Characterizing and Predicting TCP Throughput on the Wide Area Network

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

Abstract

DualPats exploits the strong correlation between TCP throughput and flow size, and the statistical stability of Internet path characteristics to accurately predict the TCP throughput of large transfers using active probing. We propose additional mechanisms to explain the correlation, and then analyze why traditional TCP benchmarking fails to predict the throughput of large transfers well. We characterize stability and develop a dynamic sampling rate adjustment algorithm so that we probe a path based on its stability. Our analysis, design, and evaluation is based on a large-scale measurement study.

1 Introduction

Application developers often pose an age-old question: "what is the *TCP throughput* of this path?" The question is more subtle than it appears. This paper motivates, describes, and evaluates DualPats, an algorithm and system for answering it. We define TCP throughput as $\frac{D}{T}$ where D is the flow size and T is the flow duration, starting at connection establishment and ending at teardown. For a file transfer [3, 39], D is the file size.

The *available bandwidth* of a path—the maximum rate of a new flow that will not reduce the rate of existing flows [15, 16]—has been thoroughly investigated. Keshav's packet pair [20], Crovella's cprobe [8], IGI [15], and spruce [33] attempt to measure the available bandwidth accurately, quickly, and non-intrusively. Other tools, such as nettimer [21], pathchar and pchar [13], pathload [16], NCS and pipechar [19], pathrate [12] and delphi [31] measure either the bottleneck link capacity or the available bandwidth.

The available bandwidth is different from the TCP throughput that an application can achieve, and that difference can be significant. For example, Jain's pathload paper [16] showed the bulk transfer capacity [25] of a path is higher than the measured available bandwidth, while Lai's Nettimer paper [21] showed many cases where the TCP throughput is much lower than their measurement. Jin, et al confirmed that available bandwidth is not an indicator of what an application can actually obtain [17]. Tools for estimating available bandwidth have a further issue: their slow convergence, on the order of at at least 10s of seconds, makes them too slow for many applications.

The most widely used real time TCP throughput prediction framework is the Network Weather Service [37] (NWS). NWS applies benchmarking techniques and time series models to measure TCP throughput and provide predictions to applications. NWS is widely used in grid computing and other application contexts.

Unfortunately, recent work [36, 35] has shown that NWS, and by implication, current TCP benchmarking techniques in general, have difficulty predicting the throughput of large file transfers on the high speed Internet. Sudharshan, et al [36] showed that NWS was predicting less than 1/10 of the actual TCP throughput achieved by GridFTP. In response, they proposed using a log of large file transfers to predict future file transfers. A key problem with this idea is that the log is updated only at application-chosen times, and thus changes in TCP throughput are only noticed after the application uses the path.

Taking dynamic changes into consideration, Sudharshan, et al [35] and Swany, et al [34] separately proposed regression and CDF-matching 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 accordingly. These techniques enhanced the accuracy of log based predictors. However, they remain limited to those host pairs that have logs of past transfers between them, and the logs must still be kept fresh. Zhang, et al [42] showed it is misleading to use history older than one hour. Furthermore, due to the strong correlation between TCP flow size and throughput [41], a log for one TCP flow size is not directly useful for predicting throughput for another.

The passive measurement approach [7, 19, 40] avoids the overhead of active probes by observing existing network traffic. Unfortunately, similar to the log-based prediction techniques, the passive approach is limited to periods of

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.

time when there is traffic on the network between the hosts of interest. Also, these systems measure available bandwidth, not TCP throughput.

There is an extensive literature on analytic TCP throughput models [26, 28, 4]. However, these models are limited in practice due to the difficulty in obtaining accurate model parameters such as TCP loss rate and RTT. Goyal et al [14] concluded that it is not easy to obtain accurate estimates of network loss rates as observed by TCP flows using probing methods, and that polling SNMP MIBs on the routers can do much better. However, because these statistics are aggregated and it is well known that TCP has a bias against connections with high RTT [30], this approach is limited to paths where the bottleneck router provides common loss rates, such as with RED. Furthermore, this approach has to determine the bottleneck router on the end-to-end path (a difficult problem) and have SNMP access to it (rarely available today).

DualPats follows from these questions, which we address via a large-scale measurement study:

- 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:

- Additional causes for the observed strong correlation between TCP flow size and throughput [42],
- A characterization of TCP throughput stability and statistics,
- A novel yet simple TCP benchmark mechanism,
- A dynamic sampling rate adjustment algorithm to lower active probing overhead, and
- DualPats and its evaluation.

2 Experimental Setup

Our experimental testbed includes PlanetLab and several additional machines located at Northwestern University and Argonne National Laboratory (ANL). PlanetLab [2] is an open platform for developing, deploying, and accessing planetary-scale services. It currently consists of more than 400 computers located at about 200 sites around the world.

We conducted four sets of experiments: Distribution Set, Correlation Set, Verification Set, and Online Evaluation Set. Each experiment set involves PlanetLab nodes from North America, Europe, Asia, and Australia. We set the TCP buffer size in a range from 1 to 3 MegaBytes in all the experiments with GridFTP, while we used default TCP buffer in other experiments. In the 2.4 Linux kernel, which we use, the socket buffer size is automatically tuned to approximately twice the estimated bandwidth delay product. More details of the experimental setup can be found elsewhere [23].

For each of Distribution Set and Correlation Set, we chose 40 nodes on PlanetLab spread across North America, Europe, Asia, and Australia. We randomly grouped those nodes into 20 pairs, each containing a client node and a server node and the Internet path between the two. A server listens and accepts incoming TCP connection requests from its client counterpart and transfers data of a particular size to the client through the established TCP connection. The client repeatedly connects to its server, requests some data, records the transfer time, and then closes the connection. To evaluate more Internet paths, we randomly changed pairings 3 times, resulting in 60 different paths for Distribution Set and another 60 for Correlation Set.

Distribution Set serves as a basis for evaluating TCP throughput stability and distributions. Since here we want to see how TCP throughput with a specific flow size varies with time and its distribution within each stable epoch, we transfer data of a particular size between pairs of clients and servers continuously for at least 3,000 times, then move on to perform the same operation on another data transfer with a different size. The trace data used in Distribution Set is mainly used in the discussion of TCP throughput stability and distributions in Section 4.

Correlation Set serves to study the strong correlation between TCP flow size and throughput, and to verify our TCP throughput benchmark mechanism, as discussed in Section 3. We define a *run* in Correlation Set and Verification Set as a procedure conducting a sequence of TCP transfers with increasing flow sizes between two hosts. For example, in Correlation Set, the client first requests 100 KB data, followed by a 200 KB request, then 400 KB, etc, up to 10 MB; this sequence forms a run. This lets us evaluate our TCP benchmark mechanism by predicting the transfer time of larger TCP transfers based on the transfer time of two smaller TCP flows and then comparing the predicted time with the actual transfer time of the larger transfers in the run. To guarantee fair evaluation, runs were repeated approximately 4,500 times between each pair of nodes, yielding the same number of TCP throughput predictions scattered at different times during our experiments. In total we have $\sim 270,000$ runs on ~ 60 paths.

Verification Set was done to further verify the correctness of our proposed TCP benchmark mechanism, and to strengthen analysis based on Distribution Set and Correlation Set with larger TCP flow sizes. Verification Set was conducted on twenty PlanetLab nodes, one node on Northwestern University campus and one node at ANL. We used GridFTP and scp in this set because both applications re-

quire authentication before transferring effective data. Our script transfered a series of files ranging from 5 KBytes to 1 GBytes in sequence and recorded each transfer time as the flow duration.

Online Evaluation Set serves to evaluate our DualPats real time TCP throughput prediction framework. We randomly choose fifty PlanetLab nodes, and do random pairing twice, resulting in 50 distinctive paths. We use DualPats to monitor the 50 paths for a duration of about 10 days. During the experiment we randomly send a file of size 8MB or 40MB or 160MB using scp as a test case to compare with the prediction result. Online Evaluation Set contains 14000 predictions.

3 Exploiting size / throughput correlation

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

Zhang, et al [41] 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 argue that the strong correlation between flow size and throughput is the most interesting one and explained it in the following ways.

Slow start: TCP slow start could cause some correlation between flow size and flow rate [41]. The distribution of TCP flow sizes follows a power law, which, in part, tells us that the majority of flows are short. Balakrishnan, et al [5] 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 bandwidth is correspondingly large [41].

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.

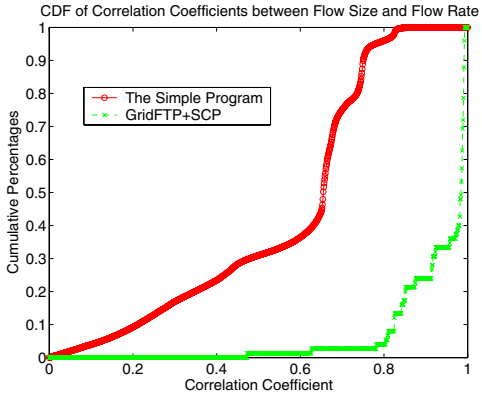

Figure 1. CDF of correlation coefficients R between flow sizes and throughput in experiments Correlation Set and Verification Set.

Let's consider the correlation between flow size and throughput in our experiments. Figure 1 gives the cumulative distribution functions (CDFs) of the correlation coefficient (Pearson's R), where each individual R value is calculated from one run of Correlation Set or Verification Set. The correlation between flow size and transfer time is large for the majority of transfers using our simple test program, and even larger for GridFTP and scp transfers.

3.2 Further explanations

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

Non-negligible 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 2 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 3 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. The linear

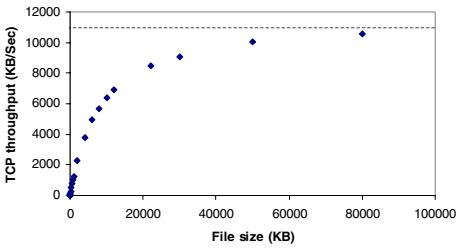

Figure 2. TCP throughput versus flow size (file size) with GridFTP. Transfers are between Northwestern University and Argonne National Lab. Single TCP flow with TCP buffer set. We made similar observations on all the other paths we studied.

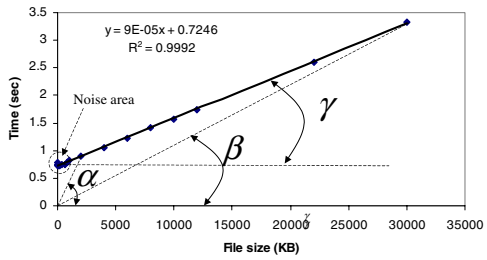

Figure 3. Transfer time versus TCP flow size with GridFTP. Transfers are between Northwestern University and Argonne National Lab. Single TCP flow with TCP buffer set. We made similar observations on all the other paths we studied.

model may not hold in the noise area.

A closer look at Figure 3 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. We represent this as

$$T = A \times x + B \tag{1}$$

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. $\frac{1}{A}$ is the steady state asymptotic TCP throughput in Figure 2.

Given Equation 1, we can easily deduce the expression for the TCP throughput in Figure 2 as

$$TP = \frac{x}{T} = \frac{x}{A \times x + B} \tag{2}$$

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

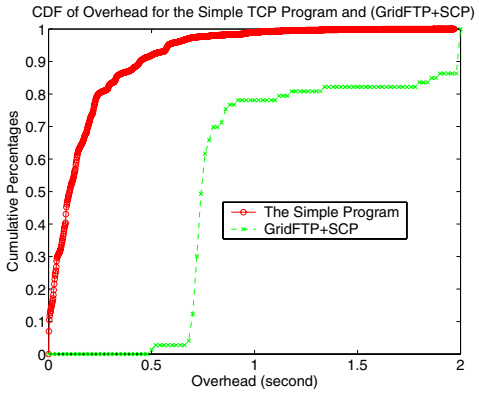

Figure 4. CDF of B, the startup overhead. Even for the simple client/server there is startup overhead likely caused by the residual slow start effect. The startup overheads of scp and GridFTP are much larger.

Residual slow start effect: Mathis, et al [26] 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 $\approx \sqrt{3/2}$. 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 rate or run out of receiver window. Zhang, et al [42] 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 1 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 Correlation Set. We did least square linear curve fitting and calculated B for each set of data. Figure 4 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.

3.3 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 [35]:

- The default probe used by NWS is too small. It will likely end up in the noise area as shown in Figure 3.
- 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 $cotangent(\alpha)$ is the TCP throughput for the flow size 2000KB, $cotangent(\beta)$ is the TCP throughput for the flow size 30000KB and $cotangent(\gamma)$ is the steady state TCP throughput. As file size increases α decreases, and when the file size is approaching infinity, the throughput will approach $cotangent(\gamma)$.
- The TCP buffer is not set for NWS probes while the GridFTP tests were done with adjusted buffer sizes.
- The usage of parallel TCP flows in GridFTP increases its aggregated throughput.

To verify that the linear model is valid for most Internet paths, Figure 5 shows the R^2 of the linear curve fitting for the data in Correlation Set and Verification Set. It is clear the model holds for both our simple client and server, and applications such as scp and GridFTP that require authentication.

3.4 A new TCP throughput benchmark mechanism

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 3. Using Equations 1 and 2, we can then calculate the TCP throughput for other flow sizes (file sizes).

To verify that the new technique works, we used the trace data in Correlation Set. 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 6 shows the CDF of relative prediction error for our results by flow size. $> 80\%$ of the prediction errors are below 20%.

The CDFs suggest that the relative prediction error may follow the normal distribution, so we used quantile-quantile plots to test this. In almost all cases, we can fit a straight

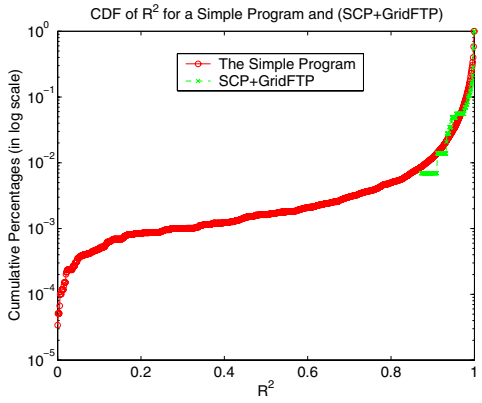

Figure 5. CDF of R^2 for linear model of Figure 3. Each R^2 is from a independent test. Both simple client/server and applications that require authentication show a strong linear property. Note that the Y axis is in log scale to show detail. Over 99% of the runs had $R^2 > 0.95$.

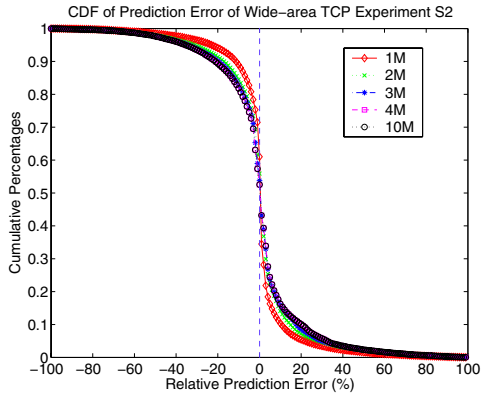

Figure 6. CDF of relative prediction error for TCP throughput with different flow sizes.

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 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 applications [11]. Time series based predictors can be applied to enhance the prediction accuracy, as covered in Section 5.

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 *dualPacket*.

As explained in Section 3.2 and 3.4, 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 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 two probe sizes of 400 KBytes and 800 KBytes as default probes, which we feel is a reasonable trade-off 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 optimal time series predictors.

4.1 Routing stability

Paxson [29] 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" [9]. Routing stability is the basis of other stabilities or consistency. If the route is changing frequently and quickly, then no other stabilities will hold.

4.2 Locality of TCP throughput

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 [6]. 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 [27]. Seshan, et al applied these findings in the development of the Shared Passive Network Performance Discovery (SPAND) system [32], 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 [42] 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 size, while SSR characterizes the steady state throughput for all flows with different sizes. We used traces from Correlation Set to characterize the SSR with steady-state TCP throughput. That is, instead of looking at the TCP throughput of a specific flow size, we estimated steady-state TCP throughput of the path using Equation 1.

Figure 7 gives the CDF of length of all SSRs modeled by steady-state TCP throughput from Correlation Set. 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. As we expected, the larger ρ is, the longer the SSRs tend to be.

For comparison purposes, we also calculated the CDF of OCR with data from Distribution Set. The comparison between ours and Zhang's results [42] 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

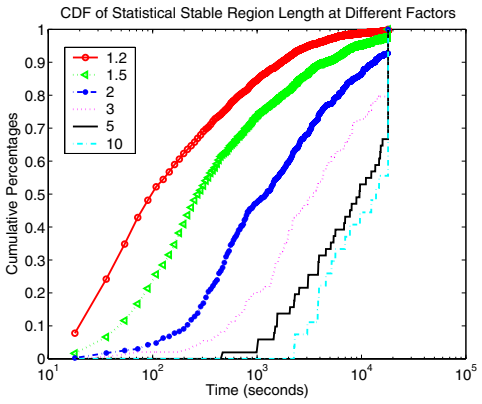

Figure 7. CDF of statistically stable region (SSR) for steady-state TCP throughput with different ρ.

fluctuations of TCP throughput. It is challenging to predict TCP throughput under a highly dynamic environment.

4.3 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. Balakrishnan, et al [6] 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. Zhang, et al verified this finding in [41].

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 Distribution Set lets us do this analysis.

Recall that in Distribution Set, for each client/server pair, we repeated the transfer of each file 3,000 times. We histogramed the throughput data for each flow size/path tuple. Almost in every case, the throughput histogram demonstrates a multimodal distribution. This suggests that it is probably not feasible to model long time 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 modeled 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 segment boundaries online). For each segment, we fit the data with several analytical distributions, and evaluate the goodness of fit using 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. The adjustment granularity can also be changed.
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 Distribution Set 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 8. It is clear that for the five distributions we compared, the normal distribution best fits the TCP throughput data. 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 [10].

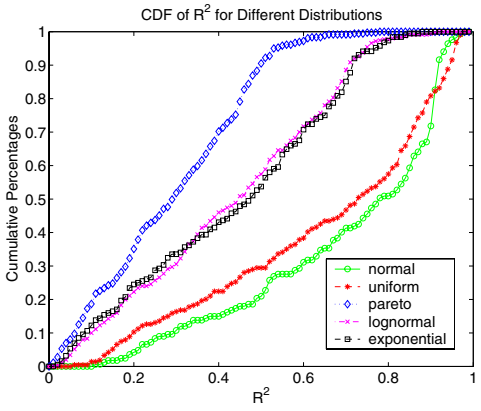

Figure 8. CDF of R^2 for five common distributions for TCP throughput characterization on segmented traces. The size of the file is 10 MBytes. Other flow sizes show similar results.

5 TCP throughput in real time

Based on our study and previous research, we have developed and evaluated DualPats, a prototype real time TCP throughput prediction service for distributed applications. DualPats actively sends out dualPackets to benchmark a path. It automatically adjusts its rate to capture the SSR on the path and therefore to minimize intrusiveness without losing sampling accuracy. The benchmarking technique is described in Section 3.

DualPats is different from all the previous available bandwidth estimation tools such as PathLoad [16]. First, instead of estimating current available bandwidth, DualPats predicts TCP throughput in the next short period of time. Secondly, DualPats monitors the paths and thus can return a prediction immediately. It takes DualPats less than 1 ms to give a prediction on a Pentium III machine, while it takes PathLoad tens of seconds to do one estimation for a path. DualPats don't send probes upon a prediction request, instead it monitors the path and sends out dualPackets according to the dynamic sampling rate adjustment algorithm as described below.

5.1 System architecture

DualPats consists of two components, a network sensor and a TCP throughput predictor. The network sensor sends out dualPackets at a self-adjusting rate as described in Section 5.2. It records the sizes and transfer times of each probe in each dualPacket. When monitoring N different TCP connections, N series of probe records are 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 executes the following:

1 Parse the query from the application and get parameters including the destination and file size.
2 Fetch the dualPacket data series for the destination from underlying network sensor. If no series exists, 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 for each of the two probes in the dualPacket.
4 Fit a linear curve as described in Equation 1 and calculate the TCP throughput for the given file size using Equation 2. (Optionally, compute a confidence interval using normality assumptions).
5 Return the estimated TCP throughput for the transfer time to the application.

We tested several prediction models for step 3, including interval-aware moving average (IAMA), exponential weighted moving average (EWMA) and simply using the last value. An IAMA is similar to a moving average except that the IAMA computes its average over only previous probe values with the same sampling interval. IAMA with window size 20 works best on average in our experiments. We believe that this is so because during each SSR, the end-to-end TCP throughput is best modeled with a normal distribution. For a normal distribution with no serial correlation, the mean is the best predictor possible, and IAMA estimates this.

5.2 Dynamic sampling rate adjustment algorithm

There are two ways to decrease the overhead caused by the dualPacket probes: decrease the sampling rate or decrease the size of the dualPacket.

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, instead of using the periodic sampling algorithm used by previous TCP throughput monitoring frameworks such as NWS [37], we designed a simple algorithm to dynamically adjust the sampling rate to the path's SSR. For stable paths with longer SSR, we send fewer probes, while for unstable paths with shorter SSR, we adapt to its dynamics by sampling the path more frequently. The algorithm is:

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 dualPacket, 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 Online Evaluation Set, $B1$, $B2$ were set to be 5% and 15%.

3 The interval must be between L and U.

We also want to minimize the size of dualPacket on the condition that none of them will fall into the noise area as shown in Figure 3. 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. Our algorithm for determining it is:

1 Set a default initial size for the dualPackets. In Online Evaluation Set, we used 400KB and 800KB. Also set an upper bound size U_S for the dualPacket.

2 If M continuous prediction errors are bigger than a threshold T_H, and with the same sign, we increase the probe size by 100KB each.

3 The size of dualPacket must be $\leq U_S$.

5.3 Evaluation

Our primary metric is the relative error:

$$err = \frac{PredValue - RealValue}{RealValue} \quad (3)$$

DualPats ran ≈ 14000 predictions on 50 monitored end-to-end paths during about 10 days. Test cases are randomly chosen 8MB, 40MB, or 160MB files. Details of the evaluation experiments can be found in the Section 2 discussion of Online Evaluation Set.

Our detailed results for each path are available elsewhere [23]. To summarize them for this paper, we use the following metrics. 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 roughly equal proportion of positive and negative mean errors. The mean $abs(err)$ is the average of the absolute value of the relative error. We consider it the most important metric in evaluating the predictions. Mean stderr is the standard deviation of relative error while mean $abs(stderr)$ is the standard deviation of the absolute value of relative error.

DualPats is accurate: Figure 9 shows the CDF of the mean error and mean $abs(err)$ of our results. Figure 10 shows the CDF of the standard deviation for relative errors. Over 70% of the predictions have mean error within [-0.1, 0.1], and all of them are within [-0.21, 0.22]. About 90% of the predictions have mean stderr smaller than 0.2. About 30% of the predictions have mean $abs(err)$ within 0.1, while over 90% of the predictions has mean $abs(err)$ below 0.2, and about 95% of them are below 0.25. Over

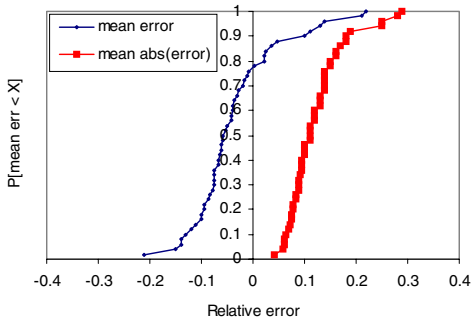

Figure 9. CDF of relative errors.

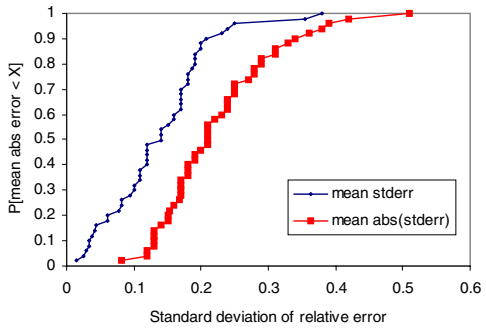

Figure 10. CDF of standard deviation of relative errors.

90% of the predictions have mean $abs(stderr)$ smaller than 0.35.

We studied the correlation among the prediction errors and several known path properties. The results are shown in Figure 11. We define $(|R| > 0.3)$ as being weakly correlated, $(0.3 \leq |R| \leq 0.8)$ being medium correlated, and $(|R| > 0.8)$ being strongly correlated. Clearly, the mean error is not related to any others, which further suggests that the predictions given by DualPats are unbiased. However, if the path is very dynamic it is hard to predict. Figure 11 shows that R between the mean absolute error and the sampling interval length (and, indirectly, the SSR) is negatively and very weakly correlated. This implies that our algorithm captured the path dynamics and effectively adjusted to its changes. The mean interval and mean standard deviation of error show the strongest correlation in Figure 11. This is because both longer mean interval and smaller mean standard deviation of error imply a stable path. Also, we can see that number of hops is weakly correlated with mean RTT.

Recall from Section 1 that Sudharshan, et al [36, 35] showed that NWS was predicting less than 1/10 of the actual TCP throughput achieved by GridFTP with large file transfers. Our evaluations show that DualPats does an effective job of predicting TCP throughput for large transfers.

	Router Hops	Mean RTT	Mean Interval
Mean abs(err)	0.24	-0.024	-0.36
Mean abs(stderr)	0.19	-0.018	-0.28
Mean err	0.10	-0.081	-0.20
Mean stderr	0.30	-0.031	-0.44
Router Hops	1.00	0.34	-0.27
Mean RTT	0.34	1.00	-0.25
Mean Interval	-0.27	-0.25	1.00

Figure 11. Correlation coefficient R among prediction error and path properties.

Also recall that log-based prediction techniques [34, 35] work only for the host pairs that have recent data exchange history, and the data to be sent is of similar sizes as in the log. Our conclusion is that DualPats achieves comparable or even better performance without such constraints.

Our evaluation is conservative: Jin, et al showed that end-system capability can have significant effects on the network bandwidth estimation algorithms [18]. They showed that resolution of the timer, the time to perform a system call, the interrupt delay and the system I/O bandwidth all can affect network bandwidth estimation. They compared packet pair dispersion against packet train based algorithm and concluded that packet train based algorithms are less sensitive to the resolution of the system timer and less affected by I/O interrupt delays. This implies that high system load has negative effects on bandwidth estimation algorithms.

We ran the experiments of Online Evaluation Set on PlanetLab, which is typically heavy loaded. Using data available from the CoDeeN project web site [1], we found that CPU load averages were very high on the machines we used. 50% of the machines had Unix load averages that exceeded 5.0. Recall that in Section 4 we compared our measured OCR with that shown by Zhang, et al [42], and found that TCP throughput on PlanetLab is more dynamic than Zhang found. This suggests that our prediction would probably do better in a more typical, lightly loaded environment, and that DualPats is robust in the face of high load conditions. We speculate that the robustness of DualPats is related to the TCP benchmarking approach being used: we are measuring the application-to-application transfer time, and thus our prediction must necessarily incorporate the end-system behavior as well.

DualPats overhead is low: The overhead of DualPats mainly comes from the dualPackets sent. In our current implementation, we use the default 800KB probe sent at the rate controlled by our dynamic sampling rate adjustment algorithm resulting in an overhead on the network of 800KB / (mean Interval). In a highly dynamic testbed like PlanetLab, close to 30% of the mean intervals achieved by DualPats are longer than 500 seconds (800KB/500Sec=1.6KB/Sec), close to 50% are longer than 180 seconds (800KB/180Sec=4.4KB/Sec), and about 90% are longer than 30 seconds (800KB/30Sec=26.7KB/Sec). The interval is bounded by our protocol to limit the maximum overhead on the network.

As DualPats is designed mainly for today's high speed networks, we believe the overhead of DualPats is reasonably small. For less dynamic paths, the overhead of DualPats will be further reduced because the dynamic sampling rate adjustment algorithm will automatically increase the sampling interval. In contrast, Strauss, et al [33] reported that Pathload generates between 2.5 and 10 MB of probe traffic per measurement, which is much larger than that of DualPats. The amount of data sent by DualPats is comparable or smaller than current available bandwidth estimators.

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, dualPacket, from which TCP throughput for different flow sizes can be derived. We described and evaluated the performance of a new TCP throughput monitoring and prediction framework, DualPats, and implemented this approach. We have recently extended our work to support throughput prediction for parallel TCP [24].

Like all benchmarking-based systems, our approach has scalability limits. We have addressed this to some extent with our dynamic sample rate adjustment algorithm. However, we are also considering combining our techniques with passive monitoring as in Wren [40], and hierarchical decomposition as in Remos [22] and NWS Clique [38]. Our evaluation of DualPats was in a conservative, heavily loaded environment, but its performance and robustness in the next generation high speed Internet are yet to be explored. An implementation of DualPats will be made available from http://plab.cs.northwestern.edu/Clairvoyance.

References

[1] http://codeen.cs.princeton.edu.

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

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

[4] E. Altman, K. Avrachenkov, and C. Barakat. A stochastic model of TCP/IP with stationary random. In *ACM SIGCOMM*, pages 231–242, 2000.

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

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

[7] J. Bolliger, T. Gross, and U. Hengartner. Bandwidth modeling for network-aware applications. In *INFOCOM (3)*, pages 1300–1309, 1999.

[8] R. Carter and M. Crovella. Measuring bottleneck link speed in packet-switched networks. *Performance Evaluation*, (28):297–318, 1996.

[9] B. Chinoy. Dynamics of internet routing information. In *SIGCOMM*, pages 45–52, 1993.

[10] P. A. Dinda. 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.

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

[12] C. Dovrolis, P. Ramanathan, and D. Moore. What do packet dispersion techniques measure? In *INFOCOM*, pages 905–914, 2001.

[13] A. B. Downey. Using pathchar to estimate internet link characteristics. In *Measurement and Modeling of Computer Systems*, pages 222–223, 1999.

[14] M. Goyal, R. Guerin, and R. Rajan. Predicting tcp throughput from non-invasive network sampling. In *IEEE INFOCOM*, 2002.

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

[16] M. Jain and C. Dovrolis. End-to-end available bandwidth: Measurement methodolody, dynamics, and relation with tcp throughput. In *ACM SIGCOMM*, 2002.

[17] G. Jin and B. Tierney. Netest: A tool to measure maximum burst size, available bandwidth and achievable throughput. In *International Conference on Information Technology*, 2003.

[18] G. Jin and B. L. Tierney. System capability effects on algorithms for network bandwidth measurement. In *ACM SIGCOMM conference on Internet measurement*, 2003.

[19] G. Jin, G. Yang, B. Crowley, and D. Agarwal. Network characterization service (ncs). In *10th IEEE Symposium on High Performance Distributed Computing, Aug. 2001.*, 2001.

[20] S. Keshav. A control-theoretic approach to flow control. *Proceedings of the conference on Communications architecture and protocols*, pages 3–15, 1993.

[21] K. Lai and M. Baker. Nettimer: A tool for measuring bottleneck link bandwidth. In *USENIX Symposium on Internet Technologies and Systems*, pages 123–134, 2001.

[22] B. Lowekamp, N. Miller, D. Sutherland, T. Gross, P. Steenkiste, and J. Subhlok. A resource monitoring system for network-aware applications. In *Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing (HPDC)*, pages 189–196. IEEE, July 1998.

[23] D. Lu, Y. Qiao, P. Dinda, and F. Bustamante. Characterizing and predicting tcp throughput on the wide area network. Technical Report NWU-CS-04-34, Northwestern University, Computer Science Department, April 2004.

[24] D. Lu, Y. Qiao, P. Dinda, and F. Bustamante. Modeling and taming parallel tcp on the wide area network. In *Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS 05)*, April 2005.

[25] M. Mathis and M. Allman. A framework for defining empirical bulk transfer capacity metrics, rfc3148, July 2001.

[26] M. Mathis, J. Semke, and J. Mahdavi. The macroscopic behavior of the tcp congestionavoidance algorithm. *Computer Communication Review*, 27(3), 1997.

[27] A. Myers, P. A. Dinda, and H. Zhang. Performance characteristics of mirror servers on the internet. In *INFOCOM (1)*, pages 304–312, 1999.

[28] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling tcp throughput: A simple model and its empirical validation. In *ACM SIGCOMM*, 1998.

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

[30] L. Qiu, Y. Zhang, and S. Keshav. On individual and aggregate TCP performance. In *ICNP*, pages 203–212, 1999.

[31] V. Ribeiro, M. Coates, R. Riedi, S. Sarvotham, B. Hendricks, and R. Baraniuk. Multifractal cross-traffic estimation, 2000.

[32] S. Seshan, M. Stemm, and R. H. Katz. SPAND: Shared passive network performance discovery. In *USENIX Symposium on Internet Technologies and Systems*, 1997.

[33] J. Strauss, D. Katabi, and F. Kaashoek. A measurement study of available bandwidth estimation tools. In *Internet Measurement Conference*, 2003.

[34] M. Swany and R. Wolski. Multivariate resource performance forecasting in the network weather service. In *ACM/IEEE conference on Supercomputing*, 2002.

[35] S. Vazhkudai and J. Schopf. Predicting sporadic grid data transfers. In *12th IEEE International Symposium on High Performance Distributed Computing (HPDC-12)*, 2002.

[36] S. Vazhkudai, J. Schopf, and I. Foster. Predicting the performance of wide area data transfers. In *The 16th Int'l Parallel and Distributed Processing Symposium (IPDPS 2002).*, 2002.

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

[38] R. Wolski, N. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Journal of Future Generation Computing Systems*, 15(5-6):757–768, 1999.

[39] T. Ylonen. SSH — secure login connections over the internet. In *Proceedings of the 6th USENIX Security Symposium*, pages 37–42, 1996.

[40] M. Zangrilli and B. B. Lowekamp. Comparing passive network monitoring of grid application traffic with active probes. In *Fourth International Workshop on Grid Computing*, 2003.

[41] Y. Zhang, L. Breslau, V. Paxson, and S. Shenker. On the Characteristics and Origins of Internet flow rates. In *ACM SIGCOMM*, 2002.

[42] Y. Zhang, N. Duffield, V. Paxson, and S. Shenker. On the Constancy of Internet path properties. In *ACM SIGCOMM Internet Measurement Workshop*, 2001.