Modeling and Taming Parallel TCP on the Wide Area Network

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

***Abstract*— Parallel TCP flows are broadly used in the high performance distributed computing community to enhance network throughput, particularly for large data transfers. Previous research has studied the mechanism by which parallel TCP improves aggregate throughput, but there doesn't exist any practical mechanism to predict its throughput. In this work, we address how to predict parallel TCP throughput as a function of the number of flows, as well as how to predict the corresponding impact on cross traffic. To the best of our knowledge, we are the first to answer the following question on behalf of a user: what number of parallel flows will give the highest throughput with less than a $p\%$ impact on cross traffic? We term this the maximum nondisruptive throughput. We begin by studying the behavior of parallel TCP in simulation to help derive a model for predicting parallel TCP throughput and its impact on cross traffic. Combining this model with some previous findings we derive a simple, yet effective, online advisor. We evaluate our advisor through simulation-based and wide-area experimentation.**

I. Introduction

Data intensive computing applications require efficient management and transfer of terabytes of data over wide area networks. For example, the Large Hadron Collider (LHC) at the European physics center CERN is predicted to generate several petabytes of raw and derived data per year for approximately 15 years starting from 2005 [6]. Data grids aim to provide the essential infrastructure and services for theses applications, and a reliable, high-speed data transfer service is a fundamental and critical component.

Recent research has demonstrated that the actual TCP throughput achieved by applications is, persistently, significantly smaller than the physical bandwidth "available" according to the end-to-end structural and load characteristics of the network [39], [26]. Here, we define *TCP throughput* as the ratio of effective data over its transfer time, also called *goodput* [35].

Parallel TCP flows have been widely used to increase throughput. For example, GridFTP [5], part of the Globus project [16], supports parallel data transfer and has been widely used in computational grids [6], [26].

A key challenge in using parallel TCP is determining the number of flows to use for a particular transfer. This number affects both the throughput that the transfer will achieve and the impact that it will have on other traffic sharing links with these data flows. While there has been significant previous work on the understanding of parallel TCP performance, no practical parallel TCP throughput prediction techniques exist and there is no analysis work or system that can support the following API call:

```
struct ParallelTCPChar {
  int    num_flows;
  double max_nondisruptive_thru;
  double cross_traffic_impact;
};

ParallelTCPChar *
TameParallelTCP(Address dest,
                double  maximpact);
```

Here, the user calls `TameParallelTCP()` with the destination of her transfer and the maximum percentage impact she is willing to have on cross traffic. The call evaluates the path and returns the number of parallel flows she should use to achieve the maximum possible throughput, while causing no more impact than the specified. We refer to this as the *maximum nondisruptive throughput (MNT)*.

The following sections address the implementation of such a function. With this in mind, we look for answers to the following questions:

- How does parallel TCP affect the throughput of the user's transfer, the throughput of cross traffic, and the combined aggregate throughput, in different scenarios?
- How can these throughputs be predicted, online and with a small set of measurements, as functions of the number of parallel TCP flows?
- How can these predictions be used to implement the `TameParallelTCP()` function?

To the best of our knowledge, we are the first to propose a practical mechanism to predict the throughput of parallel TCP flows and to answer `TameParallelTCP()`-like questions by estimating the impact on the cross traffic.

We begin by reviewing related work in Section II. In Section III we analyze parallel TCP throughput under different scenarios via simulations. We derive a prediction model for parallel TCP throughput and present results from an extensive Internet-based evaluation in Section IV. In Section V we outline a simple algorithm to estimate the effect of parallel TCP on cross traffic as a function of the number of flows. We

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).

evaluate our algorithm through simulation and later combine it with our approach to throughput prediction in order to implement the `TameParallelTCP()` call. Section VI presents our conclusions.

Throughout the paper, we use "parallelism level" interchangeably with "the number of parallel TCP flows".

A version of our `TameParallelTCP()` implementation is available from http://plab.cs.northwestern.edu/Clairvoyance/Tame.html.

II. Related work

The available bandwidth of a path is defined as "the maximum rate that the path can provide to a flow, without reducing the rate of the rest of the traffic." [21], [22]. Available bandwidth has been a central topic of research in packet networks over the years. To measure it accurately, quickly, and non-intrusively, researchers have developed a variety of algorithms and systems. Tools that measure either the bottleneck link capacity or the available bandwidth include cprobe [9], Remos [27], pathload [22], [23], NCS, and pipechar [24], among others [25], [11], [10], [41], [37], [21]. Most of these tools use packet pair or packet train techniques to conduct the measurements and typically take a long time to converge.

Previous research [25] has shown that, in most cases, the throughput that TCP achieves is considerably lower than the available bandwidth. Parallel TCP is one response to this observation. Sivakumar et al. [39] present PSockets, a library that stripes data over several sockets and evaluate its performance through wide-area experimentation. The authors concluded that this approach can enhance TCP throughput and, in certain situations, be more effective than tuning the TCP window size. Allcock et al. [6] evaluate the performance of parallel GridFTP data transfers on the wide-area, and applied GridFTP to the data management and transfer service in Grid environments.

Considerable effort has been spent on understanding the aggregate behavior of parallel TCP flows on wide area networks. Shenker et al [38] were first to point out that a small number of TCP connections with the same RTT and bottleneck can get their congestion window synchronized. Qiu et al. [35] studied the aggregate TCP throughput, goodput and loss probability on a bottleneck link via extensive ns2-based simulations. The authors found that a large number of TCP flows with the same round trip time (RTT) can also become synchronized on the bottleneck link when the average size of each TCP congestion window is larger than three packets. A detailed explanation for this synchronization was given in [35]. Therefore the parallel TCP flows will share the same loss rate and therefore the same throughput.

The work most relevant to ours is that of Hacker et al [18]. The authors observe that parallel TCP increases aggregate throughput by recovering faster from a loss event when the network is not congested. The authors go on to propose a theoretical model for the upper bound of parallel TCP throughput for an uncongested path. The model produces a tight upper bound only if the network is not congested before and after adding the parallel TCP flows. In this case the aggregated throughput increases linearly with the number of parallel TCP flows.[1] Clearly this reduces the utility of the model as networks often congested to different degrees. In Section III, we conducted simulations in six different scenarios, and only in two scenarios (Figures 7 and 8) can their model be applied.

Hacker et al also concluded that, in the absence of congestion, the use of parallel TCP flows is equivalent to using a large MSS on a single flow, with the added benefit of reducing the negative effects of random packet loss. They advise application developers not to use an arbitrary large number of parallel TCP flows, but conclude that it is difficult, if not impossible, to determine the point of congestion in the end-to-end path a priori, and therefore to decide on the proper number of parallel TCP flows.

Most TCP throughput models have limited practical utility due to the difficulty of obtaining accurate model parameters such as TCP loss rate and RTT. For example, Goyal et al [17] concluded that it is hard 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, the MIB statistics are for the aggregate traffic crossing a interface on the router while it is well-known that TCP has a bias against long round trip time connections [35]. Hence each TCP flow will have a different loss rate unless a RED-like queue management policy is used on the bottleneck router. The SNMP MIB approach is thus limited to those paths where the bottleneck router is using RED. It is also necessary in this and similar approaches to determine the bottleneck router on the end-to-end path (a difficult problem) and have SNMP access to it (rarely available today). Even if this is possible, with current models for parallel TCP we would have to know the loss rate *after* adding in n parallel TCP flows. However, even with the tools like web100 [29], [4], we can not obtain this rate by simply measuring the network.

Our work makes the following new contributions to the state of the art:

- We predict throughput for *both* congested and uncongested paths as a function of the level of parallelism.
- We estimate the impact of parallel TCP on cross traffic as a function of the level of parallelism.
- We do so using only a small number of probes and no additional tools.

It is widely believed that, under congested situations, parallel TCP flows achieve better performance by effectively behaving unfairly, stealing bandwidth from cross traffic. This has prompted some researchers to propose modifying TCP in order to make it better suited for parallel transfers by considering both efficiency and fairness [12], [19], [20]. We believe it will be difficult to persuade people to modify their TCP implementations just to use parallel TCP more fairly. By relying on our prediction tools, a user or administrator should be able to trade off a transfer's throughput and its

[1] Their model can also be applied if the loss rate of each TCP flow is known a priori. However, their work does not provide any mechanisms to obtain the a priori loss rate in practice.

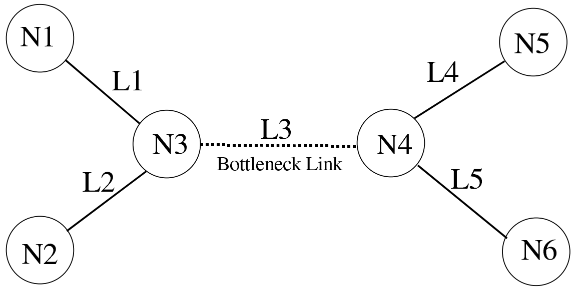


Fig. 1. Topology for simulations. Cross traffic goes from node N1 to N5, while parallel TCP flows go from node N2 to N6. Cross traffic and parallel TCPs share the same bottleneck link L_3. Each simulation lasts 100 seconds with individual TCP cross traffic flows starting randomly during the first 8 seconds, and all parallel TCPs starting simultaneously at time 10 sec.

degree of impact on cross traffic, achieving what we refer to as the *maximum nondisruptive throughput (MNT)*. All these are at application level without requiring modifications to pre-existing TCP implementations.

III. Analyzing parallel TCP throughput

In this section, we use simulation to understand the behavior of parallel TCP under different scenarios. For all our simulation-based studies we make use of the ns2 network simulator [2].

A. Simulation Setup

In a simulation study on aggregate TCP throughput on a bottleneck link, Qiu et al. [35] developed a simple yet realistic topology model for wide-area Internet connections based on the Internet hierarchical routing structure (Figure 1). We adopt this same topology for our simulations. Each simulation is 100 seconds long, with cross traffic randomly starting during the first 8 seconds and parallel TCP flows all starting at 10 seconds into the simulation. Cross traffic goes from N1 to N5, while parallel TCP flows go from N2 to N6. The bottleneck link is L3. We employ TCP Reno [13] for both cross traffic and parallel TCP flows, as this implementation makes use of the most widely deployed TCP congestion control algorithm. [2] Both DropTail and Random Early Detection (RED) [15] queue management policies are studied as they are the most commonly used queue management policies on the Internet. DropTail and RED have similar performance in most our simulations. The exception is in Scenario 1. Here, when there are more than 10 cross traffic flows, the cross traffic dominates the queue and starves the parallel TCP flows under DropTail policy. Unless otherwise noted, we show results for the DropTail policy.

We use TCP flows as cross traffic because of TCP's dominance in the current Internet, as reported in the the work by Smith et al. [40], in which TCP accounted for 90-91% of the packets and about 90-96% of the bytes transferred in traces collected in 1999-2000 from a educational institution (UNC) and a research lab (NLANR).

[2]Comparable results were obtained using TCP Tahoe.

We analyze Parallel TCP throughput under a variety of representative scenarios including a typical slow connection such as cable or DSL (Scenario 1), a coast-to-coast high-speed Internet connection (Scenario 2) and a current (Scenario 3) and next generation global-scale Internet connections (Scenario 4). Two additional scenarios (Scenarios 5 and 6) are used to represent cases where the TCP buffer has not been appropriately tuned [42]. Figure 2 summarizes the different simulation scenarios. For each scenario, we simulate from 1 to 31 parallel TCP flows with 5, 10, 15, 20, 25 and 30 random TCP cross traffic flows.

B. Simulation results

Figures 3 to 8 plot the aggregated throughput of parallel TCP as a function of the number of flows used for the different scenarios. Plots are shown both without (left graph) and with (right graph) cross traffic. In the latter case, we also plot the cross traffic's and total throughput, i.e. the sum of both the parallel TCP and cross traffic throughputs.

Figure 3 shows our results for Scenario 1, used to represent a typical slow connection. We show five cross traffic flows in this case. It is clear from the graphs that, with such a low-latency/low-bandwidth connection, the primary benefit from parallel TCP comes from being able to steal bandwidth from the existing cross traffic.

The results for Scenario 2, representing a current coast-to-coast connection with low latency and medium bandwidth, are shown in Figure 4. As it can be seen from the plots, there are some limited benefits from using parallel TCP without competition in this scenario. In the presence of cross traffic, however, parallel TCP is an even stronger competitor. Notice also how parallel TCP allows us to increase overall throughput, albeit marginally.

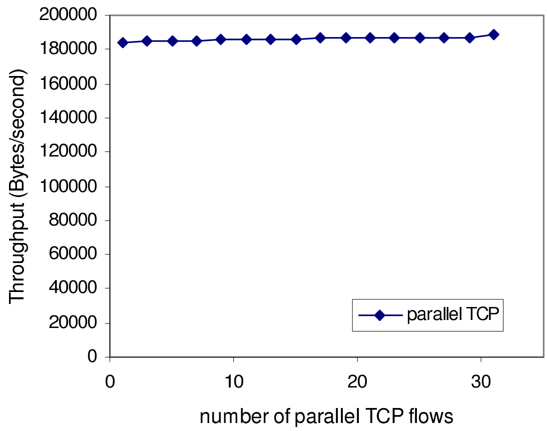
Figure 5 illustrates the benefits of parallel TCP in Scenario 3, a long latency, medium bandwidth link representing a current global-scale, fast Internet connection. In this case there are significant benefits to using parallel TCP even in the absence of cross traffic. The differences in the performance of parallel TCP under scenarios 2 and 3, without cross traffic, can be explained using Hacker's theory [18]: parallel TCP recovers faster than single TCP when there is a time out. This effect is more important as the RTT increases, because the time out will be longer and a single TCP cannot recover fast enough.

The benefits of using parallel TCP, with and without cross traffic, are very clear under Scenario 4 as Figure 6 shows. The additional throughput in the presence of cross traffic, is mainly due to the increase in overall throughput.

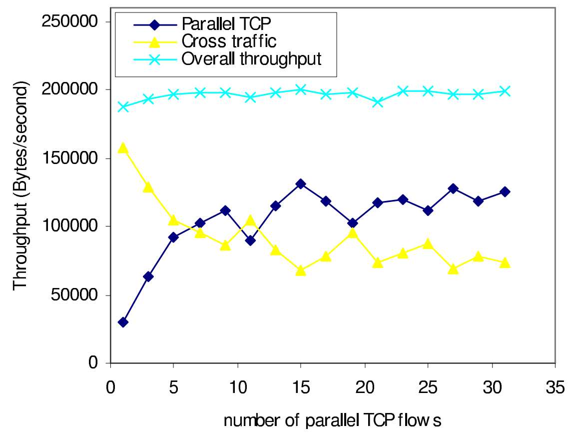
The advantage of parallel TCP is even more significant in the two scenarios representing mistuned TCP buffers. Figure 7 shows this advantage for Scenario 5, a high bandwidth and high latency link with a small socket buffer size. The benefits of parallel TCP are quite obvious, regardless of the amount of cross traffic. Furthermore, these gains come at no cost to the existing cross traffic. Parallel TCP gains performance not only by recovering faster after a time out, but also by providing an effectively larger buffer size. Note that the throughput of

Scenario	L_3 latency	L_3 Bandwidth	L_1,L_2 Bandwidth	L_4,L_5 Bandwidth	TCP buffer
1	20 ms	1.5 Mbps	10 Mbps	10 Mbps	$\geq$ Bandwidth*RTT
2	20 ms	100 Mbps	1000 Mbps	1000 Mbps	$\geq$ Bandwidth*RTT
3	50 ms	100 Mbps	1000 Mbps	1000 Mbps	$\geq$ Bandwidth*RTT
4	50 ms	1000 Mbps	10000 Mbps	10000 Mbps	$\geq$ Bandwidth*RTT
5	50 ms	1000 Mbps	10000 Mbps	10000 Mbps	60 KB
6	20 ms	100 Mbps	1000 Mbps	1000 Mbps	60 KB

Fig. 2. Bandwidth and latency configuration for different scenarios. The latency for L_1 and L_2 is fixed at 4 milliseconds, while the latency for L_4 and L_5 is fixed at 5 milliseconds. The buffer size on each node is fixed at 25 packets. Both DropTail and RED queue management policies are simulated.

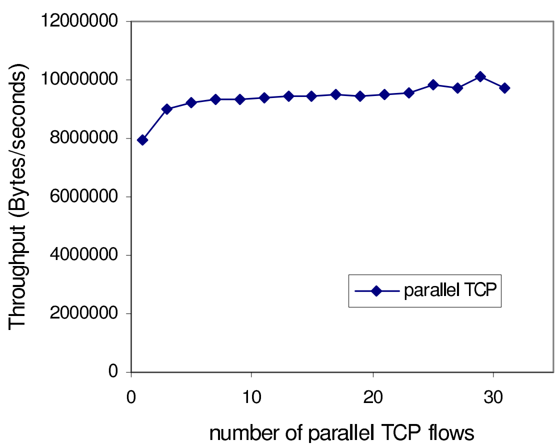


(a) No cross traffic

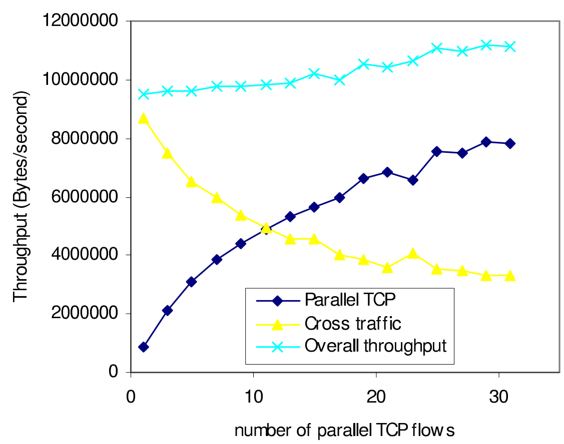


(b) 5 cross traffic

Fig. 3. Simulation results for scenario 1: latency of L_3 is 20 ms; bandwidth of L_3 is 1.5 Mbps; TCP buffer is properly tuned. Refer to Figure 2 for details.



(a) No cross traffic



(b) 10 cross traffic

Fig. 4. Simulation results for scenario 2: latency of L_3 is 20 ms; bandwidth of L_3 is 100 Mbps; TCP buffer is properly tuned. Refer to Figure 2 for details.

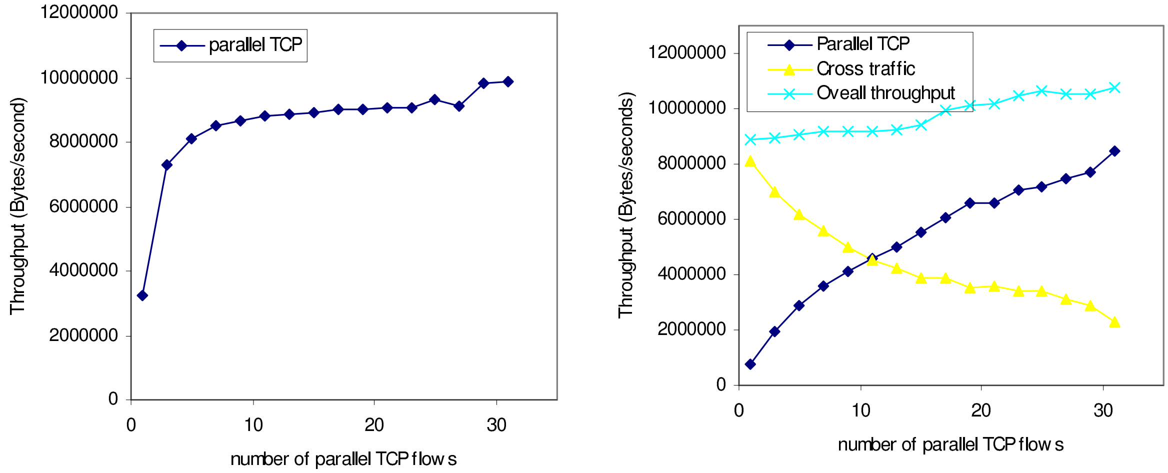
parallel TCP flows eventually flatten out as more flows are added into the simulation.

Similar benefits from parallel TCP can be observed in our last scenario (Figure 8). As with Scenario 5, parallel TCP can significantly improve throughput regardless of the degree of traffic. In this case, the impact on cross traffic increases with increasing parallelism, but remains relatively flat.

C. Observations

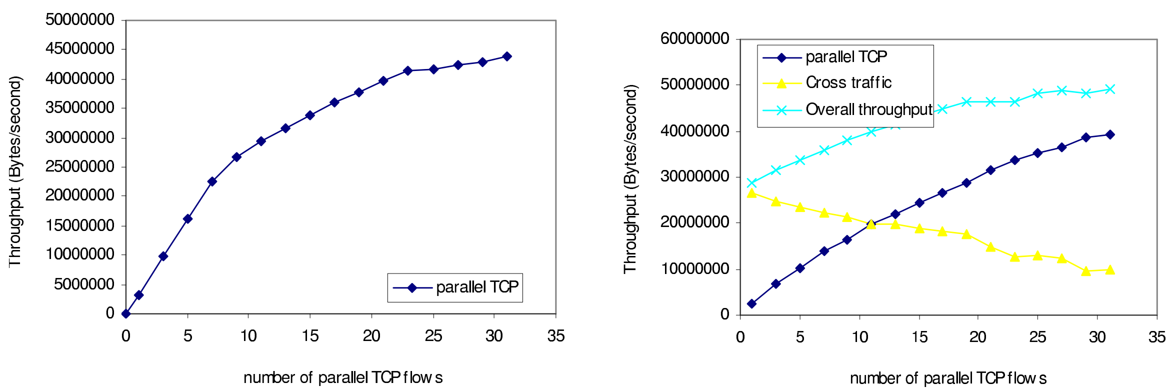
The dramatically different behaviors shown in the previous section clearly illustrate the challenges in providing a sound `TameParallelTCP()`-like call. The parallel TCP and cross traffic throughput curves adopt a wide range of forms, depending on the topology of the network and the configuration of endpoints. In addition, even if one were to disregard the almost prohibitively high costs of directly measuring these curves, the cross traffic impact would be very difficult to determine.

Without a priori knowledge of the parallel TCP loss rate, the model proposed by Hacker, et al [18] only works in uncongested networks like our Scenario 5. In contrast, our model probes the network with 2 probes with different parallelism levels and estimates the loss rate at other parallelism levels. This allows it to be applied in both congested and uncongested networks. In Section IV and Section V we evaluated the effectiveness of our model via large scale wide area experiments



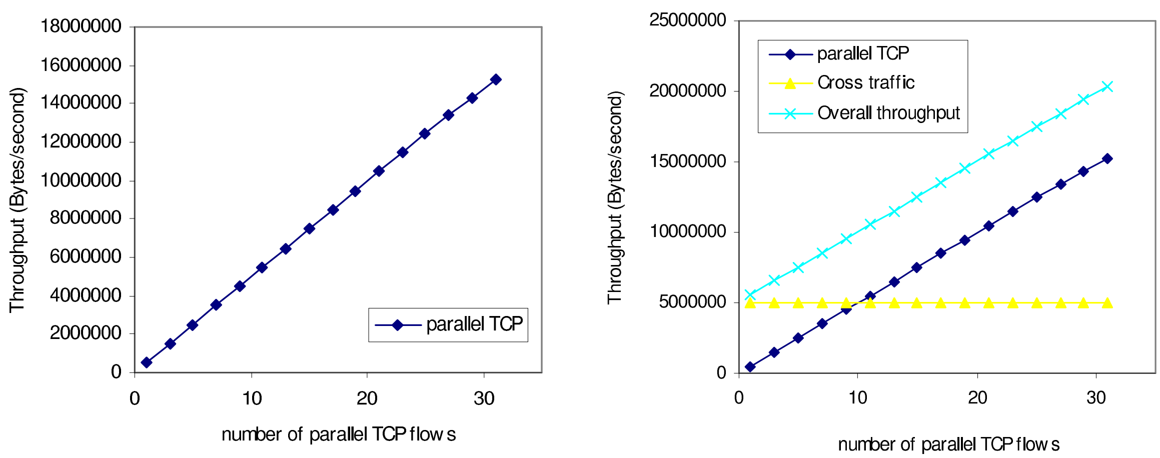
(a) No cross traffic (b) 10 cross traffic

Fig. 5. Simulation results for scenario 3: latency of L_3 is 50 ms; bandwidth of L_3 is 100 Mbps; TCP buffer is properly tuned. Refer to Figure 2 for details.



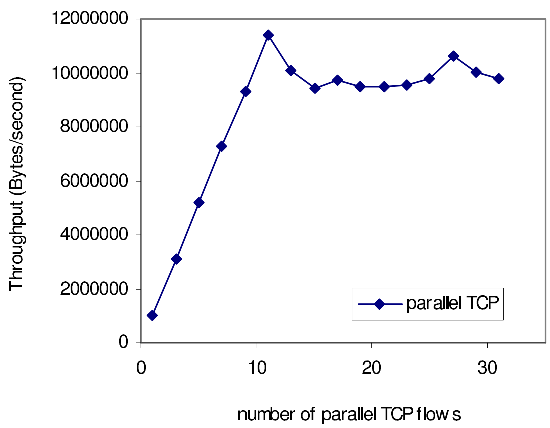
(a) No cross traffic (b) 10 cross traffic

Fig. 6. Simulation results for scenario 4: latency of L_3 is 50 ms; bandwidth of L_3 is 1000 Mbps; TCP buffer is properly tuned. Refer to Figure 2 for details.

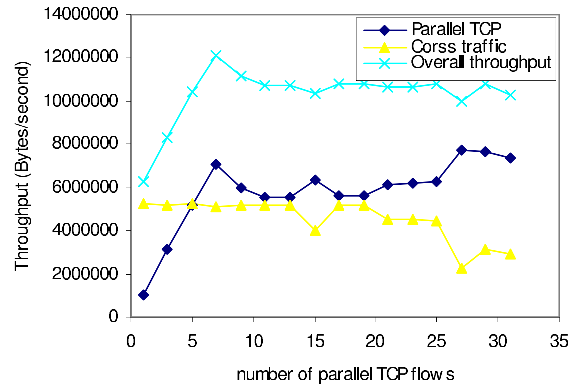


(a) No cross traffic (b) 10 cross traffic

Fig. 7. Simulation results for scenario 5: latency of L_3 is 50 ms; bandwidth of L_3 is 1000 Mbps; TCP buffer is not properly tuned. Refer to Figure 2 for details.



(a) No cross traffic



(b) 5 cross traffic

Fig. 8. Simulation results for scenario 6: latency of L_3 is 20 ms; bandwidth of L_3 is 100 Mbps; TCP buffer is not properly tuned. Refer to Figure 2 for details.

that covered various scenarios and extensive simulations.

IV. Modeling and Predicting Parallel TCP Throughput

In this section we combine our simulation work together with our analytic treatment of TCP performance to develop a model that can be used to predict the throughput of parallel TCP flows in practice. Our approach only needs to send two probes at different parallelism levels and record their throughput. We don't need any additional tools to measure the model parameters such as round trip time and loss rate, which can be hard to obtain in practice as we discussed in Section I.

A. Algorithm

Mathis et al. [30] developed a simple model for single flow TCP Reno throughput on the assumption that TCP's performance is determined by the congestion avoidance algorithm and that retransmission timeouts are avoided:

$$BW = \frac{MSS}{RTT\sqrt{\frac{2bp}{3}}} \tag{1}$$

Here, p is the loss rate or loss probability, and b is the number of packets that are acknowledged by a received message. MSS and RTT are the maximum segment size and round trip time respectively.

Padhye et al. [34] developed an improved single flow TCP Reno throughput model that considers timeout effects. Assuming that the TCP buffer is not the bottleneck (i.e., that the socket buffer size is large or "rightsized" [14]), their model is

$$BW = \frac{MSS}{RTT\sqrt{\frac{2bp}{3}} + T_0 min(1, 3\sqrt{\frac{3bp}{8}})p(1+32p^2)} \tag{2}$$

where T_0 is the timeout.

Bolliger et al [8] show that Equations 1 and 2 are essentially equivalent with packet loss rates less than 1/100, which was validated on the current Internet by Zhang et al [44]. Hacker et al. [18], based on Bollinger's findings, present a model for the upper bound of the throughput of n parallel TCP flows. The authors assume that both MSS and RTT are stable. Hacker's upper bound model can be stated as:

$$BW_n \leq \frac{MSS}{RTT}\left(\frac{1}{\sqrt{p_1}} + \frac{1}{\sqrt{p_2}} + \cdots + \frac{1}{\sqrt{p_n}}\right) \tag{3}$$

where p_i is the packet loss rate for flow i. However, the authors don't provide any mechanism to estimate the loss rate at other parallel levels for prediction purposes. Therefore, the authors acknowledge that the upper bound is tight only when the network is not congested and the loss rate doesn't increase with more parallel TCP flows. The model only has limited utility otherwise.

In our model, we introduce the notion of the number of cross traffic flows, m, and assume that m does not change dramatically over significantly large time periods[3]. Both previous work [45] and our own work on characterizing, modeling, and predicting single flow TCP throughput [28] have shown this assumption to be a valid one. It is widely believed that the TCP throughput shows statistical stability over considerable period of time. Balakrishnan et al found that end-to-end TCP 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 [7]. 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 [33]. Zhang et al [45] used Operational Constancy Region (OCR) to evaluate the temporal locality of end-to-end TCP throughput, which is defined as length of the period where the ratio between the maximum and minimum observed TCP throughput is less than a constant factor ρ. They found that $\approx 60\%$ of OCRs are longer than 1 hour when $\rho = 2$ and $> 80\%$ of all OCRs exceed 3 hours when $\rho = 10$. However, the Internet is dynamically changing and resampling is necessary when the

[3] m is only used for better presentation, but not used in our derivations. In other words, our model doesn't require the knowledge of m.

TCP throughput has significantly changed. Network Weather Service [43] periodically probes the network to resample the TCP throughput. Instead, our work dynamically resamples the path at each OCR [28]. Dynamic monitoring is beyond the scope of this paper and is addressed in our other work [28].

We also assume that all of the parallel TCP flows see the same loss rate and have the same RTT, although both are functions of n and m. These two assumptions have been independently verified [35], as discussed in Section II. We denote with p_n the loss rate after adding n parallel TCP connections, and with RTT_n the round trip time at this point.

Along different paths, the value of MSS can vary ranging from the default 536 bytes to much larger values (for example to support 9000 byte Ethernet jumbo frames on LANs). Our prediction model does not depend on the a priori knowledge of MSS. We do assume, however, that this value does not change after connection establishment. This is a valid assumption as both sides with either use path MTU discovery at connection establishment time [31] or use the default 576 byte path MTU. MSS will directly follow from this.

Based on Equation 1 and the assumptions discussed above, we developed the following parallel TCP throughput model that essentially sums n TCP flows:

$$BW_n = \frac{MSS}{RTT_n} \frac{n}{\sqrt{p_n}} \frac{c_1}{\sqrt{\frac{2b}{3}}} \tag{4}$$

The TCP flows share the same RTT and loss rate and thus the same throughput. Both p_n and RTT_n are actually functions of n and m. Given that we assume m is stable during a period of time, we treat them as functions of n alone. c_1 is a constant in the range $(0, 1]$ that we use to represent the effects of TCP timeouts. In the following, we assume that c_1 is stable for a path over at least short periods[4], so that our model is equivalent to Equation 2. This assumption is firmly supported by the numerous research on the statistical stability of TCP throughput as discussed above.

If we had a model that could compute the relationship between p_n, RTT_n and the parallelism level n based on a small set of measurements, we could then use Equation 4 to predict the throughput for any parallelism level. This is in essence what we do. We developed several parametric models for this relationship based on measurements.

Morris [32] and Qiu, et al [35], [36] independently found that the loss rate is proportional to the square of the total number of TCP connections on the bottleneck link, namely $(m+n)^2$. Through wide area experiments, Hacker, et al [18], [19] showed that RTT on a given path is stable and can be treated as constant. Similarly, we also assume that RTT is a constant during a short period of time. Therefore we have

$$p_n \times RTT_n^2 = a \times (m+n)^2 + b_1 \tag{5}$$

where $b1$ is a constant. Given that m is also a constant, Equation 5 is equivalent to a full order 2 polynomial:

$$p_n \times RTT_n^2 = a \times n^2 + b_2 \times n + c_2 \tag{6}$$

[4] c_1 will be canceled in the following derivations, therefore our model doesn't require the knowledge of c_1.

where $b_2 = 2am$ and $c_2 = am^2 + b_1$. To use Equation 6, we need to send three probes at different parallelism levels to determine the value of a, b_2 and c_2. Clearly, there is a trade-off between the sophistication of the model and the number of measurements needed to fit it. Recognizing this trade-off, we simplified the full order 2 polynomial to a partial order 2 polynomial as shown in Equation 7. This model requires only two probes to determine the parameters a and b.

$$p_n \times RTT_n^2 = a \times n^2 + b \tag{7}$$

Here a and b are parameters to be fit based on measurements. We could further simplify the partial order 2 model to a linear model that also requires two probes.

$$p_n \times RTT_n^2 = a \times n + b \tag{8}$$

We measured the performance of these three alternatives in a wide-area testbed [3], and found that

1) Equations 6 and 7 are better models than Equation 8.
2) The full order two polynomial model (Equation 6) is not significantly better than the partial order 2 polynomial (Equation 7) and can occasionally be worse due to its sensitivity to sampling errors caused by small network fluctuations. Another problem with the full order two polynomial model is that it is sensitive to the choice of probe parallelism.
3) The full order 2 model requires three probes instead of the two needed for the linear and partial polynomial models.

As a result, we use Equation 7 for our system and the discussion in the rest of the paper, unless otherwise noted.

In order to use the model in practice, we have to actively probe a path at two different parallelism levels. The procedure is derived as follows.

We denote $\frac{c_1}{\sqrt{\frac{2b}{3}}}$ in Equation 4 as C. Note that C and MSS are all constants under our assumptions. We define a new variable p'_n:

$$p'_n = p_n \frac{RTT_n^2}{C^2 MSS^2} = a'n^2 + b' \tag{9}$$

Combining Equations 4 and 9, we obtain:

$$BW_n = \frac{n}{\sqrt{p'_n}} \tag{10}$$

Based on Equation 10, we could use the TCP throughput at two different parallelism levels to predict the TCP throughput at other levels. Let n_1 and n_2 be the two parallelism levels that are probed:

$$BW_{n_1} = \frac{n_1}{\sqrt{p'_{n_1}}} = \frac{n_1}{\sqrt{a' {n_1}^2 + b'}} \tag{11}$$

and

$$BW_{n_2} = \frac{n_2}{\sqrt{p'_{n_2}}} = \frac{n_2}{\sqrt{a' {n_2}^2 + b'}} \tag{12}$$

From which we can determine:

$$a' = \frac{\frac{{n_2}^2}{{BW_{n_2}}^2} - \frac{{n_1}^2}{{BW_{n_1}}^2}}{{n_2}^2 - {n_1}^2} \tag{13}$$

and

$$b' = \frac{{n_1}^2}{{BW_{n_1}}^2} - a'{n_1}^2 \quad (14)$$

By substituting a' and b' in Equation 9 with the expressions in Equations 13 and 14, we can now predict the TCP throughput for other levels of parallelism based on Equation 10.

Notice how our prediction requires only *two* TCP throughput probes, one for each of the two different parallelism levels (n_1 and n_2). Both the probing and the calculation process are simple and incur little overhead, the majority of which lies in the communication cost of the two probes. We could use least squares fit or spline fit if we have enough probes, but the overhead is too high to be practical.

B. Evaluation

We evaluated our model extensively through online experimentation on PlanetLab [3], a planetary-scale testbed. We randomly choose 41 distinct end-to-end paths with end nodes located in North America, Asia, Europe and Australia. For each path, we conduct 10 rounds of experiments using Iperf [1] to obtain our measurements. A round of experiment starts with two probes for prediction purposes, immediately followed by parallel TCP transfers with up to 30 parallel TCP flows.

We adopt the *mean relative error* as our performance metric. Relative error is defined as:

$$relativeerror = \frac{prediction - measurement}{measurement} \quad (15)$$

Mean relative error on a path is the average of all the relative prediction errors on the path. Mean relative error for a given number of parallel TCP flows is the average of the relative prediction errors of all the experiments for that number of parallel TCP flows.

Figure 9 shows two examples of prediction using our model. The graphs show the actual and predicted throughput (based on measurements at $n_1 = 1$ and $n_2 = 10$). It can be seen that, for Example 1, predictions made based on the partial order 2 and full order 2 polynomials are virtually identical and have similar accuracy, while the prediction curve derived using the linear model deviates significantly from the measurement curve. In our second example, the prediction made using the partial order 2 polynomial and the linear model are virtually identical and equally accurate. The prediction curve generated by the full order 2 polynomial, however, deviates significantly from the measurement curve.

Figure 10 shows the performance of our parallel TCP throughput predictor using two probes at parallelism levels $n_1 = 1$ and $n_2 = 10$ for a wide range of PlanetLab pairs located all over the world. Only the partial order 2 polynomial model is used here. Each row in the table shows our relative prediction error for a particular Internet path between two hosts. The prediction quality is characterized by the mean and standard deviation of the relative errors at each of the different parallelism levels (ranging between 1 and 30). The results presented in this table are quite encouraging: in most cases, our predictions guarantee us a small mean and standard deviation of relative prediction errors.

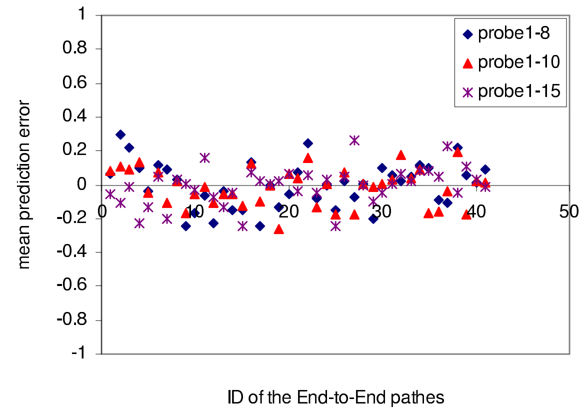


Fig. 11. Prediction sensitivity to the selection of probes.

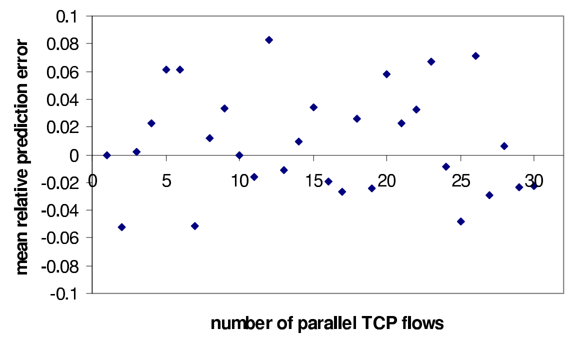


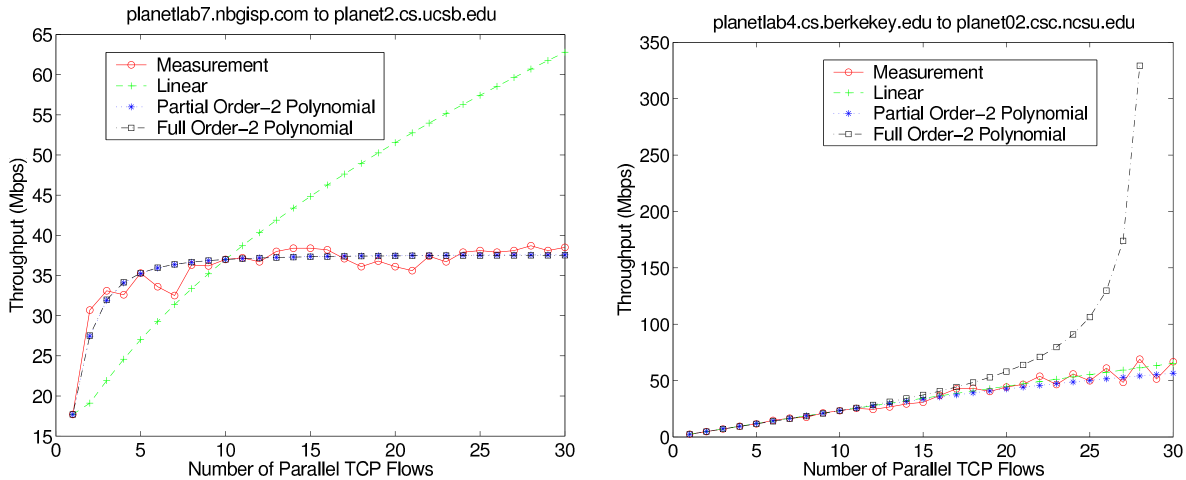
Fig. 12. Relative prediction error for parallel TCP throughput as a function of number of parallel TCP flows.

Our predictor is relatively insensitive to the particular level of parallelism for the probes. Figure 11 shows the mean relative error for our predictor using $(1, 8)$, $(1, 10)$ and $(1, 15)$ parallel probes. We can see that we obtain similar performance in all cases. Of course, it is important not to use parallelism levels that are too close together (such as $(1, 2)$), as such probes are very sensitive to small fluctuations in the network or the existing cross traffic.

As it can be seen from Figure 12, the mean relative error for a given number of parallel TCP flows is not related to the number of parallel TCP flows. The figure, a scatter plot of the mean relative error versus the number of parallel TCP flows, shows no clear trend. The correlation coefficient R between the mean relative prediction error and the number of parallel TCP flows is less than 0.1.

C. Outcome

Our experimental results have shown how, using the model derived in this section, one can effectively predict the throughput of parallel TCP for a wide range of parallelism relying only on two active probes at different levels of parallelism. In the following section we try to estimate the effect of parallel TCP in the existing cross traffic for a given level of parallelism, the last “piece” necessary to make the `TameParallelTCP()` call possible.



(a) Example 1: from nbgisp.com to ucsb.edu (b) Example 2: from berkeley.edu to ncsu.edu

Fig. 9. Parallel TCP throughput prediction. In example 1, the full order 2 polynomial model and partial order 2 polynomial model work well while the linear model deviates from the measurements. In example 2, the partial order 2 polynomial model and linear model work well while the full order 2 polynomial model deviates away from the measurements

Source Host	Destination Host	Mean	Standard Deviation
planetlab2.postel.org	planetlab-1.cmcl.cs.cmu.edu	0.0822	0.1264
planetlab2.it.uts.edu.au	planetlab1.diku.dk	0.1110	0.4061
planetlab1.millennium.berkeley.edu	planetlab3.cs.uoregon.edu	0.0910	0.1325
planetlab-1.it.uu.se	planet2.cs.ucsb.edu	0.1302	0.2599
ds-pl1.technion.ac.il	planetslug3.cse.ucsc.edu	-0.0453	0.1084
planetlab-2.cs.princeton.edu	planetlab9.cs.berkeley.edu	0.07049	0.15274
planetlab2.flux.utah.edu	planet1.cs.ucsb.edu	-0.1081	0.2583
planetlab2.chin.internet2.planet-lab.org	planetlab5.millennium.berkeley.edu	0.0211	0.0594
planetlab2.frankfurt.interxion.planet-lab.org	planet1.cc.gt.atl.ga.us	-0.1676	0.2887
planetlab2.bgu.ac.il	planetlab1.it.uts.edu.au	-0.0533	0.1264
planetlab2.cs.berkeley.edu	planetslug2.cse.ucsc.edu	-0.0138	0.1033
planet1.calgary.canet4.nodes.planet-lab.org	planetlab2.sanjose.equinix.planet-lab.org	-0.1088	0.1331
planetlab2.cs-ipv6.lancs.ac.uk	planetlab4.cs.berkeley.edu	-0.0518	0.1514
planetlab2.cs.uoregon.edu	planet02.csc.ncsu.edu	-0.0522	0.0895
planetlab2.postel.org	planetlab2.it.uts.edu.au	-0.1208	0.2825
planetlab2.flux.utah.edu	planetlab2.chin.internet2.planet-lab.org	0.1235	0.4334
planetlab2.frankfurt.interxion.planet-lab.org	planetlab2.bgu.ac.il	-0.0956	0.2232
planetlab9.millennium.berkeley.edu	planetlab2.cs.berkeley.edu	-0.0070	0.0111
planetlab2.cs-ipv6.lancs.ac.uk	planetlab2.cs.uoregon.edu	-0.2581	0.1828
planetlab-2.it.uu.se	planetlab3.sanjose.equinix.planet-lab.org	0.0664	0.0848
pli1-pa-3.hpl.hp.com	planetlab7.millennium.berkeley.edu	0.0400	0.1542
planetlab1.cs.berkeley.edu	planetlab01.ethz.ch	-0.1546	0.1937
planetlab3.cs.uoregon.edu	planetlab02.cs.washington.edu	-0.1346	0.1068
planetlab7.nbgisp.com	planet2.cs.ucsb.edu	0.0033	0.0394
planetslug3.cse.ucsc.edu	planetlab9.cs.berkeley.edu	-0.2750	0.2743
planet1.cs.ucsb.edu	planetlab5.millennium.berkeley.edu	0.0743	0.2118
planet1.cc.gt.atl.ga.us	planetlab1.it.uts.edu.au	-0.1753	0.3964
planetlab4.millennium.berkeley.edu	planetslug2.cse.ucsc.edu	5.1483e-04	0.0028
planetlab4.cs.berkeley.edu	planet02.csc.ncsu.edu	-0.0136	0.0882
planetlab2.postel.org	planetlab2.cs.berkeley.edu	0.002 9	0.1051
planetlab-2.cmcl.cs.cmu.edu	planetlab2.cs.uoregon.edu	0.0258	0.0664
planetlab2.flux.utah.edu	planetlab3.sanjose.equinix.planet-lab.org	0.1742	0.1491
planetlab2.frankfurt.interxion.planet-lab.org	planetlab2.tau.ac.il	0.03886	0.2650
planetlab9.millennium.berkeley.edu	planetlab1.flux.utah.edu	0.0922	0.1430
planetlab2.cs-ipv6.lancs.ac.uk	planetlab7.millennium.berkeley.edu	-0.1643	0.1345
planetlab-2.it.uu.se	planetlab1.enel.ucalgary.ca	-0.1604	0.1833
s2_803.ie.cuhk.edu.hk	planetlab01.ethz.ch	-0.0375	0 .5193
planet2.pittsburgh.intel-research.net	planetlab02.cs.washington.edu	0.190	0.4300
planetlab2.millennium.berkeley.edu	planetlab1.it.uts.edu.au	-0.1769	0.1695
planetlab1.cs.berkeley.edu	planetslug2.cse.ucsc.edu	0.0200	0.0912
planetlab7.nbgisp.com	planetlab5.millennium.berkeley.edu	0.0093	0.0747

Fig. 10. Relative Prediction Error Statistics for Parallel TCP Throughput.

V. Taming parallel TCP

There are a number of considerable challenges when trying to estimate the effect on cross traffic with an online system running on the end points:

1) The available bandwidth on the bottleneck link(s) is unknown.
2) The number of cross traffic flows and their loss rates and bandwidths on the bottleneck link(s) (the offered load) are unknown.
3) Making use of an additional network measurement tool (such as Pathload [22], [23]) to determine the current load on the path is problematic since it can take a long time to converge. In addition, the measurement accuracy cannot be guaranteed. One would like to avoid any additional overhead beyond the required two active probes necessary to predict the throughput of parallel TCP flows.

In what follows, we make simplifying assumptions about the cross traffic's view of the shared links on the path in order to provide an estimate of impact on the cross traffic from the same two probes from which we derived the throughput curve in the previous section.

A. Algorithm

We assume that all TCP connections, including our parallel TCP flows and the cross traffic, share the same loss rate on a bottleneck link. This assumption is valid as long as one of the two following conditions can be satisfied:

1) The cross traffic has an RTT similar to our parallel TCP flows. In that case, all connections are very likely to have their congestion window synchronized, and thus share the same loss rate. This fact has been independently verified by other research groups [38], [35], [36].
2) The router on the bottleneck link is using Random Early Detection (RED) [15] as its queue management policy, something that is becoming increasingly more common. Research has demonstrated that with RED, different flows roughly experience the same loss rate (the RED rate, which depends on the queue occupancy) under steady state [15], [36].

Our approach to determining the effect of parallel TCP on cross traffic is based on our algorithm to estimate the parallel TCP throughput (Section IV). The key idea is to estimate $p_n \times RTT_n^2$ as a function of the number of parallel TCP flows. Based on the assumption that cross traffic shares the same loss rate as parallel TCP flows, we can then use the simple TCP throughput model (Equation 1) to estimate the relative change to the cross traffic throughput.

Recall in Section IV that we model $p_n \times RTT_n^2$ with a partial order 2 polynomial function $a \times n^2 + b$ (Equation 7). After having obtained the two necessary measurements, we can calculate the value of a and b and are now able to estimate the loss rate as a function of the number of parallel TCP flows.

Relying on our assumptions, we have also obtained the loss rate of the cross traffic as a function of the number of parallel TCP flows n given there are m cross traffic flows (recall that m is relatively stable, see Section IV).

Thus, based on Equation 1, we can now estimate the relative change on each of the individual TCP throughputs without knowing m using the following equation:

$$relc = \frac{\frac{MSS\times C}{RTT_{n1}\times\sqrt{p_{n1}}} - \frac{MSS\times C}{RTT_{n2}\times\sqrt{p_{n2}}}}{\frac{MSS\times C}{RTT_{n1}\times\sqrt{p_{n1}}}} \quad (16)$$

$$= 1 - \sqrt{\frac{p_{n1}}{p_{n2}}} \quad (17)$$

$$= 1 - \sqrt{\frac{a \times {n_1}^2 + b}{a \times {n_2}^2 + b}} \quad (18)$$

Here, $relc$ is the relative throughput change for each flow. Equation 17 shows that all the flows share the same $relc$, namely the relative throughput change. MSS and C are constants as described in Section IV, and RTT_n is stable as was shown by Hacker, et al [18]. $\sqrt{\frac{p_{n1}}{p_{n2}}}$ can be estimated using Equation 7. Both a and b can be obtained with two probes as we discussed in Section IV. Note that n_1 and n_2 can be any parallelism levels. In practise, however, we are most interested in estimating the relative throughput change after adding in n_2 parallel TCP flows in comparison with adding in only one TCP flow, therefore n_1 equals 1 in this case.

In practice, we add another constraint to the `TameParallelTCP()` function to avoid the potential "diminishing returns" problem where more parallel TCP flows bring only marginal benefits. With the `TameParallelTCP()` function, we can estimate the aggregate throughput at any parallelism level. We then check to ensure that the performance gain is over an administrator-determined threshold after adding in an additional TCP flow. If the performance gain is below the threshold, we do not add more flows even when the impact on cross traffic is within the user's limit. This is important because we can avoid the system overhead and network overhead by avoiding unnecessary TCP flows.

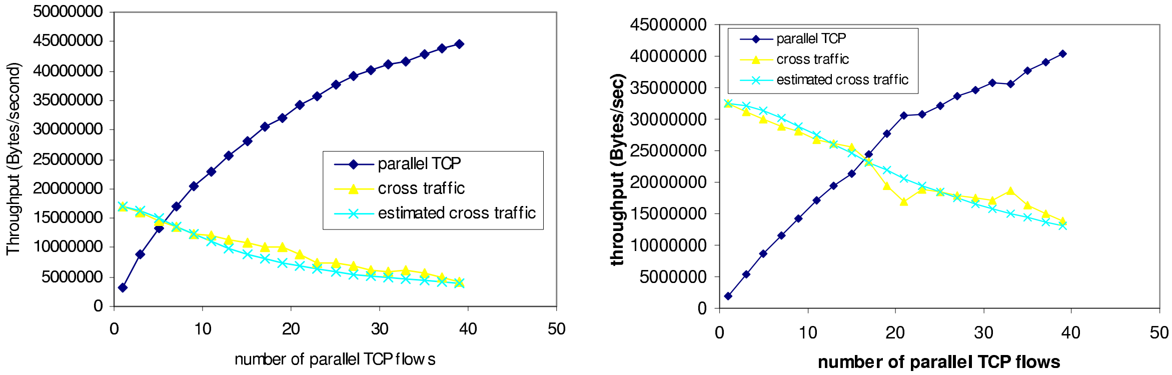
B. Evaluation

We have done a thorough ns2-based evaluation of our cross traffic estimator. Simulation experiments allow us to analyze our estimator under controlled, reproducible settings including bottleneck bandwidth and cross traffic characteristics.

Our simulation configuration was already introduced in Section III. We consider the same set of the scenarios presented there. As in Section III, we employ Qiu et al's [35] simulation topology (Figure 1). The representativeness of this topology was discussed in previous research [35], [36].

Figure 13 shows two examples, for Scenarios 4 and 6, of the performance of our estimator. In these cases we can accurately predict the impact on cross traffic as a function of the parallelism level using only two probes, the same probes we use to predict the throughput of the parallel flows as a function of parallelism level.

We summarize our prediction results as a CDF of the relative error in predicting the impact on cross traffic across all of our scenarios in Figure 14. We can see that 90% of predictions have relative prediction error less than 0.25.



(a) Scenario 4 with 5 cross traffic (b) Scenario 6 with 15 cross traffic

Fig. 13. Examples of cross traffic estimation with simulations in section IV.

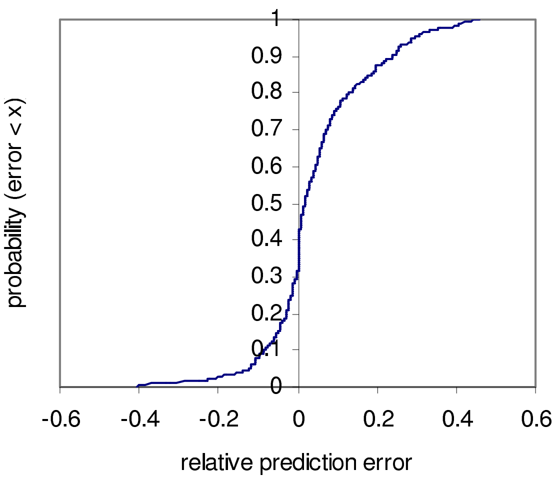


Fig. 14. Cumulative distribution function of relative prediction error for cross traffic estimation for all the simulations with 6 scenarios as described in Figure 2.

The cross traffic estimator is slightly biased. It conservatively predicts a greater impact on the cross traffic on average.

To further evaluate our cross traffic estimation algorithm, we designed a more complicated topology with two groups of cross traffic. The topology and the simulation configuration is shown in Figure 15. Each simulation is 100 seconds long with cross traffic starting randomly between 0 and 8 seconds and all the parallel TCP flows starting at 10 seconds. We applied our same algorithm for estimation of cross traffic. The results are presented in Figure 16, which clearly shows the effectiveness of our approach.

We also tested the cross traffic estimator for scenarios in which different TCP flows have different RTTs, and where RED is not used on the routers. Our estimator shows the right trend of the cross traffic throughput change, though accurate prediction cannot be guaranteed as flows with longer RTT tend to have higher loss rate than parallel TCP flows and vice versa. In essence, in situations in which cross traffic RTT and loss rate is unknown, our estimator is less accurate.

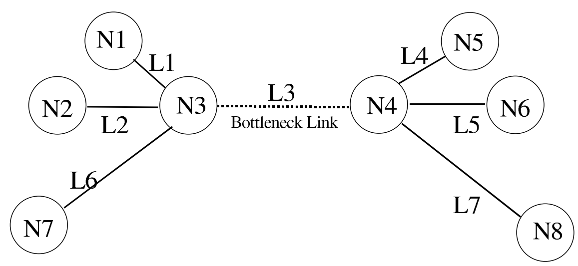


Fig. 15. Simulation topology for further evaluations of cross traffic estimation. L1 and L4 have latency 3ms, L2 and L5 have latency 6ms, L6 and L7 have latency 10ms. L3 have latency 50ms and bottleneck bandwidth 1000Mbits. N3 is using RED queue management policy. Parallel TCP flows go from N2 to N6. Cross traffic group 1 goes from N7 to N8. Cross traffic group 2 goes from N1 to N5.

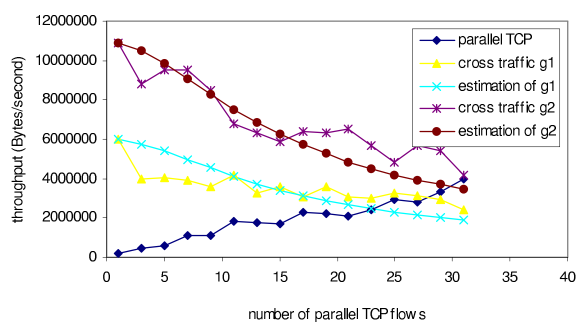


Fig. 16. Estimation results with 14 TCP flows in cross traffic group 1 (g1) and 14 TCP flows in cross traffic group 2 (g2).

C. Outcome

In this section, we have demonstrated the feasibility of predicting the impact on cross traffic of a parallel TCP transfer as a function of the degree of parallelism. Under the assumption that all flows share the same loss rate, we can accurately predict the relative impact using the same two measurement probes used to predict the throughput of the parallel TCP transfer as a function of the degree of parallelism.

Combining these two predictions, we can implement the `TameparallelTCP()` API call:

1) Execute two probes at different parallelism levels.
2) Using the probe results, estimate the parallel TCP throughput as a function of the number of parallel TCP

flows n using the techniques of the previous section.

3) Using the probe results, estimate the relative impact on cross traffic as a function of n using the techniques of this section.
4) Conduct a binary search on the cross traffic impact function, looking for the degree of parallelism, l, that has the largest impact less than that permitted in the API call.
5) Return l, and the impact and throughput predictions at parallelism l.

The cost of this implementation is dominated by executing the two probes.

VI. Conclusions and Future Work

We have shown how to predict both parallel TCP throughput and its impact on cross traffic as a function of the degree of parallelism using only two probes at different parallelism levels. Both predictions are monotonically changing with parallelism levels. Hence, the `TameParallelTCP()` function can be implemented using a simple binary search. To the best of our knowledge, our work is the first to provide a practical parallel TCP throughput prediction tool and to estimate the impact on the cross traffic.

We have made a few simplifying assumptions about the cross traffic in order to predict impact on it while having no knowledge of the actual cross traffic. While these assumptions are reasonable in many cases, we are now working on how to relax them. An implementation of a version of our `TameParallelTCP()` function is available from http://plab.cs.northwestern.edu/Clairvoyance/Tame.html.

Although the Internet paths show statistical stability, the transient stability won't hold over the long term. Either periodical resampling as what NWS [43] does or the dynamic sampling rate adjustment algorithm [28] can be applied for the long term monitoring.

References

[1] http://dast.nlanr.net/projects/iperf/.

[2] http://www.isi.edu/nsnam/ns/.

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

[4] http://www.web100.org.

[5] 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.

[6] Allcock, W., Bester, J., Bresnahan, J., Chervenak, A., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D., and Tuecke, S. Data management and transfer in highperformance computational grid environments. *Parallel Computing 28* (2002).

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

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

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

[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 *ACM Sigcomm* (1999).

[12] Eggert, L., Heidemann, J., and Touch, J. Effects of ensemble-TCP. *ACM Computer Communication Review 30*, 1 (2000).

[13] Fall, K., and Floyd, S. Simulation-based comparisons of Tahoe, Reno and SACK TCP. *Computer Communication Review 26*, 3 (July 1996), 5–21.

[14] Fisk, M., and Feng, W. Dynamic right-sizing: Tcp flow-control adaptation. In *Supercomputing (SC01)* (2001).

[15] Floyd, S., and Jacobson, V. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking 1*, 4 (1993), 397–413.

[16] Foster, I. Globus web page. Tech. Rep. http://www.mcs.anl.gov/globus, Argone National Laboratory.

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

[18] Hacker, T., Athey, B., and Noble, B. The end-to-end performance effects of parallel tcp sockets on a lossy wide-area network. In *16th IEEE/ACM International Parallel and Distributed Processing Symposium (IPDPS)* (2002).

[19] Hacker, T. J., Noble, B. D., and D.Athey, B. The effects of systemic packet loss on aggregate tcp flows. In *IEEE/ACM Supercomputing* (2002).

[20] Hacker, T. J., Noble, B. D., and D.Athey, B. Improving throughput and maintaining fairness using parallel TCP. In *IEEE Infocom* (2004).

[21] 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).

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

[23] Jain, M., and Dovrolis, C. Pathload: A measurement tool for end-to-end available bandwidth. In *Passive and Active Measurement Workshop* (2002).

[24] 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).

[25] 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.

[26] Lee, J., Gunter, D., Tierney, B., Allcock, B., Bester, J., Bresnahan, J., and Tuecke, S. Applied techniques for high bandwidth data transfers across wide area networks. In *International Conference on Computing in High Energy and Nuclear Physics, Beijing, China, September 2001.*

[27] 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.

[28] Lu, D., Qiao, Y., Dinda, P. A., and Bustamante, F. E. Characterizing and predicting tcp throughput on the wide area network. Tech. Rep. NWU-CS-04-34, Northwestern University, Department of Computer Science, 4, 2004.

[29] Mathis, M., Heffner, J., and Reddy, R. Web100: Extended tcp instrumentation for research, education and diagnosis. *ACM Computer Communications Review 33*, 3 (Julyt 2003).

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

[31] Mogul, J., and Deering, S. A framework for defining empirical bulk transfer capacity metrics, rfc3148, November 1990.

[32] Morris, R. TCP behavior with many flows. In *ICNP* (1997), pp. 205–211.

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

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

[35] Qiu, L., Zhang, Y., and Keshav, S. On individual and aggregate TCP performance. In *ICNP* (1999), pp. 203–212.

[36] Qiu, L., Zhang, Y., and Keshav, S. Understanding the performance of many TCP flows. *Computer Networks 37*, 3–4 (2001), 277–306.

[37] 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).

[38] SHENKER, S., ZHANG, L., AND CLARK, D. Some observations on the dynamics of a congestion control algorithm. *ACM Computer Communication Review* (1990).

[39] SIVAKUMAR, H., BAILEY, S., AND GROSSMAN, R. L. PSockets: The case for application-level network striping for data intensive applications using high speed wide area networks. In *Supercomputing* (2000).

[40] SMITH, F. D., HERNANDEZ-CAMPOS, F., JEFFAY, K., AND OTT, D. What TCP/IP protocol headers can tell us about the web. In *SIGMETRICS/Performance* (2001), pp. 245–256.

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

[42] TIERNEY, B. Tcp tuning guide for distributed application on wide area networks. *USENIX & SAGE Login 26*, 1 (2001).

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

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

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