SPAND: Shared Passive Network Performance Discovery

Srinivasan Seshan+
srini@watson.ibm.com

Mark Stemm, Randy H. Katz*
{stemm,randy}@cs.berkeley.edu

+IBM T.J. Watson Research Center
Yorktown Heights, NY 10598

**Computer Science Division*
University of California at Berkeley
Berkeley, CA 94720

Abstract

In the Internet today, users and applications must often make decisions based on the performance they expect to observe. For example, many Web pages are available in low-bandwidth and high-bandwidth versions, while other pages present users with long lists of mirror sites to chose from. Current techniques to perform these decisions are often ad-hoc and/or poorly designed. The most common solution used today is to require the user to manually decide based on experience and information provided by the application. Previous efforts to automate this process have focused on *isolated, active* network probes from a host. Unfortunately, this method of making measurements has several problems. Active probing introduces unnecessary network traffic that can quickly become a significant part of the total traffic handled by busy web servers. Probing from a single host results in less accurate information and additional redundant network probes than a system that shares information with nearby hosts. In this paper, we propose a system called SPAND (Shared Passive Network Performance Discovery) that determines network characteristics by making *shared*, *passive* measurements from a collection of hosts. In this paper, we show why passive measurements from a collection of hosts has advantages over active measurements from a single host. We also show that sharing measurements can dramatically increase the accuracy and timeliness of predictions. In addition, we present a initial prototype design of SPAND, a plan for incremental deployment, and the current implementation status of our system.

1. Introduction

In today's Internet, it is currently impossible to determine in advance what the network performance (e.g. available bandwidth and packet loss probability) between a pair of internet hosts will be. This is a feature that is severely lacking in today's suite of internet services, and there are many applications that could benefit from such a service:

- Applications that are presented with a choice of hosts that replicate the same service. Specific examples of this are FTP and Web mirror sites and Harvest caches that must contact the "closest" peer cache. Today, these applications rely on statistics such as hop count/routing metrics [7], round-trip latency [6], or geographic location [8], or active network probing [4]. However, each of these techniques has significant weaknesses.
- Web servers that have a choice of content *fidelity* to present to a web client, for example, a full graphics representation for high-bandwidth links or a text-only representation for low-bandwidth links. Today, the user must manually select the fidelity of the content that they wish to view.
- Applications that provide feedback to the user that indicates the expected performance to a distant site. For example, Web browsers could insert an informative icon next to a hyperlink indicating the expected available bandwidth to the remote site referred to by the hyperlink.

These applications require the creation of a system that predicts the expected network performance between a pair of internet hosts. Previous work in this area has relied on *isolated, active probing* from a single host. There are two major problems with this approach:

- Active probing requires the introduction of unnecessary traffic into the network. We show later that this unnecessary traffic can quickly grow to become a non-negligible part of the traffic reaching a busy Web server.
- Probing from a single host prevents a client from using the past information of nearby clients to predict future performance. Recent studies [2][15] have shown that performance from a client to a server is stable for many minutes and is identical to the performance observed by other nearby clients. In this paper, we show examples where using shared rather than isolated information increases the likelihood that previously collected network characteristics are valid.

We are developing a system called SPAND (Shared Passive Network Performance Discovery) that avoids the problems of probing by collecting network performance information *passively* from a collection of hosts, *caching* it for some time and *sharing* this information between them. This allows a group of hosts to obtain timely and accurate network performance information in a manner that does not introduce unnecessary network traffic.

The rest of this paper is organized as follows. In Section 2, we describe related work in more detail. In Section 3, we point out the advantages and challenges of using passive shared measurements over isolated active measurements. In Section 4, we present a detailed design of SPAND, including mechanisms for incremental deployment. In Section 5, we conclude and discuss the current status of the system and future work.

2. Related Work

In this section, we describe in more detail previous work in network probing algorithms and server selection systems.

2.1 Probing Algorithms

A common technique to estimate expected performance is to quickly probe the network with packets. The objective of these probes is to measure the round trip latency, peak bandwidth or available "fair-share" bandwidth.

Probes to measure round-trip latency and peak bandwidth are typically done by sending groups of back-to-back packets to a server which echoes them back to the sender. These probes are referred to as NetDyn probes in [3], packet pair in [11], and bprobes in [5]. As pointed out in earlier work on TCP dynamics [10], the spacing between these packets at the bottleneck link is preserved on higher-bandwidth links and can be measured at the sender.

If the routers in the network do not implement fair queuing, the minimum of many such measurements is likely to be close to the raw link bandwidth, as assumed in [3][5][14]. Pathchar [14] combines this technique with traceroute [16] to measure the link bandwidth and latency of each hop along the path from one Internet host to another.

If routers in the network implement fair queuing, then the bandwidth indicated by the back-to-back packet probes is an accurate estimate of the "fair share" of the bottleneck link's bandwidth [11]. Cprobes [5] send a short sequence of echo packets from one host to another as a simulated connection (without any flow or congestion control). By assuming that "almost-fair" queuing occurs over the short sequence of packets, cprobe provides an estimate for the available bandwidth along the path from one host to another. Combined with information from bprobes, cprobes can estimate the competing traffic along the bottleneck link.

The problem with these tools is that they introduce significant amounts of traffic that is not "useful" to any application. For example, pathchar sends (at least) tens of kilobytes of probe traffic per hop, and cprobes send 6 kilobytes per cprobe. This amount of probe traffic is a significant fraction (approximately 20%) of the mean transfer size for many WWW connections ([1], [2]) as well as a significant fraction of the mean transfer size for many WWW sessions.

2.2 Server Selection Systems

The primary application of the of network probing algorithms described in the previous section has been in dynamic wide-area server selection. [4] uses cprobes and bprobes to select the best of a number of candidate mirror sites. Harvest [6] uses round-trip latency to select the best peer cache. Requests are initiated to each peer cache, and the first to begin responding with a positive answer is selected and the other connections are closed. [8] relies on geographic location for push-caching of WWW documents.

There are also preliminary designs for network-based services to aid in server selection. IPV6's anycast [9][13] service provides a mechanism that directs a client's packets to one of a number of hosts that represent the same IP addresses. The Host Proximity Service (HOPS) [7] uses routing metrics such as hop counts to select the closest one of a number of candidate mirror sites.

The problem with many of these approaches is that one-way latency and hop count are poor estimates of actual completion time. [4][12] show that hop count is poorly correlated with available bandwidth, and one-way latency does not take available bandwidth into account at all. In addition, current systems that provide better performance metrics [4] rely on each end host independently measuring network performance.

Table 1 summarizes the previous work in this area. The significant shortcomings of existing network performance discovery and server selection systems are:

- Introduction of new traffic into the network that can quickly become significant when compared to "useful" traffic.
- Reliance on measurements from a single host, which are more often redundant and inaccurate than measurements from a collection of hosts.
- Use of imprecise metrics such as hop count, latency, and geographic location as estimates of available bandwidth.

We discuss these shortcoming further in the next section.

3. Passive and Cooperative Measurements

The goal of our work is to provide a single unified repository of real end-to-end performance information for services to consult when they wish to determine the network performance to distant Internet hosts. Our approach addresses the shortcomings of previous work by relying solely on passive rather than active measurements of the network and sharing measurement information for all hosts in an area.

3.1 Benefits and Challenges of Cooperative Measurements

Using cooperative rather than isolated measurements allows us to eliminate redundant network probes. If two hosts are nearby each other in terms of network topology, it is likely that they share the same bottleneck link to a remote site. As a result, the available bandwidth they measure to this site is likely to be identical [2]. Therefore, it is unnecessary and undesirable for each these hosts to independently probe to find this information.

To quantify how often information can be shared between nearby hosts, we examined Internet usage patterns by analyzing client-side WWW traces. From these traces, we

System	What Measured/Used for Server Selections	Additional Traffic Introduced	Notes
Bprobes, Cprobes	Available Bandwidth	Significant	Cprobes have no flow or congestion control
Packet Pair	Available Bandwidth	Little	Assumes per-flow fair queuing
Pathchar	Hop-by-hop link bandwidth, latency	Significant	
IPV6 Anycast	Not Specified	Not Specified	
Harvest	Latency	Significant	
HOPS	Routing Metric	Little	
SPAND	Available Bandwidth, Packet Loss Probability	Little	All traffic confined to local domain

TABLE 1. Summary of Previous Work compared to SPAND

determined how often a client would need to probe the network to determine the network performance to a particular WWW server when shared information was and was not available.

More formally, for a single web server, we can represent the list of arrival times from a single client (or a shared collection of clients) as a sequence (t_1, t_2,..., t_n). If the difference between t_{i+1} and t_i is extremely small (less than ten seconds), we merge the events together into a single web browsing "session". Clearly, the first arrival always requires a probe of the network. In addition, if we assume that the time between significant network changes is a fixed value Δ, then if $t_{i+1}-t_i>\Delta$, then the client must make a probe to determine the new network characteristics. If $t_{i+1}-t_i<\Delta$, then no probe is necessary. Previous studies [2][15] have shown that a appropriate value for Δ is on the order of tens of minutes.

Figure 1 shows the results of this analysis for a particular client-side trace consisting of 404780 connections from approximately 600 users over an 80 hour time period [17]. The x axis represents the time Δ between network changes, and the y axis represents the number of network probes that are necessary. There are two curves in the figure. The upper curve represents the number of probes that are necessary if no sharing between clients is performed, and the lower curve represents the number of probes that are necessary if clients share information between them. The upper curve begins at Δ=10 seconds because of the "sessionizing" of individual connections described above. We see that the number of probes that are necessary when clients share network information is dramatically reduced. This is evidence that a collection of hosts can eliminate many redundant network probes by sharing information.

The use of cooperative measurements has challenges that must be overcome, however. Measurements from arbitrary hosts in a region cannot be combined. For example, it is necessary to separate modem users within a local domain from LAN users in a local domain, because the two sets of users may not share the same bottleneck link. The challenge is that it is often difficult to determine who the set of "similarly connected" hosts within an local domain are. We can use the topology of the local domain along with post-processing on past measurements to determine which network subnets exhibit significantly different performance. The system can then avoid aggregation of measurements from dissimilar hosts.

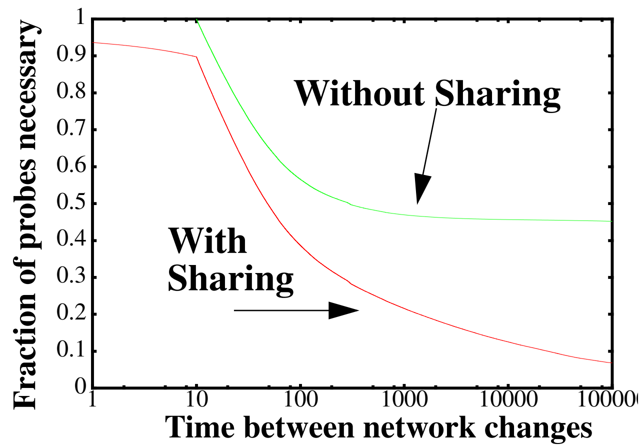

Figure 1. The benefit of sharing. Figure shows the fraction of network probes that are necessary as a function of the time between network state changes

3.2 Benefits and Challenges of Passive Measurements

The use of passive measurements makes the job of measuring available bandwidth more difficult, but avoids the introduction of useless traffic into the network. Probe traffic is a

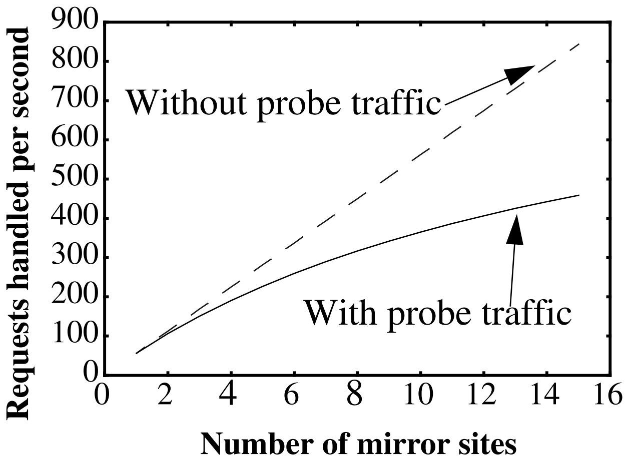

Figure 2. The effect of probe traffic on scalability. Figure shows requests/second that mirrors can serve as a function of the number of mirror sites

problem when considering the scenario of mirror sites that replicate the same content. In an active probing system, a client must first contact each of the mirror sites to determine which mirror is the "best". This slows down servers with probe-only traffic and limits the scalability of such a system.

The following example shows why. Consider a web server with a variable number of mirror sites. Assume that each mirror site is connected to the internet via a 45 Mbit/second T3 link and assume that the mean transfer size is 100 kbytes and the mean probe size is 6 kbytes. From a network perspective, an estimate of the number of requests per second that the collection of mirrors can support is the aggregate bandwidth of the mirrors' internet links divided by the sum of the average web transfer size and any associated probe traffic for the transfer. Figure 2 shows the number of requests per second that such a system can support as a function of the number of mirror sites for two systems: one without probe traffic, and one with probe traffic. We see that the system without probe traffic scales perfectly with the number of mirrors. For the system with probe traffic, however, for each web request that is handled by a single mirror, a network probe must be sent to all of the other mirrors. On the server side, this means that for each web request a particular mirror site handles, it must also handle a probe request from clients being serviced at every other mirror location. As the number of mirrors increases, the number of requests served per second becomes limited by the additional probe traffic.

However, using passive rather than active measurements is difficult for several reasons:

- Passive measurements are uncontrolled experiments, and it can be difficult to separate network events from those occurring at the endpoints, such as a rate-limited transmission, a slow or unreachable server, etc.
- Passive measurements are only collected when a host contacts a remote site. In order to have timely measurements, hosts in a local domain must visit distant hosts often enough to obtain timely information. If this is not true, the client may obtain either out of date or no information.

For our purposes, there is no need to distinguish between network events and endpoint events. If a remote site is unreachable or has poor connectivity because it is down or overloaded, that information is just as useful. In addition, we can distinguish between rate-controlled and bulk transfer transmissions by using TCP and UDP port numbers and the application classes described in Section 4.

To identify if passive measurements can provide timely information, we must analyze typical Internet usage patterns and determine how often passive techniques lead to out-of-date information. We can use the same results shown in Figure 1 to see this. We can model the arrival pattern of clients as a sequence of times ($t_1...t_n$) just as before. In the passive case, when $t_{i+1}-t_i>\Delta$, instead of saying that an active probe is necessary, we say that the passively collected information has become out of date. So the fraction of time that an active probe is necessary is exactly the same as the fraction of time that passive measurements become out of date. As mentioned earlier, the appropriate value for Δ is on the order of 10's of minutes. We see that even a relatively small collection of hosts can obtain timely network information when sharing information between them. If we assume that network conditions change approximately every 15 minutes, then the passive measurements collected from this relatively small collection of 600 hosts will be accurate approximately 78% of the time. For larger collections of hosts (such as domain-wide passive measurements), the accuracy will be even greater.

4. Design of our System

In this section, we describe the design for SPAND, including steps for incremental deployment in existing networks.

Figure 3 shows a diagram of the components of SPAND. SPAND is comprised of *clients*, *performance servers, and packet capture hosts*. Clients have modified network stacks that transmit *performance reports* to performance servers that indicate the performance of the network path between the client and distant hosts. The format of a performance report is shown in Figure 4, and includes variables such as connection bandwidth and packet loss statistics. The transport protocol field indicates the type of transport connection (UDP or TCP) used by the initiator of the connection. The optional Application Class field is a hint as to the way in which the application that is using the transport connection. If an application class is not provided, the performance server can use the port number and transport protocol fields to determine an application class. The application class field is desirable because not all applications use transport connections in the same way. Some applications (such as Web browsers and FTP) use TCP connections for bulk transfers and depend on the reliability, flow control, and congestion control abstractions that TCP connections provide. Other applications such as telnet primarily use TCP connections for reliability and not for flow or congestion control and

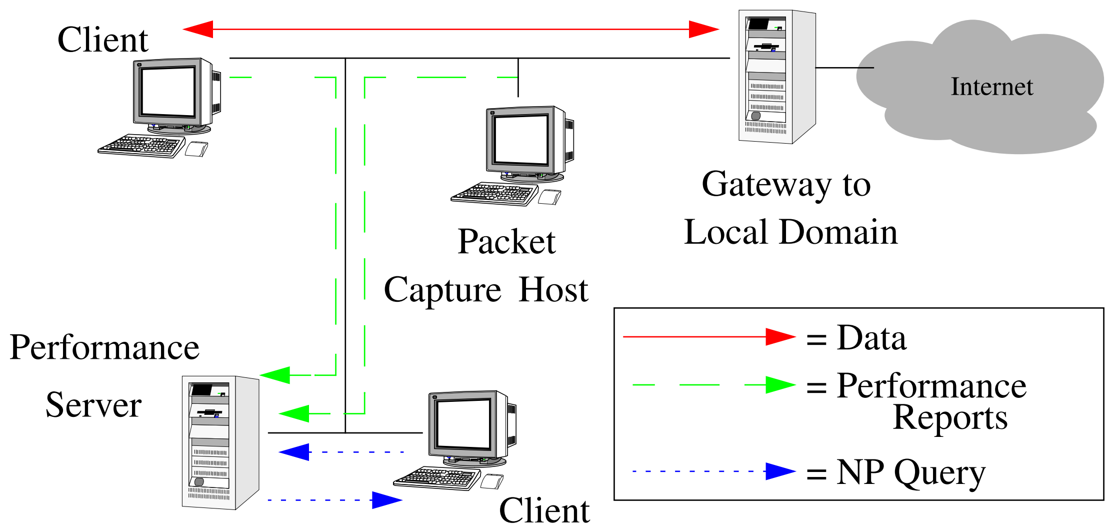

Figure 3. Design of SPAND

0	15	16	31
Version	Type	Prot.	App.Class
Source IP Address			
Source Port		Dest Port	
Dest IP Address			
NTP Timestamp, most sig word			
NTP Timestamp, least sig word			
Length of Sample in octets			
Duration of Sample in ms			
Total Packets Received			
Total Packets Lost			
Packet Size in octets			

Figure 4. Format of a Performance Report

0	15	16	31
Version	Type	Protocol	App. Class
Request IP Address			

Figure 5. Format of a Performance Request

0	15	16	31
Version	Type	Protocol	App. Class
Response IP Address			
Expected Available Bandwidth (kbits/sec)			
Std Dev of Available Bandwidth (kbits/sec)			
Expected Packet Loss Probability			
Std Dev of Expected Loss Probability			

Figure 6. Format of a Performance Response

other applications such as RealAudio in TCP mode use TCP connections for different reasons. The transport level network performance reported from these connections may vary widely depending on the way the transport connection is used, and we want to separate these applications into distinct classes.

An *performance server* receives performance reports from all clients in a local area and must incorporate them appropriately into its performance estimates. The performance server must maintain different estimates for different classes of applications as well as different classes of connectivity in a domain. In addition, the performance server can also identify reports that have possibly inaccurate information and discard them. Later, clients can query this information by contacting the performance server for the local area. A client query provides an performance request consisting of an (Address, Application Class) pair and the response returns a performance response for that pair, if one exists. The format of the performance request and response is shown in Figure 5 and Figure 6, and includes the performance server's estimates of available bandwidth and packet loss probability from the local domain to the specified foreign host.

4.1 Scalability through Local Communication

An important feature of our system is that all communication is localized to a local domain. Clients make reports about the connectivity to distant Internet hosts to a local performance server, and clients contact the local network performance server for answers to performance queries. This eliminates communication between domains and increases the scalability of the system.

4.2 Mechanisms for Incremental Deployment

The system described in the previous section is an ideal endpoint we would like to reach. In practice, it may be difficult to immediately modify all of the end clients to send performance reports. To quickly capture performance from a large number of end clients, a packet capture host can be deployed that uses a packet capture tool such as tcpdump to measure the network performance from end clients and send performance reports to the performance server. This allows a large number of performance reports to be collected while end clients are slowly upgraded.

5. Conclusions and Future Work

There are many classes of Internet applications that need the ability to predict in advance the network performance between a pair of internet hosts. Previous work providing this information has depended on isolated, active measurements from a single host. This does not scale to many users and does not provide the most accurate and timely information possible. In this paper, we have proposed a system called SPAND (Shared Passive Network Performance Discovery) that uses passive measurements from a collection of hosts to determine wide-area network characteristics. We have justified the design decisions behind SPAND and presented a detailed design of SPAND and mechanisms for incremental deployment.

We are currently in the process of implementing the system and have completed the implementation of toolkits that implement the Performance Reports. We are also working on the implementation of the packet capture host and modifications to BIND to enable it to handle NP Resource Records. We expect measurements of an operational SPAND prototype to be available soon.

To demonstrate the effectiveness of SPAND, we plan to write two applications using this framework. The first application is one that allows a system administrator to quickly browse the performance from the local domain to all foreign domains. The second is a small client-side WWW proxy that consults the performance server and places user-visible "performance hints" next to hyperlinks, as an estimate to the user of the expected performance to the site named in the hyperlink.

6. References

[1] M Arlitt and C. L. Williamson. Web Server Workload Characterization: The Search for Invariants. In *Proc. ACM SIGMETRICS '96*, May 1996.

[2] H. Balakrishnan, S. Seshan, M. Stemm, and R.H. Katz. Analyzing Stability in Wide-Area Network Performance. In *Proc. ACM SIGMETRICS '97*, June 1997.

[3] J.C Bolot. End-to-End Packet Delay and Loss Behavior in the Internet. In *Proc. ACM SIGCOMM '93*, San Francisco, CA, Sept 1993.

[4] R. L. Carter and M. E. Crovella. Dynamic server selection using bandwidth probing in wide-area networks. Technical Report BU-CS-96-007, Computer Science Department, Boston University, March 1996.

[5] R. L. Carter and M. E. Crovella. Measuring bottleneck-link speed in packet switched networks. Technical Report BU-CS-96-006, Computer Science Department, Boston University, March 1996.

[6] A. Chankhunthod, P. Danzig, C. Neerdaels, M.F. Schwartz, and K.J. Worrell. A Hierarchical Internet Object Cache. In *Proceedings 1996 USENIX Symposium*, San Diego, CA, Jan 1996.

[7] P. Francis. *http://www.ingrid.org/hops/wp.html*, 1997.

[8] J. Gwertzman and M. Seltzer. The Case for Geographical Push-Caching. In *Proc. Fifth IEEE Workshop on Hot Topics in Operating Systems*, May 1995.

[9] R. Hinden and S. Deering. *IP Version 6 Addressing Architecture*. RFC, Dec 1995. RFC-1884.

[10] V. Jacobson. Congestion Avoidance and Control. In *Proc. ACM SIGCOMM 88*, August 1988.

[11] S. Keshav. Packet-Pair Flow Control. *IEEE/ACM Transactions on Networking*, February 1995.

[12] J. C. Mogul. Network Behavior of a Busy Web Server and its Clients. Technical Report 95/5, Digital Western Research Lab, October 1995.

[13] C. Partridge, T. Mendez, and W. Milliken. *Host Anycasting Service*. RFC, Nov 1993. RFC-1546.

[14] pathchar – A Tool to Infer Characteristics of Internet Paths. ftp://ee.lbl.gov/pathchar, 1997.

[15] V. Paxson. *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, U. C. Berkeley, May 1996.

[16] W. R. Stevens. *TCP/IP Illustrated, Volume 1*. Addison-Wesley, Reading, MA, Nov 1994.

[17] UC Berkeley Annex WWW Traces. http://www.cs.berkeley.edu/ gribble/traces/index.html, 1997.