On the Performance Benefits of Multihoming Route Control

Aditya Akella, *Member, IEEE*, Bruce Maggs, *Member, IEEE*, Srinivasan Seshan, *Member, IEEE*, and Anees Shaikh, *Member, IEEE*

***Abstract*—Multihoming is increasingly being employed by large enterprises and data centers to extract good performance and reliability from their ISP connections. Multihomed end networks today can employ a variety of *route control* products to optimize their Internet access performance and reliability. However, little is known about the tangible benefits that such products can offer, the mechanisms they employ and their trade-offs. This paper makes two important contributions. First, we present a study of the potential improvements in Internet round-trip times (RTTs) and transfer speeds from employing multihoming route control. Our analysis shows that multihoming to three or more ISPs and cleverly scheduling traffic across the ISPs can improve Internet RTTs and throughputs by up to 25% and 20%, respectively. However, a careful selection of ISPs is important to realize the performance improvements. Second, focusing on large enterprises, we propose and evaluate a wide-range of route control mechanisms and evaluate their design trade-offs. We implement the proposed schemes on a Linux-based Web proxy and perform a trace-based evaluation of their performance. We show that both passive and active measurement-based techniques are equally effective and could improve the Web response times of enterprise networks by up to 25% on average, compared to using a single ISP. We also outline several "best common practices" for the design of route control products.**

***Index Terms*—Multihoming, performance, reliability.**

I. Introduction

MULTIHOMING to multiple Internet Service Providers (ISPs) has traditionally been employed by end-networks to ensure reliability of Internet access. However, over the past few years, multihoming has been increasingly leveraged for improving wide-area network performance, lowering bandwidth costs, and optimizing the way in which upstream links are used [1]. A number of products provide these route control capabilities to large enterprise customers which have their own public AS number and advertise their IP address prefixes to to upstream ISPs using BGP [2]–[4]. Recognizing that not all enterprises are large enough to warrant BGP peering with ISPs, another class of products extends these advantages to smaller multihomed organizations which do not use BGP [5]–[7]. All of these products use a variety of mechanisms and policies for route control but aside from marketing statements, little is known about their quantitative benefits.

Manuscript received December 15, 2005; revised July 19, 2006; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor Z.-L. Zhang.

A. Akella is with the Computer Sciences Department, University of Wisconson-Madison, Madison, WI 53706 USA (e-mail: akella@cs.wisc.edu).

B. Maggs and S. Seshan are with the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA 15213 USA.

A. Shaikh is with the IBM T. J. Watson Research Center, Hawthorne, NY 10532 USA.

Digital Object Identifier 10.1109/TNET.2007.899068

In this paper, we present an in-depth study of the performance benefits of multihoming route control products. Specifically, we seek to address the following two questions:

1) What tangible improvements in Internet performance (e.g., RTTs, throughput) can multihomed end networks expect from route control products?
2) What practical mechanisms must route control products employ to extract the benefits in real deployments?

Note that multihoming route control does not require any modification to Internet routing protocols, and relies solely on end-network decisions. Therefore, if our research shows that route control can offer tangible performance improvements in practice, this will imply that good performance can still be extracted from the network by making clever use of available Internet routes. On the other hand, if the improvement is insignificant, this may indicate that there is something fundamentally wrong with routing in the Internet and, to support good performance in the future Internet, we may need to replace the Internet routing protocol suite altogether.

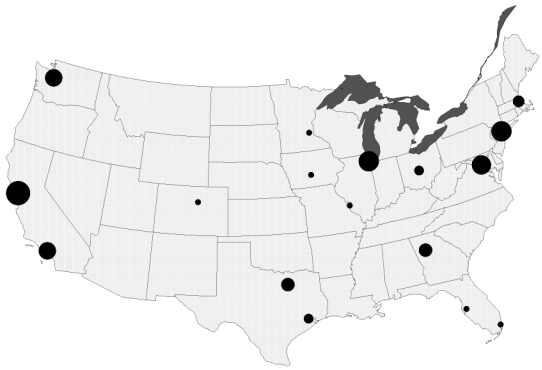
To answer the first question, we analyze active probe data collected over the servers in the Akamai content distribution network (CDN) infrastructure. We then compute the potential performance improvements from choosing ISPs from several available options. In general, we use the term *k-multihoming* to refer to the setting in which the subscriber network employs k ISPs and controls how traffic is sent or received along the ISP links (at the granularity of individual connections). To compute the potential benefits of k-multihoming, we assume that the multihomed end-network has perfect information of the performance of the k ISPs at all time instances, and can change the way traffic is routed arbitrarily often.

Our analysis of k-multihoming shows that RTT performance can potentially improve by up to 25% when an end-network connects to two well-chosen ISPs. Similarly, we observe 20% higher transfers speeds from multihoming to two or three ISPs. By studying the composition of the best set of ISPs to multihome to, we make observations on how an end network must select its ISP to obtain the maximum possible performance benefits.

The second question asks if, and how, the above potential benefits can be realized in practical multihoming scenarios. To this end, we explore several design alternatives for extracting performance benefits from multihoming in practice. Our focus is on enterprise networks with multiple ISP connections. We primarily consider mechanisms used for inbound route control, since enterprises are mainly interested in optimizing network performance for their own clients who download content from the Internet (i.e., sink data). However, our mechanisms can also be extended to multihomed content provider networks which source more data than they sink.

City	ISPs/tier				
	1	2	3	4	5
Atlanta, GA	2	0	1	1	0
Bay Area, CA	5	0	3	1	2
Boston, MA	1	0	1	0	1
Chicago, IL	6	1	0	1	0
Columbus, OH	0	1	0	1	0
Dallas, TX	3	0	0	1	0
Denver, CO	1	0	0	0	0
Des Moines, IO	0	1	0	0	0
Houston, TX	1	1	0	0	0
Los Angeles, CA	3	0	3	0	0
Miami, FL	1	0	0	0	0
Minneapolis, MN	0	0	1	0	0
New York, NY	3	2	2	1	0
Seattle, WA	2	0	2	1	1
St Louis, MO	1	0	0	0	0
Tampa, FL	0	1	0	0	0
Washington DC	3	0	3	0	2

(a) Testbed ISPs



(b) Node locations

Fig. 1. (a) Tested ISPs. The cities and distribution of ISP tiers for nodes in our measurement testbed are shown. (b) Geographic location of nodes. The area of each dot is proportional to the number of nodes in the region.

We evaluate a variety of active and passive measurement strategies for multihomed enterprises to estimate the instantaneous performance of their ISP links. We employ NAT-based techniques to control the inbound ISP link used by enterprise connections. We address a number of practical issues such as the usefulness of past history to guide the choice of the best ISP link, the impact of sampling frequency on measurement accuracy, and the overhead of managing performance information for a large number of destinations. We evaluate these policies using several client workloads, and an emulated wide-area network testbed where delay characteristics are based on a large set of real network delay measurements.

Our evaluation shows that active and passive measurement-based techniques are equally effective in extracting the performance benefits of using multiple ISPs, both offering about 15%–25% improvement compared to using a single ISP. We show that the most current sample of the performance to a destination via a given ISP is a reasonably good estimator of the near-term performance to the destination. We show that the overhead of collecting and managing performance information for various destinations is negligible.

This paper is structured as follows. We discuss the RTT and throughput improvement from route control in Section II. In Section III, we describe our enterprise multihoming solution and the various strategies for estimating ISP performance and for route control. Section IV describes our route control implementation in further detail. In Section V, we discuss the experimental set-up and results from our evaluation of the solution. Section VI discusses some limitations inherent to our approach. Related work is presented in Section VII. Finally, Section VIII summarizes the contributions of this paper.

II. Multihoming Improvements

We first study the potential performance improvements from multihoming route control via RTT and throughput measurements taken over a large testbed consisting of nodes belonging to the server infrastructure of the Akamai CDN. The key technique we use in our measurements and analyses is to *emulate* a k-multihoming scenario by selecting a few nodes in a metropolitan area, each singly-homed to a different ISP, and use them collectively as a stand-in for a multihomed network. This is similar to the approach adopted in [8]. Our testbed consists of 68 Akamai CDN server nodes spanning 17 U.S. cities, averaging about four nodes per city. The nodes are connected to commercial ISPs of various sizes. To enable emulation of multihoming, we choose the nodes in each metro area so that no two servers in a city are attached to the same ISP. The cities we measure at, and the tiers of the corresponding ISPs (derived from [9]) are shown in Fig. 1(a). The geographic distribution of the testbed nodes is illustrated in Fig. 1(b). We emulate multihomed networks in 9 of the 17 metropolitan areas where there are at least three ISPs—Atlanta, Bay Area, Boston, Chicago, Dallas, Los Angeles, New York, Seattle, and Washington DC.

We note that, by the choice of our measurement nodes (namely well-provisioned Akamai servers), our measurements will reflect the performance benefits for large enterprises or campus networks with good connectivity to the Internet. We stress that the measurement results we report may not automatically apply to other settings, such as multi-homed home users, at least not quantitatively. We do note that our results will hold for home users with very high-speed broadband connections. This is common in East Asian countries; market studies predict that broadband speeds in the US will raise to several tens of Mbps by 2010.

Next, we describe our data collection methodology. Then, we present the key measurement observations in the following order. First, we present the improvements in RTT and throughput performance from using k-multihoming. Second, we explore whether the improvements due to multiple are skewed by certain destinations, time of the day or day of the week. Finally, we explore the impact of a suboptimal choice of ISPs on observed subscriber performance.

A. Data Collection

We draw our observations from two datasets collected on the testbed described above. The first data set consists of active HTTP downloads of small objects (10 KB) to measure the *turnaround times* between the pairs of nodes. The turn-around time for such HTTP requests is the time between the transfer of the last byte of the request from the Akamai node and the

receipt of the first byte of the response from the origin server. Hence, the turnaround time offers a reasonable estimate of network delay. Since Akamai servers are well-provisioned, we expect that the observed turnaround time is constituted mainly by network delay, with almost no delay due to the Web server itself. Every 6 minutes, we collect turnaround time samples between all pairs of nodes in our testbed.

The second data set contains throughput measurements from active downloads of 1 MB objects between the same set of node-pairs. These downloads occur every 30 minutes between all node-pairs. Throughput is the size of the transfer (1 MB) divided by the time between the receipt of the first and last bytes of the response data from the server (source). This may not reflect the steady-state TCP throughput along the path.

Since our testbed nodes are part of a production infrastructure, we limit the frequencies of all-pairs measurements. To ensure that all active probes between pairs of nodes observe similar network conditions, we scheduled them to occur within 30s of each other for the RTT data set, and within a 2 mins of each other for the throughput data set. For the latter, we also ensure that an individual node is involved in at most one transfer at any time so that our probes do not contend for bandwidth at the source or destination network. The transfers may interfere elsewhere in the Internet. Also, since our testbed nodes are all located in the U.S., the routes we probe are U.S.-centric. The RTT data was collected from Thursday, Dec 4, 2003 through Wednesday, Dec 10, 2003. The throughput measurements were collected between Thursday, May 6, 2004 and Tuesday, May 11, 2004 (both days inclusive).

B. k-Multihoming Improvements

To understand performance benefits of k-multihoming, we adopt the following simple methodology: For each download, we compare the client-perceived turnaround time achieved by using the best ISP among all those available in the city, with that from using the best ISP in a candidate multihoming option. We average this ratio over transfers to all clients, and report the minimum normalized performance metric (the minimum is taken over all candidate options). We compare only those transactions for which there was a successful transfer over all ISPs at roughly the same time.

Let $M_{\text{best}}(A_i, t)$ denote the best turnaround time for a transfer to $A_i (i = 1, \ldots, 68)$ at time t, across all ISPs in a city. For a k-multihoming option OP_k, let $M_{\text{OP}_k}(A_i, t)$ be the best turnaround time across ISPs in the set OP_k. Then, the RTT performance benefits from the option OP_k is

$$\text{RTT}_{\text{OP}_k} = \frac{\sum_{i,t}(M_{\text{OP}_k}(A_i, t)/M_{\text{best}}(A_i, t))}{\text{Numvalid}(t)}.$$

The sum is over all t when transfers occur from all ISPs in the city to A_i. $\text{Numvalid}(t)$ is the number of such instances. We compute throughput benefits in a similar fashion:

$$\text{Thru}_{\text{OP}_k} = \frac{\sum_{i,t}(M_{\text{best}}(A_i, t)/M_{\text{OP}_k}(A_i, t))}{\text{Numvalid}(t)}.$$

The difference in the definition of the RTT and throughput metrics arises from the fact that we are interested in how multihoming can *lower* RTTs and *increase* transfers speeds.

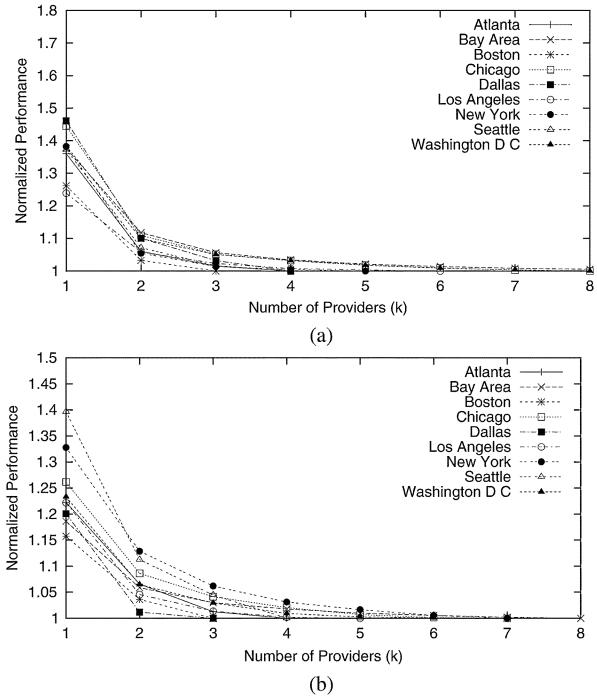


Fig. 2. (a) Turnaround time. The RTT improvements from k-multihoming are shown. (b) Throughput improvements.

In Fig. 2, we plot the above RTT and throughput metrics due to k-multihoming as a function of the number of ISPs. Two key facts are apparent from Fig. 2(a). First, the average RTT improves dramatically when the subscriber uses the best set of two or more ISPs, relative to using the single best ISP. The metric is lowered by 0.4, reflecting an average 25% improvement in RTTs. Intuitively, this occurs because a second, well-chosen ISP could potentially double the diversity in paths to various destinations. This improved choice in paths could in turn help ISPs avoid serious performance problems along any single ISP's paths. Second, there is strong evidence of diminishing returns. Beyond three or four ISPs the marginal benefits from additional ISPs is small. Again, this occurs because a fourth or a fifth ISP can provide very little additional diversity above what a well chosen set of three or four ISPs already provides. In terms of throughput, multihoming improves performance by as much as 20% relative to a single ISP [see Fig. 2(b)].

C. Unrolling the Averages

Next, we present the underlying distributions in the performance improvements to understand if the averages are particularly skewed by: 1) certain destinations, or 2) a few measurement samples on which multihoming offers significantly better performance than a single ISP, or 3) by time-of-day or day-of-week effects.

Performance per destination. In Fig. 3(a), for each city, we show the distribution of the average difference^Jbetween the best 3-multihoming path and the path via the single best ISP (i.e., a single point represents one destination). To illustrate, for a subscriber in Seattle, 3-multihoming improves the average RTT per destination by more than 10 ms for about 60% of the destinations, and more than 15 ms for about 30% of the destinations. In Los Angeles, however, the improvement due to multihoming is less dramatic. For about 60% of the destinations, the

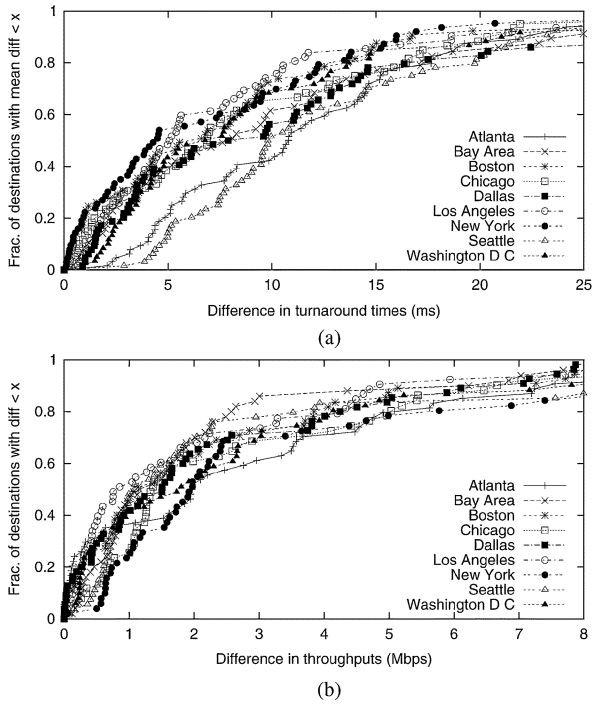


Fig. 3. These figures show the improvements, per-destination, from 3-multihoming relative to 1-multihoming. (a) Turnaround time. (b) Throughput.

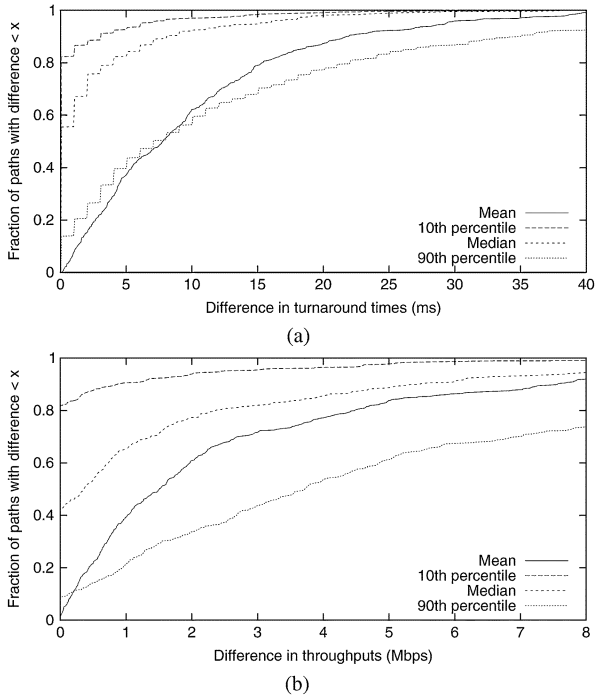


Fig. 4. Mean, median, 10th percentile, and 90th percentile RTT improvements. (a) Turnaround time. (b) Throughput.

improvement in the average RTT is under 5 ms. The key point to notice is that, for the nine cities we consider, there exist a few destinations to which multihoming can offer significantly improved RTT performance.

In Fig. 3(b), we consider the distribution of the average throughput difference of the best 3-multihoming path and the best single ISP. We see the throughput difference is more than 3 Mbps for 15%–40% of the destinations. We also note that, for 1%–10% of the destinations, the difference is in excess of 8 Mbps. As with RTT, these observations imply that the transfer speeds to certain destinations could be substantially higher when the subscriber is multihomed.

Other statistics. Fig. 4(a) and (b) plots the average, median, and the 10th and 90th percentiles of the difference in RTT and throughput between 3-multihoming and 1-multihoming. In Fig. 4(a), we see that the median difference is fairly small. More than 90% of the median RTT differences are less than 10 ms. However, the 90th percentile of the difference is much higher with roughly 25% greater than 20 ms. The 90th percentile throughput differences in Fig. 4(b) are also significant—more than 8 Mbps about 25% of the time. We see that a significant fraction of the median throughput differences (about 20%) are greater than 3 Mbps. These observations suggest that while multihoming improves the overall performance of all transfers by modest amounts, the performance of a small yet significant fraction could improve substantially when traffic is scheduled carefully across ISP links.

Time-of-day and day-of-week effects. It might be expected that 1-multihoming would perform particularly worse than 3-multihoming during peak periods. In Fig. 5(a), we examine time-of-day effects on the average difference in round-trip times. Notice that the RTT performance improvement does show a correlation with the time of the day. While the improvement due to careful route selection is minimal in the evenings

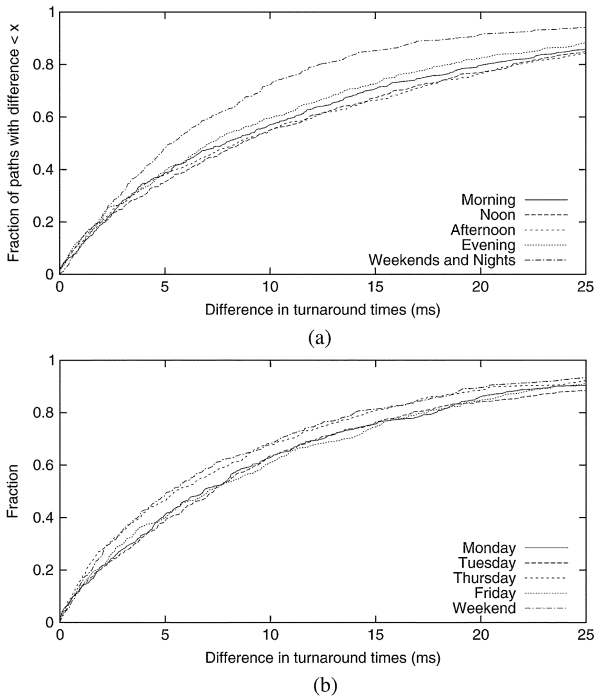


Fig. 5. The effect of the time-of-day and day-of-week on RTTs. All times are in EDT. (a) Time-of-day effects. (b) Day-of-week effects.

and weekends, the differences are more pronounced during the remaining time periods. We also examine weekly patterns to determine whether the differences are greater during particular days of the week [Fig. 5(b)]. The correlation between the performance improvements and the days of the week is not as significant. As expected, we observe the improvements to be inferior during weekends. However, the improvements for the other days of the week are not substantially different.

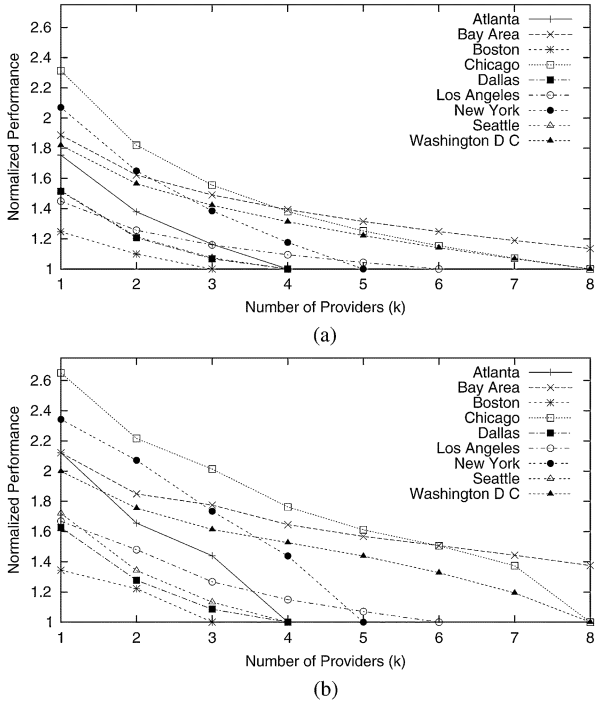


Fig. 6. The effect of random and worst-case ISP choices on RTT performance. (a) Performance from a random choice. (b) Performance from a poor choice.

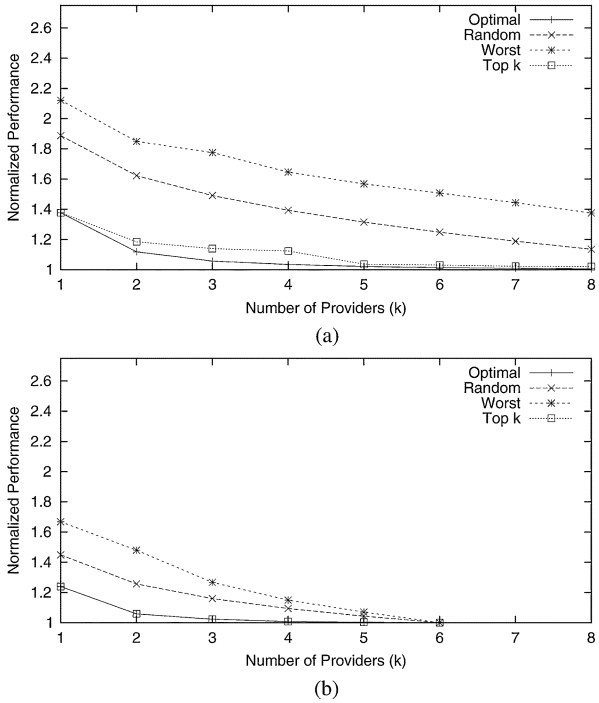


Fig. 7. RTT performance from various ISP selection policies for (a) San Francisco and (b) Los Angeles.

D. Impact of the Choice of ISPs

Fig. 6 illustrates the impact of choosing a sub-optimal set of ISPs for k-multihoming. We assume that, given a choice of ISPs, a subscriber always uses the best ISP among the available set for its transfers. Comparing Figs. 6(a) and 2(a), for $k \leq 4$, we see that the average RTT performance metric due a *random* choice of k ISPs is more than 50% higher (e.g., $k = 2$ for Chicago). The difference between optimal and random choices of ISPs is substantial even for higher values of k. In Fig. 6(b) we show the performance metric of the *worst* k-multihoming option. A poor choice of upstream ISPs could result in performance that is at least twice as bad as the optimal choice [compare, for example, $k = 2$ for Chicago, in Fig. 6(b) and 2(a)]. Therefore, while multihoming offers potential for significant performance benefits, it is crucial to carefully choose the right set of upstream ISPs.

Finally, we explore the relative RTT performance from various strategies for selecting ISPs. In particular, Fig. 7(a) compares the RTT performance of optimal, random and worst-case choice of multihoming ISPs for a subscriber in San Francisco. In addition, we show the RTT performance metric for the case when the subscriber multihomes to the top k individual ISPs (in terms of their average RTT performance). Not only does selecting the top k individual ISPs out-perform a random choice, it also provides similar RTT performance as the optimal choice. Nevertheless, a more informed selection of ISPs (than simply choosing the top k) could yield up to 5%–10% better RTT performance on average [see, for example, $k = 3, 4$ in Fig. 7(a)]. We show a similar set of results for Los Angeles in Fig. 7(b). In this case, choosing the top k ISPs yields identical RTT performance as the optimal choice.

III. Practical Route Control

So far, we studied the potential improvements from multihoming by analyzing an ideal form of multihoming that was driven by certain key assumptions: First, the end-network had perfect information of the performance of ISP links for each destination. Second, the end-network did no incur any overhead in moving traffic across ISPs over time. Third, the end network was able to control the ISP link taken by traffic entering the network. To realize these potential performance benefits in practice, we must address the following issues:

(1.a) How should end-networks monitor the performance of ISP links? Is active probing better than passive observation?

(1.b) A related question is which destinations to monitor. Should the end network probe all possible destinations via each ISP link? Does this give rise to scalability issues?

(2) How should the end network estimate the future performance of an ISP to a destination? This is key to determining which ISP the end-network must use for the destination. Should it simply rely on the most recent performance estimate as being indicative of the ISP's future performance to the destination? Or, does the end-network stand to gain more from tracking the historical performance of the ISP?

(3) Finally, how should the end-network direct traffic to use the chosen ISP links for a destination? The are two issues here: outbound control—scheduling outgoing traffic on the right output interface—and inbound control—ensuring that incoming traffic arrive on the right input interface.

Fig. 8 illustrates the three sets of issues outlined above. We stress that answering these issues is key to determining the usefulness of route control in practical settings. In the rest of this section, we discuss the functional design of these steps. We discuss the implementation details and analyze design trade-offs for each step in Section IV. We evaluate the design trade-offs using emulation experiments, and lay out best common route control design practices in Section V.

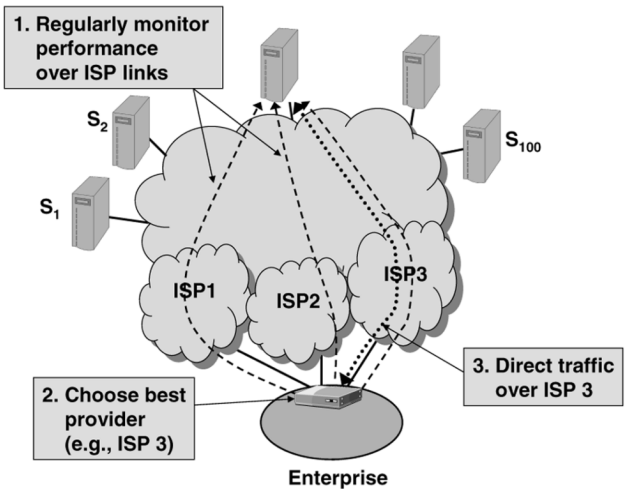


Fig. 8. Three main operations of a route control system.

A. Monitoring ISP Links

The first issue is selecting the right ISP link to direct each transfer. This choice depends on the time-varying performance of each ISP link to each destination being accessed. However, network performance could fluctuate, very substantially on some occasions [10]. A multihomed enterprise, therefore, needs effective mechanisms to monitor the performance for most, if not all, destinations over each of its ISP links. There are two further issues in monitoring performance over ISP links: *what* to monitor and *how*. In the enterprise case, one would ideally like to monitor the performance from every possible content provider over each ISP link. However, this may be infeasible in the case of a large enterprise which accesses content from many different sources. A simpler, more scalable solution to this problem is to monitor only the most important or popular destinations. But, how can we track the most popular destinations in a scalable and efficient manner, while also accommodating temporary shifts in popularity (e.g., due to events such as flash crowds)? In Section IV, we outline several approaches to track the popularity of destinations in a timely and scalable manner.

For the second question, two common approaches are active and passive monitoring. In active monitoring, the multihomed enterprise performs out-of-band measurements to or from specific destinations. These measurements could be simple pings (ICMP `ECHO_REQUEST`) or TCP `SYN` packets. The measurements are to be taken over each ISP at regular intervals. Passive measurement mechanisms rely on observing the performance of ongoing transfers (i.e., in-band) to destinations, and using these observations as samples for estimating performance over the given ISP. To ensure that there are enough samples over all ISPs, it may be necessary to explicitly direct transfers over particular links. In Section IV, we outline simple techniques to achieve fine-grained control over active or passive probes.

Another important factor in monitoring performance is the *time interval* of monitoring. A long interval between samples implies using stale information to estimate ISP performance. This might result in a suboptimal choice of the ISP link for a particular destination. While using smaller time intervals could address this issue, it could have a negative impact as well. In active monitoring, frequent measurements inflate the out-of-band measurement traffic causing additional bandwidth and processing overhead; some destinations might interpret this traffic as a security threat. In passive monitoring, frequent sampling may cause too many connections to be directed over sub-optimal ISPs in an attempt to obtain performance samples. As such, a careful choice of the interval size is crucial. We evalute this choice in Section V.

B. Choosing the Best ISP

The next component is selecting the best ISP. This choice must be made on a per-destination basis at fine time-scales. An important issue is whether historical data about ISP performance to a given destination should be employed. In general, the performance of an ISP to a destination can be tracked using a smoothed, time-weighted estimate of the performance, e.g., an Exponentially-Weighted Moving Average (EWMA). If performance of using an ISP P to reach destination D at time t_i is s_{t_i} (as obtained from active or passive measurement) and the previous performance sample was from time t_{i-1}, then the EWMA metric at time t_i is

$$\mathrm{EWMA}_{t_i}(P,D) = \left(1 - e^{-(t_i - t_{i-1})/\alpha}\right) s_{t_i} + e^{-(t_i - t_{i-1})/\alpha} \mathrm{EWMA}_{t_{i-1}}(P,D)$$

where $\alpha > 0$ is a constant. A smaller value of α attaches less weight to historical samples. $\alpha = 0$ implies no reliance on history. At any time, the ISP with the best performance as calculated above could be chosen for a given transfer. When no history is employed ($\alpha = 0$), only the most recent performance sample is used to evaluate the ISPs and select the best.

C. Directing Traffic Over Selected ISPs

The next step is to direct the traffic from the destination over the chosen link. Controlling the outbound direction of traffic is easy and well-studied. Our focus, rather, is on the *inbound route control* mechanism. Inbound control refers to selecting the right ISP or *incoming* interface on which to *receive* data. For an enterprise network, the primary mechanisms available are route advertisements and use of different addresses for different connections.

If an enterprise has its own IP address block, it can advertise different address ranges to its upstream ISPs. Consider a site multihomed to two ISPs which owns a $/19$ address block. The site announces part of its address block on each ISP link (e.g., a $/20$ sub-block on each link). Then, depending on which of the two ISP links is considered superior for traffic from a particular destination, the site would use a source address from the appropriate $/20$ address block. This ensures that all incoming packets for the connection would traverse the appropriate ISP link. In cases where the enterprise is simply assigned an address block by its upstream ISP, it may be necessary to also send outbound packets via the desired ISP to ensure that the ISP forwards the packets. Notice that different techniques must be employed for handling connections that are initiated from the enterprise, and for those that are accepted into the site from external clients. These are discussed next.

Initiated Connections: Handling connections initiated from an enterprise site requires ensuring that the remote content

provider transmits data such that the enterprise ultimately receives it over the chosen ISP. Inbound control can be achieved by the edge router translating the source addresses on the connections initiated from its network to those belonging to the chosen ISP's address block (i.e., the appropriate $/20$ block in the example above) via NAT-like mechanisms. This ensures that the replies from the destination will arrive over the appropriate ISP. We elaborate on this in Section IV.C.

Accepted Connections: Inbound control over connections accepted into a site is necessary when the enterprise also hosts Internet servers accessed from outside. In this case, inbound control amounts to controlling the ISP link on which a client is forced to send request and acknowledgment packets to the Web server. This key challenge here lies in predicting client arrivals and forcing them to use the appropriate server address. Techniques based on DNS or deploying multiple versions of Web pages are commonly used to address this challenge. For example, the enterprise can use a different version of a base Web page for each ISP link. The hyperlinks for embedded objects in the page could be constructed with IP addresses corresponding to a given ISP. Then, arriving clients would be given the appropriate base HTML page such that subsequent requests for the embedded objects arrive via the selected ISP. On the other hand, the essential function of the DNS-based technique is to provide the address of the "appropriate" interface for each arriving client. Our focus in this paper, however, is on enterprise-initiated connections.

IV. Implementation Details

We extend a simple open source Web proxy called *TinyProxy* [11] to implement the above multihoming route control functions. TinyProxy is a transparent, non-caching forward Web proxy that manages the performance of Web requests made by clients in a moderately-sized, multihomed enterprise. Below, we present the details of our implementation of the three basic multihoming components in TinyProxy. For the sake of simplicity, we assume that the proxy is deployed by a multihomed end-network with three ISP links.

A. Performance Monitoring Algorithms

Passive Measurement. The passive measurement module tracks the performance to destinations of interest by sampling ISP links using Web requests initiated by clients in the enterprise. This module uses new requests to sample an ISP's performance to a destination if the performance estimate for that ISP is older than the predefined sampling interval. If the module has current performance estimates for all links, then the connection is directed over the best link for the destination. The module maintains a performance hash table keyed by the destination. A hash table entry holds the current estimates of the performance to the destination via the three ISPs, along with an associated timestamp indicating the last time performance to the destination via the ISP was measured. This is necessary for updating the EWMA estimate of performance.

Notice that without some explicit control, the hash table maintains performance samples to all destinations, including those rarely accessed. This could cause a high overhead of measurement. Also connections to less popular destinations may all used up for obtaining performance samples. While maintaining explicit TTLs per entry might help flush out destinations that have not been accessed over a long period of time, it does not guarantee a manageable measurement overhead. Also, TTLs require maintaining a separate timer per entry, which is an additional overhead. In view of this, we limit performance sampling to connections destined for the most popular sites, where popularity is measured in terms of aggregate client request counts, as follows: Hash entries also hold the number of *accesses* made to the corresponding destinations. Upon receiving a connection request for a given destination, we update the access count for the destination using an exponentially weighted moving average (EWMA). The EWMA weight is chosen so that the access count for the destination is reset to ~ 1 if it was not accessed for a long time, say 1 hour. We use a hard threshold and monitor performance to destinations for which the total number of requests exceeds the threshold. This can be done by looking for live entries in the table with the access counts exceeding the threshold. In a naive hash table implementation for tracking the frequency counts of the various elements, identifying the popular destinations may take O(hash table size) time.

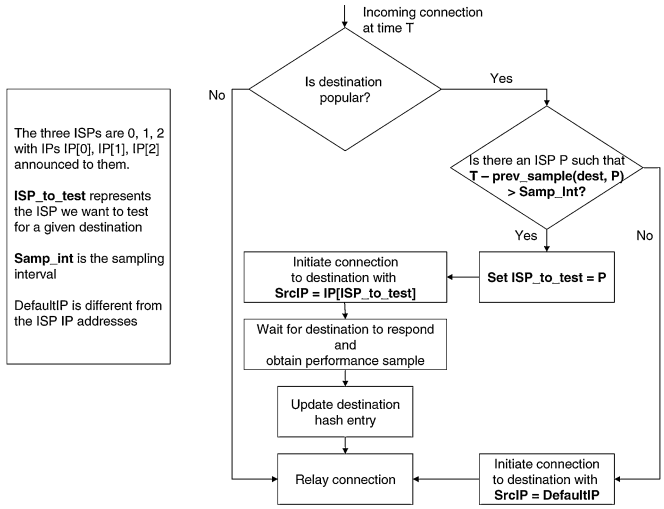


Fig. 9. Passive ISP monitoring scheme.

Other ways of tracking top destinations such as Iceberg Queries [12] or Sample-and-hold [13], may not incur such an overhead. Nevertheless, we stick with our approach for its simplicity of implementation. Also, as we will show later, the overhead from looking for the popular hash entries in our implementation is negligible. Note that this approach does not necessarily limit the actual number of popular destinations, for example in the relatively unlikely case that a very large number of destinations are accessed very often.

Fig. 9 shows the basic operation of the passive monitoring scheme. When an enterprise client initiates a connection, the scheme first checks if the destination has a corresponding entry in the performance hash table (i.e., it is labeled popular). If not, the connection is simply relayed using an ISP link chosen randomly, in a load-balancing fashion. If there is an entry for the destination, the passive scheme scans the timestamps for the three ISPs to see if the elapsed time since the last measurement on any of the links exceeds the predefined *sampling interval*. If so, the current connection is used to sample the destination along one such ISP link.

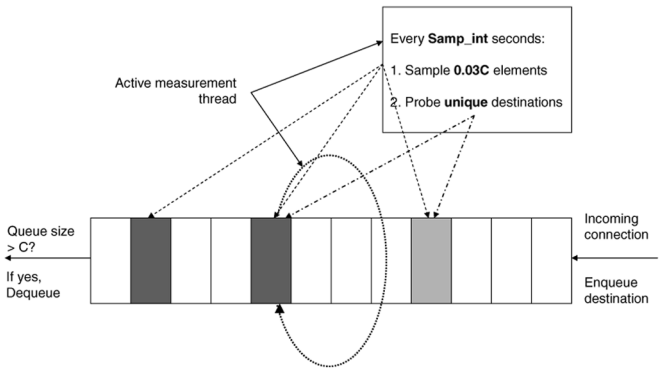


Fig. 10. *SlidingWindow* monitoring scheme.

In order to obtain a measurement sample on an ISP link, the scheme initiates a connection to the destination using a source IP address set such that the response will return via the link being sampled. Then, it measures the *turn-around time* for the connection, defined as the time between the transmission of the last byte of the client HTTP request, and the receipt of the first byte of the HTTP response from the destination. The observed turn-around time is used as the performance sample to the destination, and the corresponding entry in the hash table is updated using the EWMA method (Section III.B). The remainder of the Web request proceeds normally, with the proxy relaying the data appropriately. If all of the ISP links have current measurements (i.e., within the sampling interval), the proxy initiates a connection using the best link for the destination by setting the source IP address appropriately. We discuss these details in Section IV.C.

Active Measurement. Similar to passive measurement, the active measurement scheme also maintains a hash table of the performance estimates to candidate destinations over the three ISPs. For active measurement, we use two techniques to identify which destinations should be monitored.

FrequencyCounts. Again, in this scheme we track the number of client requests directed to each destination. Every T seconds, we initiate active probes to destinations with at least a threshold number of requests.

SlidingWindow. This scheme maintains a window of size C that contains the C most recently accessed destinations. The window is implemented as a fixed size FIFO queue, in which destinations from newly initiated connections are inserted. If this causes the number of elements to exceed C, then the oldest in the window is removed. Every T seconds (the sampling interval), an active measurement thread scans the window and chooses $m\%$ of the elements at random. After discarding duplicate destinations from this subset, the active-measurement scheme measures the performance to the remaining destinations along the ISPs (see Fig. 10).

The two active schemes offer distinct trade-offs. Notice that both the schemes effectively sample the performance to destinations that are accessed more often relative to others. However, there are a few key differences. First, *FrequencyCounts* is deterministic since it works with a reasonably precise set of popular destinations. *SlidingWindow*, on the other hand, may either miss a few popular destinations, or sample a few unpopular destinations. Second, *FrequencyCounts* in its simplest form, cannot easily track small, short-term shifts in the popularity of the destinations. These new, temporarily-popular destinations may not receive enough requests to exceed the threshold and force performance sampling for them, even though they are popular for a short time. *SlidingWindow*, on the other hand, can effectively track small shifts in the underlying popularity distribution of the destinations.

Active Probe operation. Once a destination is selected for active probing, the active measurement scheme sends three probes, with different source IP addresses, corresponding to the three ISPs, and waits for the destination to respond. Since we found that a large fraction of popular Web sites filter ICMP `ECHO_REQUEST` packets, we employ a TCP-based probing mechanism. Specifically, we send a TCP `SYN` packet with the `ACK` bit set to port 80 and wait for an RST packet from the destination. We use the elapsed time as a sample of the turn-around time performance. We found that most sites respond promptly to the $\mathrm{SYN} + \mathrm{ACK}$ packets. When a response is received, we update the performance estimates to the destination for the corresponding ISP, along with the measurement timestamp. If no response is received from a destination (which has an entry in the performance hash table), then a large positive value is used as the current measurement sample of the performance.

B. Switching ISPs

After updating all ISP entries for a destination in the performance hash, we switch to a new ISP only if it offers at least a 10% bettter RTT performance over the current best ISP for the destination. Since the hash entries are updated at most once every T seconds (in either the passive or active measurement schemes), the choice of best ISP per destination also changes at the same frequency.

C. NAT-Based Inbound Route Control

Our inbound control mechanism is based on manipulating NAT tables at the Web proxy to reflect the current choice of best ISP. We use the iptables packet filtering facility in the Linux 2.4 kernel to install and update NAT tables at the proxy. The NAT rules associate destination addresses with the best ISP link such that the source address on packets directed to a destination in the table are translated to an address that is announced to the chosen ISP.

For example, suppose ISP 1 is selected for transfers involving destination 1.2.3.4 and the addresses 10.1.1.1 was announced over the link to ISP 1. Then we insert a NAT rule for the destination 1.2.3.4 that: (1) matches packets with a source IP of `defaultIP` and destination 1.2.3.4, and (2) translates the source IP address on such packets to 10.1.1.1. Notice that if the NAT rule blindly translates the source IP on all packets destined for 1.2.3.4 to 10.1.1.1, then it will not be possible to measure the performance to 1.2.3.4 via ISP 2, assuming that a different IP address, e.g., 10.1.1.2, was announced over the link to ISP 2. This is because the NAT translates the source address used for probing 1.2.3.4 across ISP 2 (i.e., 10.1.1.2) to 10.1.1.1, since ISP 1 is considered to be the best for destination 1.2.3.4. To get around this problem in our implementation, we simply construct the NAT rule to only translate packets with a specific source IP address (in this case `defaultIP`). Measurement packets that belong to probes (active measurement) or client connections (passive measurement) are sent with the appropriate source address, corresponding to the ISP being measured.

V. Experimental Evaluation

In this section, we describe our experimental evaluation of the design alternatives proposed in Section IV. These include the performance of passive versus active monitoring schemes, sensitivity to various measurement sampling intervals, and the overhead of managing performance information for a large set of target destinations. We focus on understanding the benefits each scheme offers, including the set of parameters that result in the maximum advantage.

A. Experimental Set-Up

We describe our testbed and discuss how we emulate realistic wide-area delays. Then we discuss key characteristics of the delay traces we employ in our emulation. Finally, we discuss the performance metrics for comparing various schemes.

We use the simple testbed topology of Fig. 11(b). Our goal is to emulate a moderately-sized enterprise with three ISP connections and a client population of about 100 (shown in Fig. 11(a)). Node S in the topology runs a simple lightweight Web server and has one network interface configured with 100 different IP aliases—10.1.1.1 through 10.1.1.100. Each alias represents an instance of a Web server—10.1.1.1 is the most popular and 10.1.1.100 the least popular. C runs 100 instances of clients which make requests to the Web sites 10.1.1.1 through 10.1.1.100. The inter-arrival times between requests from a single client are Poisson-distributed with a mean of λ seconds. Notice that this mean inter-arrival rate translates into an average request rate of $(100)/(\lambda)$ requests per second at the S. Each client request is for the ith destination where i is sampled from the set $\{10.1.1.1, \ldots, 10.1.1.100\}$ according to a Zipf distribution with an exponent ≈ 2. In our evaluation, we set the parameters of the monitoring schemes (passive and active) so that the average rank of the destinations probed is 20, meaning that we explicitly track the top 40 most popular sites during each experiment. The object sizes requested by the client are drawn from a Pareto distribution with an exponent of 2 and a mean size of 5 KB.

Node P in the topology runs the Web proxy (TinyProxy). It is configured with one "internal" interface on which the proxy listens for connections from clients within the emulated enterprise. It has another interface with three IP aliases, 10.1.3.1, 10.1.3.2 and 10.1.3.3, each representing addresses announced over the three ISP links. Node D is a *delay element*, running WaspNet [14], a loadable kernel module providing emulation of wide-area network characteristics on the Linux platform. We modify WaspNet to enforce packet delays (along with drops, and bandwidth limits) on a per-$<$ `source`IP, destination IP $>$ pair basis. We also modify it to support trace-based network delay emulation as illustrated in Fig. 11(b).

To recreate realistic network delays between the clients and the servers in the testbed, we collect a set of wide area delay measurements using the Akamai content distribution network. We pick three Akamai server machines in Chicago, attached to unique ISPs. We run pings at regular intervals of 10s from these nodes to 100 other Akamai servers located in various US cities and attached to a variety of ISPs. The measurements were taken over a one-day period on Dec 7th, 2003. The three Akamai machines in Chicago collectively act as a stand-in for a multihomed network with three ISP links. The 100 Akamai servers probed represent destinations contacted by end-nodes in the multihomed network. We use the colected delay samples as inputs to the WaspNet module to emulate wide-area delays.

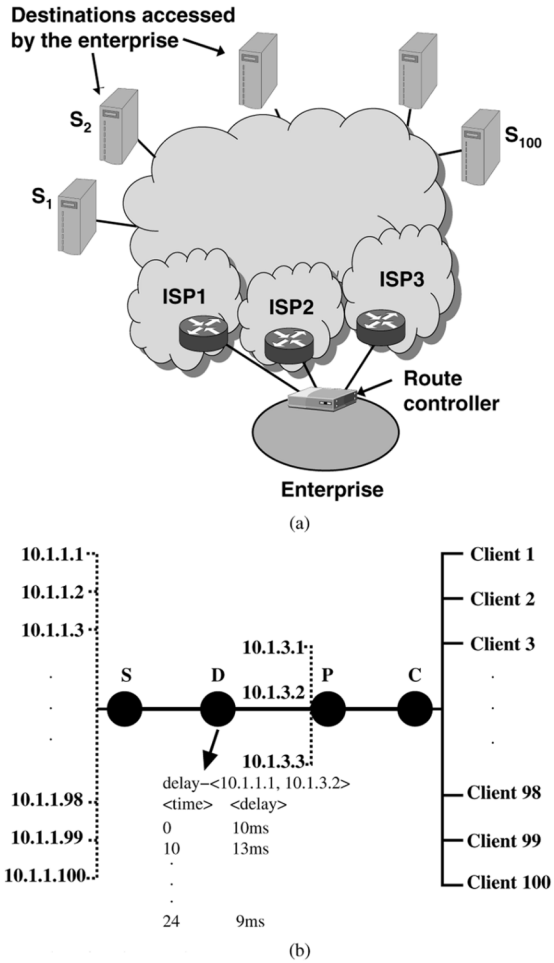


Fig. 11. (a) Multihomed enterprise. (b) Testbed topology. The simple testbed, shown in (b), is used to emulate the route control scenario shown in (a).

Compressing time. It is quite time-consuming to emulate the entire day's worth of delays, multiple times over, to test and tune the parameters in each scheme. One work-around could be to choose a smaller portion of the delay traces (e.g., 2 hours). However, a quick analysis of the delay traces we collect shows that there is not much variation in the delays along the probed paths on a 2-hour timescale. Since our goal is to understand how effective each scheme is over a wide range of operating conditions, it is important to test how well the schemes handle frequent changes in the performance of the underlying network paths. With this in mind, we compress the 24-hour delay traces by a factor of 10, to 2-hour delay traces and use these as the actual inputs to the WaspNet delay module. In these 2-hour traces, performance changes in the underlying paths occur roughly 10 times more often when compared to the full 24-hour trace. The characteristics of the 2-hour delay traces collected from the nodes in Chicago are shown in Table I, column 2.

To ensure that the delays measured from Chicago were not significantly different from other major cities, we collect similar traces from sources located in New York and Los Angeles. These traces were collected on March 20th, 2004. The statistics for these latter traces are shown in columns 2 and 3 of Table I.

TABLE I
CHARACTERISTICS OF THE DELAY TRACES

	Chicago	NYC	LA
Mean time between performance changes	79s	101s	105s
Standard deviation of time between changes	337s	487s	423s
Mean extent of performance change	$\pm 33\%$	$\pm 28\%$	$\pm 34\%$
Standard deviation of extent of change	$\pm 26\%$	$\pm 22\%$	$\pm 27\%$
Mean time between performance changes of 30%	298s	261s	245s

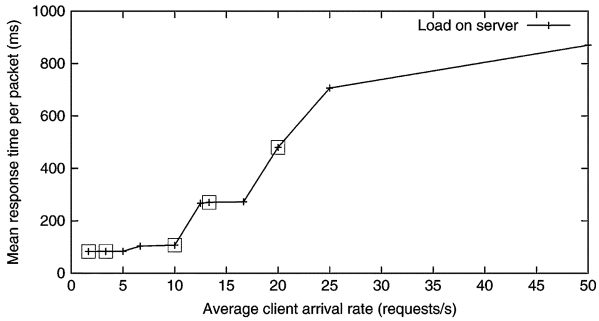


Fig. 12. Response time, per KB of the request, as a function of the client arrival rate at the server in our testbed.

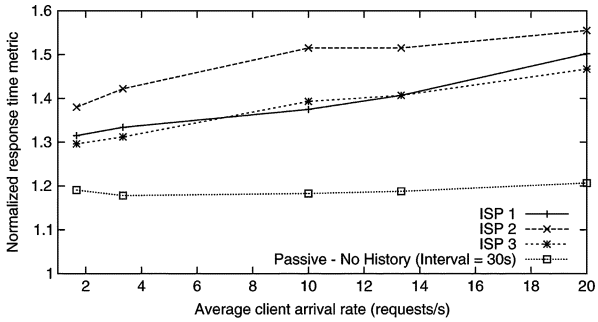


Fig. 13. $\mathcal{R}$ for the passive scheme with EWMA parameter $\alpha = 0$ and sampling interval of 30 s.

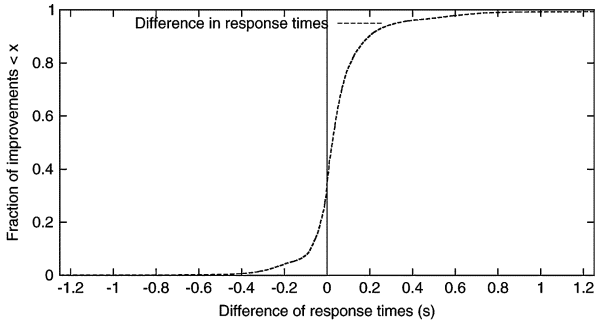


Fig. 14. Response times from using the passive scheme relative to using just ISP 3. The client arrival rate is 13.3 requests/s.

These statistics show that the Chicago-based traces we use in our experiments have roughly the same characteristics as those collected at the other metros. We also conducted s subset of the evaluations on the New York and Los Angelestraces. We note that the results obtained from the latter traces (not presented in this paper) were qualitatively similar to those obtained from Chicago traces.

Comparison Metric. We compare the response time of transfers when using a particular scheme (i.e., $\mathrm{Resp}_{(x,\mathrm{scheme})}$, for a transfer x), with the response time when the best of the three ISPs is employed for each transfer $(\min_i\{\mathrm{Resp}_{(x,\mathrm{ISP}_i)}\})$ to compute the following "performance metric" for the scheme:

$$\mathcal{R}_{\mathrm{scheme}} = \frac{1}{||x||}\sum_x \frac{\mathrm{Resp}_{(x,\mathrm{scheme})}}{\min_i\{\mathrm{Resp}_{(x,\mathrm{ISP}_i)}\}}$$

$||x||$ is the total number of transfers. The closer $\mathcal{R}$ is to 1, the better the performance of the scheme. We compute response times for a transfer from using the best ISP (terms in the denominator above) in an offline manner: we simply force the transfer to use the three ISPs in turn, and select the ISP offering the best response time.

B. Experimental Results

Our experiments are conducted on Emulab [15] using 600 MHz Pentium III machines running Red Hat 7.3.

Selecting the Client Workloads. In Fig. 12 we show the average response time per KB of client requests (i.e., the completion time for a request divided by the size of the request in KB), as a function of the average arrival rate of clients at the server S (i.e., $(100/\lambda)$ requests/s). The response time quickly degrades beyond an arrival rate of about 15 requests/s beyond which it increases only marginally with the request rate. We select five different points on this load curve (highlighted), corresponding to arrival rates of 1.7, 3.3, 10, 13.3 and 20 requests/s, and evaluate the proposed schemes under these workloads. These workloads represent various stress levels on the server S, while also ensuring that it is not overloaded. The high variability in response times in overload regimes might impact the confidence or accuracy of our comparison of the proposed schemes. Next, we present results from our experimental evaluation of the route control schemes.

Improvements from Route Control. The aggregate performance improvement from the passive measurement-based schemes is shown in Fig. 13. Here, we set the EWMA parameter $\alpha = 0$ so that only the current measurement samples are used to estimate ISP performance, and select a sampling interval of 30 s. The figure plots the performance for the five client workloads. In addition, we show the performance from using the three ISPs individually. The performance improvement relative to the best individual ISP is significant—about 20%–25% for the heavy workloads (right end of the graph) and about 10%–15% for the light workloads (left end of the graph). The performance is still about 15%–20% away from the optimal value of 1. The results for other sampling intervals (60 s, 120 s, 300 s and 450 s) are similar, and are omitted for brevity.

Fig. 14 illustrates the distribution of the absolute response time improvements offered by the passive measurement scheme (for $\alpha = 0$ and sampling interval $= 30$ s) relative to being singly-homed to the best ISP from Fig. 13, i.e., ISP 3. The passive measurement scheme improves the response time performance for over 65% of the transfers. Notice that the scheme can improve the response time by more than 1s for some transfers. Notice also that the passive measurement-based scheme

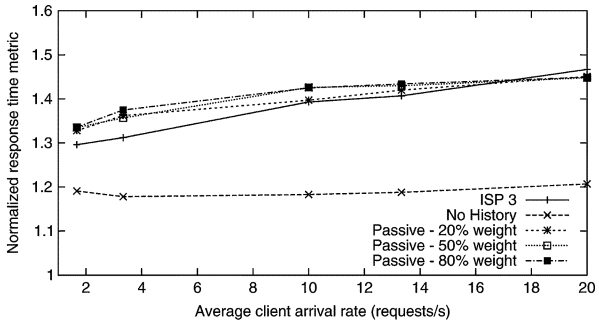


Fig. 15. Impact of history (passive strategy interval = 30 s).

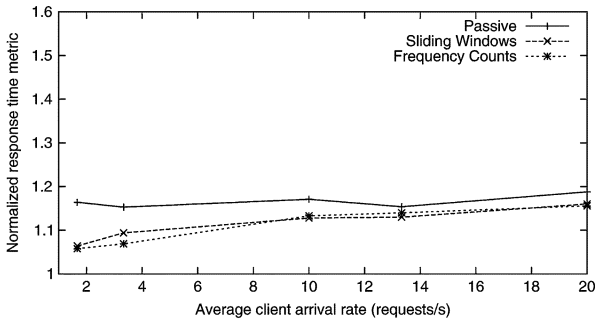
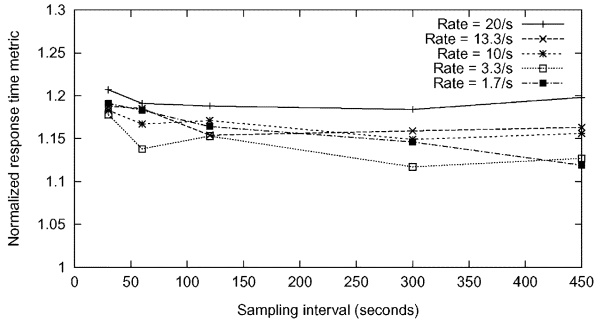
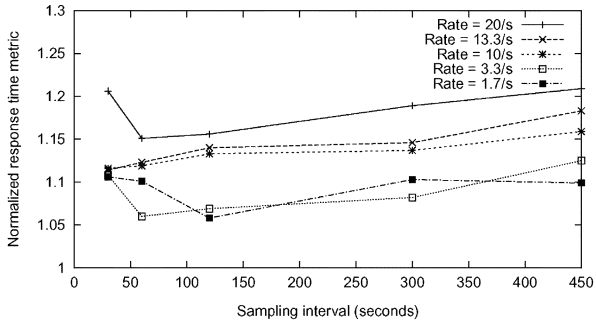


Fig. 16. Active versus passive schemes (interval = 120 s).



(a)



(b)

Fig. 17. The impact of sampling interval size. (a) Passive measurement. (b) FrequencyCounts.

ends up offering sub-optimal performance for about 35% of the transfers.

Employing History to Estimate Performance. Fig. 15 plots the performance of the passive measurement scheme for different values of α. These correspond to assigning 80%, 50%, and 20% weight to the current measurement sample and the remaining weight to the past samples. Although we only show results for a sampling interval of 30 s, the performance from other interval sizes are similar. The figure also plots the performance when no history is employed ($\alpha = 0$) and the performance from using ISP 3 alone. Notice that the performance from employing history is uniformly inferior in all situations, relative to employing no history. In fact, historical samples only serve to bring performance close to that from using the single best ISP. These results show that the best way to estimate ISP performance is to just use the current performance sample as an estimate of near-term performance.

Active versus Passive Measurement. Fig. 16 compares the performance from the two active techniques (i.e., *SlidingWindow* and *FrequencyCounts*) with the passive measurement approach. Since our earlier results showed that history does not help in improving performance, henceforth we present results in which no history is employed. We compare the performance of the three measurement schemes for a common sampling interval of 120 s across the five client workloads. Note that the two active measurement schemes offer comparable performance. Unfortunately, the workloads we selected do not bring out other underlying trade-offs of these schemes (A detailed comparison of these active measurement schemes is future work). Fig. 16 also shows that the active measurement-based schemes offer slightly better performance than the passive measurement scheme: about 8%–10% for the light workloads and 2%–3% for the heavier workloads. This occurs because the passive scheme uses existing transfers to obtain samples across potentially sub-optimal ISP links.

Frequency of Monitoring. Fig. 17 shows the impact of measurement frequency on the performance for the passive scheme (Fig. 17(a)) and the *FrequencyCounts* active measurement scheme (Fig. 17(b)). Each figure plots the results for the five client workloads. From Fig. 17(a) we notice that longer sampling intervals surprisingly offer slightly better performance for passive measurement. To understand this better, consider the curve for the 10 requests/s workload. This arrival rate implies that an average of $10T$ connections are made by the clients every T seconds, where T is the sampling interval. In order to obtain samples for a fraction f of the 100 destinations over the three ISPs, the passive measurement scheme will have to force $300f$ connections across the ISP links. This leaves a fraction $1-(30f/T)$ which are not employed for measurement, and could be routed along the optimal ISP, assuming that the passive measurement yields reasonably accurate estimate of performance (About a third of the connections employed for measurement can be expected to be routed along their optimal ISPs). As T increases, the fraction of connections routed over the optimal path also increases, resulting in a marginal improvement in performance. This explains the downward slopes in Fig. 17(a). At the same time, infrequent sampling (i.e., large values of T) can have a negative impact on the overall performance. This is not immediately clear from Fig. 17(a). However, Fig. 17(b), which plots the performance from *FrequencyCounts* as a function of the sampling interval, sheds more light. A sampling interval of 450 s suffers a 5%–8% performance penalty relative to a smaller interval such as 60 s. Notice that in the case of *FrequencyCounts* too, aggressive sampling (e.g., an interval of 30 s) could slightly impact overall performance on some occasions due to the increased software overheads at the proxy.

TABLE II
ANALYSIS OF PERFORMANCE OVERHEADS

	Passive	FreqCount	SlidingWin
Total penalty	18%	14%	17%
Penalty from inaccurate estimation	16%	12%	14%
Penalty from measurement and NAT	2%	2%	3%

Overhead. Both passive and active measurement are about 10%–20% away from the optimal performance. Three key factors contribute to this gap: 1) the accuracy of measurement techniques, and correspondingly, the accuracy of ISP choices; 2) overhead of measurement; and 3) software overhead, specifically, the overhead of making frequent updates to the NAT table and employing NAT rules on a significant fraction of packets. To quantify the overhead in our implementation due to these factors, we compare the performance derived from the choices made by the route control proxy, with the performance when the best ISP choices are made in an offline manner for each connection. Recall that in order to compute the performance metric $\mathcal{R}$, we evaluated the response time of each ISP for every transfer offline so that the best ISP link for each connection was known, independent of the route control mechanisms. By combining these offline values with the decisions made by the proxy, we can estimate the performance penalty due to incorrect choices, independent of the software overheads (i.e., #2 and #3 above). The difference between the resulting performance metric and 1 gives us the performance penalty, excluding overheads of the implementation.

The penalties from the above analysis for the three proposed schemes are shown in Table II, row 2. The client arrival rate is 13.3 requests/s and the sampling rate is 30 s. In this table, the numbers in row 1 show the actual performance penalties suffered by the schemes in our implementation, taking all overheads into account (from Fig. 17(a) and (b)). Notice that a large portion of the overall penalty is contributed by the inaccuracies in measurement and ISP selection (rows 1 and 2 are almost identical). Measurement and software overheads themselves result in a performance penalty of 2%–3% (difference between rows 1 and 2, shown in row 3).

VI. ADDITIONAL OPERATIONAL ISSUES

The key findings from our study of the benefits of multihoming and of practical route control strategies are as follows:

Potential improvements: Multihoming to 2–3 ISPs could improve RTTs by 25% and throughputs by 20%, on average.

Choice of ISPs: The improvement in performance from employing more than 3 ISPs is marginal. A good heuristic to select ISPs is to simple to pick the top 3 individual ISPs.

Benefits in practice: The route control schemes we describe can significantly improve the performance of client transfers at a multihomed site, by up to 25% in our experiments.

Employing history: Using historical samples to monitor ISP performance could prove detrimental. Also, the current sample is a good estimator of near-term ISP performance.

Active versus passive: Both passive and active measurement-based schemes offer competitive performance, with the latter offering better performance for lighter client workloads.

Sampling interval: The overhead due to aggressive performance sampling may slightly reduce the overall benefit of route control schemes. Sampling on minutes' timescale (e.g., 60 s) seems to offer very good performance overall.

Low overhead: The overhead from measurements and updates to the NAT table are low. Most of the performance penalty arises from inaccuracies in measurement and estimation.

A. Route Control Deployment Issues

The above mechanisms are a first attempt at understanding how to extract good performance from multiple ISP connections in practice. There are a number of ways in which they can be improved. Also, we do not address several important issues, such as ISP costs and the interplay of performance and reliability optimization. We discuss these issues next.

Hybrid passive and active measurements. The accuracy of passive measurement can be improved by sending active probes immediately after a failed passive probe, for example when a connection ends unexpectedly. This increases confidence that the failed connection is due to a problem with the ISP link, as opposed to a transient effect. Also, in our implementation, paths to less popular destinations are not explicitly monitored. As a result, we may have to rely on passive observations of transfers to unpopular destination to ensure quick fail-over. For example, whenever the proxy observes several failures on connections to an unpopular destination, it can immediately switch the destination's default ISP to one of the remaining two ISPs for future transfers.

Balancing performance and resilience. A key function of most route control products is to respond quickly to ISP failures. One of our findings is that even a relatively long sampling interval offers good performance benefits. But, a long interval can also slow the end-network's reaction to path failures, however. This can be addressed by sampling each destination with a sufficiently high frequency, while still keeping the probing overhead low. For example, a sampling interval of 60s with active measurement works well in such cases, providing reasonably low overhead, good performance (Fig. 17(b)), and a failover time of about one minute.

ISP pricing structures. In our study, we ignore issues relating to the the cost of the ISP links. Different ISP connections may have very different pricing policies. One may charge a flat rate up to some committed rate, while another may use purely usage-based pricing or charge differently depending on whether the destination is "on-net" or "off-net". A more formal and thorough discussion of techniques for optimizing ISP usage costs as well as performance may be found in [16]. While we do not explicitly consider how to optimize overall bandwidth costs, we believe that our evaluation of active and passive monitoring, and the utility of history, are central to general schemes that optimize for both cost and performance.

About externally-initiated connections. Our implementation primarily considered handling connections initiated from within the enterprise, as these are common for current enterprise applications (e.g., to contact content ISPs). A route control product must also handle connections from outside clients, however, to enable optimized access to servers hosted in the enterprise network. As mentioned in Section III.C, a common mechanism to achieve inbound control in such situations is to employ DNS. However, preliminary measurements regarding the usefulness

DNS for externally-initiated connections (presented in [17]) show that a large fraction of end-clients do not obey TTLs on DNS records. This impacts the effectiveness of DNS-based network control. The authors in [17] also discuss mechanisms to improve the responsiveness of end-clients to DNS-based mechanisms.

Impact on routing table sizes. Announcing small, non-aggregateable address sub-blocks to different upstream ISPs (Section III.C) could affect the size of routing tables in the core of the network. This problem can be overcome, however, if multihomed end-networks obtain *provider-assigned* address: instead of buying a single individual IP address block, an end-network simply acquires equal-sized independent IP address blocks from each of its ISPs. These address blocks could then be further aggregated by the ISPs.

Global effects of route control. Another important issue is the potential impact of the interactions between many enterprises deploying route control mechanisms. This will likely have an effect not only on the marginal benefits of the route control solutions themselves, but also on the network as a whole. A recent simulation-based study of this problem by Qiu *et al.* in [16] has shown that the the impact of multiple end-networks employing route control on any single multihoming user is very minimal, at the equilibrium of the interactions. Similarly, the impact, at equilibrium, on singly-homed users is also negligible. While these are positive observations, the issues of whether end-networks can reach an equilibrium, and, how stable the equilibrium is, still remain open.

VII. Related Work

In this paper, we study mechanisms for improving Internet performance of enterprise networks via route control. Several research studies and products have considered other benefits of multihoming route control. In a study closely related to ours, the authors conduct trace-driven experiments to evaluate several design options using a commercial multihoming device [6], [18]. The evaluation focuses on the ability of several algorithms to balance load over multiple broadband-class links to provide service similar to a single higher-bandwidth link. The authors find that the effectiveness of hash-based link selection (i.e., hashing on packet header fields) in balancing load is comparable to load-based selection. Andersen et al. similarly consider various mechanisms for improving the reliability of Web access for DSL clients in [19].

A number of vendors have recently developed dedicated networking appliances [5], [7], [20] or software stacks [21], [22] for optimizing the use of multihomed connectivity in enterprises settings where BGP is not used. Most of these products use techniques similar to those we evaluate in our study, though their focus is geared more toward balancing load and managing bandwidth costs across multiple ISP links, rather than optimizing performance. All of these use NAT-based control of inbound traffic and DNS to influence links used by external client-initiated connections. They also ensure, by tracking sessions or using policy-based routing, that the same ISP link is used in both directions.

Another class of products and services are targeted at settings where BGP is employed, for example, large data centers or campuses [3], [23]. These products mainly focus on outbound control of routes and, as such, are more suited for content providers which primarily source data. Details of the algorithms used by any of the above commercial products to monitor link performance or availability are generally proprietary, and little information is available on specific mechanisms or parameter settings. Here, we review the general approaches taken in enterprise route control products.

Most commercial products employ both ICMP ping and TCP active probes to continuously monitor the health of upstream links, enabling rapid response to failure. In some cases, hybrid passive and active monitoring is used to track link performance. For example, when a connection to a previously unseen destination is initiated from an enterprise client, active probes across the candidate links sample performance to the destination. Connections to known destinations, on the other hand, are monitored passively to update performance samples. Another approach is to use active probing for monitoring link availability, and passive monitoring for performance sampling. Some products also allow static rules to dictate which link to use to reach known destinations networks.

Finally, some products and research efforts [19] suggest using "race"-based performance measurements. In these cases, SYN packets sent by enterprise clients to initiate connections are replicated by the route control device on all upstream ISPs (using source NAT). The link on which the corresponding SYN-ACK arrives from the server first is used for the remainder of the connection. The route control device sends RST packets along the slower paths so that the server can terminate the in-progress connection establishment state. The choice of best link is cached for some time so that subsequent connections that arrive within a short time period need not trigger a new race unless a link failure is detected.

A final note about the benefits of multihoming route control is due here. Our study of multihoming shows that by increasing the flexibility in the choice of routes at end-networks, their Internet performance can be substantially improved. A natural follow-up question is whether the performance of end-networks can be further improved by enabling even greater router flexibility, e.g., by employing Overlay Routing [24]. In [25], Akella *et al.* compare overlay routing against multihoming and show that route control can indeed offer roughly the same performance as overlay-based approaches.

VIII. Summary

Our goal was to quantify the performance improvements from route control and evaluate practical mechanisms and policies for realizing the performance benefits in practice. We used measurements on the Akamai's server network to show that multihoming to 2 or 3 ISPs can significantly improve Internet download performance. We discussed how to choose ISPs for multihoming. Further, we evaluated the Web performance of a real Linux-based enterprise route control implementation. We employed an emulated wide-area network testbed and experimentally evaluated several design alternatives. Our evaluation shows that both active and passive measurement-based route control schemes offer significant performance benefits in practice, between 15% and 25%, when compared with using the single best-performing ISP. Our experiments also show that the most current measurement sample gives a good estimate of ISP performance. We also showed that the performance penalty from collecting and managing performance data across various destinations is negligible.

REFERENCES

[1] P. Morissey, "Route optimizers: Mapping out the best route," Network Computing Dec. 2003 [Online]. Available: http://www.nwc.com/showitem.jhtml?docid=1425f2

[2] J. W. Stewart, *BGP4: Inter-Domain Routing in the Internet*. Reading, MA: Addison-Wesley, 1999.

[3] *RouteScience Technologies: PathControl*, [Online]. Available: http://www.routescience.com/products

[4] *Internap Network Services: Flow Control Platform,* [Online]. Available: http://www.internap.com

[5] *Nortel Networks: Alteon Link Optimizer*, , 2003 [Online]. Available: http://www.nortelnetworks.com/products/01/alteon/optimizer/collateral/n n102801-010203.pdf

[6] *Rether Networks: Internet Service Management Device,* [Online]. Available: http://rether.com/ISMD.htm

[7] *F5 Networks: BIG-IP Link Controller,* 2003 [Online]. Available: http://www.f5.com/f5products/pdfs/SS_BIG-IP_LC.pdf

[8] A. Akella, B. Maggs, S. Seshan, A. Shaikh, and R. Sitaraman, "A measurement-based analysis of multihoming," in *Proc. ACM SIGCOMM 2003*, Karlsruhe, Germany, Aug. 2003.

[9] L. Subramanian, S. Agarwal, J. Rexford, and R. H. Katz, "Characterizing the internet hierarchy from multiple vantage points," in *Proc. IEEE INFOCOM*, Jun. 2002.

[10] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker, "On the constancy of internet path properties," in *Proc. ACM SIGCOMM Internet Measurement Workshop (IMW)*, Nov. 2001.

[11] *Tinyproxy: A Simple Lightweight Web Proxy,* Nov. 2003 [Online]. Available: http://tinyproxy.sourceforge.net

[12] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D. Ullman, "Computing iceberg queries efficiently," in *Proc. VLDB 1998*, New York, Aug. 1998.

[13] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," in *Proc. ACM SIGCOMM 2002*, Pittsburgh, PA, Aug. 2002.

[14] E. M. Nahum, M. Rosu, S. Seshan, and J. Almeida, "The effects of wide-area conditions on WWW server performance," in *Proc. ACM SIGMETRICS*, Cambridge, MA, Jun. 2001.

[15] *Emulab: Network Emulation Testbed*, [Online]. Available: http://www.emulab.net

[16] D. Goldenberg, L. Qiu, H. Xie, Y. Yang, and Y. Zhang, "Optimizing cost and performance for multihoming," in *Proc. ACM SIGCOMM 2004*, Portland, OR, 2004.

[17] J. Pang, A. Akella, B. Krishnamurty, S. Seshan, and A. Shaikh, "On the responsiveness of dns-based network control," in *Proc. 2nd Internet Measurement Conf. (IMC) 2004*, Taormina, Italy, Oct. 2004.

[18] F. Guo, J. Chen, W. Li, and T. Chiueh, "Experiences in building a multihoming load balancing system," in *Proc. IEEE INFOCOM*, Hong Kong, Mar. 2004.

[19] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Rao, "Improving web availability for clients with MONET," in *Proc. 2nd Networked Systems Design and Implementation Conf.*, May 2005.

[20] *Radware: Peer Director*, [Online]. Available: http://www.radware.com/content/products/pd/

[21] *Stonesoft: Multi-Link Technology,* Oct. 2001 [Online]. Available: http://www.stonesoft.com/files/products/StoneGate/SG_Multi-Link_Technology_Whitepaper.pdf

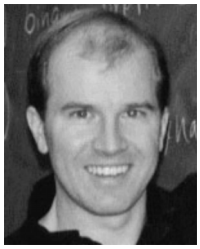
[22] *Rainfinity: Overview of RainConnect,* 2004 [Online]. Available: http://www.rainfinity.com/products/wp_rc_overview.pdf

[23] *Sockeye Networks*, [Online]. Available: http://www.sockeye.com

[24] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris, "Resilient overlay networks," in *Proc. 18th Symp. Operating System Principles*, Banff, Canada, Oct. 2001.

[25] A. Akella, J. Pang, B. Maggs, S. Seshan, and A. Shaikh, "A comparison of overlay routing and multihoming route control," in *Proc. ACM SIGCOMM 2004*, Portland, OR, Aug. 2004.

Aditya Akella (M'06) received the B.Tech. degree in computer science and engineering from IIT Madras in May 2000, and the Ph.D. degree in computer science from Carnegie Mellon University, Pittsburgh, PA, in September 2005. After spending an year at Stanford University as a Postdoctoral Associate, he joined the Computer Sciences faculty at the University of Wisconsin-Madison in August 2006. His research interests include Internet routing, network protocol design, Internet security, and wireless networking. (http://www.cs.wisc.edu/∼akella).



Bruce Maggs (M'92) received the S.B., S.M., and Ph.D. degrees in computer science from the Massachusetts Institute of Technology in 1985, 1986, and 1989, respectively.

After spending one year as a Postdoctoral Associate at MIT, he worked as a Research Scientist at NEC Research Institute, Princeton, NJ, from 1990 to 1993. In 1994, he moved to Carnegie Mellon University, Pittsburgh, PA, where he is now an Associate Professor in the Computer Science Department. His research focuses on networks for parallel and distributed computing systems.

In 1986, he became the first winner (with Charles Leiserson) of the Daniel L. Slotnick Award for Most Original Paper at the International Conference on Parallel Processing, and in 1994 he received an NSF National Young Investigator Award. He was co-chair of the 1993–1994 DIMACS Special Year on Massively Parallel Computation and has served on the program committees of SPAA, SODA, STOC, PODC, and many other technical conferences. (http://www.cs.cmu.edu/∼bmm).

Srinivasan Seshan (M'93) received the Ph.D. degree from the Computer Science Department, University of California, Berkeley, in 1995.

He is currently an Associate Professor and holds the Finmeccanica Chair in the Computer Science Department, Carnegie Mellon University, Pittsburgh, PA. From 1995 to 2000, he was a Research Staff Member at IBM's T. J. Watson Research Center. His primary interests are in the broad areas of network protocols and distributed network applications. In the past, he has worked on topics such as transport/routing protocols for wireless networks, fast protocol stack implementations, RAID system design, performance prediction for Internet transfers, Web server benchmarking, new approaches to congestion control, firewall design and improvements to the TCP protocol. His current work explores new approaches in overlay networking, sensor networking, online multiplayer games and wide-area Internet routing. (http://www.cs.cmu.edu/∼srini).

Anees Shaikh (M'99) received the B.S. and M.S. degrees in electrical engineering from the University of Virginia, Charlottesville, in 1994, and the Ph.D. degree in computer science from the University of Michigan, Ann Arbor, in 1999.

In 1999, he joined the IBM T. J. Watson Research Center as a Research Staff Member in the Network Software and Services group. His research focused on Internet services infrastructure, particularly the areas of content distribution, Web and network performance, and Internet measurement and traffic management. He has co-authored more than 35 technical papers in these and other areas. In 2006, he moved to IBM's Services Research division in which he leads a group focused on systems and network management problems related to IT service delivery. (http://www.research.ibm.com/people/a/aashaikh) .