

Server-centric View of Internet Performance: Analysis and Implications

Lili Qiu and Venkata N. Padmanabhan

September 2001

Technical Report
MSR-TR-2001-78

Microsoft Research
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
<http://www.research.microsoft.com>

Abstract

In this paper, we present a study of wide-area Internet performance as observed from a busy Internet server site. We explore the question of predicting the network performance that a client would observe based on topological metrics and past performance data. Specifically, we seek to answer the following questions: (a) can network performance for a client be predicted based on past performance for “nearby” clients?, (b) how well does performance correlate with topological metrics such as router hop count or AS hop count?, and (c) is it possible to identify sections of the network (e.g., congested links) that significantly impact network performance for downstream clients? We discuss the implications of our findings on wide-area server selection and performance prediction.

1 Introduction

The rapid growth of the Internet has resulted in ever-increasing heterogeneity of hosts, networks, and links that comprise the Internet. From the viewpoint of an Internet service, such as a Web site, this heterogeneity presents interesting challenges. No two clients that connect to the service are identical. Specifically, the clients may have very different network connectivities to the Internet in general (e.g., a LAN client and a dialup client) and/or to the server in particular (e.g., a client in San Francisco and one in New Delhi connecting to the Microsoft server in Seattle). Furthermore, the quality of network connectivity can vary with time because of network congestion.

Our work is motivated by this heterogeneity in network connectivity. We analyze wide-area Internet performance by tracing connections between servers in the *microsoft.com* site [12] and clients distributed across the Internet. We are especially interested in the question of estimating the network performance that clients connecting to the site would see, based on past performance data. We believe that this is a significant issue because being able to predict client performance has a bearing on two important techniques to accommodate heterogeneity in client connectivity:

1. The server may maintain multiple version of the content (say of different sizes) and might

serve the version that is most appropriate given the expected network connectivity between itself and the client. To do such *content negotiation* effectively, we need to be able to accurately estimate the expected network performance to the client.

2. The server site and/or content might be replicated at multiple locations (say on a content distribution network (CDN)). A client is typically directed to the replica to which it is likely to have the best network connectivity (modulo factors such as server load). Strictly speaking we only care about determining the *relative* ordering of replicas based on network performance rather than estimating the actual network performance.

We seek to answer several questions: (a) can network performance for a client be predicted based on past performance for “nearby” clients?, (b) how well does performance correlate with topological metrics such as router hop count or AS hop count?, and (c) is it possible to identify sections of the network (e.g., congested links) that significantly impact performance for downstream clients?

One question is what “network performance” means. Clearly, the performance metrics that matter depend on the application. Latency may be most critical in the case of gaming servers while throughput may be the most important metric for software download servers. In our study, we consider two metrics: the round-trip time (RTT) and the packet loss rate. We view these metrics as being more fundamental than throughput since the latter is affected by factors such as the workload (e.g., bulk transfers versus short Web transfers) and the transport protocol (e.g., the specific variant of TCP and TCP options employed by the server and the client). Furthermore, it is possible to obtain a rough estimate of throughput knowing the packet loss rate and RTT, using an analytical model of TCP such as [14].

Here is an overview of the rest of this paper. In Section 2, we discuss related work, and in Section 3, we describe our experimental setup and methodology. We begin our analysis in Section 4 by investigating the relationship between network performance metrics, such as RTT and loss rate, and topological server-to-client “distance” metrics,

such as router hop count. Our results indicate that there is little or no correlation between topological distance and performance. One reason is that not all network links are “equal”; a few lossy links may be responsible for much of poor performance seen by clients. In Section 5, we present a scheme for identifying lossy network links by *passively* observing end-to-end performance of existing traffic between a server and its clients. While our scheme is not perfect, the results are promising. In Section 6, we evaluate the effectiveness of sharing network performance information across clients that are close to each other topologically. We investigate the impact of spatial and temporal issues on the accuracy of performance prediction. Finally, we present our conclusions in Section 7.

2 Related Work

There have been numerous studies of Internet performance. We can broadly classify these studies as either *active* or *passive*. Active studies involve measuring Internet performance by injecting traffic (in the form of pings, traceroutes, TCP connections, etc.) into the network. In contrast, passive studies analyze existing traffic obtained from server logs, packet sniffers, etc. Our study is a passive one.

Several studies (e.g., [15], [16]) have examined the temporal stability of Internet performance metrics through active measurements. [15] reports that observing (no) packet loss along a path is a good predictor that we will continue to observe (no) packet loss along the path. However, the magnitude of the packet loss rate is a lot less predictable. [16] examines the stationarity of packet loss rate and available bandwidth. They find that the correlation in the loss process mainly comes only from back-to-back loss episodes, and not from “nearby” losses. Throughput has a close coupling with the loss process, and can often be modeled as a stationary IID process for periods of hours.

Several studies have also examined similar issues by studying traces gathered using a packet sniffer. The authors in [2] used traces from the 1996 Olympic Games Web site to analyze the spatial and temporal stability of TCP throughput. Using traceroute data, they constructed a tree rooted at the server and extending out to the client hosts. Clients were clustered together based on how far

apart they are in the tree. The authors report that clients within 2-4 tree-hops of each other tend to have similar probability distributions of TCP throughput. They also report that throughput to a client host tends to remain stable (i.e., within a factor of 2) for time scales of many tens of minutes.

Packet-level traces have also been used to characterize other aspects of network traffic. [1] uses traces gathered at the NASA Glenn Research Center Web server to study issues such as TCP and HTTP option usage, RTT and packet size distributions, etc. [13] uses packet-level traces to study the effectiveness of delta compression for HTTP.

Our study is similar to [2] in that it is based on packet sniffer traces gathered passively at a busy server. However, our analysis is different in many ways. We focus on packet loss rate and RTT rather than TCP throughput for the reasons mentioned previously. Our analysis of spatial locality considers operationally meaningful entities such as autonomous systems (ASes) and BGP prefix clusters [11] rather than treating the network as an undifferentiated tree. We try to infer the characteristics of internal links in the network rather than just the end-to-end characteristics.

This last aspect of our work has some similarities to previous work on inferring the loss rate of network links using end-to-end measurements. MINC [5] bases its inference on losses experienced by multicast probe packets while [7] does so using closely-spaced unicast probe packets striped across multiple destinations. A common feature of both of these techniques is that they are based on detailed observations of loss events for packets *actively* injected into the network. In contrast, we try to infer link loss rates based on *passively* observing the end-to-end loss rate for existing traffic between a server and its clients.

3 Experimental Setup and Methodology

We now describe the experimental setup and methodology used in our study. The packet traces were gathered at the *microsoft.com* site. We used the *tcpdump* tool [10] on a Pentium-III 550 MHz PC running the Windows 2000 Server OS to do the packet capture. This machine was connected to the spanning (replication) port of a Cisco Cat-

Date	Duration	# packets	# clients
20 Dec 2000	2.12 hours	100.0 million	134,475
24 Jan 2001	1.23 hours	20.38 million	53,811

Table 1: Summary of the two traces analyzed in this paper.

alyst 6509 switch via a 100 Mbps Ethernet link. With port replication turned on, our packet sniffer saw traffic to/from several server nodes that were connected either to the same Catalyst switch or to a sister switch in a two-switch cluster.

For our study, we only captured (the headers of) TCP packets since we are able to estimate packet loss rate and RTT by observing TCP data packets and the corresponding ACKs. With the setup described above, we were able to capture a portion of the traffic entering and leaving the *microsoft.com* site. This included Web traffic, software download traffic, and streaming media traffic (including streaming media over TCP to traverse firewalls). Due to cluster-level load balancing, our packet sniffer did not necessarily see all connections to/from a particular client. For a particular connection, however, it either saw *all* of the packets or none at all. So we are able to derive meaningful estimates of packet loss rate and RTT from the subset of connections that were captured. Table 1 summarizes the two traces we analyze in this paper.

We used the *traceroute* [9] tool to determine the network path from the *microsoft.com* site to each of the clients seen in the traces. The traceroute data was collected in December 2000 and January 2001. Due to security and administrative concerns, the packet sniffer machine located in the data center was configured to be in “listen-only” mode. So the traceroutes were run from a FreeBSD PC located on a separate Microsoft network. While the first few hops within Microsoft’s network were different, the entire path outside of Microsoft’s network was identical to the path that packets from the server nodes located in the data center would have taken. So these traceroutes help us determine the wide-area Internet path from the server cluster to the clients.

In addition, we obtained BGP tables from a BB-NPlanet (Genuity) router [4] on Jan. 24, 2001. We used the routing table to perform BGP prefix clustering for our data analysis.

3.1 Estimating Network Metrics

Since our packet sniffer is located very close to the server nodes, we estimate the RTT using an algorithm akin to that a TCP sender would use. We record the time that a particular data segment was sent. We then wait for an ACK that acknowledges that segment (as possibly more). The time of receipt of the ACK minus the time the data segment was sent yields a sample data point for the RTT. We then start the process over for the next data segment.

One potential problem is that our estimates of RTT could be inflated because the receiver employs the delayed-ACK algorithm or because ACKs are dropped in the network. Therefore, in our analysis we consider both the mean RTT and the minimum RTT for each client. The latter presumably does not suffer from the effects of delayed ACKs or lost ACKs. We filter out the RTT samples greater than 1 second, since we believe these are likely to be erroneous.

We detect packet losses by looking for packet retransmissions by the sender. The underlying assumption is that (a) the TCP sender only retransmits a packet if the original transmission was lost, and (b) no packets are lost on the network path between the server node and the packet sniffer. The former assumption is reasonable since TCP is conservative about retransmissions. The latter assumption is likely true because the local network is over-engineered so that it is rarely, if ever, a point of congestion. We compute the loss rate for client node as the ratio of the number of retransmitted packets to the total number of packets sent to it within a window of time. (As explained in Section 6.1, we varied the size of this window in our experimental analysis.)

4 Correlation between Internet Distance Metrics and Network Performance

In this section, we study the impact of Internet “distance” between a server and a client on the network performance perceived by the client. We consider several different notions of Internet distance between the server and client: (i) router hop count, (ii) AS hop count, and (iii) address prefix

(AP) hop count. Our goal is to understand if such static topological metrics correlate well with network performance. For instance, can we use such distance metrics to select the best replica for a particular client?

As explained in Section 3, we determined the network path from the server to each client using traceroute. For each client, we determined the number of router hops. We determined the AS number corresponding to each router by querying the *Whois* database and thereby computed the AS hop count for the path. (We ignored paths for which we were unable to determine the AS number for even one router. 31.6% of paths were ignored as a result of this.) Likewise, we used BGP prefix information in the table to determine the address prefix corresponding to each router and determined the corresponding AP count. Since BGP prefixes tend to be more fine-grained than ASes, the AP hop count for a path was generally larger than the AS hop count but smaller than the router hop count.

We analyze the impact of Internet distance on the (minimum) RTT and the loss rate.

4.1 Minimum Round-trip Time

Figure 1, Figure 2, and Figure 3 show the minimum RTT observed for each client (in the December 2000 trace) versus the corresponding router hop count, AS hop count, and AP hop count, respectively. As we can see, there is only a weak correlation between RTT and the hop count. The correlation coefficients between RTT and router hop count, AS hop count, and AP hop count are 0.035, 0.021, and 0.042, respectively. The absence of significant correlation suggests that the hop count metric cannot be used to obtain a reliable estimation of RTT. Our conclusion is in agreement with results of a recent study [3] done in a very different context.

4.2 Loss Rate

Figure 4, Figure 5, and Figure 6 show the average loss rate for each client versus its router hop count, AS hop count, and AP hop count, respectively. Just as in the case of RTT, there is little correlation between loss rate and hop count. The correlation coefficients between loss rate and router hop count,

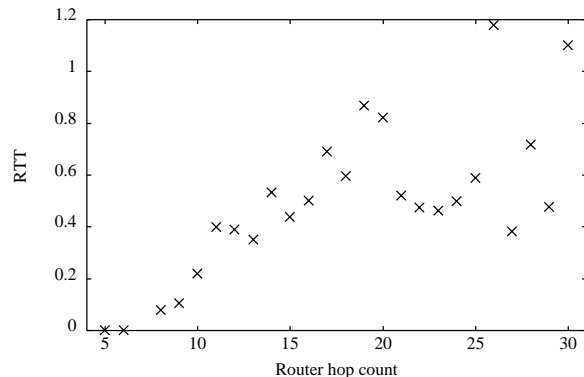


Figure 1: RTT versus router hop count.

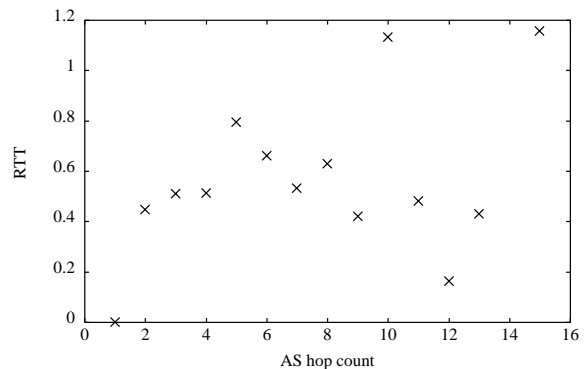


Figure 2: RTT versus AS hop count

AS hop count, and AP hop count are 0.05, 0, and 0.03, respectively. Filtering out the IP addresses that have loss rate below 10% improves the correlation coefficient somewhat. The corresponding correlation coefficients become 0.112, 0.011, and 0.104, respectively. The correlation is still pretty weak, which implies that the hop count is not a reliable indicator of the loss rate either.

4.3 Discussion

Our results indicate that the correlation between network performance and topological distance, measured in terms of hop count, is weak. The number of hops, be they router hops or AP hops or AS hops, cannot be used as a reliable indicator of network performance. This suggests that making wide-area replica selection decisions based on (static) topological metrics (as, for example, in certain configurations of the Cisco Distributed Director [6]) may not be appropriate.

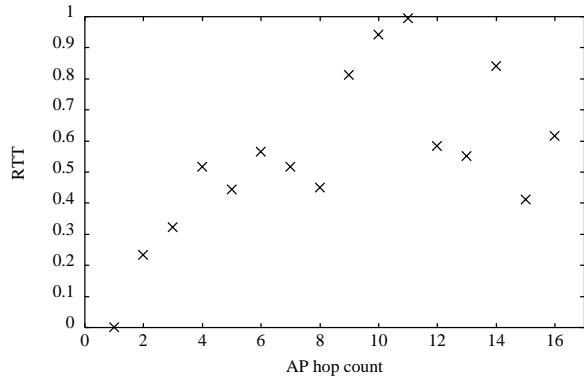


Figure 3: RTT versus AP hop count

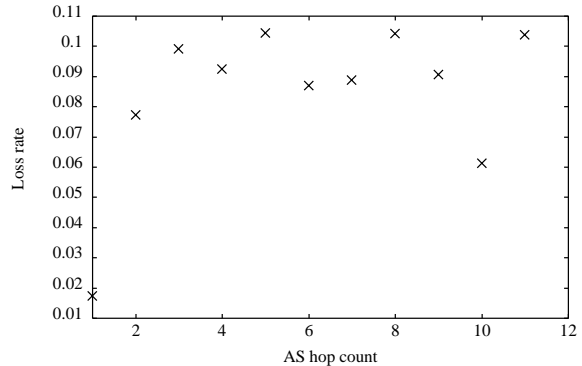


Figure 5: Loss rate versus AS hop count

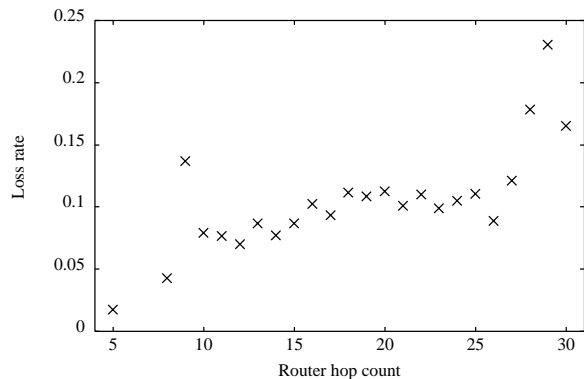


Figure 4: Loss rate versus router hop count

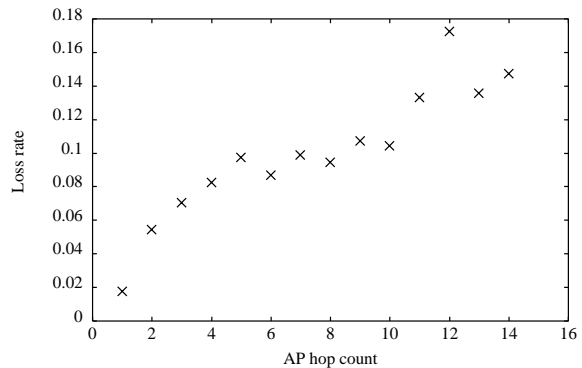


Figure 6: Loss rate versus AP hop count

In light of this finding, we consider two different (and orthogonal) approaches to estimating the quality of network connectivity between a server and a client.

The first approach is motivated by the observation that all links are not “equal” (which is in contrast to the implicit assumption made in metrics such as router hop count that all hops are the same). In other words, it is likely that poor end-to-end performance is caused by a few bad (i.e., lossy) links. If the network path from the server to a client traverses one or more of the lossy links, then it is likely that the client would see poor performance (e.g., a high packet loss rate) even if the number of hops in the path is small. Therefore it is important to identify the lossy links. In Section 5, we present a scheme for doing this identification.

The second approach we consider is motivated by the observation that clients that are topologically close to each other may see similar performance to the server (perhaps because they share a bottleneck

link). Although our results indicate that topological metrics such as server-to-client hop count are not good indicators of performance, it may still be possible to fruitfully use client-to-client topological information to share performance information across clients. Previous work (e.g., [2]) has shown the usefulness of topological clustering using simple forms of clustering. In Section 6.1, we investigate this issue by considering a more extensive set of clustering techniques.

5 Identifying Lossy Links

In this section, we attempt to identify lossy links based on observations (made at the server) of end-to-end packet loss rates to different clients. Figure 7 depicts the scenario. We have a single server, from which clients download files. In the simple case, when there are no overlapping or transient routes, the paths from the server to the clients form a tree. Our goal is to identify the lossy links

given the end-to-end loss rate between the server and each client.

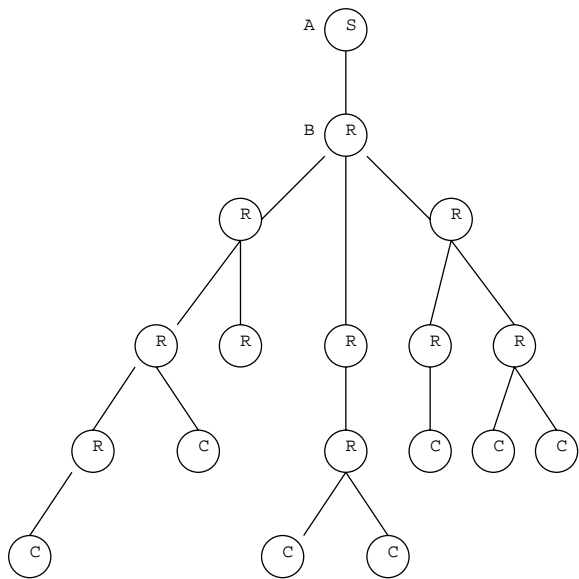


Figure 7: Network topology from the Web server point of view, where S denotes a server, C denotes a client, and R denotes a router.

Identifying lossy links is challenging for the following reasons. First, network characteristics change over time. Without knowing the temporal variation of the network link performance, it is hard to correlate different clients’ performance. Second, even when the loss rate of each link is constant, there may be no unique solution to the problem. Given M clients and N links, we have M constraints in N variables (i.e., loss rates of the individual links). For each client C , there is a constraint of the form $1 - \prod_{i \in P} (1 - l_i) = l_C$ where P is the set of links on the path from the server to the client, l_i is the loss rate of link i , and l_C is the end-to-end loss rate between the server and client C . (We can turn these into linear constraints by taking the logarithm of both sides and defining the variable corresponding to link i to be $x_i = \log(1 - l_i)$.) There is not a unique solution to this set of constraints if $M < N$, as is often the case.

As noted in Section 2, the approaches previously proposed for estimating the loss rate of network links (e.g., [5, 7]) involve actively injecting probe packets into the network. This offers the advantage of greater control — we can send multicast probes or closely-spaced striped unicast probes. In contrast, our goal here is to work with *existing* traffic,

which makes the problem even more challenging.

We now describe our initial approach to the problem. We also discuss some concerns and potential limitations of our approaches and alternative techniques that we are considering for further investigation.

To make the problem tractable, we make the simplifying assumption that the loss rate of each link is constant. Although this is not a very realistic assumption, it is a reasonable simplification according to the recent Internet measurement studies. For example, Zhang et al. found that about two-thirds of the time, a loss process stays in the same loss category for at least an hour in duration, where a loss category can be “no loss”, “minor loss”, “tolerable loss”, or “unacceptable loss” etc.

Furthermore, since there is not, in general, a unique assignment of loss rates to network links, our goal here is to identify links that are likely to have a high loss rate rather than compute a specific loss rate for each link.

Our high-level approach is to randomly sample the solution space. The way we find random solutions to the problem is as follows. Consider a tree as shown in Figure 7. We first assign a loss rate of zero to each link. The loss rate of link AB is bounded by the minimum of the loss rate among clients downstream of the link. (A client is considered to be downstream of a link if the path from the server to the client traverses the link. By this definition, all clients in Figure 7 are downstream of link AB .) We assign a random number between 0 and $\min(L)$ to be the loss rate l_{AB} of the link AB . We define the residual loss rates of a client to be the loss rate that is not accounted for by the links whose loss rates have already been assigned. We update the residual loss rate of a client C to $1 - \frac{1 - l_C}{\prod_{i \in P'} (1 - l_i)}$ where P' is the subset of links along the path from the server to the client C for which a loss rate has been assigned. Then we iterate the procedure to compute the loss rate at the next level of the tree by considering the residual loss rate of each client in place of its original loss rate. Figure 8 shows the pseudo-code for our algorithm.

There are a number of details to the algorithm. First, if there are no branches along (a section of) a path, it is impossible to estimate the loss rates of the links on (that section of) the path solely using end-to-end measurements. Therefore, we coalesce

```

done = 0;
currLevel = 0;
while (!done)
{
  done = 1;
  currLevel ++;
  foreach leaf
  {
    if (level[leaf] ≥ currLevel) {
      done = 0;
      parentLevel = level[leaf] - currLevel + 1;
      parent = leaf2path[leaf][parentLevel];
      child = leaf2path[leaf][parentLevel - 1];
      if (residualLoss[leaf] < lossBound[parent : child])
        lossBound[parent : child] = residualLoss[leaf];
    }
  }
  foreach parent : child {
    if (IsLeaf(child))
      loss = lossBound[parent : child];
    else
      loss = random(0 .. lossBound[parent : child]);
      lossRate[parent : child] = loss;
  }
  update residualLoss at all the leaves
}

```

Figure 8: Pseudo-code of computing link loss given the loss rate at the leaves.

such a path into a single “link”, as shown in Figure 9, before running the algorithm. Second, we need a large sample of packets to get an accurate estimate of the loss rate at clients. So we filter out the leaves (clients) to which sender sends fewer than a threshold number of packets. We set the threshold to be 1000 packets in our analysis.

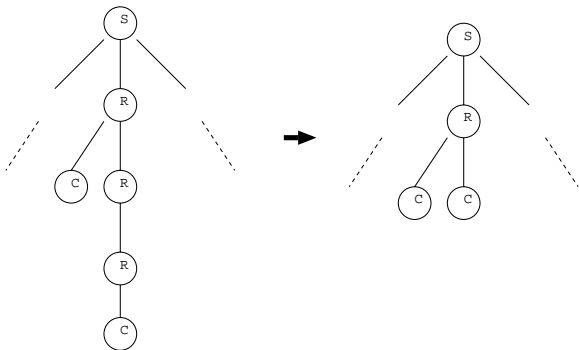


Figure 9: Coalescing a linear path with no branches to a single edge.

We ran 500 independent iterations of the randomized algorithm, each time constructing a (likely different) feasible solution to the link loss rate estimation problem. By examining these solutions, we identify links whose average loss rate over all iterations is above 10% and terms them as “lossy

links”¹.

For each lossy link, we compute the RTT of the lossy link by subtracting the RTT reported by *traceroute* for the near end of the link from that for the far end. While this is not a very accurate calculation, it suffices for our purpose since we are only trying to broadly classify links as having a large RTT or not. Figure 10 shows the cumulative distribution of the round-trip time of these links. As we can see, a significant fraction of the links have large RTT (e.g. around 30% of the links have RTT larger than 500 msec).

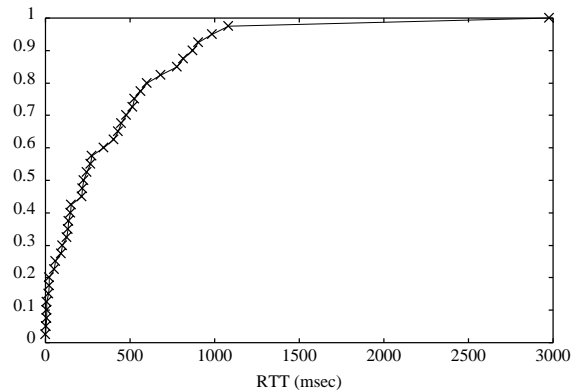


Figure 10: Cumulative distribution of the identified lossy links’ round-trip time.

We also determine the AS corresponding to either end of the link by querying the *Whois* database [8]. If and only if the two ends are in different ASes, we classify the link as crossing an inter-AS boundary (e.g., the boundary between two ISPs).

Table 2 summarizes the characteristics of the identified lossy links using the Web traffic traces collected on Dec. 20, 2000 and traceroute data gathered soon thereafter. As we can see, in 45% of the cases, the identified lossy link crosses inter-AS boundaries and has a high latency. Some examples of such boundaries include the connection from AT&T to IndoInternet in Indonesia, from Teleglobe in Canada to Telemar in Brazil. 10% of the paths have low latency but cross ISP boundaries, e.g. from Sprint to Trivalent. 20% of the links do not cross ISP boundary but have high latency. These are usually international links within an ISP

¹Note that because of the coalescing procedure depicted in Figure 9, a lossy “link” may actually correspond to a sequence of network links.

	RTT \leq 100 ms	RTT $>$ 100 ms
AS crossing	4	18
No AS crossing	8	8
Unknown	0	2

Table 2: Characteristics of the identified lossy links.

(e.g., ChinaNet). Only 20% of the lossy links neither cross ISP boundaries nor have a high latency. (This may be due to a sampling bias in our algorithm, which we discuss in Section 5.1.) Thus in the vast majority of cases, the identities of the lossy links are consistent with our intuition.

5.1 Discussion

Our algorithm for determining the identities of lossy links is akin to the Monte Carlo method in that we repeatedly sample the solution space and draw conclusions based on the statistics of the sampled solutions. It is important that the sampling procedure be unbiased, i.e., the solution space should be sampled uniformly. In this respect, our algorithm suffers from a limitation because the sampling procedure is not unbiased. Since we start assigning loss rates to links starting at the root, there is a bias towards apportioning more of the losses to links close to the root.

Our experimental results suggest that the impact of the bias is not very great. Indeed, the vast majority (80%) of the lossy links discovered are links with a large delay and/or links that cross ISP boundaries, both of which are likely candidates to be lossy. It is possible that some fraction of the remaining 20% of lossy links were victims of sampling bias although it is hard for us to know for sure.

A richer data set would tend to alleviate the impact of the bias. A topology tree that includes a larger number of paths and would tend to place a tighter bound on the loss rate of links, especially ones close to the root of the tree. This is because the bound on the loss rate of a link is determined by the “best” downstream client, i.e., the one with the smallest loss rate.

We could modify the sampling procedure to randomize the order in which links are picked rather than always start at the root and march towards the leaves. Each time we pick a link at random,

we assign it a loss rate value picked uniformly between the bounds for the link. We then adjust the residual loss rate for the affected clients, recompute the loss rate bound for the affected links, and repeat the process. While we are unable to prove mathematically that this sampling procedure would be unbiased, it does appear to be less biased than the procedure we used previously. We are presently experimenting with the modified algorithm. We are also planning to do simulations on large-scale topologies to validate (or invalidate!) our approach.

6 Sharing Performance Information Across Clients

As noted in Section 4.3, it is interesting to consider whether clients that are topologically close to each other can share network performance information to make informed predictions of future performance. The prediction for a client would be made based on past network performance information either for the same client or other “nearby” clients. This leads us to examine two separate questions:

1. **Spatial locality:** How effectively can we predict the future performance for a client using past performance information for “nearby” clients?
2. **Temporal locality:** How does the age of past network performance information impact our ability to make accurate predictions?

The larger the set of nearby clients that can be used to make predictions, the greater the chance that we will have past performance information for one or more of those clients. Likewise, the larger the time window of past information that is still useful for making predictions, the greater the chance we will have useful performance information available for making predictions.

In the remainder of this section, we analyze spatial and temporal locality of network performance.

6.1 Spatial Locality of Network Performance

In this section, we study the spatial locality of network performance, i.e., whether clients that are

topologically close experience similar network performance. We examine whether it is possible to predict the network performance for a client using spatial clustering, i.e., by considering the performance experienced by nearby clients. We use a moving window prediction scheme. In the first window, we compute the performance of a client cluster (i.e., round-trip time or loss rate) by averaging over all clients in the cluster. We then predict the performance of a given client in the next window using the estimates for the corresponding client cluster. We repeat this procedure for subsequent windows of time.

We evaluate the accuracy of the predictions using the *prediction error*, which is defined as

$$predictionError = \frac{\max(actual, prediction)}{\min(actual, prediction)}$$

When computing the prediction error for the loss rate, we filter out all the samples whose $\min(actual, prediction)$ is 0.

In our analysis, we consider the following clustering schemes: (i) clients clustered by IP address (i.e., one IP address per cluster), (ii) clients clustered by subnet address assuming a 24-bit network prefix, (iii) clients clustered by the address prefix (AP) in BGP routing tables [11], (iv) clients clustered by autonomous system (AS) number, (v) clients clustered randomly, and (vi) all clients placed in a single cluster (abbreviated as 1-cluster).

6.1.1 Round-trip Time

We first consider the problem of predicting the minimum RTT for each client. The advantage of considering the minimum RTT is that it minimizes the impact of delays due to queuing and the delayed-ACK mechanism. Figure 11 and Figure 12 show the cumulative distribution of prediction error for the December 2000 and January 2001 traces, where the prediction error for a given IP address is computed as the average error over all the windows in which there is performance data corresponding to the address. The graphs show the results using window sizes of 1 minute and 10 minutes.

We make the following observations. First, as we would expect, random clustering and 1-cluster yield significantly larger prediction error compared to the other clustering techniques. For example, in the January trace, only in 30% of the cases do

the random clustering and 1-cluster schemes have a prediction error within a factor of 1.5, whereas the corresponding fraction for the other clustering schemes is 55% or larger. The performance gap between the random clustering and 1-cluster schemes versus the other clusterings is smaller in the December trace, but is still significant.

Second, we note that the prediction error tends to decrease as the clustering becomes more fine-grained. The list of clustering schemes sorted from best to worst in terms of prediction error is: IP address based clustering, subnet-based clustering, AP clustering, and AS clustering. Clustering by IP address generally yields the lowest prediction error. For example, its prediction error is below a factor of 1.5 in 60-80% of cases, and below a factor of 2 in 75-85% of cases. On the flip side, though, as the clustering becomes more fine-grained, it is less likely that we will have past performance information associated with the cluster that can be used to make predictions. We will come back to this issue in Section 6.2.

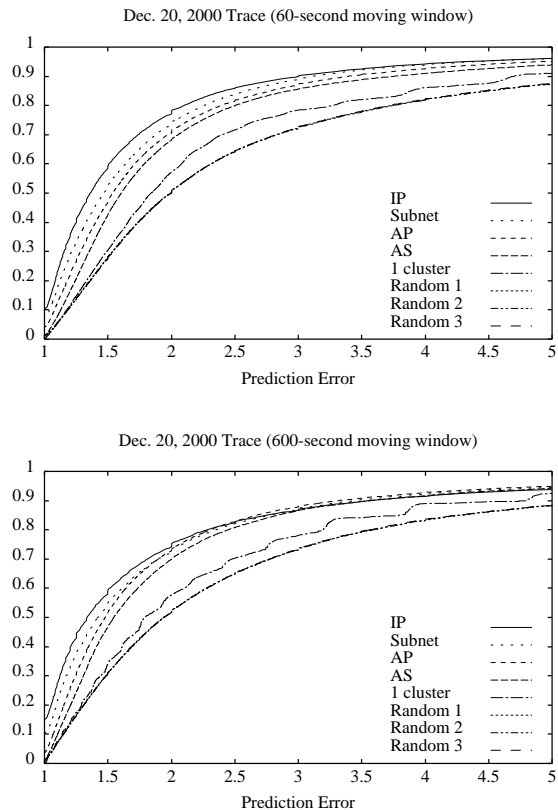


Figure 11: Prediction error of the minimum RTT.

Now we turn to predicting the average RTT for a client. We evaluate the same set of clustering methods as before. Figure 13 and Figure 14 summarizes the results for the December and January traces, respectively. As in the minimum RTT case, random clustering and 1-cluster perform worse than the others. However, IP address based clustering no longer performs the best. Its CDF curve starts off as the best among all, but then it converges to 1 very slowly, even becoming worse than 1-cluster near the tail. Our conjecture for its poor performance is that in order to accurately predict the average RTT we need a large number of samples, which may not be available when clustering on a very fine granularity, such as the client IP address. The subnet-based clustering has the lowest error in predicting the average RTT, with AP clustering and then AS clustering being the next best.

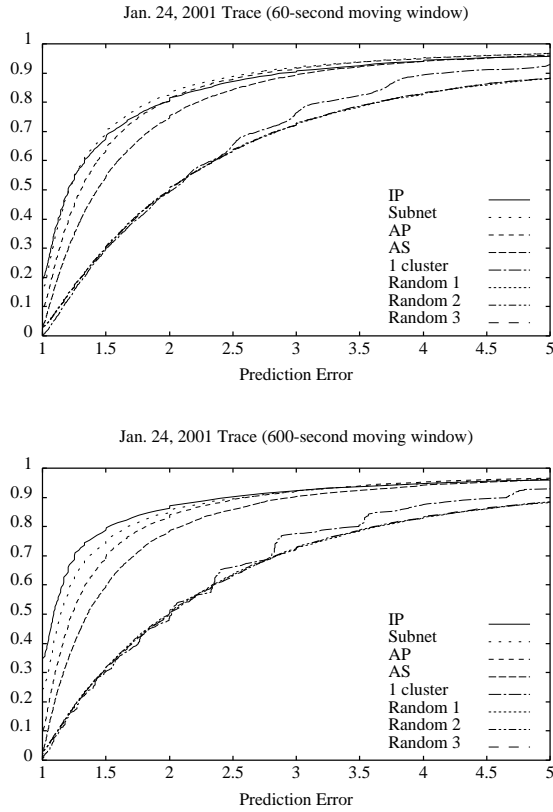


Figure 12: Prediction error of the minimum RTT.

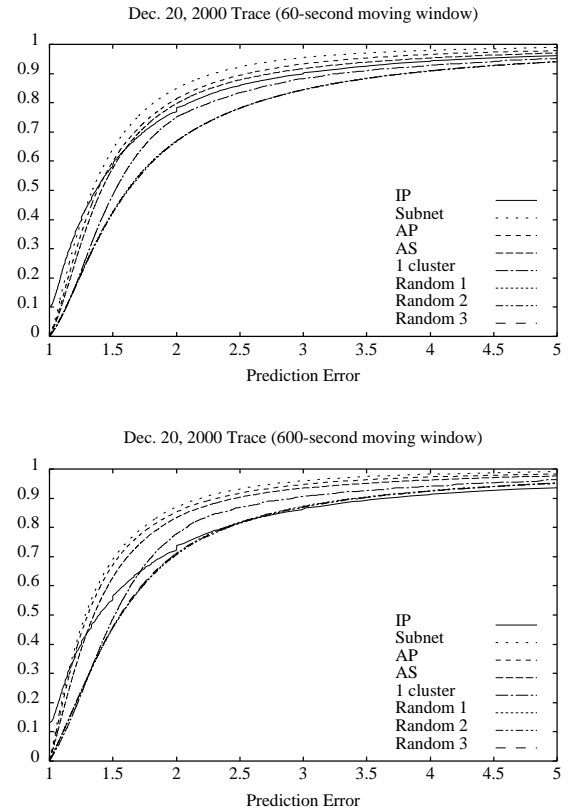


Figure 13: Prediction error of the average RTT.

6.1.2 Loss Rate

We now turn to evaluating the prediction accuracy for the loss rate. Figure 15 shows the results for

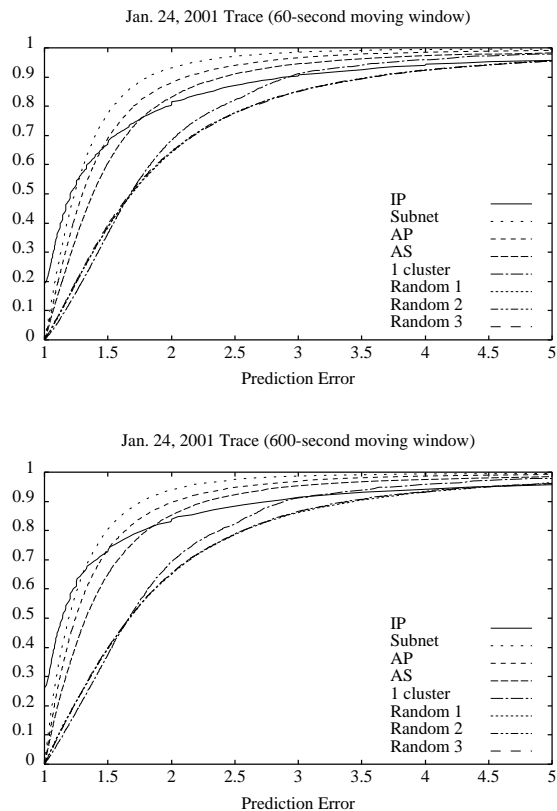


Figure 14: Prediction error of the average RTT.

the December trace. (The results for the January traces are similar.) We observe that the prediction error decreases with the window size. This is especially true for the IP address based clustering. For example, when using IP address based clustering with a 1-second moving window, there are often not enough samples to yield an accurate estimate of the loss rate, so the prediction error is significantly worse than for the other clustering schemes. However, once the window size is large enough that there are many samples, IP address based clustering yields more accurate estimates than the other schemes. As in the case of RTT, the prediction error tends to increase as the clustering becomes more coarse-grained.

6.1.3 Discussion

In summary, our results suggest that the more fine-grained the clustering, the better the prediction accuracy is (unless the fine-grained clusters have insufficient network performance samples). However, as we make the clustering more fine-grained, the lower the hit rate is. This suggests that when predicting performance, we should use as fine-grained cluster as possible, and go to the coarse-grained cluster only when there are an insufficient number of samples in the fine-grained cluster.

So far, we have only considered the absolute performance. In many applications, such as replica selection, it is the *relative* performance that matters. (The question is whether replica R_1 better than R_2 , not necessarily what the actual network performance to each replica is.) Although fine-grained clustering is preferable for predicting absolute performance, it may well be the case that coarse-grained clustering is sufficient for predicting relative performance. For example, consider two clients on a university campus, one on a LAN and the other behind a DSL link. (Note that the two clients are likely to share the same AP and AS although they are probably on different subnets.) Sharing performance estimates across the two clients may not be meaningful, so fine-grained clustering (such as at the IP address or subnet level) is desirable for predicting absolute performance. However, if the bottleneck link is close to the server/replicas, the preferred replica may be the same for the two clients. So coarse-grained clustering may be meaningful for predicting rela-

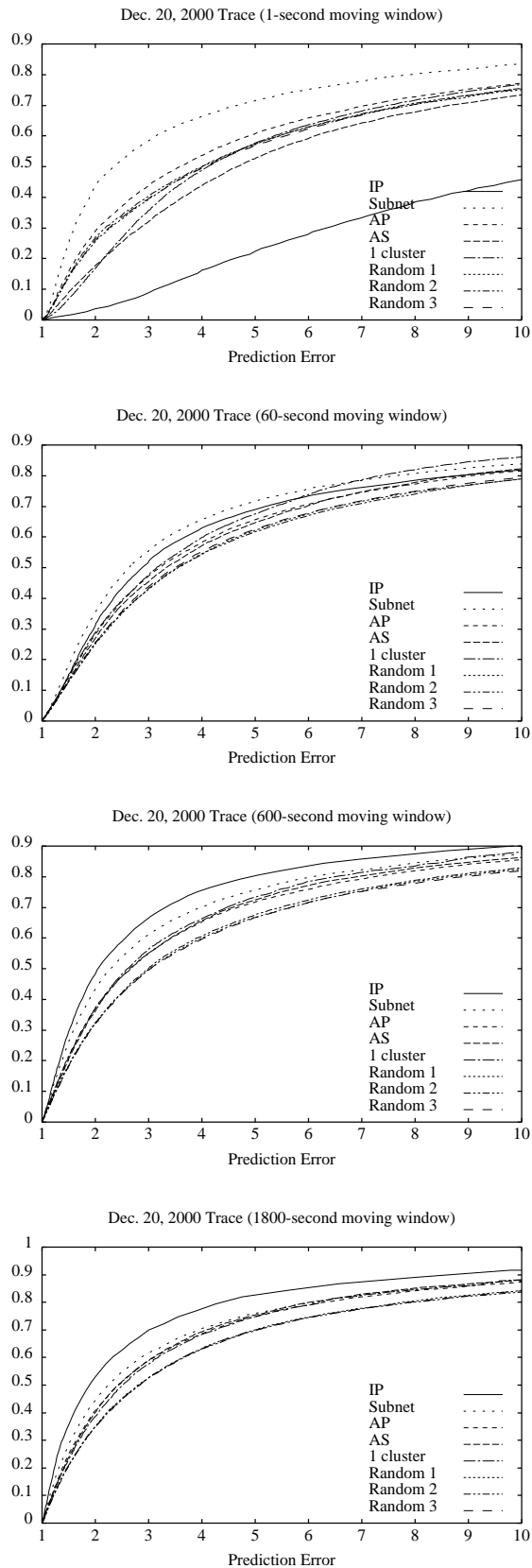


Figure 15: Prediction error of the loss rate.

tive performance.

6.2 Temporal Locality of Network Performance

In this section, we study the temporal locality of network performance. We try to understand the following issues: (i) how the prediction error varies with different window sizes, (ii) how much performance history should be used for prediction, and (iii) how the age of the history information affects the prediction. In this section, we use BGP prefix clusters for our analysis unless otherwise specified.

6.2.1 Amount of history to be used for prediction

When predicting the performance, how much performance history should be used? Using too little performance history may be vulnerable to transient variation in performance and also has small prediction hit rate (where hit rate refers to the probability that we have the history information to be able to estimate performance for a client). On the other hand, using too much performance history may cause errors because of stale information. In our analysis here, we vary the size of the history window (i.e., the window during which performance data is gathered for making predictions), while fixing the size and offset of the target window (i.e., the window for which we seek to predict performance). The results presented here correspond to a target window size of 20 minutes. We vary the history window size from 20 to 100 minutes. Since our traces are at most two hour long, we are unable to investigate the impact of using a larger amount of history to make prediction.

Figure 16 and Figure 17 shows the cumulative distribution of the prediction error for RTT using different history window sizes. As we can see, the prediction error hardly changes with the window size. This is not surprising since the RTT tends to remain relatively stable, with significant (and persistent) changes happening only when there is a relatively rare event, such as route change. So there is little improvement in prediction accuracy when we consider more than say 20 minutes of history information. (There is, however, an improvement in the hit rate, as we discuss below.) Our results for the loss rate are similar (i.e., 20 minutes of history

information appears to yield a stable estimation of loss rate, and more history does not improve accuracy).

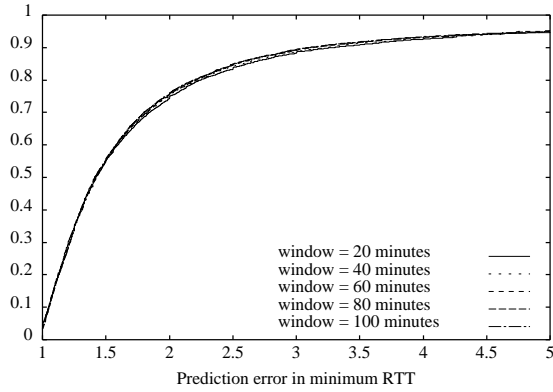


Figure 16: Error in predicting the minimum RTT using different window sizes.

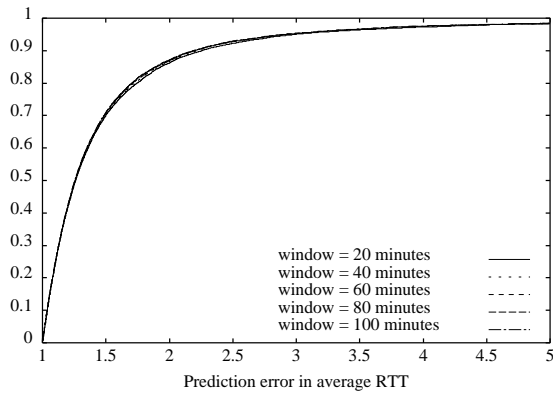


Figure 17: Error in predicting the average RTT using different window sizes.

Figure 18 plots the hit rate versus the history window size for different clustering schemes. As we can see, the hit rate remains stable for AS based clustering, but increases with window size for the other clustering schemes. It is interesting to note that subnet-based clustering sees the largest increase in hit rate. A possible explanation is that spatial locality in client access pattern increases as we consider a larger history window, i.e., the larger the history window, the more likely it is that other clients from a given subnet will connect to the server. On the other hand, an individual client usually stays at a Web site for some time and then leaves. So growing the history window

beyond say 20 minutes does not improve the hit rate by very much. At the other extreme, AS and AP clusterings already have a high hit rate when the window size is small, say 20 minutes. So the incremental benefit of growing the window is small.

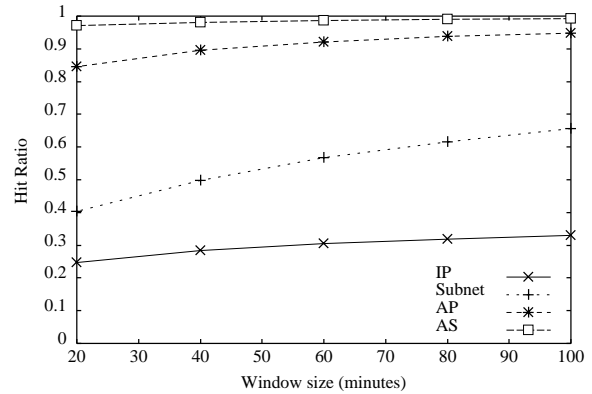


Figure 18: Hit rate for different window sizes.

6.2.2 Age of history information

In this section, we examine how the prediction accuracy varies with the freshness (i.e., age) of the performance data. Our approach is as follows. We fix the size and offset of the target window for which predictions are to be made. We fix the size of the history window but vary its offset (i.e., we vary the age of the data that we consider). Again, since our traces are only two hours long at most, we are unable to explore the impact of history information older than 2 hours.

Figure 19 and Figure 20 shows the error in predicting the minimum and average RTT using different time gaps. It is clear from the graphs that the prediction error for RTT changes very little when considering different freshness of data, presumably for reasons similar to those mentioned in Section 6.2.1.

Figure 21 shows the cumulative distribution of the prediction error for loss rate. Compared to RTT, we see that the age of history information has a greater impact. The prediction error tends to increase as the history information used for prediction becomes older. For instance, at the 70th percentile, the prediction error is a factor of 4 when history information from the immediate past is used, whereas it is a factor of 5 when we use

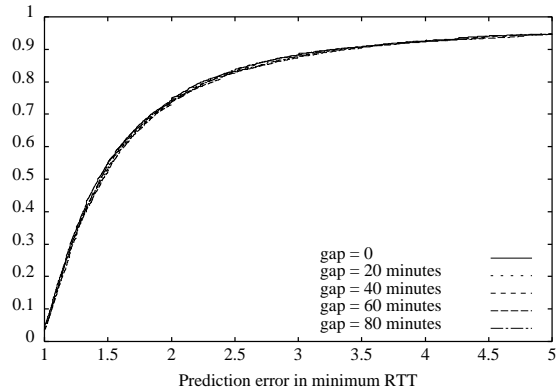


Figure 19: Error in predicting the minimum RTT using different time gaps.

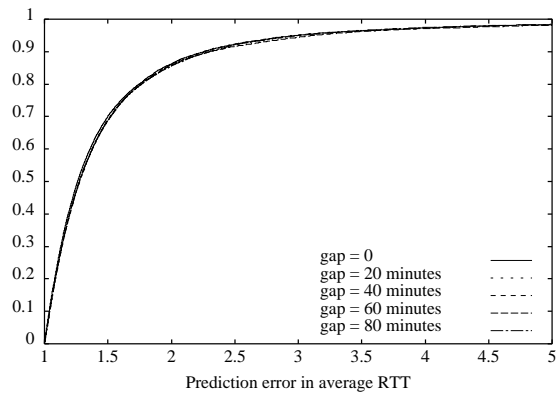


Figure 20: Error in predicting the average RTT using different time gaps.

history information that is 80 minutes old.

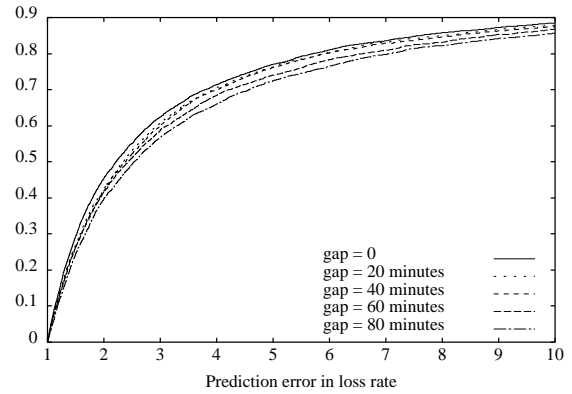


Figure 21: Error in predicting the loss rate using different time gaps.

Figure 22 plots the hit rate for different values of age. The hit rate for AS and AP clustering remains the same. In contrast, the hit rate for IP and subnet based clustering decreases by around 10%. Most of the decrease in the hit rate happens initially, and beyond that the hit rate stabilizes. The reason for this is that the cluster size is small in these cases (in fact it is just 1 in the case of IP address based clustering). So the probability of finding past performance data for a cluster diminishes as we go further back in time (for instance, a client machine may have been turned on only 10 minutes ago, so if we use IP address based clustering there would likely be no performance history for the cluster beyond 10 minutes in the past).

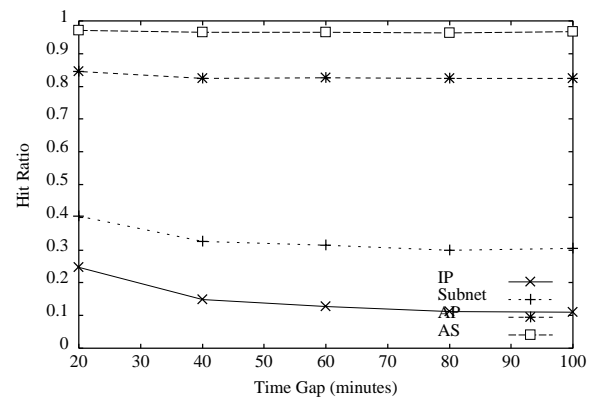


Figure 22: Hit rate for different time gaps.

6.2.3 Discussion

To summarize, we find that the size of the history window (i.e., the amount of history considered) has little effect on the prediction accuracy when varying from tens of minutes to over an hour. Similarly, the age of history information has little effect on the prediction accuracy for RTT, and only a slight effect on that for loss rate.

7 Summary and Conclusions

In this paper, we present a study of wide-area Internet performance as observed from a busy Internet server site. We explore the question of predicting the network performance that a client would observe based on topological metrics and past performance data.

We found the correlation between topological metrics such as router hop count and network performance is weak. Rather than all links being “equal”, it is more likely that a few bad (lossy) links cause poor end-to-end performance. Identifying such links would be very useful for performance prediction.

However, inferring link loss rates from end-to-end measurements is a very challenging problem. To the best of our knowledge, previous work on this problem has involved active probing. Inferring link loss rates based on *passively* observing the end-to-end loss rate for existing traffic is a new research issue, and one that is relevant in many different settings (client-server communication, peer-to-peer communication, etc.) We have proposed an initial approach to the problem. Our approach is based on repeatedly sampling the solution space and drawing conclusions based on the statistics of the sampled solutions. Our experimental results indicate that 80% of the lossy links identified by this approach either cross an inter-AS boundary or are long-haul international links, which is consistent with our intuition of where packet loss is likely to happen. Our sampling procedure has a limitation in that it is not unbiased. We are looking at ways to reduce or remove the sampling bias, and plan to evaluate them through large scale simulations as well as by gathering and analyzing more trace data. We are also interested in studying longer traces to determine the accuracy of the loss rate

trends we have found.

We also studied how well we can predict network performance for a client by using past performance for “nearby” clients. We find that it is very hard to predict performance accurately, especially only through passive measurements. In 80% of the cases, we can predict RTT within a factor of 2, and in 60-80% of cases, we can predict loss rate within a factor of 4.

The prediction accuracy depends on both the spatial scale and the temporal scale of performance data used for making predictions. The more fine-grained the clustering of clients, the better the prediction accuracy is. However there is a trade-off between the better prediction accuracy and the hit rate.

8 Acknowledgement

We would like to thank Scott Hogan, Rob Emanuel, Chris Darling, and Al Lee for their help in setting up the packet tracing machine, and thank Dimitris Achlioptas for useful discussions on the loss rate estimation problem.

References

- [1] M. Allman. A Web Server’s View of the Transport Layer. *ACM Computer Communication Review*, October 2000
- [2] H. Balakrishnan, S. Seshan, M. Stemm, and R.H. Katz. Analyzing the Stability in Wide-Area Network Performance. *ACM SIGMETRICS*, June 1997
- [3] G. Ballintijn, M. van Steen, and A.S. Tanenbaum. Characterizing Internet Performance to Support Wide-area Application Development. *Operating Systems Review*, 34(4):41-47, October 2000.
- [4] telnet://ner-routes.bbnplanet.net.
- [5] R. Caceres, N.G. Duffield, J. Horowitz, and D. Towsley. Multicast-based Inference of Network Internal Loss Characteristics. *IEEE Transactions on Information Theory*, November 1999
- [6] K. Delgado. Cisco Distributed Director. Cisco White Paper, 1997. Available at <http://www.cisco.com>.
- [7] N.G. Duffield, F.L. Presti, V. Paxson, and D. Towsley. Inferring Link Loss Using Striped Unicast Probes. *IEEE Infocom*, April 2001
- [8] K. Harrenstien, M. Stahl, E. Feinler, NICKNAME/WHOIS, *RFC-954, IETF*, October 1985.

- [9] V. Jacobson, Traceroute software, 1989, <ftp://ftp.ee.lbl.gov/traceroute.tar.gz>
- [10] V. Jacobson, C. Leres, and S. McCanne. tcpdump - dump traffic on a network.
- [11] B. Krishnamurthy and J. Wang. On Network Aware Clustering of Web Clients. *ACM SIGCOMM*, August 2000
- [12] <http://www.microsoft.com>.
- [13] J.C. Mogul, F. Douglis, A. Feldman, and B. Krishnamurthy. Potential Benefits of Delta-encoding and Data Compression for HTTP. *ACM SIGCOMM*, 1997
- [14] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling TCP Throughput: A Simple Model and its Empirical Validation. *ACM SIGCOMM*, August 1998
- [15] V. Paxson. Measurements and Analysis of End-to-End Internet Dynamics. *ACM SIGCOMM*, September 1997
- [16] Y. Zhang, V. Paxson, and S. Shenker. On the Constancy of Internet Path Properties. *ACM SIGCOMM Internet Measurement Workshop*, November 2001.