USENIX Association

Proceedings of the General Track: 2004 USENIX Annual Technical Conference

Boston, MA, USA
June 27–July 2, 2004

Multihoming Performance Benefits: An Experimental Evaluation of Practical Enterprise Strategies

Aditya Akella, Srinivasan Seshan
Dept. of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Anees Shaikh
Network Software and Services
IBM T.J. Watson Research Center
Hawthorne, NY 10532

Abstract

Multihoming is increasingly being employed by large enterprises and data centers as a mechanism to extract good performance from their provider connections. Today, multihomed end-networks can employ a variety of commercial *route control* products to optimize performance over multiple ISP links. However, little is known about the mechanisms employed by such products and their relative trade-offs.

In this paper, we propose and evaluate a wide range practical schemes that could go into the design of a route control device and analyze their trade-offs. We implement the proposed schemes on a Linux-based Web proxy and perform a trace-based emulation of their relative performance benefits. We show that both passive and active monitoring based techniques are equally effective and could improve Web performance by about 25% when compared to using a single provider. Another key observation is that the conventional practice of employing historical measurement samples to monitor and predict ISP performance could, in fact, result in sub-optimal performance.

1 Introduction

Large enterprises, campuses, and data centers have traditionally used multihoming to multiple ISPs as a way of ensuring continued operation during connectivity outages or other ISP failures. While increased resilience and availability remain primary objectives of multihoming, there is increasing interest in deriving other benefits from multiple ISP connections. In particular, multihoming can be leveraged for improving wide-area network performance, lowering bandwidth costs, and optimizing the way in which upstream links are used [12].

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 providers using BGP [20, 18, 10]. Recognizing that not all enterprises are large enough to warrant full BGP peering with upstream ISPs, another class of products extends these advantages to smaller multihomed organizations which do not use BGP [14, 17, 7]. All of these products use a variety of mechanisms and policies for route control but aside from marketing statements, little is known about the relative quantitative benefits of these mechanisms.

In an recent measurement study to quantify the performance benefits from multihoming, it was shown that performance could potentially improve by more than 40% when multiple upstream providers are employed [4]. In that study, the focus was on the the maximum achievable benefits, assuming that the multihomed network had perfect information about the performance across all providers at any time and could change routes arbitrarily often. Hence, it is still unclear if, and how, these benefits can be realized in a more practical multihoming scenario.

In this paper we explore design alternatives to realize performance benefits from multihoming in practice, particularly for enterprises with multiple ISP connections. We focus primarily on 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).

We evaluate a variety of active and passive measurement strategies for multihomed enterprises to estimate the instantaneous performance of their provider links and pick the best provider for a given transfer. These strategies are evaluated in the context of a NAT-based implementation 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 provider link, the effects of sampling frequency on measurement accuracy, and the overhead of managing performance information for a potentially very large set of target destinations. We evaluate these policies using several client workloads, and an emulated wide-area network 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 providers, both offering about 15-25% improvement when compared to using a single provider. We also show that the most current sample of the performance to a destination via a given provider 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. We also conduct an initial study of mechanisms to control the ISP link used by external Internet clients who initiate connections to servers hosted in the enterprise.

The rest of this paper is structured as follows. In Section 2, we describe our enterprise multihoming solution and the various strategies for estimating ISP performance and for route control. Section 3 describes our implementation in further detail. In Section 4, we discuss the experimental set-up and results from our evaluation of the solution. Section 5 discusses some limitations inherent to our approach. Related work is presented in Section 6. Finally, Section 7 summarizes the contributions of this paper.

2 Solution Overview

In order to realize the performance benefits of multihoming, a route control solution requires three key functions: (1) monitoring provider links, (2) choosing the best provider link at a given instant, and (3) directing traffic over the best provider links. Figure 1 illustrates each of the functions. We discuss the functional design of each of these below. We discuss the actual implementation details in Section 3.

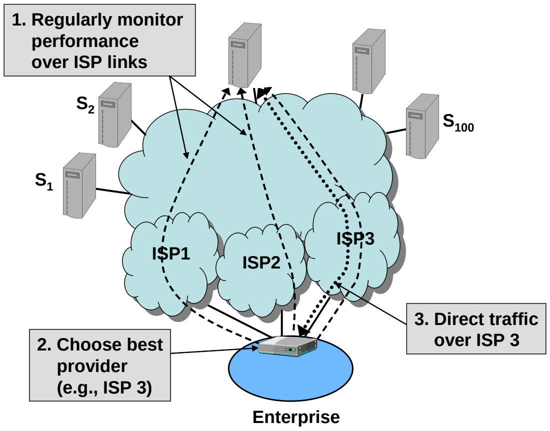

1. Regularly monitor performance over ISP links
S$_2$
S$_1$
S$_{100}$
ISP1
ISP2
ISP3
3. Direct traffic over ISP 3
2. Choose best provider (e.g., ISP 3)
Enterprise

Figure 1: **Solution steps:** This figure illustrates the three main operations of an enterprise route control system.

2.1 Monitoring Provider Links

Selecting the right provider link over which to direct each transfer is crucial to realizing the performance benefits of multihoming from the enterprise network's perspective. The choice of the right ISP clearly depends on the time-varying performance of each provider link to each destination being accessed. However, network performance could vary over small timescales, very drastically on some occasions [4, 22]. A multihomed enterprise, therefore, needs effective mechanisms to monitor the performance for most, if not all, destinations over each of its providers links.

There are two further issues in monitoring performance over provider 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 simple solution to this problem is to monitor only the most important destinations on the basis of the volume of requests made from the enterprise (e.g., the top 100 most frequently accessed destinations). This would ensure that a significant fraction of all flows will experience good performance.

For the second question (i.e., how to monitor), two common approaches are active monitoring and passive monitoring. Active monitoring works by having the multihomed enterprise perform out-of-band measurements of performance to or from the destinations selected by the policy used to determine what to monitor. These measurements could be simple pings involving, for example, ICMP `ECHO_REQUEST` or TCP `SYN` packets to the destinations. These measurements are to be taken over each provider at regular intervals.

On the other hand, 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 provider. However, in order to ensure that there are enough samples over all providers, it may be necessary to *explicitly direct* some transfers over particular links.

An important component of monitoring performance is the *time interval* of monitoring. A long interval between performance samples implies using stale information to estimate provider performance. This might result in a suboptimal choice of the provider link for a particular destination. While using smaller time intervals would address this, 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 providers in an attempt to obtain performance samples. As such, a careful choice of the interval size is crucial.

2.2 Choosing the Best Provider

The next component is to select the best provider for a destination at a given time based on past measurement samples from monitoring provider links. The core issue here is whether, and how, historical data about ISP performance to a given destination should be used at all. The performance of an ISP link to a destination can be tracked by keeping a smoothed, time-weighted estimate of the performance, for example 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:

$$\begin{aligned} EWMA_{t_i}(P, D) \quad &= \quad (1 - e^{-(t_i - t_{i-1})/\alpha}) s_{t_i} \\ &\quad + e^{-(t_i - t_{i-1})/\alpha} EWMA_{t_{i-1}}(P, D) \end{aligned}$$

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

2.3 Directing Traffic Over Selected Providers

Once the best-performing provider for a transfer is identified, the traffic from the destination must be directed over the chosen link. This is the main 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. Here, we discuss how these controls can be implemented.

If an enterprise has its own IP address block, it can advertise different address ranges to its upstream providers. Consider a site multihomed to two ISPs which owns a $/19$ address block. The site announces part of its address block on each provider link (e.g., a $/20$ sub-block on each link). Then, depending on which of the two provider links is considered superior for incoming 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 provider link. In cases where the enterprise is simply assigned an address block by its upstream provider, it may be necessary to also send outbound packets via the desired provider to ensure that the ISP forwards the packets.[1]

The process of ensuring that a connection uses a particular address must be handled differently for connections that are initiated from the enterprise than for those that are accepted into the site from external clients, as discussed below.

Initiated Connections: Handling connections initiated from an enterprise site amounts to ensuring that the remote content provider transmits data such that the enterprise ultimately receives it over the chosen provider. Inbound control can be achieved by having the edge router translate the source addresses on the connections initiated from its network to those belonging to the chosen provider's address block (i.e., the appropriate $/20$ block in the example above) via simple NAT-like mechanisms. This ensures that the replies from the destination will arrive over the appropriate provider.

Accepted Connections: Inbound route control over connections accepted into a site is necessary when the enterprise also hosts Internet servers which are accessed from outside. In this case, inbound control amounts to controlling the path (or the provider link) on which a given client is forced to send request and acknowledgment packets to the Web server. This is not easy since predicting client arrivals and forcing them to use the appropriate server address is generally not possible.

However, techniques based on DNS or deploying multiple versions of Web pages can help to achieve inbound control for externally initiated connection. For example, the enterprise can use a different version of a base Web page for each provider link. The hyperlinks for embedded objects in the page could be written with IP addresses corresponding to a given provider. Then, arriving clients would be given the appropriate base HTML page such that subsequent requests for the embedded objects arrive via the selected provider. 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. A preliminary study of its effectiveness is discussed in Section 5. In this paper, we focus primarily on the case of enterprise-initiated connections.

3 Implementation Details

We implement the multihoming route control functions discussed above by extending a simple open source Web proxy called *TinyProxy* [3]. 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 being employed by a multihomed end-network with three ISP links.

3.1 Performance Monitoring Algorithms

We implement both the active and passive measurement mechanisms, described in Section 2.1, for monitoring the performance of upstream provider links.

3.1.1 Passive Measurement

The passive measurement module tracks the performance to destinations of interest by sampling provider links using Web requests initiated by clients in the enterprise. The basic strategy is to use new requests to sample an ISP's performance to a given destination if the performance estimate for that ISP

[1] In fact, like most enterprise route control products, we enforce outbound route control by transmitting packets *to* a destination along the same provider as the one on which the traffic *from* the destination is received.

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 (i.e., either the IP address or the domain name of the destination). A hash table entry holds the current estimates of the performance to the destination via the three providers, along with an associated timestamp indicating the last time performance to the destination via the provider was measured. This is necessary for updating the EWMA estimate of performance (Section 2.2).

Notice that without some explicit control, the hash table maintains performance samples to all destinations, including those rarely accessed. One concern is that this could cause a high overhead of measurement, with connections to less popular destinations being 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 $\sim$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 (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 $O(\text{hash table size})$ time.

Other ways of tracking top destinations such as Iceberg Queries [8] or Sample-and-hold [6], 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.

Figure 2 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 a provider link chosen randomly, in a load-balancing fashion.

If there is an entry for the destination, the passive scheme scans the measurement timestamps for the three providers to see if the elapsed time since the last measurement on any of the links exceeds the predefined *sampling interval*. If so, the performance to the destination along one of these providers links is sampled using the current connection.

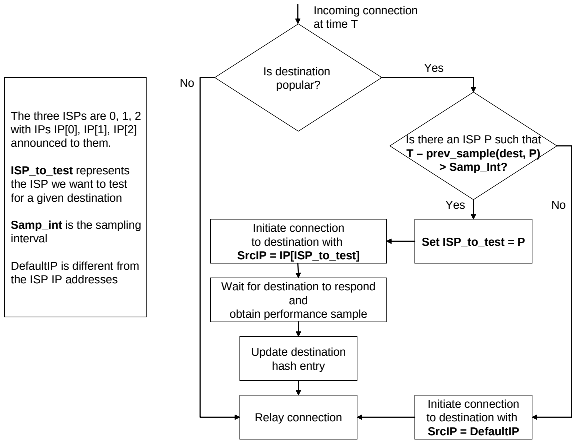

Incoming connection
at time T
Is destination popular?
Yes
No
Is there an ISP P such that T – prev_sample(dest, P) > Samp_Int?
Yes
No
Set ISP_to_test = P
Initiate connection to destination with SrcIP = IP[ISP_to_test]
Wait for destination to respond and obtain performance sample
Update destination hash entry
Relay connection
Initiate connection to destination with SrcIP = DefaultIP
The three ISPs are 0, 1, 2 with IPs IP[0], IP[1], IP[2] announced to them.
ISP_to_test represents the ISP we want to test for a given destination
Samp_int is the sampling interval
DefaultIP is different from the ISP IP addresses

Figure 2: **Monitoring provider performance:** The passive measurement scheme.

In order to obtain a measurement sample on a provider 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 2.2). 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 3.3.

3.1.2 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 providers. For active measurement, we use two techniques to identify which destinations should be monitored.

FrequencyCounts. Just like the passive measurement mechanism, in this scheme we track the number of client requests directed to each destination. Every T seconds (the sampling interval), we initiate active probes to those destinations for which the number of requests exceeds a fixed threshold.

SlidingWindow. This scheme maintains a window of size C that contains the C most recently accessed destinations. The

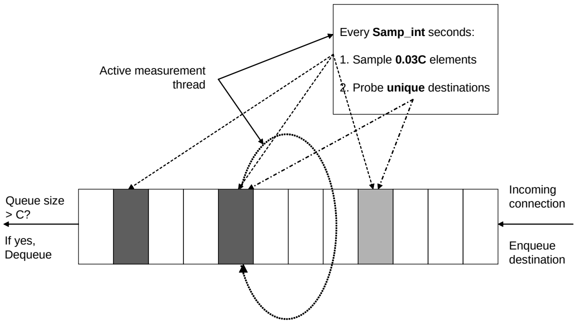


Figure 3: **Monitoring provider performance:** The *Sliding-Window* active measurement scheme.

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 providers. This is illustrated in Figure 3.

The two active measurements schemes have their respective advantages and disadvantages. 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 the top destinations by popularity. *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* can effectively track small shifts in the underlying popularity distribution of the destinations and try to optimize performance to such temporarily popular destinations.

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 providers 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 `SYN+ACK` packets.

When a response is received, we update the performance estimates to the destination for the corresponding provider, along with the measurement timestamp. As described above, we update the performance estimate using the EWMA computation. 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, and the performance is updated accordingly.

3.2 Switching Providers

After updating all provider entries for a destination in the performance hash, we switch to a new provider only if it offers at least a 10% performance improvement over the current best provider 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 provider per destination also changes at the same frequency.

3.3 NAT-based Inbound Route Control

Our inbound route control mechanism is based on manipulating NAT tables at the Web proxy to reflect the current choice of best provider. 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 provider 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 provider.

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 to be measured.

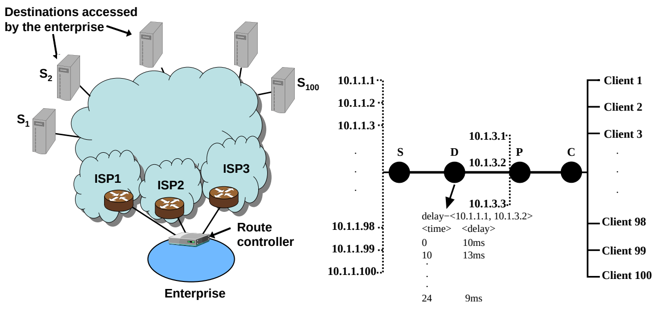
4 Experimental Evaluation

In this section, we describe our experimental evaluation of the various design alternatives proposed in Section 3. 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.

4.1 Experimental Set-up

We first describe our testbed setup and discuss how we emulate realistic wide-area network delays. Then we discuss key characteristics of the delay traces we employ in our emulation. Finally, we discuss the performance metrics we use to compare the proposed schemes.

4.1.1 Testbed topology



(a) Multihomed enterprise (b) Testbed topology

Figure 4: **Testbed topology:** The simple test-bed, shown in (b), is used to emulate the route control scenario shown in (a).

We use the simple testbed topology shown in Figure 4(b). Our goal is to emulate a moderately-sized enterprise with three provider connections and a client population of about 100 (shown in Figure 4(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 being the most popular and 10.1.1.100 being the least popular.

Node C runs 100 instances of clients in parallel, each of which makes requests to the Web sites 10.1.1.1 through 10.1.1.100 as follows. 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 $\frac{100}{\lambda}$ requests per second at the server S. Each client request is for the i^{th} 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 5KB.

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 provider links.

Node D is a *delay element*, running WaspNet [13], 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 Figure 4(b).

In order 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, each attached to a unique provider. We then run pings at regular intervals of 10s from each of them 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.

In this measurement, the three Akamai machines in Chicago collectively act as a stand-in for a multihomed network with three provider connections. The 100 Akamai servers probed represent destinations contacted by end-nodes in the multihomed network. We use the series of delay samples between the three Akamai sources and the 100 destination servers as inputs to the WaspNet module to emulate delays across each provider link.

4.1.2 Compressing time

It is quite time-consuming to emulate the entire day's worth of delays, multiple times over, to test and tune the different 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 collected shows that there is not much variations 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 changes in the performance of the underlying network paths. With this in mind, our approach is to compress the 24-hour delay traces by a factor of 10, to 2-hour delay traces and use these as the real 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 1, column 2. We use these delay traces in our emulation.

We also wanted to ensure that the delays observed from the Chicago source nodes were not significantly different from typical delays experienced by a well-connected, multihomed network located in a major U.S. metropolitan area. Hence, we collected similar traces from 3 source nodes located in two other cities, namely 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 1. 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.

	Chicago trace	**NYC trace**	**LA trace**
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

Table 1: Characteristics of the delay traces. Here "performance" refers to the delay along a given path.

4.1.3 Comparison Metric

To evaluate the benefit from using various route control schemes we compare the response time of transfers when using the scheme (i.e., $Resp_{(x,scheme)}$, for a transfer x), with the response time when the best of the three providers is employed for each transfer ($min_i\{Resp_{(x,ISP_i)}\}$):

$$\mathcal{R}_{scheme} \quad = \quad \frac{1}{|x|}\sum_x \frac{Resp_{(x,scheme)}}{min_i\{Resp_{(x,ISP_i)}\}} \qquad (1)$$

Where, $|x|$ is the total number of transfers. We call $\mathcal{R}$ the "performance metric" or the "normalized response time". The closer $\mathcal{R}$ is to 1, the better the performance of the scheme.

In the above computation, the response times from employing the best provider for any transfer (the terms in the denominator above) are computed in an offline manner for each transfer by forcing it to use each of the three providers and selecting the provider offering the best response time.

4.2 Experimental Results

We perform our experiments on the Emulab [5] testbed. We use 600MHz Pentium III machines with 256MB RAM, running Red Hat 7.3. We first describe how we select different client workloads in our evaluation, and then move on to the evaluation of different route control strategies.

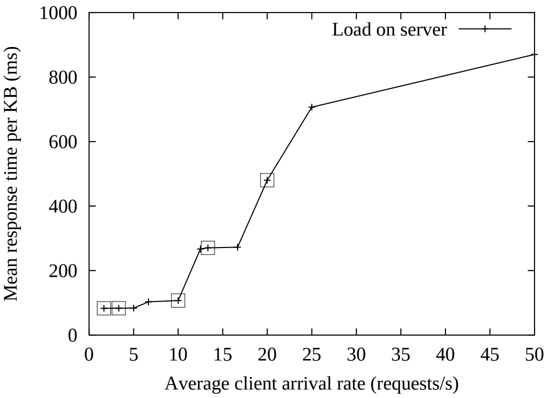


Figure 5: **Web server load profile:** Average response time in ms, per KB of the request, as a function of the average client arrival rate at the server in our topology (Figure 4(b)).

4.2.1 Selecting the Client Workloads

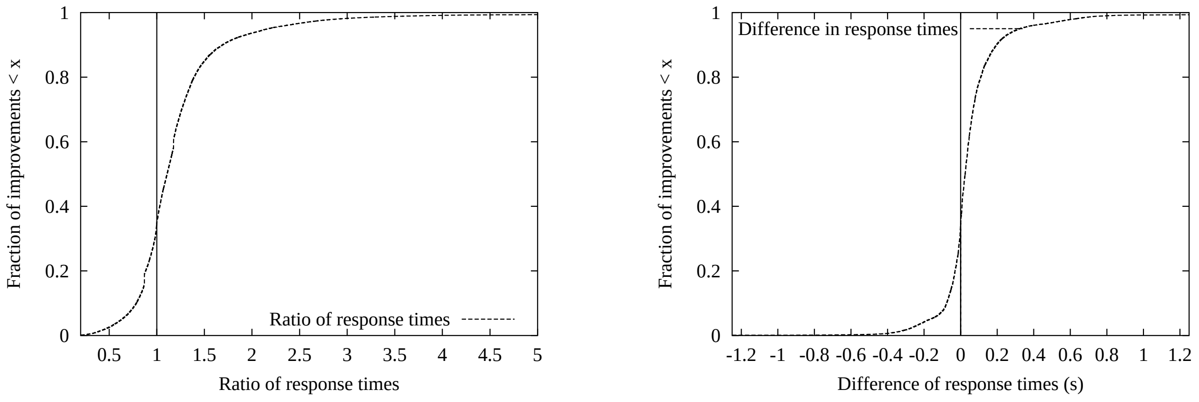
In Figure 5 we show the average response time per KB of client requests (i.e., the completion time for a request divided by the size of the request in KB), as a function of the average arrival rate of clients at the server S (i.e., $\frac{100}{\lambda}$ requests/s). The response time quickly degrades 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.

In the remainder of the evaluation we focus on addressing the following questions:

- To what extent do the route control schemes improve the performance of the multihomed site, relative to using the single best provider alone?
- Does employing historical samples help in better estimating future provider performance?
- How do active and passive measurement schemes compare in terms of the performance improvement they offer? Which of the two active measurement schemes – *SlidingWindow* or *FrequencyCounts* – works better?
- At what time intervals should samples for provider performance be collected?
- What overheads do the proposed mechanisms incur?

4.2.2 Improvements from Route Control

The aggregate performance improvement from the passive measurement-based schemes is shown in Figure 6. Here, we



(a) Ratio of response times

(b) Difference in response times

Figure 7: **Unrolling the averages:** Ratio and the difference in the response times from using just ISP 3 for all transfers relative to using the passive measurement scheme. The average client arrival rate in either case is 13.3 requests/s.

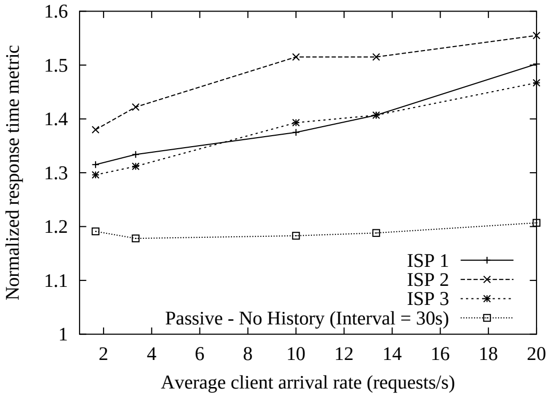


Figure 6: **Performance improvement:** The performance metric $\mathcal{R}$ for the passive measurement scheme with EWMA parameter $\alpha = 0$ (no history employed) and sampling interval of 30s. The graph also shows the performance from the three individual providers.

set the EWMA parameter $\alpha = 0$ so that only the current measurement samples are used to estimate provider performance, and select a sampling interval of 30s. The figure plots the performance for the five client workloads. In addition, we show the performance from using the three providers individually.

The performance improvement relative to the best individual provider 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, however. The results for other sampling intervals (60s, 120s, 300s and 450s) are similar, and are omitted for brevity. The performance improvements from using the active measurement-based schemes are also similar and are discussed later.

Figures 7(a) and (b) illustrate the distribution of the response time improvements offered by the passive measurement scheme (for $\alpha = 0$ and sampling interval = 30s) relative to being singly-home to the best provider from Figure 6, i.e., ISP 3. Figure (a) plots the CDF of the ratio of the response time from using ISP 3 to the response time from the passive measurement scheme across all transfers. These results are for the specific instance where the client arrival rate is 13.3 requests/s at the server. Figure (b) similarly plots the *difference* in the response times for the same client workload.

Notice, from either figure, that the passive measurement scheme improves the response time performance for over 65% of the transfers. Figure 7(a) shows that this route control scheme improves the response times by factors as large as 5 for a small fraction of transfers (about 1%), relative to being singly-homed. Similarly, Figure 7(b) shows that the scheme can improve the response time by more than 1s for some transfers. Notice also, from either figure, that the passive measurement-based scheme ends up offering sub-optimal performance for about 35% of the transfers.

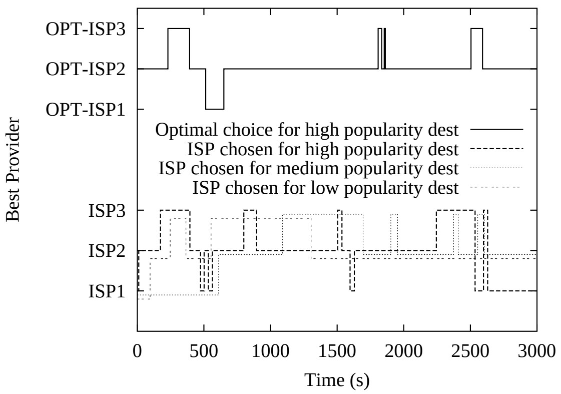


Figure 8: **Route control at work:** The providers chosen by the passive measurement-based route control scheme for destinations with different popularity levels.

Figure 8 illustrates the operation of the passive measurement-based scheme. In this figure, we show the providers used over time for transfers to three different destinations – a popular destination (10.1.1.4), a moderately popular destination (10.1.1.16), and a less popular destination (10.1.1.38). Recall that the passive measurement-based scheme explicitly tracks and controls candidate paths to the 40 most popular destinations. The sampling interval is 30s and the client arrival rate is about 13.3 requests/s.

From this figure, we see that changes to the route for the popular destination is made every 160s on an average. For the moderate and less popular destinations, the intervals are 300s and 550s respectively. For the passive scheme, the number of route changes depends on the popularity of the destinations – the more popular a destination is, the higher the frequency of its route changes. Figure 8 also shows the optimal choice of providers for the popular destination as a function of time, as determined from the underlying delay traces. Comparing this with the ISPs actually selected by the scheme for this destination illustrates cases where the scheme sometimes makes a sub-optimal choice (e.g., between 750-800s, around 1500s, and 2250-2450s).

4.2.3 Employing History to Estimate Performance

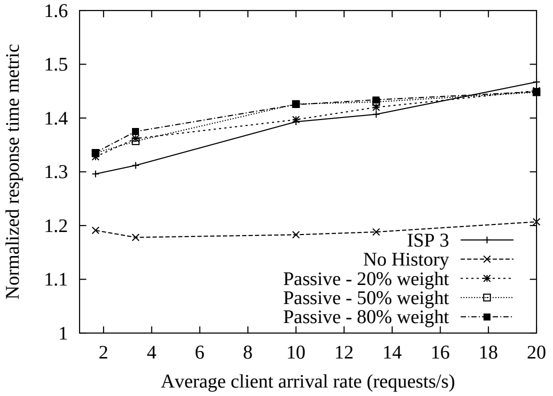


Figure 9: **Impact of history:** The performance achieved by relying on historical samples to varying degrees. These results are for the passive measurement-based strategy with a sampling interval of 30s.

Figure 9 plots the performance of the passive measurement scheme for three different values of the parameter 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 30s, 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 provider. These results show that the best way to estimate provider performance is to just use the current performance sample as an estimate of near-term performance.

4.2.4 Active vs Passive Measurement

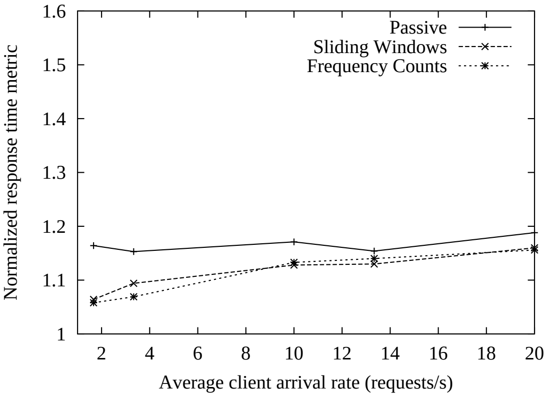


Figure 10: **Active vs passive measurement:** The performance of the two active measurement-based schemes, and the passive measurement scheme for a sampling interval of 120s.

In Figure 10, we compare the performance from the two active measurement based 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 120s across the five client workloads.

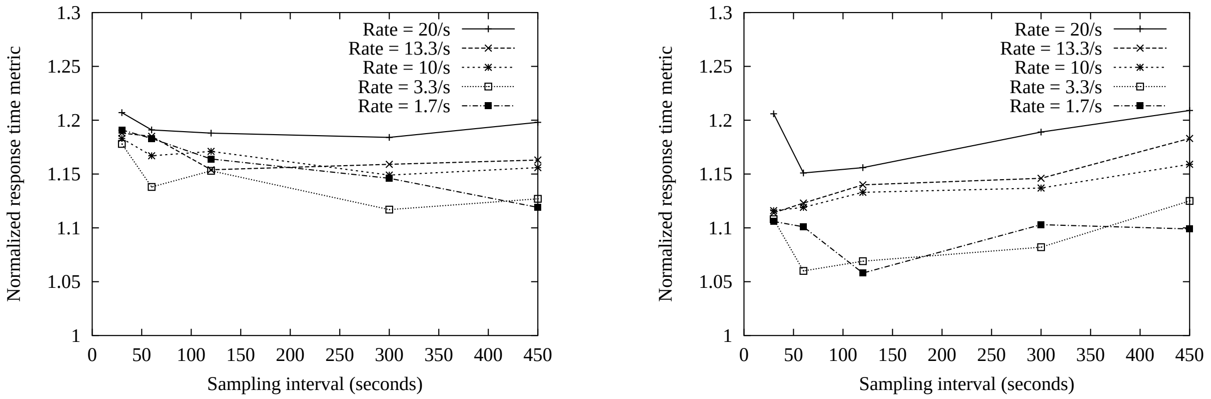
Note that the two active measurement schemes offer comparable performance. Unfortunately, the workloads we selected do not bring out other underlying trade-offs of these schemes (discussed earlier in Section 3.1.2). A detailed comparison of these active measurement schemes is future work.

Figure 10 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 is expected, since the passive scheme uses existing transfers to obtain samples of performance across the, potentially sub-optimal, ISP links.

4.2.5 Frequency of Performance Monitoring

Figure 11 shows the impact of the measurement frequency on the aggregate performance for the passive measurement scheme (Figure 11(a)) and the *FrequencyCounts* active measurement scheme (Figure 11(b)). Each figure plots the results for the five client workloads.

From Figure 11(a) we notice that longer sampling intervals surprisingly offer slightly better performance for passive measurement. To understand this better, consider the curve for the client arrival rate of 10 requests/s. A client arrival



(a) Passive measurement

(b) *FrequencyCounts*

Figure 11: **Impact of the sampling interval:** The performance from using different sampling intervals from passive measurement-based and the *FrequencyCounts* active measurement-based schemes.

rate of 10 requests/s implies that an average of $10T$ connections are made by the clients every T seconds, where T is the sampling interval. However, in order to obtain samples for a fraction f of the 100 destinations over the three providers, the passive measurement scheme will have to force $300f$ connections across the provider links. This leaves a fraction $1 - \frac{30f}{T}$ which are not employed for measurement, and could be routed along the optimal provider, assuming that the passive measurement yields reasonably accurate estimate of performance[2]. As T increases, the fraction of connections routed over the optimal path is likely to increase, resulting in a marginal improvement in performance. This explains the slight downward slopes in Figure 11(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 Figure 11(a). However, Figure 11(b), which plots the performance from the *FrequencyCounts* scheme as a function of the sampling interval, sheds more light on this effect. A sampling interval of 450s suffers a 5-8% performance penalty relative to a smaller interval such as 60s. Notice that in the case of *FrequencyCounts* too, aggressive sampling (e.g, an interval of 30s) could slightly impact overall performance on some occasions due to the increased software overheads at the proxy.

4.2.6 Analysis of overheads

As the performance results show, both passive and active measurement are still 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 provider choices, (2) overhead of performing measurement, and (3) software overhead, specifically, the overhead of making frequent updates to the NAT table[3] and employing NAT rules on a significant fraction of packets. In this section, we analyze the contribution of these factors on the eventual performance of the different schemes.

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

Table 2: **Analysis of performance overheads.** Here "penalty" is defined as the value of $\mathcal{R} - 1$ in each case.

Our approach to quantify the overhead of our implementation is to 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 (the terms in the denominator in Equation 1). If we combine these offline response time 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, $\mathcal{R}$, and 1 gives us the performance penalty, not including overheads of the implementation.

[2] About a third of the connections employed for measurement can be expected to be routed along their optimal providers

[3] We could allow routes to change less frequently than the sampling interval, T, (e.g., every $T' > T$ seconds) but since we do not use performance history, this would be equivalent to sampling and updating every T' seconds.

The penalties from the above analysis for the three proposed schemes are shown in Table 2, row 2. The client arrival rate is 13.3 requests/s and the sampling rate is 30s. 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 Figure 11(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).

5 Implications and Discussion

The key findings from our evaluation are as follows:

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

2. We show that relying on historical samples to monitor performance of ISPs (e.g., using EWMA) is not very useful, and sometimes may be detrimental to performance. The most current measurement sample is a very good estimator of near-term performance of an ISP link.

3. Both passive and active measurement-based schemes offer competitive performance, with the latter offering better performance for lighter client workloads. For the generic Web workloads we tested with, both active measurement implementations – *SlidingWindow* and *FrequencyCounts* – showed similar performance benefits.

4. The overhead introduced by aggressive performance sampling may slightly reduce the overall performance benefit of route control schemes. A sampling interval on a minutes timescale, e.g., 60s, seems to offer very good performance overall.

5. The overhead from measurements and frequent updates to the NAT table are negligible. Most of the performance penalties arise from the inaccuracies of the measurement and estimation techniques.

5.1 Additional Issues

The route control mechanisms we presented and analyzed are a first attempt at understanding how to extract good performance from multiple provider connections in practice. There are clearly a number of ways in which they can be improved, however. Also, we do not address several important issues, such as ISP costs and the interplay of performance and reliability optimization. Below, we briefly discuss some of these potential improvements and issues.

Handling lost probes. In our implementation of the active probing schemes, we send just one probe when collecting a performance sample for a (*ISP link*, *destination*) pair. It is therefore possible that lost probes, e.g., due to transient congestion or even timeouts, may be misinterpreted for poor performance of the provider path to the destination. This can in turn cause unwanted changes in the ISP choice for the destination. We can mitigate this by sending a short burst of, say, three probes per (*ISP link*, *destination*) pair. Then the performance reported by all three probes can be used to estimate the quality of the ISP link, perhaps with a weighting to account for any observed losses.

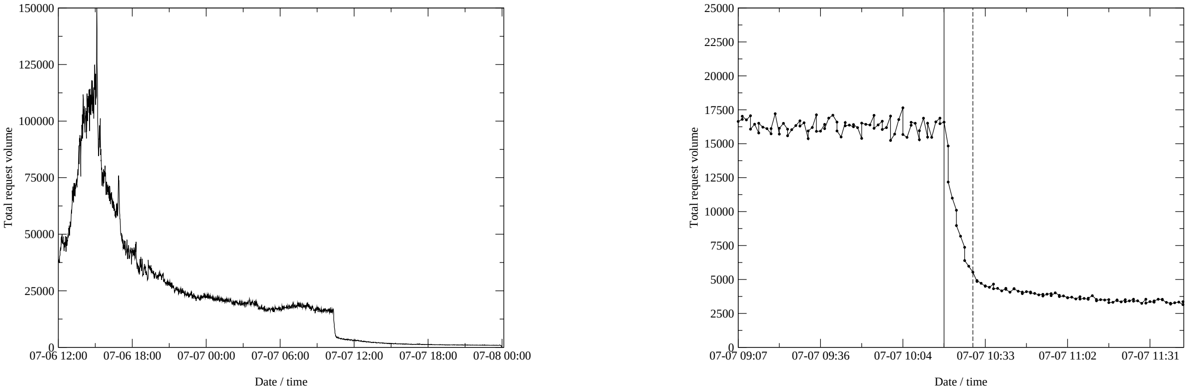
Hybrid passive and active measurements. The accuracy of passive measurement can be improved by sending active probes immediately after a failed passive probe, for example when the observed connection ends unexpectedly. This increases confidence that the failed connection is due to a problem with the provider link, as opposed to a transient effect.

In our implementation, paths to less popular destinations are not explicitly monitored (in both active and passive schemes). 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 a number of failures on connections to an unpopular destination, it can immediately switch the destination's default provider to one of the remaining two providers for future transfers.

Balancing performance and resilience. The goal of most current multihoming deployments is to provide resilient connectivity in the face of network failures. Hence, one of the main functions of a route control product is to respond quickly to ISP failures. One of our findings is that even with a relatively long sampling interval, the performance advantages of multihoming can be realized. 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 and good performance (Figure 11(b)), while ensuring a failover time of about one minute.

ISP pricing structures. In our study, we ignore issues relating to the the cost of the provider 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." Though we do not consider how to optimize overall bandwidth costs, our evaluation of active and passive monitoring, and the utility of history, are central to more general schemes that optimize for both cost and performance.

Long-lived TCP flows. In our route control schemes, an update to a NAT entry for a destination in the midst of an ongoing transfer involving that destination could cause the transfer to fail (due to the change in source IP address). We did not observe many failed connections in our experiments,



(a) 2-day trace

(b) 2-hour portion

Figure 12: **DNS responsiveness:** This figure shows traffic volume over time just before and after a DNS change. The left graph (a) shows a 2-day period around the end of the event, while (b) focuses on a 2-hour period around the time of the DNS update.

however, and most of the flows were very short. However, this effect is nevertheless likely to have a pronounced impact on the performance of long-lived flows. It is possible to address this problem by delaying updates to NAT table until after ongoing large transfers complete. However, this increases the complexity of the implementation since it involves identifying flow lengths, and checking for the existence of other long-lived flows at the time of update. It may also force other short flows to the same destination to traverse sub-optimal ISPs while the NAT update is delayed.

Issues for further study. We do not address the impact that announcements of small address sub-blocks to different upstream ISPs (Section 2.3) has on the on the inflation of the routing table size in the core of the network. We also do not consider the potential impact of interactions when many enterprises deploy intelligent route control to each optimize their own multihomed connectivity. This will likely have an affect on the marginal benefits of the route control solutions themselves, and on the network as a whole. We leave these issues for future consideration.

Our implementation primarily considered handling connections initiated from within the enterprise, as these are common for current enterprise applications (e.g., to contact content providers). A route control product must also handle connections from outside clients, however, to enable optimized access to servers hosted in the enterprise network. Next, we describe some preliminary measurements regarding the usefulness DNS for externally-initiated connections.

5.2 DNS for Inbound Route Control

When deploying Internet servers in a multihomed environment, it is useful to be able to transparently direct connections initiated by external clients over a specific link, according to performance or other metrics. Recently, several route control device vendors have introduced features to use Domain Name System (DNS) resolution requests as a means to direct inbound client traffic over the desired link. In this scheme, it is assumed that the destination IP address used by the client determines which ISP link is used for the connect request. Hence, by responding with the appropriate IP address when the client makes a request to resolve a service name (e.g., www.service.com), the inbound link can be selected. This is very similar to using DNS as a server selection mechanism in content distribution networks [19].

While DNS is a convenient and relatively transparent mechanism, it is unclear whether it can respond quickly enough for dynamic route control. Responses to name resolution requests have an associated time-to-live (TTL) value that determines how long the response should be cached by the client's local nameserver. Ideally, by setting the TTL to a very small value (e.g., 10s or even zero), it is possible to force external clients to resolve the IP address frequently, thus providing fast responsiveness. In practice, however, this is complicated by the behavior of the wide variety of applications and DNS servers deployed in the Internet. Many applications perform their own internal DNS caching that does not adhere to the expected behavior, and some older implementations of DNS software have been reported to ignore low TTL values. These artifacts make it difficult to predict how quickly clients will respond to changes communicated via DNS responses.

In order to quantify the responsiveness of DNS in practice, we perform a simple analysis of client behavior in response to DNS changes during a large Web event. We collect logs from a set of Web caches that served requests for content related to a Summer 2003 sporting event with global audience. During the event when the request rate was very high, the authoritative nameservers directed all clients to the set of caches with a 10min TTL. After the event, the nameservers were updated to direct clients to lower capacity origin servers. Ideally, all traffic to the caches should subside after 10min.

Figure 12 shows the aggregate request volume to all caches over time, just before and after the DNS change, where requests were gathered into 1-minute intervals. During the one-hour period after the DNS change, requests came from about 13400 unique client IP addresses and 5600 unique IP subnets. The number of subnets is computed by clustering client IP addresses using BGP tables obtained from [11, 2].

Figure 12(a) shows the last part of the trace, with a clear peak occurring on the last day of the event, followed by a period of relatively constant and sustained traffic, and finally a sharp dropoff corresponding to the time when the DNS is updated. Figure 12(b) focuses on the time around the DNS update; the solid line denotes the time of the update and the dashed line is the time when the 10min TTL expires. Between these times, the request volume decreases by 66%. The remaining third of the traffic decays very slowly over a period of more than 12 hours. While this analysis is not definitive, it does suggest that DNS is at best a coarse-grained mechanism for controlling traffic.

6 Related Work

In a previous study we considered the extent to which multihoming can be leveraged by enterprises and Internet data centers to improve network performance [4]. This measurement-based study revealed the maximum benefits attainable in a variety of scenarios, but provided only limited guidance as to how to extract those improvements. In this paper, we perform an experimental evaluation of a number of practical techniques for using multihoming to improve performance.

In a study closely related to ours, the authors conduct a few trace-driven experiments to evaluate several design options using a commercial multihoming device [9, 17]. 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. In addition, their results show that managing load at a connection-level granularity is only slightly less effective than per-packet load balancing. They also show that using knowledge of the asymmetric nature of some applications (e.g., Web connections) can be useful in improving traffic balance, although it requires additional application-specific information.

A number of vendors have recently developed dedicated networking appliances [15, 7, 14] or software stacks [21, 16] 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 examples large data centers or campuses [18, 1]. 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 use “race”-based performance measurements, in which 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 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.

7 Summary and Ongoing Work

Our goal in this paper was to quantitatively evaluate a variety of practical mechanisms and policies for realizing performance benefits from ISP multihoming. We focused on the scenario of multihomed enterprises that wish to leverage multiple providers to improve the response time performance for clients who download content from Internet Web servers. Using a real Linux-based route control implementation and an emulated wide-area network testbed, we experimentally evaluated several design alternatives. These included the performance of passive versus active monitoring schemes, sensitivity to various measurement sampling intervals, and techniques to manage performance information for a potentially very large set of target destinations.

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 provider. Our experiments also show that using historical performance to choose the best ISP link is not necessary – the most current measurement sample gives a good estimate. We showed that the performance penalty from collecting and managing performance data across various destinations is negligible.

Although our evaluation was done using an emulated wide-area network and actual delay traces, it is valuable to deploy our implementation in a multihomed site for further experimentation and evaluation. To this end, we are planning to install our route control proxy device in a commercial multihomed data center in which we can perform additional experiments and uncover other wide-area effects.

Acknowledgments

We are grateful to Prof. Bruce Maggs (CMU) for his support and assistance in facilitating measurement experiments on the Akamai network. We are grateful to Herbie Pearthree, Rance Smith (IBM Global Services), Jehan Sanmugaraja and Paul Dantzig (IBM Research), for their assistance in collecting data for the DNS analysis and discussions regarding the practical usage of route control and DNS. Our work has greatly benefited from discussions with Ashwin Bharambe, Dave Maltz, Hui Zhang (CMU), Dave Andersen (MIT) and Sridhar Machiraju (UC Berkeley). Finally, we thank our shepherd, Carl Staelin, and our anonymous reviewers for their invaluable feedback on the presentation of this paper.

References

[1] Sockeye Networks, Inc. `http://www.sockeye.com`.

[2] University of Oregon, RouteViews Project. `http://www.routeviews.org`.

[3] Tinyproxy: A Simple Ligh-weight Web Proxy. `http://tinyproxy.sourceforge.net`, November 2003.

[4] Akella, A., Maggs, B., Seshan, S., Shaikh, A., and Sitaraman, R. A Measurement-Based Analysis of Multihoming. In *Proc. ACM SIGCOMM 2003* (Karlsruhe, Germany, August 2003).

[5] Emulab: Network Emulation Testbed. `http://www.emulab.net/`.

[6] Estan, C., and Varghese, G. New Directions in Traffic Measurement and Accounting. In *Proc. ACM SIGCOMM 2002* (Pittsburgh, PA, August 2002).

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

[8] Fang, M., Shivakumar, N., Garcia-Molina, H., Motwani, R., and Ullman, J. D. Computing Iceberg Queries Efficiently. In *Proc. VLDB 1998* (New York, August 1998).

[9] Guo, F., Chen, J., Li, W., and cker Chiueh, T. Experiences in Building a Multihoming Load Balancing System. In *INFOCOM 2004* (Hong Kong, March 2004).

[10] Internap Network Services: Flow Control Platform. `http://www.internap.com`.

[11] Krishnamurthy, B., and Wang, J. On Network-Aware Clustering of Web Clients. In *Proc. ACM SIGCOMM 2000* (Stockholm, Sweden, August 2000).

[12] Morissey, P. Route Optimizers: Mapping Out the Best Route. *Network Computing* (December 2003). `http://www.nwc.com/showitem.jhtml?docid=1425f2`.

[13] Nahum, E. M., Rosu, M., Seshan, S., and Almeida, J. The Effects of Wide-Area Conditions on WWW Server Performance. In *Proc. ACM SIGMETRICS* (Cambridge, MA, June 2001).

[14] Nortel Networks: Alteon Link Optimizer. `http://www.nortelnetworks.com/products/01/alteon/optimizer/collateral/n%n102801-010203.pdf`, 2003.

[15] Radware: Peer Director. `http://www.radware.com/content/products/pd/`.

[16] Rainfinity: Overview of RainConnect. `http://www.rainfinity.com/products/wp_rc_overview.pdf`, 2004.

[17] Rether Networks: Internet Service Management Device. `http://rether.com/ISMD.htm`.

[18] RouteScience Technologies: PathControl. `http://www.routescience.com/products`.

[19] Shaikh, A., Tewari, R., and Agrawal, M. On the Effectiveness of DNS-based Server Selection. In *Proc. IEEE INFOCOM* (Anchorage, AK, April 2001).

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

[21] Stonesoft: Multi-Link Technology. `http://www.stonesoft.com/files/products/StoneGate/SG_Multi-Link_Technol%ogy_Whitepaper.pdf`, October 2001.

[22] Zhang, Y., Duffield, N., Paxson, V., and Shenker, S. On the Constancy of Internet Path Properties. In *Proc. ACM SIGCOMM Internet Measurement Workshop (IMW)* (November 2001).