followed by related work in Section 9. Finally, we conclude the paper in Section 10.

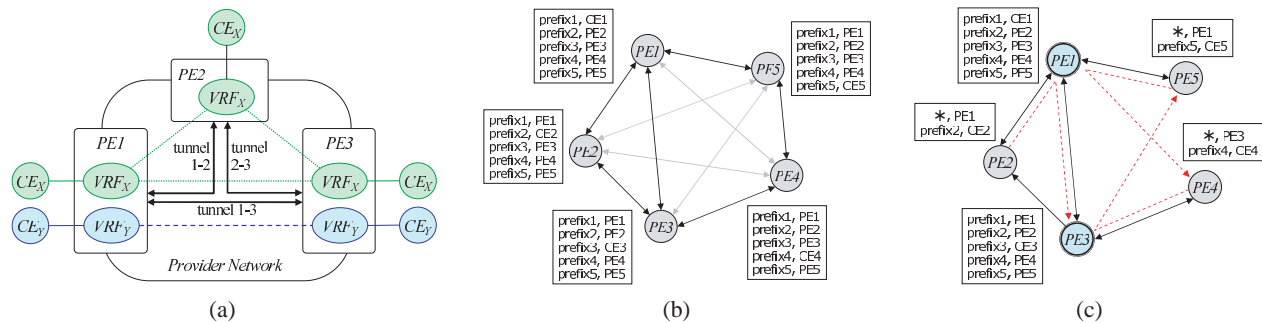
## 2. BACKGROUND

In this section, we first provide some background on MPLS VPN and then introduce terms we use in later sections. We also describe what properties a memory saving solution for VPNs should possess. Finally we briefly justify our Relaying architecture.

### 2.1 How MPLS VPN works

Layer 3 MPLS VPN is a technology that creates virtual networks on top of a shared MPLS backbone. As shown in Figure 1a, a PE can be connected to multiple Customer Edge (CE) routers of different customers. Isolating traffic among different customers is achieved by having distinct Virtual Routing and Forwarding (VRF) instances in PEs. Thus, one can conceptually view a VRF as a virtual PE that is specific to a VPN<sup>1</sup>. Given a VPN, each VRF locally populates its VPN routing table either with statically configured routes (i.e., subnets) pointing to incident CE routers, or with routes that are learned from the incident CE routers via BGP [13]. Then, these local routes are propagated to other VRFs in the same VPN via Multi-Protocol Border Gateway Protocol (MP-BGP) [1]. Once

<sup>1</sup>We also use “PE” to denote “VRF” when we specifically discuss about a single VPN.



**Figure 1: (a) MPLS VPN service with three PEs; two customer VPNs (X, Y) exist, (b) Direct reachability, (c) Reachability under Relaying.**

routes are disseminated correctly, each VRF learns all customer routes of the VPN. Then, packets are directly forwarded from a source to a destination VRF through a label-switched path (i.e., tunnel). PEs in a region are physically located at a single POP (Point of Presence) that houses all communication facilities in the region.

Figure 1b illustrates an example VPN provisioned over five PEs. Each PE’s routing table is shown as a box by the PE. We assume that PE  $i$  is connected to CE  $i$  which announces prefix  $i$ . PE  $i$  advertises prefix  $i$  to the other PEs via BGP, ensuring reachability to CE  $i$ . To offer the direct any-to-any reachability, each PE stores every route advertised by the other PEs in its local VRF table. In this example, thus, each PE keeps five route entries, leading to 25 entries in total across all PEs. The arrows illustrate a traffic matrix. Black arrows represent active communications between pairs of PEs that are monitored during a measurement period, whereas gray ones denote inactive communications.

Specifically, our Relaying architecture aims to reduce the size of a FIB (Forwarding Information Base), a data structure storing route entries. A FIB is also called a forwarding table and is optimized for fast look-up for high speed packet forwarding. Due to performance and scalability reasons, routers are usually built with several FIBs each of which is located in a very fast memory on a line card (i.e., network interface card). Unfortunately, the size of a line-card memory is limited, and increasing its size is usually very hard due to various constraints, such as packet forwarding rate, power consumption, heat dissipation, spatial restriction, etc. For example, some line-card models use a special hardware, such as TCAM (Ternary Content Addressable Memory) or SRAM [3], which is much more expensive and hard to be built in a larger size than regular DRAMs are. Even if a larger line-card memory was available, upgrading all line cards in the network with the large memory may be extremely costly. In MPLS VPN, a VRF is a virtual FIB specific to a VPN and resides in a line-card memory along with other VRFs configured on the same card. Beside the VRFs, line-card memory also stores packet filtering rules, counters for measurement, and sometimes the routes from the public Internet as well, which collectively make the FIB-size problem even more challenging.

## 2.2 Desirable properties of a solution

To ensure usefulness, a router memory saving architecture for VPNs should satisfy the following requirements.

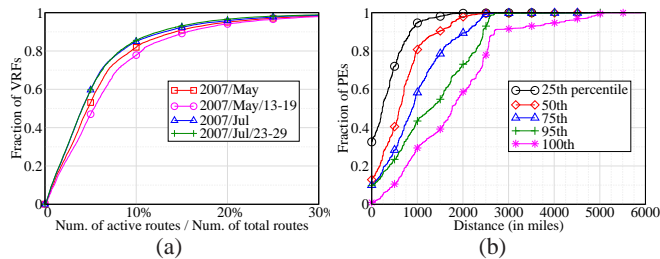
- 1. Immediately deployable:** Routing table growth is an imminent problem to providers; a solution should make use of router functions (either in software or hardware) and routing protocols that are available today.
- 2. Simple to implement:** A solution must be easy to design and implement. For management simplicity, configuring the solution should be intuitive as well.
- 3. Transparent to customers:** A solution should not require modifications to customer routers.

We deliberately choose Relaying as a solution because it satisfies these requirements. Relaying satisfies goal 1 because the provider can implement Relaying only via router configuration changes (see Section 8 for details). It also meets goal 3 since a hub maintains full reachability, allowing spoke-to-spoke traffic to be directly handled at a hub without forwarding it to a customer site that is directly connected to the hub. Ensuring goal 2, however, shapes some design principles of Relaying which we will discuss in the following sections. Here we briefly summarize those details and justify them.

Relaying classifies PEs into just two groups (hubs and spokes) and applies a simple “all-or-one” table construction policy to the groups, where hubs maintain “all” routes in the VPN, and spokes store only “one” default route to a hub (the details are in Section 4). Although we could save more memory by allowing each hub to store a disjoint fraction of the entire route set, such an approach inevitably increases complexity because the scheme requires a consistency protocol among PEs.

For the same reason, we do not consider incorporating cache-based optimizations. When using route caching, each spoke PE can store a small fraction of routes (in addition to the default route, or without the default route) that might be useful for future packet delivery. Thus any conversation whose destination is found in the cache does not take an indirect path. Despite this benefit, a route caching scheme is very hard to implement because we have to modify routers, violating goal 1. Specifically, we need to design and implement *i*) a resolution protocol to handle cache misses, and *ii*) a caching architecture (e.g., route eviction mechanism) running in router interface cards. Apart from the implementation issues, the route caching mechanism itself is generally much harder to correctly configure than Relaying is, violating goal 2. For example, to actually reduce memory usage, we need to fix a route cache’s size. However, a fixed-sized cache is vulnerable to a sudden increase of the number of popular routes due to the changes in the customer side or malicious attempts to poison a cache (e.g., scanning). To avoid thrashing in these cases we have to either dynamically adjust cache size, or have to allow some slackness to buffer the churns; neither is satisfactory because the former introduces complexity, and the latter lowers memory saving effect.

Goal 2 also leads us to another important design decision, namely individual optimization of VPNs. That is, in our Relaying model, a set of Relaying configuration (i.e., the set of hubs) for a VPN does not depend on other VPNs. Thus, for example, we do not select a VRF in a PE as a hub at the expense of making other VRFs in the same PE spokes, neither do we choose a VRF as a spoke to make other VRFs in the same PE hubs. This design decision is critical because VPNs are dynamic. If we allowed the dependency among different VPNs, having a new VPN customer or deleting an existing customer might alter the Relaying configuration of other VPNs, leading to a large re-configuration overhead. Moreover, this independence condition also allows network admin-



**Figure 2:** (a) CDFs of the proportion of active prefixes in a VRF, (b) CDFs of the distance to the  $i$ -th percentile closest VRF

istrators to customize each VPN differently by applying different optimization parameters to different VPNs.

### 3. UNDERSTANDING VPNS

In this section, we first briefly describes the data set used throughout the paper. Then we present our measurement results from a large set of operational VPNS. By analyzing the results, we identify key observations that motivate Relaying.

#### 3.1 Data sources

**VPN configuration, VRF tables, and network topology:** We use configuration and topology information of a large VPN service provider in the U.S. which has, at least, hundreds of customers. VPNS vary in size and in geographical coverage; smaller ones are provisioned over a few PEs, whereas larger ones span over hundreds of PEs. The largest VPN installs more than 20,000 routes in each of its VRFs. Specifically, from this VPN configuration set, we obtain the list of PEs with which each VPN is provisioned, and the list of prefixes each VRF advertises to other VRFs. We also obtain the list of routes installed in each VRF under the existing routing configuration. From the topology, we obtain the location of each PE and POP, the list of PEs in each POP, and inter-POP distances.

**Traffic matrices:** We use traffic matrices each of which describes PE-to-PE traffic volumes in a VPN. These matrices are generated by analyzing real traffic traces captured in the provider backbone over a certain (usually longer than a week) period. The traffic traces are obtained by monitoring the links of PEs facing the core routers in the backbone using Netflow [4]. Thus, the source PE of the flow is obvious, while the destination is also available from the tunnel end point information in flow records. Unless otherwise specified, the evaluation results shown in the following sections are based on a week-long traffic measurements obtained in May, 2007.

#### 3.2 Properties enabling memory saving

Through the analysis of the measurement results, we make the following observations about the MPLS VPNS. These properties allow us to employ Relaying to reduce routing tables.

**Sparse traffic matrices:** A significant fraction of VPNS exhibit “hub-and-spoke” traffic pattern, where a majority of PEs (i.e., spokes) communicate *mostly* with a small number of highly popular PEs (i.e., hubs). Figure 2a shows the distributions of the number of active prefixes (i.e., destination address blocks that are actually used during a measurement period) divided by the number of total prefixes in a VRF. We measure the distributions during four different measurement periods, ranging from a week to a month. The curves show that, for most VRFs, the set of active prefixes is much smaller than the set of total prefixes. Across all measurement periods, roughly 80% (94%) of VRFs use only 10% (20%) of the total prefixes stored. The figure also confirms that the sets of active prefixes are stable over different measurement periods. By processing

these results, we found out that the actual amount of memory required to store the active route set is only 3.9% of the total amount. Thus, if there was an ideal memory saving scheme that precisely maintain only those prefixes that are used during the measurement period, such a scheme would reduce memory usage by 96.1%. This number sets a rough guideline for our Relaying mechanism.

**Spatial locality of customer sites:** Sites in the same VPN tend to be clustered geographically. Figure 2b shows the distributions of the distance from a VRF to its  $i$ -th percentile closest VRF. For example, the 25th percentile curve shows that 80% of VRFs have 25% of the other VRFs in the same VPN within 630 miles. According to the 50th percentile curve, most (81%) VRFs have at least half of the other VRFs in the same VPN within 1,000 miles. Thus, a single hub can serve a large number of nearby PEs, leading to the decrease of additional distances due to Relaying.

**PE’s Freedom to selectively install routes:** A PE can choose not to store and advertise every specific route of a VPN to a CE as long as it maintains reachability to all the other sites (e.g. via a default route). Indeed, this does not affect a CE’s reachability to all other sites because a CE’s only way to reach other sites is via its adjacent PE(s) of the same (and sole) VPN backbone. Furthermore, this CE does not have to propagate all the routes to other downstream customers. However, a CE might still be connected to multiple PEs for load-balancing or backup purpose. In that case, the same load-balancing or backup goals are still achieved if all the adjacent PEs are selected as hubs or all are selected as spokes at the same time so that all the PEs announce the same set of routes to the CE. Note that this property does not hold for the routers participating in the Internet routing, where it is common for customers to be multi-homed to multiple providers or to be transit providers themselves.

## 4. OVERVIEW OF RELAYING

The key properties of VPN introduced in the previous section collectively form a foundation for Relaying. In this section, we first define the Relaying architecture, and then introduce detailed variations of the Relaying mechanism.

### 4.1 Relaying through hubs

In Relaying, PEs are categorized into two different groups: *hubs* and *spokes*. A hub PE maintains full reachability information, whereas a spoke PE maintains the reachability for the customer sites that are directly attached to it and a *single default route* pointing to one of the hub PEs. When a spoke needs to deliver packets destined to non-local sites, the spoke forwards the packets to its hub. Since every hub maintains full reachability, the hub that received the relayed packets can then directly deliver them to correct destinations. Multi-hop delivery across hubs is not required because every hub maintains the same routing table.

This mechanism is illustrated in Figure 1c. Assuming the traffic pattern shown in Figure 1b is stable, one may choose PE1 and PE3 as hubs. This leads to 16, rather than 25, route entries in total. Although the paths of most active communications remain unaffected (as denoted by solid edges), this Relaying configuration requires some communications (dotted edges) be detoured through hubs, offering *indirect* reachability. This indirect delivery obviously inflates some paths’ length, leading to the increase of latency, additional resource consumption in the backbone, and larger fate sharing. Fortunately, reducing this side effect is possible when one can build a set of hubs that originates or receive most traffic within the VPN. Meanwhile, reducing the memory footprint of routing tables requires the hub set to be as small as possible. In the following sections, we show that composing such a hub set is possible.

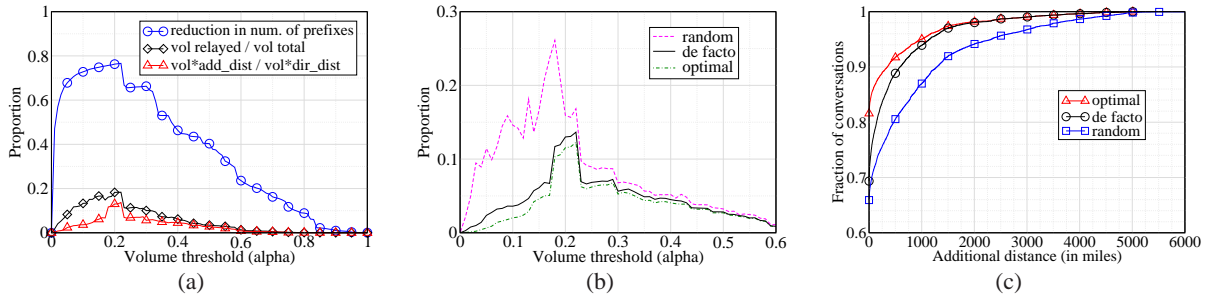


Figure 3: (a) Gain and cost (de facto asgn.), (b) Sum of the products of volume and additional distance, (c) CDF of additional distances ( $\alpha = 0.1$ )

## 4.2 Hub selection vs. hub assignment

Relaying is composed of two different sub-problems: *hub selection* and *hub assignment* problems. Given a VPN, a *hub selection* problem is a decision problem of selecting each PE in the VPN as a hub or a spoke. On the other hand, a *hub assignment* problem is a matter of deciding which hub a spoke PE should use as its default route. A spoke must use a single hub consistently because, by definition, a PE cannot change its default route for each different destination. To implement Relaying, we let each hub advertise a default route (i.e., 0.0.0.0/0) to spoke PEs via BGP. Thus, in practice, the BGP routing logic running at each PE autonomously solves the hub assignment problem. Since all updates for the default route are to be equivalent for simplicity, each PE chooses the closest hub in terms of IGP (Interior Gateway Protocol) distance. We call this model the *de facto* hub assignment strategy.

In order to assess the effect of the de facto strategy on path inflation, we compare it with some other hub assignment schemes, including random, optimal, and algorithm-specific assignment. The *random* assignment model assigns a random hub for each non-local destination. In the *optimal* assignment scheme, we assume that each PE chooses the best hub (i.e., the one minimizing the additional distance) for each non-local destination. Note that this model is impossible to realize because it requires global view on routing. Finally, the *algorithm-specific* assignment is the assignment plan that our algorithm generates. This plan is realistic because it assumes a single hub per spoke, not per destination.

## 5. BASELINE PERFORMANCE OF RELAYING

To investigate fundamental trade-off relationship between the gain (i.e., memory saving) and cost (i.e., increase of path lengths and the workload in the backbone due to detouring), we first explore a simple, light-weight strategy to reduce routing tables. Despite its simplicity, this strategy saves router memory substantially with only moderate penalties. Note that the Relaying schemes we propose in later sections aim to outperform this baseline approach.

### 5.1 Selecting heavy sources or sinks as hubs

The key problem of Relaying is building a right set of hubs. Fortunately, spatial locality of traffic matrices hints us that forcing a spoke PE to communicate only through a hub might increase memory saving significantly, without increasing the path lengths of most conversations. Thus, we first investigate the following simple hub selection strategy, namely *aggregate volume-based hub selection*, leveraging the sparsity of traffic matrices.

For a PE  $p_i$  in VPN  $v$ , we measure the aggregate traffic volume to and from the PE. We denote  $a_i^{in}$  to be the aggregated traffic volume received by  $p_i$  from all customer sites directly connected to  $p_i$ , and  $a_i^{out}$  to be the aggregated traffic volume sent by  $p_i$  to all customer sites directly attached to  $p_i$ . In VPN  $v$ , if  $a_i^{in} \geq \alpha \sum_j a_j^{in}$  or

$a_i^{out} \geq \alpha \sum_j a_j^{out}$  where  $\alpha$  is a tunable parameter between 0 and 1 inclusively, then we choose  $p_i$  as a hub in  $v$ .

Although we could formulate this as an optimization problem to determine the optimal value of  $\alpha$  (for a certain VPN or for all VPNs) minimizing a multi-objective function (e.g., a weighted sum of routing table size and the amount of traffic volume relayed via hubs), this approach lead us to two problems. First, it is hard to determine a general, but practically meaningful multi-objective utility function especially when each of the objectives has a different meaning. Second, the objectives (e.g., memory saving) are not convex, making efficient search impossible. Instead, we perform numerical analysis with varying values of  $\alpha$  and show how table size and the amount of relayed traffic volume varies across different  $\alpha$  values. Since there are hundreds of VPNs available, exploring each individual VPN with varying  $\alpha$  values broadens the solution space impractically large. Thus we apply a common  $\alpha$  value for all VPNs.

### 5.2 Performance of the hub selection

**Performance metrics:** To assess Relaying performance, we measure four quantities: *metric-i*) the number of routing table entries reduced, *metric-ii*) the amount of traffic that is indirectly delivered through hubs, *metric-iii*) the sum of the products of traffic volume and additional distance by which the traffic has to be detoured, and *metric-iv*) the additional distance of each conversation's forwarding path. For easier representation, we normalize the first three metrics. *Metric-i* is normalized by the total number of routing entries before Relaying, *metric-ii* is normalized by the amount of total traffic in the VPN, and *metric-iii* is normalized by the sum of the products of traffic volume and direct (i.e., shortest) distance. We consistently use these metrics throughout the rest of the paper.

The meanings of these metrics are as followings. *Metric-i* quantifies our scheme's *gain* in memory saving, whereas *metric-ii*, *iii* and *iv* denote its *cost*. Specifically, *ii* and *iii* show the increase of workload on the backbone. On the other hand, *iv* shows the latency inflation of individual PE-to-PE communications. Note that we measure the latency increase in distance (i.e., miles) because, in the backbone of a large tier-one network, propagation delay dominates a path latency. Due to the speed of light and attenuation, a mile in distance roughly corresponds to 11.5 usec of latency in time. Thus, increasing a path length by 1000 (or 435) miles lead to the increase of unidirectional latency roughly by 11.5 (5) msec.

**Relaying results:** Figure 3a shows the gain and cost of the aggregate volume-based hub selection scheme across different values of the volume threshold  $\alpha$ . As soon as we apply Relaying (i.e.,  $\alpha > 0$ ), all three quantities increase because the number of hubs decreases as  $\alpha$  increases. Note, however, that the memory saving increases very fast, whereas the amount of relayed traffic and its volume-mile product increases modestly. If we assume a sample utility function that is an equally weighted sum of the memory saving and the relayed traffic volume, the utility value (i.e., the gap between the gain and the cost curves) is maximized where  $\alpha$  is

around 0.1 to 0.2. When  $\alpha$  passes 0.23, however, the memory saving begins to decrease fast because a large value of  $\alpha$  fails to select hubs in some VPNs, making those VPNs revert to the direct reachability architecture between every pair of PEs. This also makes the cost values decrease as well.

Figure 3b shows how different hub assignment schemes affect the cost (specifically, the increase of workload on the backbone manifested by the sum of the products of volume and additional distance). Note that we do not plot the gain curve because it remains identical regardless of which hub assignment scheme we use. First, the graph shows that the overall workload increased by Relaying with either the de facto or the optimal assignment is generally low (less than 14% for any  $\alpha$ ). Second, the de facto assignment only slightly increases the workload on the backbone (around 2%) in the sweet spot ( $0.1 < \alpha < 0.2$ ), compared to the optimal (but impractical) scheme. The increase happens because the de facto scheme forces a spoke to use the closest hub consistently, and that closest hub might not be the spoke's popular communication peer. Nevertheless, this result indicates that choosing the closest hub is effective in reducing the path inflation.

Although the sum of the volume-mile products is reasonably small, the increased path lengths can be particularly detrimental to some *individual traffic flows*. Figure 3c shows, for all communicating pairs in all VPNs, how much additional distances the Relaying scheme incurs. The figure shows latency distributions when using Relaying (with  $\alpha = 0.1$ ) for three different hub assignment schemes: optimal, de facto, and random. For example, when using Relaying with the de facto assignment scheme, roughly 70% of the communicating pairs still take the shortest paths, whereas around 94% of the pairs experience additional distances of at most 1000 miles (i.e., the increase of unidirectional latency by up to 11.5 msec). Unfortunately this means that some 6% of the pairs suffer from more than 1000 miles of additional distances, which can grow in the worst case larger than 5000 miles (i.e., additional 60 msec or more unidirectionally). To those communications, this basic Relaying scheme might be simply unacceptable, as some applications' quality may drastically drop. Unfortunately, the figure also shows that even the optimal hub assignment scheme does not help much in reducing the particularly large additional path lengths. *To remove the heavy tail, we need a better set of hubs.*

## 6. LATENCY-CONSTRAINED RELAYING

Relaying requires spoke-to-spoke traffic to traverse an indirect path, and therefore increases paths' latency. However, many VPN applications such as VoIP are particularly delay-sensitive and can only tolerate a strictly-bounded end-to-end latency (e.g., up to 250 ms for VoIP). SLAs routinely specify a tolerable maximum latency for a VPN, and violations can lead to adverse business consequences, such as customers' loss of revenue due to business disruptions and corresponding penalties on the provider.

The simple baseline hub selection scheme introduced in Section 5 does not factor in the increase of path latencies due to relaying. Thus, we next formulate the following optimization problem, namely *latency-constrained Relaying (LCR) problem*, of which goal is to minimize the memory usage of VPN routing tables subject to a *constraint on the maximum additional latency of each path*. Note that we deliberately bound individual paths' additional latency, rather than the overall stretch, because guaranteeing a certain hard limit in latency is more important for applications. For example, increasing a path's latency from 30 msec (a typical coast-to-coast latency in the U.S.) to 60 leads to the stretch of only 2, whereas the additional 30 msec can intolerably harm a VoIP call's quality. On the other hand, increasing a path's latency from 2 msec

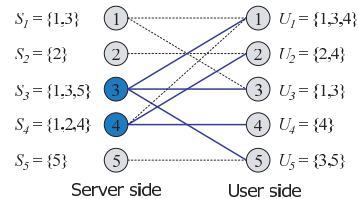


Figure 4: A sample serve-use relationship

to 10 may be nearly unnoticeable to users, even though the stretch factor in this case is 5.

### 6.1 LCR Problem formulation

We first introduce the following notation. Let  $P$  denote the set of PE routers  $P = \{p_1, p_2, \dots, p_n\}$  in VPN  $v$ . We define two matrices: *i*) Conversation matrix  $C = (c_{i,j})$  that captures the historical communication between the routers in  $P$ , where  $c_{i,j} = 1$  if  $i \neq j$  and  $p_i$  has transmitted traffic to  $p_j$  during the measurement period, and  $c_{i,j} = 0$  otherwise; and *ii*) latency matrix  $L = (l_{i,j})$  where  $l_{i,j}$  is unidirectional communication latency (in terms of distance) from  $p_i$  to  $p_j$ .  $l_{i,i} = 0$  by definition. Let  $H = \{h_1, \dots, h_m\}$  ( $m \leq n$ ) be a subset of  $P$  denoting the hub set. Finally, we define mapping  $M : P \rightarrow H$  that determines a hub  $h_j \in H$  for each  $p_i \in P$ .

LCR is an optimization problem of determining a smallest  $H$  (i.e., hub selection) and a corresponding mapping  $M$  (i.e., hub assignment), such that in the resulting Relaying solution, every communications between a pair of VRFs adhere to the maximum allowable additional latency (in distance) threshold  $\theta$ . Formally,

$$\begin{aligned} \min \quad & |H| \\ \text{s.t.} \quad & \forall s, d \text{ whose } c_{sd} = 1, \\ & l_{s,M(s)} + l_{M(s),d} - l_{s,d} \leq \theta \end{aligned}$$

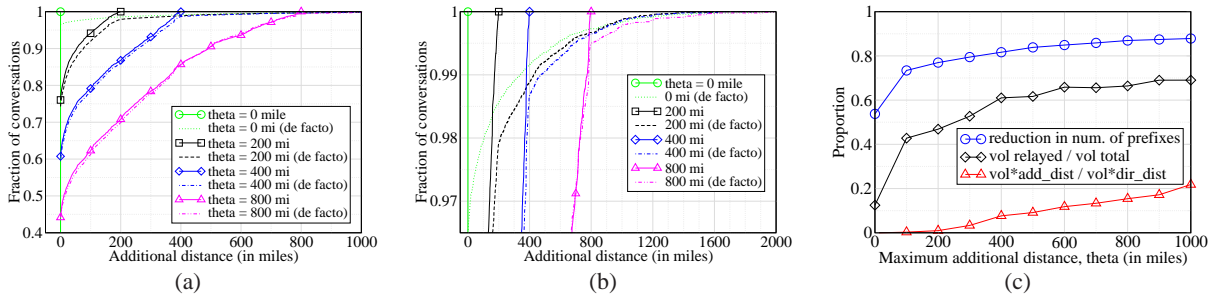
Other variations of the above formulation include bounding either the maximum total one-way distance, or both the additional and the total distances. In this paper, we do not study these variations due to the following reasons. First, bounding the additional distance is a stricter condition than bounding only the total distance is. Thus, our results in the following sections provide lower bounds of memory saving and upper bounds of indirection penalties. Second, when bounding total and additional distances, the total distance threshold must be larger than the maximum direct distance. However, this maximum direct distance often results from a small number of outlier conversations (e.g., communication between Honolulu and Boston in the case of the U.S.), making the total distance bound ineffective for most common conversations.

Considering the any-to-any reachability model of MPLS VPNs, we could accommodate the possibility that any PE can potentially communicate with any other PEs in the VPN, even if they have not in the past. Thus, we can solve the LCR problem using a *full-mesh* conversation matrix  $C^{full} = (c_{i,j}^{full})$ , where  $\forall i, j$  ( $i \neq j$ )  $c_{i,j}^{full} = 1$ ,  $c_{i,i}^{full} = 0$ . There is trade-off between using the usage-based matrices ( $C$ ) and full-mesh matrices ( $C^{full}$ ). Using  $C^{full}$  imposes stricter constraints, potentially leading to lower memory saving. The advantage of this approach, however, is that the hub selection would be oblivious to the changes in communication patterns among PEs, obviating periodical re-adjustment of the hub set.

Unfortunately, the LCR problem is NP-hard, and we provide this proof in the extended version of this paper [10]. Hence, we propose an approximation algorithm explained in the following subsection.

### 6.2 Algorithm to solve LCR

In this section, we outline our solution to the LCR problem. The pseudo-code of this algorithm is also given in our extended ver-



**Figure 5: LCR performance with usage-based  $C$**  (a) CDF of additional distances, (b) CDF of additional distances (zoomed in), (c) Gain and cost

sion [10]. We solve a LCR problem through a two-staged approach. The first stage builds *serve-use* relationships among PEs. Given a pair of PEs  $(p_i, p_j) \in P$ , we say that  $p_i$  can *serve*  $p_j$ , or conversely, that  $p_j$  can *use*  $p_i$ , if all conversations in  $C$  from  $p_j$  to other PEs in the VPN can be routed via  $p_i$  as a hub without violating the latency constraint. For each PE  $p_i$  in  $P$ , we build two sets: *i*) the *serve* set  $S_i$  composed of PEs that  $p_i$  can serve as a hub, and *ii*) the *use* set  $U_i$  composed of PEs that  $p_i$ , as a spoke, can use for relaying. The serve-use relationships among PEs can be represented as a bipartite graph. Figure 4 shows a sample serve-use relationship graph of a VPN with five PEs. Each node on the left bank represents  $p_i \in P$  in its role as a possible hub, and on the right bank, each node represents  $p_j$  in its role as a potential spoke. The existence of serve-use relationship between  $(p_i, p_j) \in P$  is represented by an edge between  $p_i$  on the left bank and  $p_j$  on the right bank.

Our original LCR problem now reduces to finding the smallest number of nodes on the left bank (i.e., hubs) that can serve every node on the right bank. This is an instance of the set cover problem, which is proven to be NP-complete [9], and we use a simple greedy approximation strategy to solve this. At each step, the LCR algorithm *i*) greedily selects  $p_i$  from the left bank whose serve set  $S_i$  is the largest, *ii*) remove  $p_i$  from the left bank, *iii*) remove all  $p_j$  ( $j \in S_i$ ) from the right bank and update the mapping function  $M$  to indicate that  $p_j$ 's assigned hub is  $p_i$ , and *iv*) revise the serve-use relationships for the remaining PEs in the graph. The above step is repeated until no PE remains on the right bank.

The LCR algorithm can be easily extended to solve the alternative problems mentioned above (i.e., bounding total latency, or both additional and total latency) only by re-defining the semantics of “serve” and “use” relationship. Note also that the LCR algorithm assigns a single hub for each spoke PE, and each spoke PE is assumed to use the single hub consistently for all packets. Thus, the hub assignment plan generated by the LCR algorithm is implementable, if not as simple as the de facto assignment scheme based on BGP anycasting as described in Section 4. For example, a VPN provisioning system can configure each spoke PE’s BGP process to choose a specific route advertisement from a corresponding hub.

### 6.3 Solutions with usage-based matrices

We run the LCR algorithm individually for each VPN. The results shown in later sections are the aggregate of individual solutions across all VPNs.

**Bounding additional distances:** Figure 5a shows the CDFs of additional distances of all conversation pairs in  $C$  across all VPNs for varying  $\theta$  values. Solid lines show the additional distance distributions when using the algorithm-specific hub assignment decisions (i.e., assignment plans computed by the LCR algorithm), whereas the dotted lines represent the distributions when spoke PEs choose the closest hubs in terms of the IGP metric (i.e., the de facto hub assignment). Note, however, that for a given  $\theta$ , both solid and dotted

curves are generated with the same hub set. For some spoke PEs, the hub determined by the LCR algorithm might be different from the closest hub that the de facto assignment scheme would choose. Thus, when one uses the hub sets selected by the LCR algorithm accompanied with the de facto hub assignment scheme, some pairs of PEs might experience additional distances larger than  $\theta$ . This is why solid lines always conform the  $\theta$  values, whereas dotted lines have tails extending beyond theta.

However, the fraction of PE pairs experiencing additional distances larger than  $\theta$  is small. We re-scale Figure 5a to magnify the tails and present it as Figure 5b. For a reasonable value of  $\theta$  (e.g., additional 400 miles, which are commensurate with 5 msec increase in latency), only 1.3% of PE pairs experience additional distances larger than  $\theta$ . Meanwhile, the fraction of pairs exposed to unbearably larger additional distances (e.g., 1000+ miles, or more than 11.5 msec in latency) is only 0.2%, which amounts to a few tens of pairs. These results suggest that in practice, administrators can use the simpler de facto assignment scheme in general and can configure only those small number of exceptional PEs as per specific assignment plans dictated by the LCR algorithm.

**Memory saving:** We next investigate the memory saving from our latency constrained hub selection. Figure 5c shows the gain (i.e., memory saving) and cost (i.e., the increase of workload in the backbone) under LCR for a range of  $\theta$ . Both the gain and cost curves increase with  $\theta$  because a larger value of  $\theta$  makes it possible for a smaller number of hubs to serve all the spokes.

The results convey a number of promising messages. For example, to achieve roughly 80% memory saving, a conversation (i.e., VRF pairs) need to tolerate additional distance of only up to 320 miles, which corresponds to the increase of unidirectional latency by just 3 ms. Moreover, when conversations can bear at most 1000 miles of path inflation, Relaying can reduce routing table memory by roughly 88%. Note that this amount of memory saving is only 8% worse than that of the ideal memory saving scheme mentioned in Section 3.2, and is better than that of the aggregate volume-based scheme (Figure 3a) with any choices of the volume threshold  $\alpha$ .

A surprising result is that, even if we do not allow any additional distance (i.e.,  $\theta = 0$ ), the relaying scheme can reduce routing table memory consumption by around a hefty 54%. By specifically analyzing these penalty-free cases, we found out three reasons for this huge, free-of-charge benefit. First, a significant number of PEs communicate mostly with a hub PE. In the case of the traffic whose source or destination is a hub, our relaying scheme does not introduce any additional distance penalty as long as the hubs are correctly identified and assigned. This case accounts for roughly 45% of the entire penalty-free conversations. Second, a hub can sometimes be located on the shortest path from a source to a destination (38%). For example in the United States, relaying traffic between San Diego and San Francisco through Los Angeles might not incur any detour in effect because physical connectivity between the two cities might be already configured that way. Third, for availability

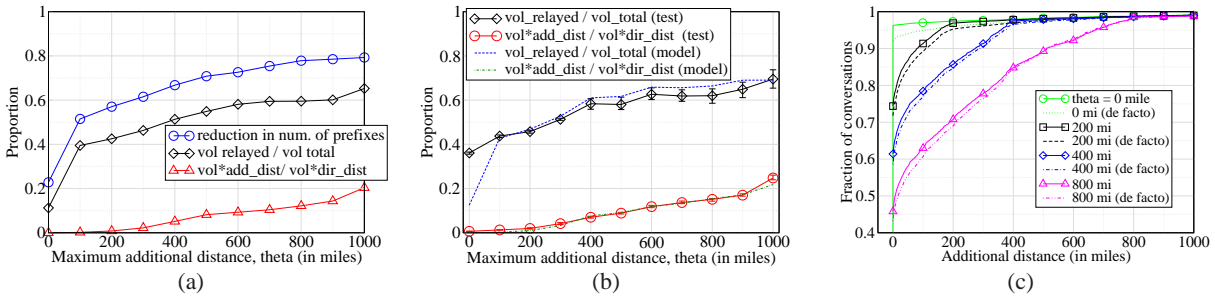


Figure 6: (a) LCR performance with  $C^{full}$ , (b) Robustness results (costs), (c) Robustness results (CDF of additional distances with  $C$ )

purposes, a VPN is often provisioned over multiple PEs located at the same POP (17%). Thus if one of the PEs in a POP is chosen as a hub, all the other PEs at the same POP can use the hub PE for relaying and avoid any (practically meaningful) distance penalties.

**Indirection penalty:** Figure 5c shows that the latency-constrained Relaying requires substantially more traffic to be relayed compared to the base line relaying scheme shown in Figure 3a. (Note that we use the same metrics described in Section 5.2.) This is because the LCR algorithm does not take into account the relative traffic volumes associated with the different PE-to-PE conversations while making either the hub selection or hub assignment decisions; the LCR algorithm selects a PE as a hub as long as the PE can relay the largest number of other PEs without violating the latency constraint. However, Figure 5c also shows that the sum of the relayed volume and additional distance products is a relatively small compared to the corresponding sum of the volume and direct distance products. This is because, even when relaying is needed, the algorithm limits the additional distances below the small  $\theta$  values. Hence, the *practical impact* of relaying (e.g., the increase of resource usage in the provider network) is much less severe than it is suggested by the sheer amount of relayed traffic. Also, we confirmed that using the de facto hub assignment, rather than the LCR algorithm’s hub assignment plan, increases the aggregate costs very little. This is because the LCR algorithm does not necessarily choose heavy sources or sinks as hubs, leaving only little room to improve/worsen the indirection penalties via different hub assignments.

## 6.4 Solutions with full-mesh matrices

We next consider the performance of the LCR when using the full-mesh conversation matrices. Figure 6a shows the gain and the cost curves. While we select the hubs based on the full-mesh matrices, to evaluate the penalties due to relaying we use the historical PE-to-PE conversations (including volumes) that are monitored during the measurement period (May 13 - 19, 2007).

The results are encouraging, even though the conversation matrices are much denser in this case. At the expense of incurring additional distances of up to roughly 480 miles (corresponding only to roughly 5 ms in unidirectional latency), we can reduce the memory consumption by nearly 70%. Interestingly, we can still save roughly 23% of memory even with no additional distance. This is because, given a PE, it is sometimes possible to have a hub lying on every shortest path from the PE to each other PE. For example, on a graph showing the physical connectivity of a VPN, if a spoke PE has only one link connecting itself to its neighbor, and the neighbor is the hub of the spoke PE, delivering traffic through the neighbor does not increase any paths’ lengths originating from the spoke PE. We also note that the aggregate costs (i.e., relayed traffic volume and its volume-mile product) are slightly reduced compared to the results derived from usage-based matrices. This is because hub sets for full-mesh matrices are bigger than those for

usage-based matrices. We also confirmed that the lengths of individual paths are correctly bounded within  $\theta$  for all cases.

For network administrators, these full-mesh results are particularly encouraging for the following reasons. First, even when customers’ networking patterns drastically change from today’s hub-and-spoke style to a more peer-to-peer-like style (e.g., by pervasively deploying VoIP or P2P-style data sharing applications), the relaying scheme can still save significant amount of memory. Second, it has the additional attraction that the relaying solution needs to be computed and implemented only once and for all. There would be no need to track the changing conversation matrices, unlike the usage-based case.

## 6.5 Robustness

We next explore how our solution performs under traffic dynamics. Figure 6b shows the results when we apply a *fixed* solution set derived from the traffic measurements in a particular *model* week (May 13 - 19, 2007) to the traffic for 8 different *test* weeks in May to August, 2007. We assume that VRFs added later on after the model week maintains full reachability.

The solid curves in Figure 6b depict the aggregate cost during the test weeks. We apply the usage-based solutions – both the hub selection and assignment results – from the model week to the traffic matrices of the test weeks. Error bars represent the maximum and minimum cost across all 8 weeks with each value of  $\theta$ , whereas the curve connects the averages. For comparison, we also plot the cost curves for the model week using dotted lines. We do not plot the gain curves because we use the same hub set for all the weeks. The results are promising; the aggregate cost curves for the test weeks are close to those of the model week for all choices of  $\theta$  except 0. The exception when  $\theta = 0$  occurs because the strict latency condition leads to a solution set that is very sensitive to traffic pattern changes. The tight error bars show that the aggregate cost is stable with little variance across the 8 test weeks. We found similar results with the full-mesh solution of the model week, and omit the graphs in the interest of space.

Figure 6c shows the distribution of additional distances when we apply the usage-based solutions from the model week to one (July 23 - 29, 2007) of the test weeks. The solid lines show the additional distances when we use both the hub selection and assignment plans of the model week. The dotted lines represent the distances when combining the hub selection results with de facto hub assignment. The graph is similar to Figure 5a, meaning that the site-to-site latency distribution remains fairly stable over the test weeks. However, the fraction of communication pairs whose additional distance is larger than the specified  $\theta$  increases by at most 3%, leaving heavier tails. Note that, due to traffic dynamics, just using the hub assignment results of the model week (i.e., solid curves) does not guarantee the conformance to  $\theta$  in subsequent weeks because the conversation matrix changes. However, the fraction of such cases is small. In the case of  $\theta = 400$ , only less than 2.5% of conversation pairs do not meet the latency constraint. We verified that

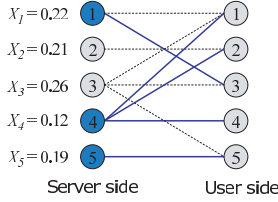


Figure 7: A sample serve-use relationship with penalties

these tails are removed when using the full-mesh solutions of the test week. In conclusion, the latency-constrained hub selection and assignment scheme generates robust solutions.

## 7. LATENCY-CONSTRAINED, VOLUME-SENSITIVE RELAYING

In addition to bounding additional latency within a threshold, we also want to reduce additional resource consumption in the backbone required for Relaying. We view the sum of volume and additional distance products is one of the relevant metrics to measure the load a backbone has to bear. This requirement motivates another optimization problem, namely *latency-constrained, volume-sensitive Relaying (LCVSR)*.

### 7.1 LCVSR Problem formulation

To formulate this problem, we define a volume matrix  $V = (v_{i,j})$  of a VPN where  $v_{i,j}$  denotes the amount of traffic volume that  $p_i$  sends to  $p_j$ . Obviously,  $v_{i,j} = 0$  when  $c_{i,j} = 0$ . Now we consider the following problem:

$$\begin{aligned} \min \quad & |H|, \sum_{s,d} v_{s,d} \cdot (l_{s,M(s)} + l_{M(s),d} - l_{s,d}) \\ \text{s.t.} \quad & \forall s,d \text{ whose } c_{s,d} = 1, \\ & l_{s,M(s)} + l_{M(s),d} - l_{s,d} \leq \theta \end{aligned}$$

To provide robust solutions for the worst case, we can also utilize the full-mesh conversation matrix  $C^{full}$ . Note, however, that we still need to use the usage-based volume matrix  $V$  as well because we cannot correctly assume the volumes of the conversations that have never taken place. Thus, a hub set generated by this formulation can serve traffic between any pairs of PEs without violating the latency constraint, and keeps the amount of relayed volume relatively small as long as the volume matrices considered are similar to the usage-based matrices. Hence, assuming that the future communication between two PEs who have never spoken to each other in the past generates relatively small amount of traffic, this full-mesh solution might still work effectively.

### 7.2 Algorithm to solve LCVSR

We outline our LCVSR algorithm in this section. The pseudocode is given in [10]. Minimizing  $|H|$  conflicts with minimizing  $\sum_{s,d} v_{s,d} \cdot (l_{s,M(s)} + l_{M(s),d} - l_{s,d})$ , making the problem hard to solve. For example, to reduce the additional resource consumption to zero, every PE should become a hub, leading to the maximum value of  $|H|$ . Note, however, that this does not mean that the additional resource consumption can be minimized only and if only the hub set is maximal. In fact, for some  $C$ , it is possible to minimize the additional resource consumption with a very small  $|H|$  – one trivial example is the case where there is only one popular PE, and all the other PEs communicate only with the popular PE. Although we cannot minimize both the objectives at the same time in general, coming up with a unified objective function composed of the two objectives functions (e.g., a weighed sum of the two objectives) is not a good approach either because the two objectives carry totally different meanings. Thus, we propose a simple heuristic to build a reasonably small hub set that reduces the relayed volume.

Our algorithm for LCVSR works similarly to the algorithm for the LCR problem. Thus, the first stage of the algorithm builds the same serve-use relationships among PEs. However, during the process, the algorithm also computes a *penalty* value for each PE. The penalty  $X_i$  of PE  $p_i$  is defined to be the sum of the volume and additional distance products of all conversations in  $C$ , assuming  $p_i$  is a sole hub in the VPN. That is,  $X_i = \sum_{s,d} v_{s,d} \cdot (l_{s,i} + l_{i,d} - l_{s,d})$ . Figure 7 illustrates a sample serve-use relationship graph with five PEs, where each PE is annotated with its penalty value. With these serve-use relationships along with penalties, the algorithm chooses hubs.

Due to the two conflicting objectives, at each run of the hub selection process, the algorithm has to make a decision that reduces either one of the two objectives, but not necessarily both. We design our algorithm to choose a PE from the server side (i.e., the left bank) that has the *smallest penalty value*, rather than the largest serve set size. The intuition behind this design is that choosing a PE with the smallest penalty is often conducive to reducing *both* objectives, if not always. By definition, a penalty value  $X_i$  of PE  $p_i$  decreases when each product (i.e.,  $v_{s,d} \cdot (l_{s,i} + l_{i,d} - l_{s,d})$ ) becomes smaller. Now suppose a PE that communicates with a large number of other PEs; we call such a PE has a high *span*. The penalty of the high-span PE is usually lower because the high-span PE is on the shortest path of many conversations, making many of the volume-distance products zero. Thus, our algorithm tends to choose PEs with higher spans. The key is that, fortunately, a PE with a higher span *also* has a large serve set (i.e.,  $S_i$ ) because it can serve as a hub a large number of PEs it communicates without violating the additional distance constraint. The remaining part (removing the chosen PE, and revising the serve-use relationships) is identical to the previous algorithm. We repeat the process until no PE remains on the user side.

### 7.3 Solutions with usage-based matrices

The benefit of LCVSR over LCR is shown in Figure 8a. The figure indicates that LCVSR algorithm significantly reduces indication penalties compared to the LCR algorithm. The amount of relayed traffic volume increases much more slowly than does it with the LCR algorithm and never exceed 40% of the total volume for any  $\theta$  below 1000 miles. In comparison, in Figure 5c, the same cost curve lies above the 40% line for almost all  $\theta$ , reaching nearly 70% when  $\theta = 1000$ . This decrease in relayed volume also reduces the sum of volume and additional distance products. For a reasonable choice of  $\theta$  (e.g., 400 miles), the sum of the volume and additional mile products is only 1.2% of the corresponding total.

Figure 8a also shows the LCVSR can still substantially reduce router memory usage, and also that the amount of memory saving is marginally lower than the results of LCR. Specifically, a comparison with Figure 5c reveals that the saving for each  $\theta$  is lower by only 1 to 3%. While, the lower saving is somewhat expected given that LCVSR does not explicitly aim to minimize  $|H|$ , the small difference in saving suggests that the LCVSR is still very effective in identifying memory-efficient solutions.

In conclusion, the LCVSR scheme results in very modest increase of backbone workload, while enabling dramatic memory saving. We also confirmed that combining the LCVSR with the simpler-to-implement de facto assignment scheme, instead of the assignment dictated by the algorithm, marginally affects the aggregate cost.

We also note that the distribution of path lengths is biased towards lower values for the LCVSR compared to the LCR, as evidenced by comparing the each CDF curve in Figure 8b to the corresponding curve in Figure 5a. In particular we note that for any

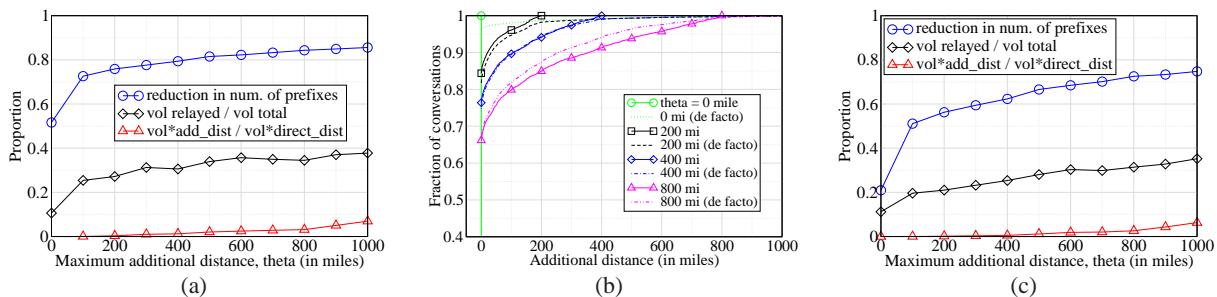


Figure 8: LCVSR performance (a) Gain and cost (with  $C$ ), (b) CDF of additional distances (with  $C$ ) (c) Gain and cost (with  $C^{full}$ )

given  $\theta$ , a significantly larger fraction of communications have no path inflation. This reduced inflation is probably a consequence of the LCVSR algorithm tending to choose PEs with higher span values, rather than those PEs whose locations are qualified to serve a larger number of other PEs, as hubs. Using the de facto assignment scheme, however, also adversely impacts a few (e.g., 0.9% of the total communication pairs in the case of  $\theta = 400$ ) individual pairs, leading to the increase of additional distances beyond  $\theta$ .

## 7.4 Solutions with full-mesh matrices

Here we compare LCVSR with LCR, both using full-mesh conversation matrices. Recall that a full-mesh conversation matrix of a VPN factor in the latency for conversations between every PE pair. Hence, for a given PE, the number of hub PEs that can serve the given PE tends to be smaller than the corresponding number with the usage-based conversation matrix. This in turn suggests that hub sets are generally larger in the full-mesh case. This intuition is confirmed in Figure 8c which shows that the hub set size indeed increases. However, we can still save a substantial 22 to 75% of memory for reasonable values of  $\theta$  in the few hundred miles range. This corresponds to marginally lower (about 1 – 5%) memory saving compared to the LCR algorithm (see Figure 6a). On the other hand, the cost of Relaying reduces significantly under LCVSR (about 50% in terms of the amount of relayed volume, and 75 to 95% in terms of the sum of volume and additional distance products) compared to LCR.

## 7.5 Robustness

We next evaluate how a specific LCVSR solution (i.e., a hub selection and assignment results generated by the LCVSR algorithm) performs as conversation and volume matrices change over time. We use data from the same model and test weeks as in Section 6.5.

Figure 9 presents the aggregate cost during the test weeks with the solid curves and error bars. We apply the usage-based solution (both hub selection and assignment results) derived from the model week to the conversation and volume matrices of the test weeks. For comparison, dotted curves show the corresponding indirection penalties for the model week. First, the figure shows that the increase of backbone workload (i.e., the sum of volume and additional distance products), while still higher than the model week, is quite small and has low variability. For all values of  $\theta$ , this quantity remains below 12% (below 9% when  $\theta \leq 800$ ). Hence, the actual additional network load in the test weeks is still very low. However, the results also indicate that the amount of relayed traffic volume itself can be substantially higher in the test weeks compared to that in the model week. For example for  $\theta = 400$ , the relayed volume is higher by 17 to 106%, depending on weeks. This higher and markedly variable amount of relayed volume can be attributed to the fact that LCVSR uses the traffic matrix of the model week in its hub selection, and that different weeks will have some variability in the volumes of common conversation pairs. Similar results are found when using the full-mesh solutions of the model week, but

we do not show them to save space. Despite the increase of relayed traffic volume, we confirmed that the distributions of additional distances during the test weeks are similar to those of the test week (i.e., curves in Figure 8b), except that tails become slightly heavier in a similar fashion shown as in the LCR results (Section 6.5 and Figure 6c).

The fact that the distributions of the additional distances do not change much over the test weeks might seem to be conflicting with the fact that the aggregate relayed volume significantly increases during the test weeks. By manually investigating the traffic patterns of the model and test weeks, we figure out that this happens because conversation matrices ( $C$ ) are more stable than volume matrices ( $L$ ) are. For example, suppose PE  $p_1$  communicates with PE  $p_2$  during the model week (i.e.,  $c_{1,2} = 1$ ), and both  $p_1$  and  $p_2$  are not hubs (i.e., traffic from  $p_1$  to  $p_2$  is relayed). Note that during the test weeks traffic from  $p_1$  to  $p_2$  never experience additional distances larger than  $\theta$  because our algorithm guarantees this for all pairs of PEs who communicated during the model week. Now let us consider the volume  $v_{1,2}$  of the traffic from  $p_1$  to  $p_2$ . When  $v_{1,2}$  increases during the test weeks, compared to  $v_{1,2}$  of the model week, the fraction of the relayed traffic volume during the test weeks eventually increases, leading to the effect shown in Figure 9 without affecting the additional distance distributions.

## 8. IMPLEMENTATION AND DEPLOYMENT

Implementing relaying is straightforward and does not introduce complexity into the existing VPN infrastructure. Given a set of hubs for each VPN, network administrators can easily migrate from the conventional VPN architecture to the relaying architecture only by slightly modifying PE routers' configuration. Meanwhile, relaying traffic through a hub is handled solely by the hub, without affecting the CEs or links that are incident to the hub. Moreover, both initial deployment and periodic re-adjustment of Relaying do not incur any service disruption.

### 8.1 Implementing Relaying with BGP

PE routers use BGP to disseminate customer prefixes. For scalability, PE routers multiplex route updates of different VPNs over a set of shared BGP sessions established among the PEs. Isolation between different VPNs is then achieved via *route targets* (RTs) and *import policies*. Specifically, all routes of VPN  $p$  are tagged with its own route target  $RT_p$  when they are advertised. When PEs receive route updates tagged with  $RT_p$ , they import those routes only into  $p$ 's VRF table  $VRF_p$  as dictated by  $p$ 's import policy. Note that implementing this conventional VPN routing architecture requires only one RT and one import policy for each VPN.

Our Relaying scheme can be easily implemented by introducing in each VPN  $i$ ) two RTs to distinguish customer routes from the default routes (0.0.0.0/0), and  $ii$ ) two different import policies for hubs and spokes. This mechanism is illustrated in Figure 10.

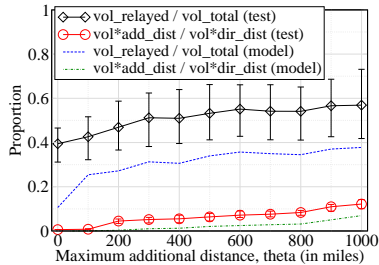


Figure 9: Robustness results during test weeks

Each PE – regardless of whether it is a hub or a spoke – in VPN  $p$  advertises customer routes tagged with  $RT_p^c$ . Additionally, each hub in  $p$  advertises a default route tagged with  $RT_p^d$ . Upon receiving updates, a spoke PE imports routes tagged with  $RT_p^d$  only, whereas a hub PE imports routes tagged with  $RT_p^c$  only. As a result, a VRF table  $VRF_p^{spoke}$  at a spoke PE is populated only with the default routes advertised by other hubs and the routes of the locally-attached customer sites. Contrastingly, a hub’s VRF table  $VRF_p^{hub}$  contains all customer routes in the VPN. Note that this routing architecture ensures for a hub to directly handle relayed traffic without forwarding it to a locally-attached customer site because hubs maintain full reachability information. Hence, implementing Relaying does not require any modification to customers.

The mechanism described above naturally implements the default hub assignment scheme because each spoke individually selects the closest hub in terms of IGP metric. However, to implement a specific hub assignment plan that our algorithm generates, we need to customize the import policy of each spoke’s VRF differently. There may be multiple options enabling such customization, including local preference adjustment.

Unfortunately there is one exceptional case where this implementation guideline should not be directly applied. When a customer site announces the default route on its own (to provide Internet gateway service to other sites, for example), the PE connected to the customer site must be chosen as a hub regardless of our algorithm’s decision. By comparing our algorithms’ results with the VPN routing configuration, we found that most (97.2%) PEs that are connected to those default-route-advertising sites are anyway chosen as hubs because those sites usually have higher span values or generate/receive large amount of traffic volume. Thus, the penalty due to those exceptional cases are very low.

## 8.2 Managing Relaying-enabled VPNs

**Run-time complexity:** The complexity of our algorithms is  $O(n^3)$  per VPN where  $n$  is the number of PEs in a VPN. A sparse usage-based traffic matrix often allows much better run times in practice –  $O(n^2)$  especially when each spoke communicates with a small constant number of hubs. Actually, given hundreds of VPNs each of which spans some tens to hundreds of PEs, running our prototype script with usage-based traffic matrices measured over one week takes less than an hour. With full-mesh matrices, the completion time goes up to tens of hours. Thus, assuming a simple configuration automation tool set [5], running our algorithms and correspondingly re-configuring PEs can be easily performed as a weekly management operation. With further optimization (e.g., coding in C language), daily adjustment might be possible as well.

**Reconfiguration overhead:** To assess the incremental reconfiguration overhead, we measured how stable hub sets are across different measurement/re-configuration windows. We used two different sizes (i.e., a week and a month) of windows. For 9 weeks beginning May 07, 2007, we measured how many hub PEs

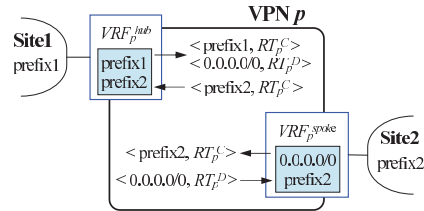


Figure 10: BGP configuration for Relaying

out of those chosen for a window are also chosen as hubs in the following window. The results are shown in Table 1. Overall, the hub sets are quite stable; more than 94% (91%) percent of PEs chosen for a week (month) remain as hubs in the following week (month). The results also confirms the followings: *i*) Using a smaller measurement window (i.e., a shorter re-configuration interval) ensures higher stability, *ii*) full-mesh solutions are more stable than usage-based solutions, and *iii*) the LCR solutions are more stable than the LCVSR solutions.

Table 1: Proportions (in percentage) of hubs that remain as hubs across two windows (averaged across all windows and  $\theta$ )

	LCR-usage	LCR-full	LCVSR-usage	LCVSR-full
weekly	96.4	99.2	94.3	98.4
monthly	91.2	97.8	91.0	97.3

**Availability:** One concern about utilizing Relaying is the potential decrease of availability due to the small number of hubs. When one of the multiple hubs in a VPN goes down, our Relaying scheme ensures that each spoke using the unreachable hub immediately switches over to another hub. Thus, by having at least two hubs located at different POPs, we can significantly increase a VPN’s availability. Although our hub selection algorithms do not consider this constraint (i.e.,  $|H| \geq 2$ ), most large VPNs almost always have multiple hubs anyway; 98.3% of VPNs which are provisioned over more than 10 PEs have at least 2 hubs. Moreover, our algorithms in itself ensure that each of those hubs are located at different POPs.

Nevertheless, extending the algorithms to explicitly factor in this additional constraint for every VPN is also straight forward. When the algorithm chooses only one hub for a VPN, we can make the algorithm additionally choose the second best hub in terms of the algorithm’s hub selection criterion that does not reside in the same POP as the first hub. We modified our algorithms this way and measured memory saving effect. The reduction of memory saving due to this modification was only 0.5% at most, and this penalty decreases as  $\theta$  becomes smaller because a smaller  $\theta$  naturally chooses more number of hubs per VPN.

## 9. RELATED WORK

To the knowledge of the authors, this paper is the first study focusing on a scalable L3 VPN routing architecture that reduces the amount of state information stored in routing tables and yet strictly bounds individual path lengths.

Designing a scalable routing architecture for *the Internet*, however, is an active research field where several interesting results are available, including CRIO [15], and LISP [6]. One key mechanism commonly used by these works is *indirection*. An indirection mechanism divides the large address space used for end host (or subnet) identification into several small fractions. Then each router stores the reachability information about one of the fractions, rather than the entire address space, leading to smaller routing tables. When a router receives a packet destined to a fraction for which the router is not responsible, the packet is first forwarded via tunneling to a router that is responsible for the fraction, and then

finally to the actual destination. Each architectural model mentioned above suggests unique mechanisms to systematically divide the original address space into fractions, and to map a fraction to a router. For example, CRIO uses large super-prefixes and BGP for these purposes, whereas LISP encompasses several variations, including super-prefix-based, DNS-based, or ICMP-based mapping models. However, none of these models suggest specific algorithms that can generate complete indirection configuration satisfying parameterized performance constraints. Also, all these models propose caching for reducing path inflation, whereas our approach avoids caching for simplicity and implementability.

Flat routing architectures, such as ROFL [2] and UIP [7], also reduces the amount of state at the expense of increasing path stretch. These works leverage on DHTs (Distributed Hash Tables) to avoid storing individual route entries for each destination. Unfortunately, these solutions are not immediately available and lacks simplicity. Also it is unclear how we can utilize these flat routing schemes in a VPN environment where names (i.e., prefixes) are not flat. For example, a router cannot determine which part of a given destination address it should hash unless it knows the destination prefix. Apart from these, the flat routing schemes are not practically suitable for realizing a constrained indirection architecture because path stretch in those architectures is unlimited in the worst case, and paths may often change as the set of destinations (i.e., prefixes) change.

Understanding the unique nature of VPNs and suggesting a better (e.g., more efficient or scalable) provisioning architecture leveraging the unique nature of VPNs has been of interest to many researchers recently. A study on estimating PE-to-PE traffic matrix from aggregate volume data gathered at the edge [12] has identified the strong “hub-and-spoke” traffic patterns. They used the estimation model to suggest a more efficient admission control scheme [11]. Unfortunately, estimated traffic matrices are not suitable for making relaying decisions since hub selection is sensitive to the conversation matrices, rather than the volumes matrices.

## 10. CONCLUSION

The large memory footprint of L3 VPN services is a critical, impending problem to network service providers. As a solution, this paper suggests Relaying, a highly scalable VPN routing architecture. Relaying enables routers to reduce routing tables significantly by offering indirect any-to-any reachability among PEs. Despite this benefit, there are two practical requirements that must be considered. First, from customer sites’ point of view, end-to-end communication latency over a VPN should not increase noticeably. Second, for the service provider’s sake, Relaying should not significantly increase the workload on the backbone.

Reflecting these requirements, we formulate two hub selection and assignment problems and suggest practical algorithms to solve the problems. Our evaluation using real traffic matrices, routing configuration, and VPN topologies draws the following conclusions: *i*) When one can allow the path lengths of common conversations to increase by a few hundred miles (i.e., a few msec in unidirectional latency) at most, Relaying can reduce memory consumption by 80 to 90%; *ii*) even when enforcing the additional distance limit to every conversation, rather than only common ones, Relaying can still save 60 to 80% of memory with the increase of unidirectional latency by around 10 msec at most; and *iii*) it is possible, at the same time, to increase memory saving, reduce the increase of workload on the backbone, and bound additional latency of individual conversations.

Our Relaying technique is readily deployable in today’s network, works in the context of existing routing protocols, requires no changes to either router hardware and software, or to the cus-

tomers’ network. Network administrators can implement Relaying by modifying routing protocol configuration only. The entire process of Relaying configuration can be easily automated, and adjusting the configuration does not incur service disruption.

In this paper, we focused on techniques that did not require any new capabilities or protocols. The space of alternatives increases if we relax this strict backwards-compatibility assumption. One interesting possibility involves combining caching with Relaying, where Relaying is used as a resolution scheme to handle cache misses. Another revolves around having hubs keep smaller non-overlapping portions of the address space, rather than the entire space, and utilizing advanced resolution mechanisms such as DHTs. We are exploring these as part of ongoing work.

## Acknowledgments

We would like to thank Jennifer Rexford for her role in formulating this project and providing insightful feedback throughout the execution of the project. We also thank Yung Yi, Matthew Caesar, Rui Zhang-Shen, Yaping Zhu, and Jiayue He for their valuable comments on the earlier draft of this paper. Finally, we thank our shepherd, Nick Feamster, and the anonymous reviewers for their suggestions on how to improve the presentation of the material.

## 11. REFERENCES

- [1] T. Bates, R. Chandra, D. Katz, and Y. Rekhter. Multiprotocol Extensions for BGP-4. RFC 2283, 1998.
- [2] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica. ROFL: Routing on Flat Labels. In *Proc. ACM SIGCOMM*, September 2006.
- [3] Cisco. Cisco line cards, engine 0,1,2,3,4. [http://www.cisco.com/en/US/products/hw/routers/ps167/products\\_tech\\_note09186a00801e1d8e.shtml](http://www.cisco.com/en/US/products/hw/routers/ps167/products_tech_note09186a00801e1d8e.shtml).
- [4] B. Claise. Cisco Systems NetFlow Services Export Version 9. Request for Comments 3954, October 2004.
- [5] W. Enck, P. McDaniel, S. Sen, et al. Configuration Management at a Massive Scale: System Design and Experience. In *Proc. USENIX Annual Technical Conference*, 2007.
- [6] D. Farinacci, V. Fuller, D. Oran, and D. Meyer. Locator/ID Separation Protocol (LISP). Internet-Draft (work in progress), November 2007.
- [7] B. Ford. Unmanaged Internet Protocol: taming the edge network management crisis. In *ACM Computer Communication Review*, volume 34, pages 93–98, 2004.
- [8] IDC. U.S. IP VPN services 2006-2010 forecast. <http://www.idc.com/getdoc.jsp?containerId=201682>.
- [9] R. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pages 85–103, 1972.
- [10] C. Kim, A. Gerber, C. Lund, D. Pei, and S. Sen. Scalable VPN Routing via Relaying. Technical Report, November 2007. AT&T TD-794M29.
- [11] S. Raghunath, S. Kalyanaraman, and K. K. Ramakrishnan. Trade-offs in Resource Management for Virtual Private Networks. In *Proc. IEEE INFOCOM*, March 2005.
- [12] S. Raghunath, K. K. Ramakrishnan, S. Kalyanaraman, and C. Chase. Measurement Based Characterization and Provisioning of IP VPNs. In *Proc. Internet Measurement Conference*, October 2004.
- [13] Y. Rekhter, T. Li, and S. Hares. A Border Gateway Protocol (BGP-4). RFC 4271, January 2006.
- [14] E. Rosen and Y. Rekhter. BGP/MPLS IP Virtual Private Networks. RFC 4364, February 2006.
- [15] X. Zhang, P. Francis, J. Wang, and K. Yoshida. Scaling IP Routing with the Core Router-Integrated Overlay. In *Proc. International Conference on Network Protocols*, 2006.