Technical Report
NWU-CS-04-34
April 19, 2004

Characterizing and Predicting TCP Throughput on the Wide Area Network

Dong Lu Yi Qiao Peter A. Dinda Fabian E. Bustamante

Abstract

The packet pair mechanism has been shown to be a reliable method to measure the bottleneck link capacity and available bandwidth on a network path, and has been widely deployed in tools such as nettimer, IGI, and PTR. However, the available bandwidth is different from the TCP throughput that an application can achieve and the difference can be huge. TCP throughput benchmarking techniques are widely used to probe the TCP throughput for applications, for example in the Network Weather Service (NWS). Unfortunately recent research shows that these techniques often cannot predict TCP throughput well for large transfers. This paper addresses this issue. We begin by statistically characterizing the TCP throughput on the Internet, exploring the strong correlation between TCP flow size and throughput, and the transient end-to-end throughput distribution. We then analyze why benchmarking fails to predict large transfers, and propose a novel yet simple prediction model based on our observations. Our prototype, dualPats, is an application level TCP throughput prediction framework that combines our model with simple time series models and a dynamic probing rate adjustment algorithm that relates intrusiveness to path dynamics. Our analysis and evaluation is based on large scale Internet-based measurements and experiments involving many sites distributed all over the world.

Effort sponsored by the National Science Foundation under Grants ANI-0093221, ACI-0112891, ANI-0301108, EIA-0130869, and EIA-0224449. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation (NSF).

Keywords: Statistical TCP throughput characterization; TCP throughput prediction; TCP throughput monitoring;

Characterizing and Predicting TCP Throughput on the Wide Area Network

Dong Lu   Yi Qiao   Peter A. Dinda   Fabian E. Bustamante
{donglu,yqiao,pdinda,fabianb}@cs.northwestern.edu
Department of Computer Science, Northwestern University

Abstract

The packet pair mechanism has been shown to be a reliable method to measure the bottleneck link capacity and available bandwidth on a network path, and has been widely deployed in tools such as nettimer, IGI, and PTR. However, the available bandwidth is different from the TCP throughput that an application can achieve and the difference can be huge. TCP throughput benchmarking techniques are widely used to probe the TCP throughput for applications, for example in the Network Weather Service (NWS). Unfortunately recent research shows that these techniques often cannot predict TCP throughput well for large transfers. This paper addresses this issue. We begin by statistically characterizing the TCP throughput on the Internet, exploring the strong correlation between TCP flow size and throughput, and the transient end-to-end throughput distribution. We then analyze why benchmarking fails to predict large transfers, and propose a novel yet simple prediction model based on our observations. Our prototype, dualPats, is an application level TCP throughput prediction framework that combines our model with simple time series models and a dynamic probing rate adjustment algorithm that relates intrusiveness to path dynamics. Our analysis and evaluation is based on large scale Internet-based measurements and experiments involving many sites distributed all over the world.

1 Introduction

The concept of available bandwidth has been of central importance throughout the history of packet networks, and researchers have been trying to create end-to-end measurement algorithms for a long time. From Keshav's packet pair [16], to Crovella's cprobe [6], and the latest work, such as IGI [12], the purpose is to measure the end-to-end available bandwidth accurately, quickly, and non-intrusively. Today's definition of available bandwidth is "the maximum rate that the path can provide to a flow, without reducing the rate of the rest of the traffic." [12, 13]. Other tools to measure either the bottleneck link capacity or the available bandwidth include nettimer [17], pathchar and pchar [11], pathload [13, 14], NCS and pipechar [15], pathrate [10], spruce [26] and pathchirp [24], and Remos [18]. Most of such tools used the packet pair or packet train techniques to conduct the measurements.

The available bandwidth is different from the TCP throughput that an application can achieve, and that difference can be huge. Lai's Nettimer paper [17] showed many cases where the TCP throughput is much lower than the available bandwidth, while Jain's pathload paper [13] showed the bulk transfer capacity [19] of a path could even be higher than the measured available bandwidth. Additionally, most of these tools take a long time to run, which make them unsuitable to be used in real time for applications and services.

The most widely used TCP throughput prediction tool is Network Weather Service [31] (NWS). NWS applies benchmarking techniques and time series models to measure TCP throughput and provide predictions to applications in real time. NWS has been broadly applied. Allen, et al [3] applied NWS to address the so called Livny and Plank-Beck problems. Swany, et al [28] applied NWS in the Grid information service.

Unfortunately, recent work [30, 29] has argued that NWS, and by implication, TCP benchmarking techniques in general, are not good at predicting large file transfers on the high speed Internet. Sudharshan, et al [30] proposed and implemented predicting large file transfers from a log of previous transfers and showed that it can produce reasonable results. However, a pure log-based predictor is updated at application-chosen times and thus neglects the dynamic nature of the Internet. Hence, when a path changes dramatically, the predictor will be unaware of it until after the application begins to use the path. To take the dynamic changes

Effort sponsored by the National Science Foundation under Grants ANI-0093221, ACI-0112891, ANI-0301108, EIA-0130869, and EIA-0224449. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation (NSF).

of Internet into consideration, Sudharshan, et al [29] and Swany, et al [27] separately proposed regression and CDF techniques to combine the log-based predictor with small NWS probes, using the probes to estimate the current load on the path and adjust the log-based predictor. These techniques enhanced the accuracy of log based predictors.

These combined techniques are limited to those host pairs that have logs of past transfers between them, and due to the dynamic nature of Internet, which only shows certain statistical stabilities, the logs can become invalid after some time. Furthermore, due to the strong correlation between TCP flow size and throughput [35], logs for certain ranges of TCP flow (file) size are not useful for the prediction of different TCP flow sizes.

The questions we address here are:

- How can we explain the strong correlation between TCP flow size and throughput, and what are its implications for predicting TCP throughput?
- How can we characterize the statistical stability of the Internet and TCP throughput, and what are its implications for predicting TCP throughput?
- How can we predict the TCP throughput with different TCP flow sizes without being intrusive?

The main contributions of this paper are:

- We explored reasons for the observed strong correlation between TCP flow size and throughput [36].
- We characterized end-to-end TCP throughput stability and distribution.
- We proposed a novel yet simple TCP benchmark mechanism.
- We proposed a dynamic sampling rate adjustment algorithm to lower active probing overhead.
- We described and evaluated dualPats, a TCP throughput prediction service based on the preceding contributions.

We define TCP throughput as $\frac{D}{T}$ where D is the TCP flow size and T is the TCP flow duration, which starts at TCP connection initialization and ends when data transfer finishes. In some of our experiments using GridFTP [2] and scp [33], we treat D as equivalent to file size, neglecting the small messages exchanged for authentication. T is equivalent to file transfer time in our experiments with GridFTP and scp. TCP throughput is directly experienced by applications and thus accurate predictions are very important for the design and implementation of distributed applications.

We begin by describing our experimental setup and measurements (Section 2). In Section 3, we use our measurements to address the strong correlation between TCP flow size and throughput, explaining the phenomenon and how it can cause benchmarking to err, and develop a new predictive model that incorporates it. Next, we consider the statistical stability of the Internet and how it affects the lifetime of measurements and predictions (Section 4). Finally, we incorporate our results into dualPats and evaluate its performance (Section 5).

2 Experiments

Our experimental testbed includes PlanetLab and several additional machines located at Northwestern and Argonne National Laboratories (ANL). PlanetLab [1] is an open platform for developing, deploying, and accessing planetary-scale services. It currently consists of 359 computers located at 147 sites around the world.

We conducted S1 and S2 mainly to characterize the TCP throughput on the Internet in which we implemented a simple C client-server program. S2 was also used to verify the new TCP benchmarking mechanism we proposed. We conducted S3 using GridFTP and scp to strengthen S1 and S2 with big TCP flows and with applications that require authentication before transferring effective data. S3 was also used to further verify the benchmarking mechanis. S4 was conducted to evaluate dualPats, our TCP throughput prediction framework.

Our experiments are summarized in Figure 1.

3 The strong correlation between TCP flow size and throughput

A surprising finding in recent TCP connection characterization is that TCP flow size and throughput are strongly correlated. This section explains the phenomenon, provides new additional explanations for it, explains why it can lead to inaccurate TCP throughput predictions, and outlines a new prediction approach.

3.1 Phenomenon

Yin Zhang, et al [35] analyzed the correlations between the TCP flow characteristics of interest, including flow duration and throughput, flow duration and size, and flow size and throughput. They pointed out that these correlations are fairly consistent across all their traces, and show a slight negative correlation between duration and throughput, a slight positive correlation between size and duration, and a strong correlation between throughput and flow size. They pointed out that the strong correlation between flow size and throughput is the most interesting one and explained it in two ways:

Name	Statistics	Main Purpose	Hosts, Paths, Repetitions	Messages, Software, procedure
S1	1,620,000 TCP transfers	To evaluate TCP throughput stability and transient distributions	40 PlanetLab nodes in North America, Europe, Asia, and Australia. Repeat random pairing 3 times, 60 distinctive paths total	Client/Server: 100 KB, 200 KB, 400 KB , 600 KB, 800 KB, 1 MB, 2 MB, 4 MB, 10 MB. Server sends a file with specific size to client continuously for 3,000 times and then start to send a another file of different size.
S2	2,430,000 TCP transfers; 270,000 runs	To study correlation between TCP throughput and flow size, and evaluate proposed TCP benchmark mechanism.	40 PlanetLab nodes in North America, Europe, Asia, and Australia. Repeat random pairing 3 times, 60 distinctive paths total	Client/Server: 100 KB, 200 KB, 400 KB , 600 KB, 800 KB, 1 MB, 2 MB, 4 MB, 10 MB. Server sends a sequence of files with increasing sizes in order, start over after each *run*.
S3	4,800 TCP transfers; 300 runs	To test proposed TCP throughput benchmark mechanism; To strengthen S1 and S2 with large TCP flow sizes and different applications	20 PlanetLab nodes in North America, Europe, Asia, and Australia, one node at Northwestern, one node at ANL, 30 distinctive paths total	GridFTP, scp: 5 KB to 1GB. Server sends a sequence of files with increasing sizes in order, start over after each *run*.
S4	2400 test cases	To evaluate the dualPats TCP throughput prediction service.	20 PlanetLab nodes in North America, Europe, Asia, and Australia, one node at Northwestern, one node at ANL, 20 distinctive paths total	GridFTP, scp: randomly send a file of size 40 MB or 160 MB. About 48 hours long

Figure 1. Summary of experiments. We define a *run* in S2 and S3 as a procedure conducting a sequence of TCP transfers with increasing flow sizes between two hosts.

- **Slow start**: TCP slow start could cause some correlation between flow size and flow rate [35]. Hari Balakrishnan, et al [4] showed that 85% of the web-related TCP packets were transfered during slow start. This implies that most web-related flows ended in slow start, before TCP had fully opened its congestion window, leading to throughput much lower than would be possible with a fully open window.

  However, after eliminating the first one second of all the flows, they found that the strong correlation between flow size and throughput remained strong.

- **User effect**: The users are estimating the underlying bandwidth, and thus transferring big files only when the estimated estimated bandwidth is correspondingly large.

These are two valid reasons, but they may be insufficient. We claim that most users do not estimate the available bandwidth before transferring data. Furthermore, that the correlation persists even when initial slow start is removed suggests that there must be some other mechanisms at work.

Let's consider the correlation between flow size and throughput in our experiments. Figure 2 gives the cumulative distribution functions (CDFs) of the correlation coefficient (Pearson's R)[1], where each individual R value is calculated from one run of S2 or S3. It is clear from the graph that for the simple C server results in S2, over 80% of all runs demonstrate strong or medium Rs between flow sizes and flow rates. Further, 64% of all runs have $R > 0.6$. The correlation is much stronger for the GridFTP and scp results in S3: $> 98\%$ of the runs shows strong correlation, $> 95\%$ show $R > 0.8$.

3.2 Explanations

Now we consider additional explanations for the surprising correlation between flow size and transfer time.

[1]Both Pearson's Correlation Coefficient R and Coefficient of Determination R^2 are used in the analysis of the paper. R^2 represents the percent of the variation that can be explained by the regression equation, therefore we use it to show how good a curve fitting is. R is widely used to measure the strength of a (linear) relationship, therefore we use R to show how strong two random variables are linearly correlated.

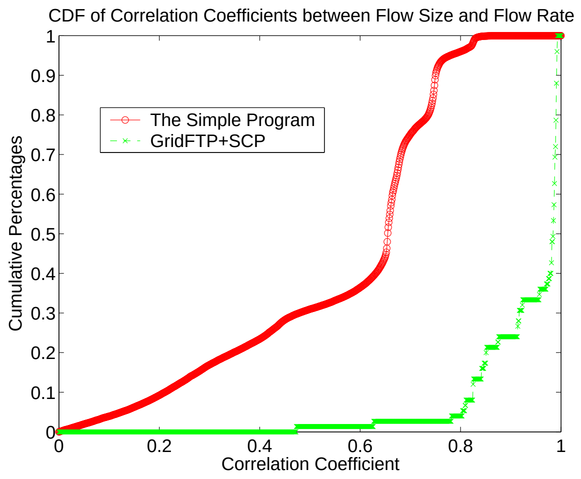

Figure 2. CDF of correlation coefficients R between flow sizes and throughput in experiments S2 and S3.

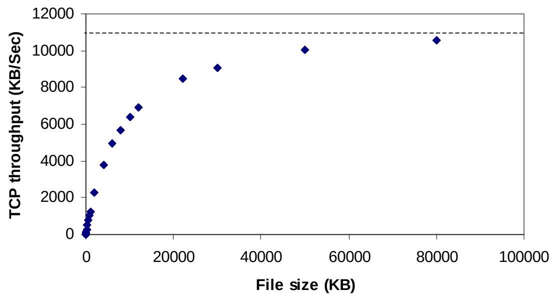

Figure 3. TCP throughput versus flow size (file size) with GridFTP. Transfers are between Northwestern University and Argonne National Lab. Single TCP flow with TCP buffer set.

Non-negligable startup overheads

Most applications have an initial message exchange. For example, GridFTP and scp require certificate or public key authentication before starting to send or receive data.

Figure 3 shows the TCP throughput as a function of TCP flow size, for transfers using GridFTP between Northwestern university and ANL. The dotted line is the asymptotic TCP throughput. We tried linear, logarithmic, order 2 polynomial, power, and exponential curve fitting, but none of them fit well.

We next considered the relationship between TCP flow duration (transfer time) and flow size (file size). Figure 4 shows that this relationship can be well modeled with a simple linear model with R^2 close to 1. The majority of the data-points missed by the linear model are located at the very beginning of the curve, which we refer to as the *noise area* in the figure. The noise area is due to startup costs and the residual slow start effect, described below.

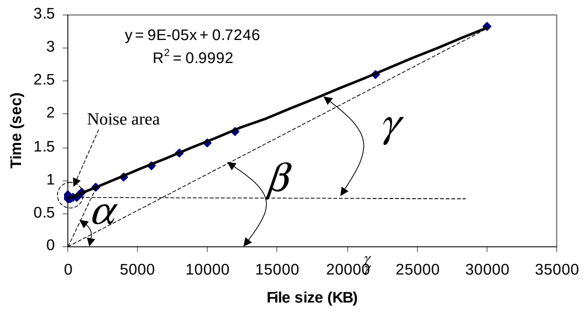

Figure 4. Transfer time versus TCP flow size with GridFTP. Transfers are between Northwestern University and Argonne National Lab. Single TCP flow with TCP buffer set.

A closer look at Figure 4 shows that the total TCP flow duration or file transfer time can be divided into two parts: the startup overhead and the effective data transfer time. In this case, the startup overhead is about 0.72 seconds. We represent this as

$$T = A \times x + B \qquad (1)$$

where T is the TCP flow duration, including both startup overhead and data transfer time, x is the TCP flow size or file size, and B is the startup overhead, which includes authentication time and the residual slow start effect as described below. $\frac{1}{A}$ equals Steady State TCP throughput, which is the asymptotic TCP throughput as shown in Figure 3.

Given Equation 1, we can easily deduce the expression for the TCP throughput in Figure 3 as

$$TP = \frac{x}{T} = \frac{x}{A \times x + B} \qquad (2)$$

where TP is the TCP throughput, and x, A, B are the same as in Equation 1.

Residual slow start effect

Mathis, et al [20] pointed out that it takes TCP some time before its throughput reaches equilibrium. Assuming selective acknowledgments (SACK), TCP will send roughly $\frac{1}{p} \times log_2 \frac{1}{C\sqrt{p}}$ packets in the unstable phase, where p is the loss rate and C is a constant $\approx \sqrt{\frac{3}{2}}$. This number can be significant given a low loss rate p. This happens because with SACK, slow start will overshoot and drive up the loss

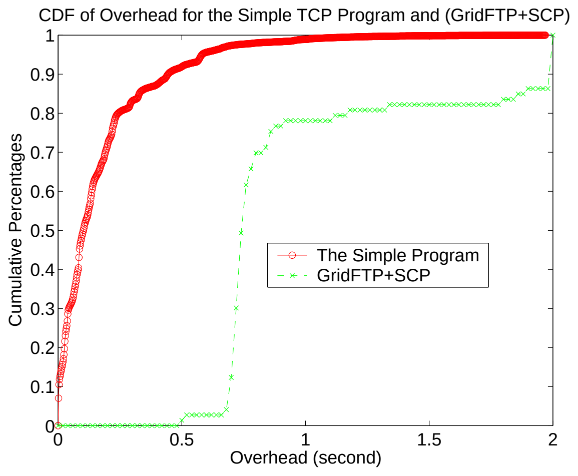

Figure 5. CDF of B, the startup overhead. Even for the simple client/server there is startup overhead likely caused by the residual slow start effect. The startup overheads of scp and GridFTP are much larger.

rate or run out of receiver window. Zhang, et al [36] showed that the mean loss rate in their traces is between 0.006 and 0.0087. Assuming the loss rate is 0.006, and each packet is 1.5 KB, roughly 800KB data has to be sent before TCP throughput reaches equilibrium.

We examined hundreds of Linux machines on the Northwestern University campus and on PlanetLab and found that all of them were using SACK. Therefore, it is likely that most TCP connections experience this slow start overshoot effect, and because TCP in slow start doesn't use bandwidth well, this residual slow start effect can be treated as another kind of startup overhead, incorporated in B as above. This can also explain why in Figure 2 the Rs for the scp and GridFTP traces are much stronger than that of the simple program.

To verify that this is the case in general for the simple applications without other startup overheads, we used the data collected in experiment S2. We did least square linear curve fitting and calculated B for each set of data. Figure 5 shows the CDF for these Bs. The effect of residual slow start is obvious in the CDF, where we see over 50% simple TCP transfers has a B value equal or larger than 0.1. For comparison purpose, we also plot the CDF of B for applications that require authentication in the same Figure, namely GridFTP and SCP. As the CDF indicates, a typical B for such applications is much larger than that of the simple application.

Why simple TCP benchmarking fails

Now we can explain why current TCP benchmarking approaches, such as implemented in NWS, have difficulty predicting the performance of large transfers such as GridFTP tests [29]:

- The default probe used by NWS is too small. It will likely end up in the noise area as shown in Figure 4.
- The TCP throughput that the probe measures is only useful to TCP flows of similar size because of the strong correlation between throughput and flow size.

  Given Equation 2, it is clear that $cot(\alpha)$ is the TCP throughput for the file size 2000KB, $cot(\beta)$ is the TCP throughput for the file size 30000KB and $cot(\gamma)$ is the steady state TCP throughput. As file size increases α decreases, and when the file size is approaching infinity, the throughput will approach $cot(\gamma)$.
- The TCP buffer is not set for NWS probes while the GridFTP tests were done with adjusted TCP buffer size.
- The usage of parallel TCP flows in GridFTP increased its aggregated throughput.

To verify that the linear model is true for most Internet paths, Figure 6 shows the R^2 of the linear curve fitting for the data in experiment S2 and S3. It is clear the model holds for both our simple client and server, and applications such as scp and GridFTP that require authentication.

3.3 A new TCP throughput benchmark mechanism and its verification

Based on the above observations, we developed a new simple TCP benchmark mechanism. Instead of using probes with the same size, we use two probes with different sizes, chosen to be beyond the noise area. We then fit a line between the two measurements, as shown in figure 4. Using Equation 1 and 2, we can then calculate the TCP throughput for other flow sizes (file sizes).

To verify that the new TCP benchmark works, we used the trace data in experiment S2. We chose a small probe with size 400KB and a bigger probe with size 800KB, and predicted the throughput of the other TCP transfers in the trace. Figure 7 shows the CDF of relative prediction error for our results by flow size. $>$ 80% of the prediction errors are below 20%.

The CDFs look normal, so we used quantile-quantile plots to test this. Figure 8 shows an example plot. In almost all cases, we can fit a straight line to these plots with $R^2 \approx 1$, which tells us that our relative error is almost always normal. Normality of prediction errors here is both

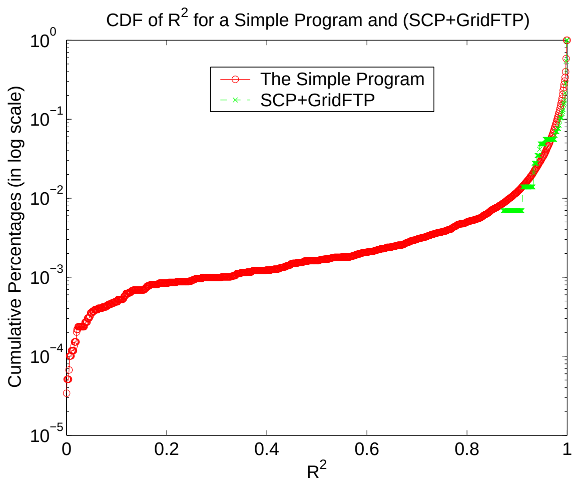

Figure 6. CDF of R^2 for linear model of Figure 4. Each R^2 is from a independent test. Both simple client/server and applications that require authentication show a strong linear property. Note that the Y axis is in log scale to show detail. Over 99% of the runs had $R^2 > 0.95$.

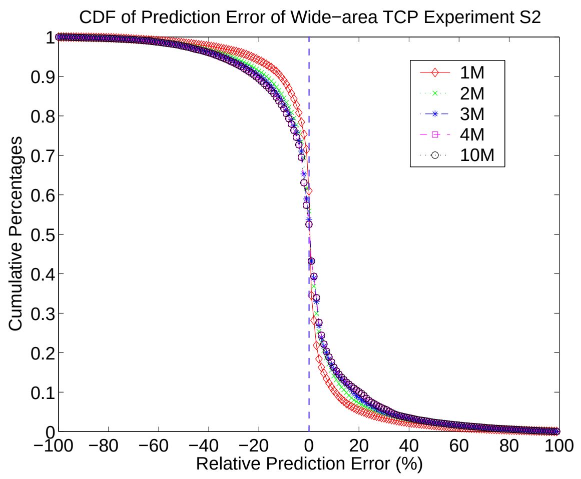

Figure 7. CDF of relative prediction error for TCP throughput with different flow sizes.

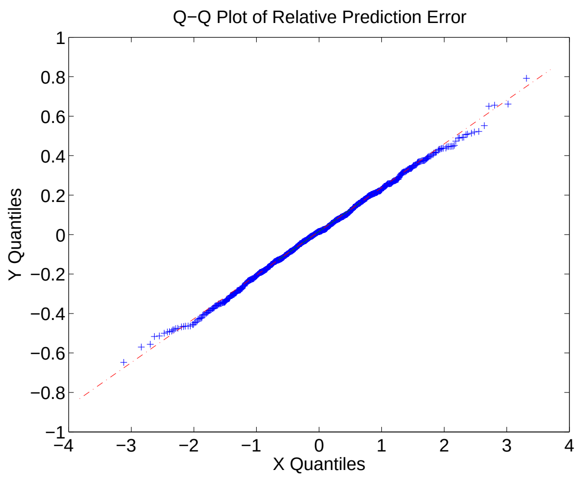

Figure 8. Quantile-quantile plot of relative prediction error with flow size 2MB against standard normal distribution. qqplot of relative prediction error for flows with different sizes looks almost identical to this figure.

surprising and extremely useful. In particular, we can simply estimate the variance of the relative prediction error as we measure and predict, and then use this information straightforwardly to create confidence intervals for our predictions. Being able to compute accurate confidence intervals is vital to using predictions in practice [9].

There are other techniques that we can apply to enhance the prediction accuracy, which we will cover in later sections.

In practice, we don't have to send two probes. Instead, we can send the larger one of the two probes, record its starting time, the time when as much data as the size of the small probe was sent and full probe's finishing time. We call such a probe a *probe pair*.

Figure 9 shows the CDFs of standard deviation of transfer time on all paths in experiment S2. Each CDF curve corresponds to a particular flow size. For most paths, the larger the flow size is, the bigger the standard deviation of the transfer time. However, in Figure 10, where we show the CDFs of the coefficient of variation (COV) of transfer time, the conclusion is the reverse: for most paths, the bigger the file is, the less the COV of the transfer time. This essentially means that in *relative* terms, the variance of transfer time of flows with larger sizes are actually smaller. The bigger the flow, the better.

As explained in Section 3.2 and 3.3, we need two probes with different sizes to determine the steady state TCP throughput. Inevitably, fluctuations of flow transfer time happen on the dynamic Internet, and have shown themselves in the standard deviation and COV we have just seen. These fluctuations are the main cause of the estimation error of steady state TCP throughput. Since flows with larger sizes actually have less variance in *relative* terms, estimating steady state throughput using larger flows will certainly be more accurate. On the other hand, probes with larger flows are more expensive. This leads us to the selection of

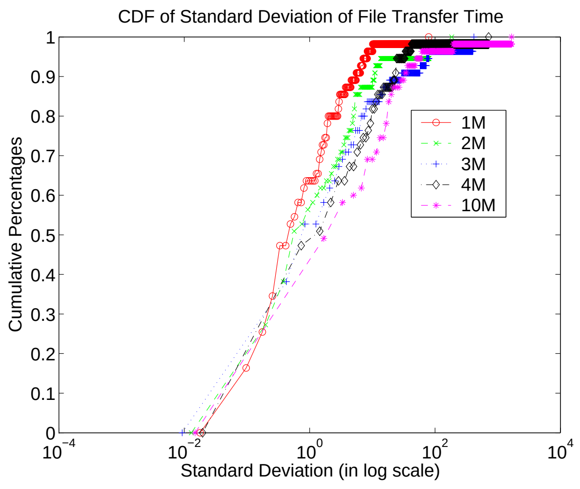

Figure 9. CDF of standard deviation of transfer time at all Internet paths for 5 different flow sizes.

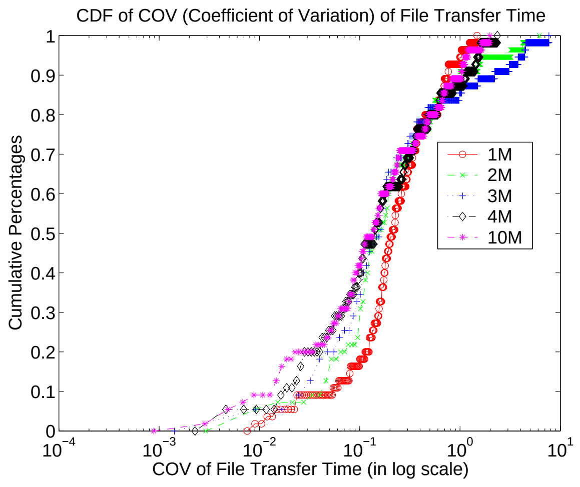

Figure 10. CDF of COV (coeffecient of variation $COV = \frac{std}{mean}$) of transfer Time at all Internet paths for 5 different flow sizes.

two probe sizes of 400 KBytes and 800 KBytes as default probes, which we feel is a reasonable tradeoff between estimation accuracy and probing cost.

4 Statistical stability of the Internet

Statistical stability or consistency is one of the most important characteristics of the Internet and is the basis that makes it possible to predict TCP throughput on the wide area network. A good understanding of stability will also help us to make decisions about prediction strategies, such as the frequency of active probing, and therefore to lower the intrusiveness of the predictors.

Routing stability: Paxson [23, 22] proposed two metrics for route stability, prevalence and persistency. Prevalence, which is of particular interest to us here, is the probability of observing a given route over time. If a route is prevalent, than the observation of it allows us to predict that it will be used again. Persistency is the frequency of route changes. The two metrics are not closely correlated. Paxson's conclusions are that Internet paths are heavily dominated by a single route, but that the time periods over which routes persist show wide variation, ranging from seconds to days. However, 2/3 of the Internet paths Paxson studied had routes that persisted for days to weeks. Chinoy found that route changes tend to concentrate at the edges of the network, not in its "backbone" [7].

Spatial locality and temporal locality: Balakrishnan, et al analyzed statistical models for the observed end-to-end network performance based on extensive packet-level traces collected from the primary web site for the Atlanta Summer Olympic Games in 1996. They concluded that nearby Internet hosts often have almost identical distributions of observed throughput. Although the size of the clusters for which the performance is identical varies as a function of their location on the Internet, cluster sizes in the range of 2 to 4 hops work well for many regions. They also found that end-to-end throughput to hosts often varied by less than a factor of two over timescales on the order of many tens of minutes, and that the throughput was piecewise stationary over timescales of similar magnitude [5]. Myers, et al examined performance from a wide range of clients to a wide range of servers and found that bandwidth to the servers and server rankings from the point of view of a client were remarkably stable over time [21]. Seshan, et al applied these findings in the development of the Shared Passive Network Performance Discovery (SPAND) system [25], which collected server performance information from the point of view of a pool of clients and used that history to predict the performance of new requests.

Zhang, et al [36] experimented by sending 1 MB files every minute between pairs of hosts, and proposed an effective way to evaluate the temporal locality of end-to-end TCP throughput of those flows. He looks at the length of the period where the ratio between the maximum and minimum observed TCP throughput is less than a constant factor ρ. This is referred to as an Operational Constancy Region (OCR). Instead of using OCR, we define a *Statistically Stable Region* (SSR) as the length of the period where the ratio between the maximum and minimum *estimated* steady state TCP throughput is less than a constant factor ρ. The difference between OCR and SSR is important because OCR is only characterizing the throughput for flows with a specific

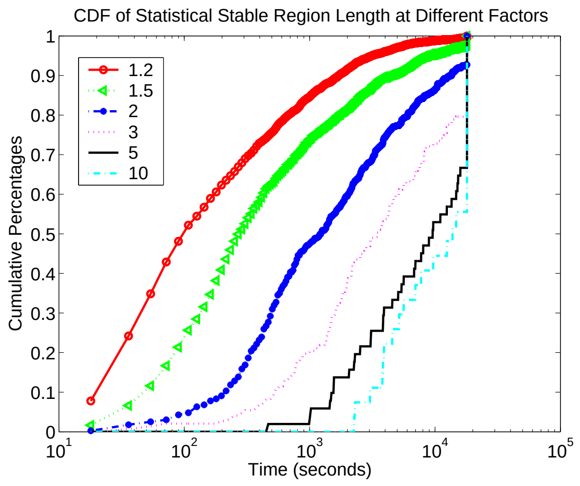

Figure 11. CDF of statistically stable region (SSR) for steady-state TCP throughput with different ρ.

size, while SSR characterizes the steady state throughput for all flows with different sizes. We used traces from experiment S2 to characterize the SSR with steady-state TCP throughput. That is, instead of looking at the TCP throughput of a specific flow size, we applied least square linear fitting to get Equation 1, and therefore the estimated steady-state TCP throughput of the path.

Figure 11 gives the CDF of length of all SSRs modeled by steady-state TCP throughput from experiment S2. Each curve in the plot corresponds to a particular value of the constant factor ρ. Under all different values of ρ, some degree of temporal locality is exhibited. Moreover, the larger ρ is, the longer the SSRs tend to be.

For comparison purposes, we also calculated the CDF of OCR with data from S1. The comparison between ours and Zhang's results [36] suggests that the temporal locality in our test environment is much weaker. For instance, Zhang found that $\approx 60\%$ of OCRs are longer than 1 hour when $\rho = 2$ and $> 80\%$ of all OCRs exceed 3 hours when $\rho = 10$. In our results, the two corresponding numbers drop to 2% and 10% respectively. TCP throughput in our testbed appears to be less stable. We suspect that this difference may largely due to the fact that Planetlab nodes often become CPU or bandwidth saturated, causing great fluctuations of TCP throughput. It is challenging to predict TCP throughput under a highly dynamic environment.

End-to-end TCP Throughput Distribution: An important question an application often poses is how the TCP throughput varies, and, beyond that, whether an analytical distribution model can be applied to characterize its distribution. Balakrishman, et al [5] studied aggregated TCP throughput distribution across all different flow sizes between each pair of Internet hosts. Their statistical analysis suggests that end-to-end TCP throughput can be well modeled as a log-normal distribution.

Since we have already seen earlier that there exists strong correlation between TCP throughput and flow size, we are therefore more interested in studying the TCP throughput distribution of a particular flow size than in getting an aggregated throughput distribution across all different flow sizes. The data from experiment S1 lets us do this analysis.

Recall that in S1, for each client/server pair, we repeated the transfer of each file 3,000 times. We histogramed throughput data for each flow size/path tuple. Almost in every case, the throughput histrogram demonstrates a multimodal distribution. This suggests that it is probably not feasible to model TCP throughput using simple distributions.

Because the collection of data for each client/server pair lasted several hours or even longer, we suspect that the multimodal feature may be partially due to the change in network conditions during the measurement period. To verify this hypothesis, we try to study throughput distribution using subsets of each dataset. A subset contains much less data and covers shorter measurement length. In other words, we hoped to find "subregions" in each dataset in which the network conditions are relatively stable and the throughput data can be better modelled unimodally.

It is very hard to predefine an optimal length or data size for such "subregions" in the throughput data; in fact, the appropriate length may vary from time to time. Therefore, we believe it is necessary to adaptively change the subregion length over time as we acquire data (or walk the dataset offline). The purpose is to segment the whole dataset into multiple subregions (or identify segement boundaries online). For each segment, we fit the data with several analytical distributions, and evaluate the goodness of fit using the values of R^2.

Our offline distribution fitting algorithm for TCP throughput has the following steps:

1. Select a trace of TCP throughput (sequence of measurements for a particular flow size on a particular Internet path).

2. Initialize the subregion length, and set the start and end point of the subregion to 1 and 100, respectively.

3. Fit the subregion data with an analytical distribution, and calculate the value of R^2.

4. Increase the subregion length by 100, that is, keep the start point as from the previous step, but increase the end point by 100. For this new subregion, fit the data with the analytical distribution model again, get a new value of R^2. Note that the granularity here, 100, can also be changed.

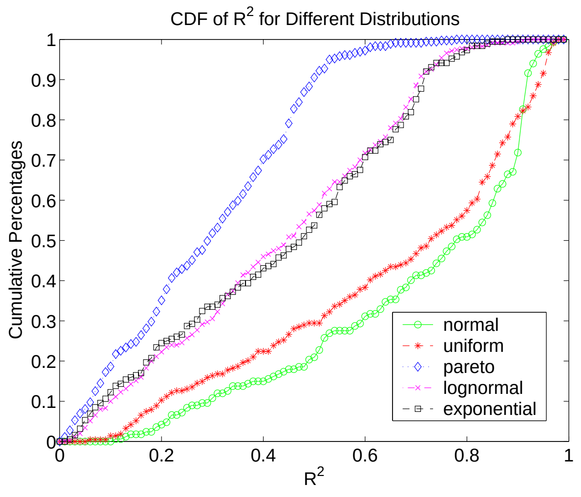

Figure 12. CDF of R^2 for five common distributions for TCP throughput characterization on segmented traces. The size of the file is 10 MBytes. Other flow sizes show similar results.

5. Compare the new R^2 with the previous one. If the new one is larger, repeat step 4, otherwise, we have found that previous subregion has the optimal length.

6. Log the start point, end point, and value of R^2 from previous subregion. Reset the subregion length to be 100, and set the start point of the subregion to be one larger than the end point of the previous subregion.

7. Go to step 3 and repeat above procedure, until all data points in the datasets are examined.

We segmented and model-fitted each path/flow size trace in S1 using this algorithm. We then considered the R^2 distribution for each of flow size and analytical distribution. The CDFs of the values of R^2 for each flow size and analytical distribution are shown in Figure 12.

It is clear from Figure 12 that for the five distributions we compared, the normal distribution best fits the TCP throughput data. For example, of all subregions, $> 50\%$ have $R^2 \geq 0.8$ with the normal distribution, but for the lognormal distribution, the percentage drops to $\approx 2.3\%$. For the uniform, exponential, and Pareto distributions, the percentages are 42%, 2.7%, and 0.0%, respectively.

This result firmly shows that for a particular flow size, its throughput is best characterized by a simple normal distribution. However, throughput is *nonstationary*, so a given normal distribution holds for only a period of time before it changes to another one. This nonstationary behavior is remarkably similar to the "epochal behavior" pattern of load on hosts that we observed in earlier work [8].

On the other hand, none of analytic distributions are *overwhelmingly* successful. Even for the normal distribution, $R^2 < 0.8$ half of the time.

5 dualPats: predicting TCP throughput on the wide area network

Based on our study and previous research, we have developed and evaluated dualPats, a prototype TCP throughput prediction service for distributed applications. dualPats actively sends out probe pairs to benchmark a path. It automatically adjusts its rate to minimize intrusiveness. The benchmarking technique is described in Section 3.

5.1 System architecture

dualPats consists of two components, a network sensor and a TCP throughput predictor. Figure 13 illustrates the two components and their relationship with applications and the underlying operating system. The whole system works at application level.

The network sensor sends out probe pairs at a self-adjusting rate as described in Section 5.2. It records the sizes and transfer times of each probe pair. When monitoring N different TCP connections, N series of probe pair records is maintained.

The TCP throughput predictor interfaces with both the network sensor and applications. Whenever an application needs a prediction, it sends a query to the TCP throughput predictor, and the predictor execute the following:

1. Parse the query from the application and get parameters including the destination and file size.

2. Fetch the probe pair data series for the destination from underlying network sensor. If no series exists, an error is returned and a probing process for the destination is started.

3. Apply a prediction model, such as moving average or EWMA, to predict the current transfer times of each of the messages in the probe pair from the series of pairs.

4. Fit a linear curve as described in Equation 1 and calculate the TCP throughput for the given file size and using equation 2. (Optionally, compute a confidence interval using normality assumptions).

5. Return the estimated TCP throughput for transfer time to the application.

We tested several prediction models in step 3, including moving average, exponential weighted moving average

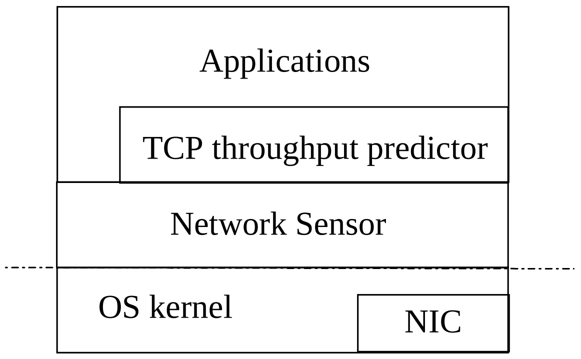

Figure 13. System architecture of dualPats.

(EWMA) and simply using the last value. A moving average with window size 20 works best on average in our experiments. We speculate that this is so because during each SSR, the end-to-end TCP throughput is best modeled with normal distribution. For a normal distribution with no serial correlation, the mean is the best predictor possible, and the windowed mean estimates this.

5.2 Dynamic sampling rate adjustment algorithm

There are two ways to decrease the overhead caused by the probe pairs of dualPats: decrease the size of the probe pair or decrease the sampling rate.

As we discussed in Section 4, each Internet path shows statistical stability in TCP throughput. However, each path is different in the length of its SSR. Therefore, we designed a simple algorithm to dynamically adjust the sampling rate to the path's SSR. The algorithm is as follows:

1. Set an upper bound U and a lower bound L for the sampling interval. They were set as 20 and 1200 seconds in our tests.

2. Set another two relative changing bounds, $B1$, $B2$, in units of percentage. After sending each probe pair, estimate the current steady-state TCP throughput. If it has changed less than $B1$, increases the sampling interval by a step of S seconds; if it changes between $B1$ and $B2$, keep the current interval; otherwise decrease the interval. In experiment S4, $B1$, $B2$ were set to be 5% and 15%.

3. The interval must be between L and U.

We also want to minimize the size of probe pairs on the condition that none of them will fall into the noise area as shown in Figure 4. However, the noise area is different for each Internet path, as discussed in Section 3. It is a function of loss rate and underlying bandwidth. We need an algorithm that can detect it automatically. For now, the algorithm uses feedback from application about its prediction error.

1. We set a default value or staring value for the probe pairs. In experiment S4, we used 400KB and 800KB. Also set a upperbound US for the probe pair.

2. If M continuous prediction errors are bigger than a threshold TH, and with the same sign, we increase the probe pair by 200KB each.

3. The probe pair can't exceed US.

5.3 Evaluation

We evaluated dualPats using the data from experiment S4, which we described in detail in Section 2. The primary metric used was the relative prediction error:

$$err = \frac{PredValue - RealValue}{RealValue} \tag{3}$$

dualPats ran $\approx$ 2400 predictions on the 20 paths 48 hours long S4 experiments. Test cases are randomly chosen 40MB or 160MB files.

The prediction results are shown in Figure 14. Mean error is calculated by averaging all of the relative errors. For an unbiased predictor, this value should be close to zero given enough test cases. We can see that in our evaluation it is quite small in most cases, and we see an equal proportion of positive and negative mean errors.

The mean absolute error is the average of the absolute value of all of the relative errors. We consider it the most important metric in evaluating the predictions. Figure 14 shows that all 20 paths have a mean absolute error $< 30\%$, 17 out of 20 paths are $< 20\%$, and 13 out 20 paths are $< 15\%$. As we commented in Section 4, PlanetLab is much more heavily loaded and dynamic than the current Internet, thus it is likely to be much harder to predict than on the current Internet.

We studied the correlation between the error and several known attributes. The results are shown in Figure 15. Clearly, the mean error is not related to any of the attributes, which further suggests that the predictions given by dualPats are unbiased. However, if the path is very dynamic it is hard to predict. Figure 15 shows that the R between the mean absolute error and the sampling interval length (and, indirectly, the SSR) is negative and pretty strong. This implies that our algorithm captured the path dynamics and tried to adjust to its changes.

Our conclusion is that dualPats does a effective job of predicting TCP throughput.

Path	Router Hops	Mean RTT	Mean err %	Mean stderr	Mean abs(err)%	Mean abs(stderr)	Mean Interval
1	20	55	-0.0073	0.11	0.069	0.13	641.97
2	18	60	0.10	0.17	0.17	0.18	29.44
3	17	33	-0.21	0.23	0.25	0.51	132.2
4	11	27.5	-0.03	0.19	0.13	0.25	71.56
5	13	31	-0.04	0.20	0.16	0.28	48.76
6	16	138	-0.079	0.19	0.14	0.29	58.18
7	16	120	0.048	0.355	0.28	0.42	21.87
8	14	51	0.021	0.12	0.095	0.168	512.64
9	18	207	-0.14	0.17	0.18	0.36	51.50
10	14	29	-0.11	0.19	0.14	0.31	180.17
11	19	110	-0.036	0.18	0.11	0.24	28.57
12	15	36	-0.038	0.14	0.078	0.18	258.16
13	17	59	0.035	0.208	0.16	0.24	32.23
14	12	23.5	-0.012	0.060	0.042	0.082	320.97
15	13	28	-0.095	0.186	0.14	0.31	511.33
16	18	100	-0.028	0.16	0.11	0.21	543.75
17	19	70	-0.083	0.030	0.083	0.17	543.63
18	14	81	-0.076	0.025	0.076	0.154	522.20
19	19	72	0.21	0.38	0.29	0.39	48.39
20	17	50	0.11	0.12	0.14	0.12	97.25

Figure 14. Prediction error statistics for experiment S4. RTT is the round trip time between the two sites in miliseconds, and Mean Interval is the average interval time between probe pairs in seconds. Mean err is the average relative error while mean abs(err) is the average of the absolute relative errors.

	Router Hops	Mean RTT	Mean Interval
Mean abs(err)	0.112	0.257	-0.62
Mean err	0.076	0.13	-0.13

Figure 15. Correlation coefficient R between prediction error and known attributes.

6 Conclusions and future work

We have characterized the behavior of TCP throughput in the wide area environment, providing additional explanations for the correlation of throughput and flow size and demonstrating how this correlation causes erroneous predictions to be made when using simple TCP benchmarking to characterize a path. In response, we proposed and evaluated a new benchmarking approach, probe pair, from which TCP throughput for different messages sizes can be derived. We described and evaluated the performance of a new predictor, dualPats, implements this approach.

In this work, we do not consider parallel TCP flows, which is a current subject for us. We also acknowledge that, like all benchmarking-based systems, our approach has scalability problems. We have addressed this to some extent with our dynamic sample rate adjustment algorithm. However, we are also considering whether our ideas can work within a passive measurement model such as that in Wren [34], and the use of hierarchical decomposition as in Remos [18] and NWS Clique [32]. We have assumed that the network path is the bottleneck for file transfer. In some cases, especially in high speed optical networks, this may not be true, and transfer time prediction would also have to take into account processor and memory system performance.

References

[1] http://www.planet-lab.org.

[2] Allcock, W., Bester, J., Bresnahan, J., Cervenak, A., Liming, L., and Tuecke, S. GridFTP: Protocol extensions to ftp for the grid. Tech. rep., Argonne National Laboratory, August 2001.

[3] Allen, M., and Wolski, R. The livny and plank-beck problems: Studies in data movement on the computational grid. In *Supercomputing 2003* (November 2003).

[4] Balakrishnan, H., Padmanabhan, V. N., Seshan, S., Stemm, M., and Katz, R. H. TCP behavior of a busy internet server: Analysis and improvements. In *INFOCOM (1)* (1998), pp. 252–262.

[5] Balakrishnan, H., Seshan, S., Stemm, M., and Katz, R. H. Analyzing Stability in Wide-Area Network Performance. In *ACM SIGMETRICS* (June 1997).

[6] CARTER, R., AND CROVELLA, M. Measuring bottleneck link speed in packet-switched networks. *Performance Evaluation*, 28 (1996), 297–318.

[7] CHINOY, B. Dynamics of internet routing information. In *SIGCOMM* (1993), pp. 45–52.

[8] DINDA, P. A. The statistical properties of host load. *Scientific Programming 7*, 3,4 (1999). A version of this paper is also available as CMU Technical Report CMU-CS-TR-98-175. A much earlier version appears in LCR '98 and as CMU-CS-TR-98-143.

[9] DINDA, P. A. Online prediction of the running time of tasks. *Cluster Computing 5*, 3 (2002). Earlier version in HPDC 2001, summary in SIGMETRICS 2001.

[10] DOVROLIS, C., RAMANATHAN, P., AND MOORE, D. What do packet dispersion techniques measure? In *INFOCOM* (2001), pp. 905–914.

[11] DOWNEY, A. B. Using pathchar to estimate internet link characteristics. In *Measurement and Modeling of Computer Systems* (1999), pp. 222–223.

[12] HU, N., AND STEENKISTE, P. Evaluation and characterization of available bandwidth probing techniques. *IEEE JSAC Special Issue in Internet and WWW Measurement, Mapping, and Modeling 21*, 6 (August 2003).

[13] JAIN, M., AND DOVROLIS, C. End-to-end available bandwidth: Measurement methodolody, dynamics, and relation with tcp throughput. In *ACM SIGCOMM* (2002).

[14] JAIN, M., AND DOVROLIS, C. Pathload: A measurement tool for end-to-end available bandwidth. In *Passive and Active Measurement Workshop* (2002).

[15] JIN, G., YANG, G., CROWLEY, B., AND AGARWAL, D. Network characterization service (ncs). In *10th IEEE Symposium on High Performance Distributed Computing, Aug. 2001.* (2001).

[16] KESHAV, S. A control-theoretic approach to flow control. *Proceedings of the conference on Communications architecture and protocols* (1993), 3–15.

[17] LAI, K., AND BAKER, M. Nettimer: A tool for measuring bottleneck link bandwidth. In *USENIX Symposium on Internet Technologies and Systems* (2001), pp. 123–134.

[18] LOWEKAMP, B., MILLER, N., SUTHERLAND, D., GROSS, T., STEENKISTE, P., AND SUBHLOK, J. A resource monitoring system for network-aware applications. In *Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing (HPDC)* (July 1998), IEEE, pp. 189–196.

[19] MATHIS, M., AND ALLMAN, M. A framework for defining empirical bulk transfer capacity metrics, rfc3148, July 2001.

[20] MATHIS, M., SEMKE, J., AND MAHDAVI, J. The macroscopic behavior of the tcp congestionavoidance algorithm. *Computer Communication Review 27*, 3 (1997).

[21] MYERS, A., DINDA, P. A., AND ZHANG, H. Performance characteristics of mirror servers on the internet. In *INFOCOM (1)* (1999), pp. 304–312.

[22] PAXSON, V. End-to-end routing behavior in the Internet. In *Proceedings of the ACM SIGCOMM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications* (New York, August 1996), vol. 26,4 of *ACM SIGCOMM Computer Communication Review*, ACM Press, pp. 25–38.

[23] PAXSON, V. End-to-end routing behavior in the Internet. *IEEE/ACM Transactions on Networking 5*, 5 (1997), 601–615.

[24] RIBEIRO, V., RIEDI, R., BARANIUK, R., NAVRATIL, J., AND COTTRELL, L. pathchirp: Efficient available bandwidth estimation for network paths. In *Passive and Active Measurement Workshop* (2003).

[25] SESHAN, S., STEMM, M., AND KATZ, R. H. SPAND: Shared passive network performance discovery. In *USENIX Symposium on Internet Technologies and Systems* (1997).

[26] STRAUSS, J., KATABI, D., AND KAASHOEK, F. A measurement study of available bandwidth estimation tools. In *Internet Measurement Conference* (2003).

[27] SWANY, M., AND WOLSKI, R. Multivariate resource performance forecasting in the network weather service. In *ACM/IEEE conference on Supercomputing* (2002).

[28] SWANY, M., AND WOLSKI, R. Representing dynamic performance information in grid environments with the network weather service. In *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)* (2002).

[29] VAZHKUDAI, S., AND SCHOPF, J. Predicting sporadic grid data transfers. In *12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12)* (2002).

[30] VAZHKUDAI, S., SCHOPF, J., AND FOSTER, I. Predicting the performance of wide area data transfers. In *The 16th Int'l Parallel and Distributed Processing Symposium (IPDPS 2002).* (2002).

[31] WOLSKI, R. Dynamically forecasting network performance using the network weather service. *Cluster Computing 1*, 1 (1998), 119–132.

[32] WOLSKI, R., SPRING, N., AND HAYES, J. The network weather service: A distributed resource performance forecasting service for metacomputing. *Journal of Future Generation Computing Systems 15*, 5-6 (1999), 757–768.

[33] YLONEN, T. SSH — secure login connections over the internet. In *Proceedings of the 6th USENIX Security Symposium* (1996), pp. 37–42.

[34] ZANGRILLI, M., AND LOWEKAMP, B. B. Comparing passive network monitoring of grid application traffic with active probes. In *Fourth International Workshop on Grid Computing* (2003).

[35] ZHANG, Y., BRESLAU, L., PAXSON, V., AND SHENKER, S. On the Characteristics and Origins of Internet flow rates. In *ACM SIGCOMM* (2002).

[36] ZHANG, Y., DUFFIELD, N., PAXSON, V., AND SHENKER, S. On the constancy of internet path properties. In *ACM SIGCOMM Internet Measurement Workshop* (November 2001).